IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 October 2023, accepted 20 November 2023, date of publication 23 November 2023,
date of current version 29 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3336287

APPLIED RESEARCH

Remote Labs Meet Computational Notebooks: An Architecture for Simplifying the Workflow of Remote Educational Experiments

OSWALDO VANEGAS-GUILLÉN[1,2], (Member, IEEE), PABLO PARRA-ROSERO[3], JAVIER MUÑOZ-ANTÓN[1], JOHANNA ZUMBA-GAMBOA[4], AND CARLOS DILLON[5]

[1]Department of Energy Engineering, School of Industrial Engineering, Universidad Politécnica de Madrid, 28006 Madrid, Spain
[2]Department of Information Technologies, Faculty of Mathematical and Physical Sciences, Universidad de Guayaquil, Guayaquil 090514, Ecuador
[3]Industrial Processes Research Group (GIPI), Universidad Politécnica Salesiana, Guayaquil 090101, Ecuador
[4]Department of Computational Systems, Faculty of Mathematical and Physical Sciences, Universidad de Guayaquil, Guayaquil 090514, Ecuador
[5]Department of Computer Science, Universidad Politécnica Salesiana, Guayaquil 090101, Ecuador

Corresponding author: Oswaldo Vanegas-Guillén (oswaldo.vanegasg@ug.edu.ec)

This work was supported in part by Universidad de Guayaquil under Grant FCI-007-2021; in part by the Industrial Processes Research Group (GIPI), Universidad Politécnica Salesiana del Ecuador; and in part by the Department of Energy Engineering, School of Industrial Engineering, Technical University of Madrid.

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Human Research Ethics Committee of the University of Guayaquil, and performed in line with the Organic Law of Personal Data Protection of Ecuador.

ABSTRACT Online laboratories and computational notebooks have established themselves as essential tools in the fields of engineering and science education, significantly enhancing the teaching and learning experience. Despite the benefits of remote laboratories, and although architectures, models, and tools have been proposed to facilitate the work involved in their implementation, challenges remain in the development lifecycle. This paper proposes a novel architecture that simplifies the development and management of remote experiments using a publisher-subscriber communication paradigm to securely and efficiently integrate WebAssembly computational notebooks. This approach eliminates the need for additional infrastructure for communication at the lab stations. It also avoids the need for servers to deploy notebooks, because all operations, calculations, and data processing run locally in the user's web browser. To achieve this, a remote laboratory management system (RLMS) interoperable with virtual learning environments was designed and implemented, including notebook-based authoring, learning scenarios, and grading tools. As a case study, remote experiments were developed to characterize and evaluate the performance of heat exchangers using a thermal fluid systems test bench built for this project. Several quality criteria categorized into technical, educational, and adaptability groups were quantitatively evaluated using the responses of 70 participants from two different universities. The analysis highlighted both the advantages and challenges of the proposal, with a significant emphasis on interaction, scalability, reusability, interoperability, and accessibility. The results confirmed the effectiveness of the notebooks in each phase of the remote experiments, demonstrating an increase in students' active and autonomous participation and exploration.

INDEX TERMS Architecture, computational notebook, engineering education, online experimentation, online laboratory, remote laboratory, thermal fluids.

I. INTRODUCTION

The onset of the COVID-19 pandemic catalyzed a significant development in infrastructure and technologies aimed at remote access to learning and research resources, including

The associate editor coordinating the review of this manuscript and approving it for publication was Nikhil Padhi.

remote laboratories, which still bring a number of challenges that are not present when accessing other types of digital-only resources online, such as accessibility, resource usage, and security, among the main ones [1]. A remote laboratory is a set of devices arranged to perform experiments at a distance by monitoring and controlling these devices over a network, usually the Internet [2]. Remote labs offer a practical way to gain hands-on and real-world experience [3], [4], which provides students with the experimental work they need to succeed in engineering, science, and technology learning [1], [4]. Remote labs are an evolving technology [2] that has proven to potentially offer benefits such as availability, observability, accessibility, shareability, safety, resource optimization, and cost advantages [4], [5].

Resource constraints are a common factor in education in numerous regions worldwide. Remote laboratory systems require not only resources to acquire equipment, but also infrastructure, development, and maintenance, which tend to increase as more users are able to access the system. Ultimately, all of these factors translate into difficulties for developers and lab administrators in providing experiments to lab users.

To develop remote laboratories, various approaches or models can be followed, with Laboratory as a Service (LaaS) [2] being the most common, which is also the one used by the IEEE Standard for Networked Smart Learning Objects for Online (IEEE Std 1876-2019) [1]. With the LaaS approach, labs must develop infrastructure to be offered through services and functionalities through a lab server, which requires access to a public Internet protocol (IP) address to offer the lab through the Internet. This can represent limitations for many laboratories with limited network infrastructure and development resources, apart from problems with institutional security protocols and policies [6], [7]. Alternatives to the self-managed lab server have been presented, such as the Laboratory Infrastructure as a Service (LIaaS) model [8] with its Experiment Dispatcher, or the NCSLab [9], [10], [11] with its Experiment Server. However, even though these architectures do not require local infrastructure with public access to offer the laboratory resources, it is still necessary to abstract laboratories as servers, a service logic must be managed, and the lab still must be adapted to be compatible with these servers. Similarly, resources for teachers and lab administrators to create, design, reconfigure, and re-engineer educational experiments [12] are generally unavailable. The educational nature of these experiments requires the sessions be supplemented with educational resources. This is usually intended to be solved with the integration of remote laboratories within virtual learning environments (VLE); however, this integration does not guarantee that appropriate learning resources will be provided to assist students in performing the required activities of the experiment. Although certain tools can be used to assist in the creation of experiments, they are external tools that are not integrated into the flow of the rest of the system, thus complicating the workflow for the experiment author. This lack of authoring tools for remote laboratory systems was documented in [12].

The main objective of this study is to present a remote lab architecture called RemoteLabo, which facilitates the process of integrating, creating, publishing, and communicating educational experiments powered by computational notebooks. This architecture has two main aspects to allow this facilitation. First, it decouples the communication mechanism of remote labs from the lab station itself, eliminating the interdependence between the lab station and the client [13], which simplifies the integration and communication of labs for developers and administrators who do not need to develop communication servers or manage services at the lab station but only communicate the equipment data. Second, it was designed to integrate computational notebooks as a tool for facilitating the entire lifecycle of an experiment for administrators and teachers. Aspects such as interactivity [14], replicability [15], shareability [14], [16], and their explorative nature make notebooks potential tools for creating educational remote lab experiments [17], [18]. For this workflow, an RLMS is proposed and implemented with tools for the development lifecycle of experiments powered by computational notebooks for teachers and students. The RemoteLabo architecture is designed to integrate both the proposed approaches, computational notebooks, and its decoupled communication that links labs with notebooks in a secure manner. The present work is substantiated by the previous authors' experiences in the TermoLabo project [19], a remote lab system for the teaching of thermal fluids. To evaluate the proposed architecture, experiments were implemented on a thermal fluid systems test bench as lab station involving students and teachers from two courses at two universities. The tool used for the evaluation was adapted from the UNE 71362:2020 standard [20], which provides a model for evaluating the quality of digital educational materials. The evaluation criteria covered educational, technological and accessibility aspects. The results obtained favor the proposed architecture, underlining its potential in the educational field.

The rest of the paper is organized as follows: Section II reviews the state of the art of remote laboratories and their different approaches and architectures. Section III describes the architecture of RemoteLabo, how its approach facilitates lab admin and developer work, and how it is designed to integrate computational notebooks. Section IV presents computational notebooks as tools for remote educational experiments. Here, we detail the workflow of the RemoteLabo experiments, from lab integration to the grading of experiment assignments by the teacher. Section V shows the implementation of the proposed architecture and a case study of a thermal fluid lab station that uses the authoring system to create its experiments. Section VI evaluates technological, educational, and accessibility aspects on students and teachers and shows the results of the evaluation and its analysis. Section VII presents the conclusion, and Section VIII discusses future work.

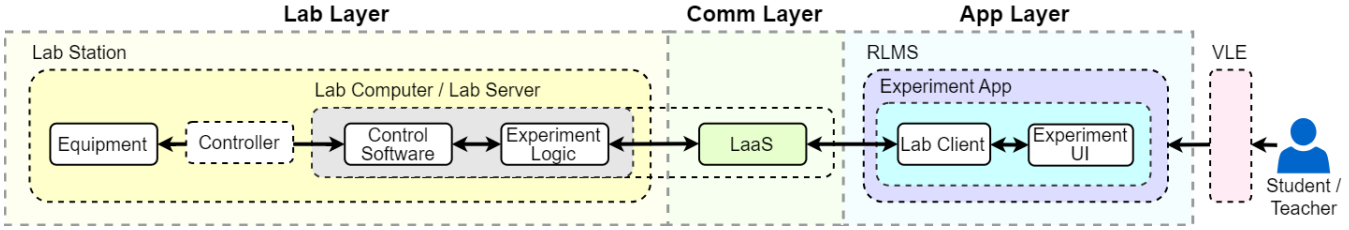

FIGURE 1. General architecture of a remote lab system integrated into a virtual learning environment (VLE): lab layer, communication layer, and application layer. In the case of LaaS, the lab communication responsibility is part of the lab computer.

II. BACKGROUND AND RELATED WORKS

To understand the general structure and functionality of remote laboratory systems we can represent them in a general architecture with three layers: laboratory layer, communication layer, and application layer, as illustrated in Fig. 1. We can identify these three basic layers in every remote laboratory system, but each specific architecture implements various approaches according to its requirements, using specific technologies, and composing the logical layers as needed.

A. LAB LAYER

The Lab Layer comprises various physical laboratories connected to the system. The key components of a physical laboratory are the collection of laboratory equipment and the lab computer. In this study, we refer to this setup as a lab station [21], which contains the physical elements required for the execution of an experiment [21]. The lab station is also referred to by various other names, such as experiment rigs [22], test rigs [3], [9], or merely remote laboratory [1]. The control of the equipment is commonly provided by the control software that runs the lab computer [23]; however, some experiments might also require a dedicated controller. As a result, some architectures unify the roles of the controller and the lab computer, commonly referred to as the lab server [1] or simply the controller [21], [22], whereas other architectures explicitly separate the controller from the lab server [3], [9], [10], [24].

B. COMMUNICATION LAYER

The communication layer plays a key role in remote lab systems by connecting the lab station to the final user. For this layer, most remote lab systems use a client-server approach [22] to communicate the lab station with the user. In this approach, the lab computer becomes a server, commonly named "lab server" which exposes the lab resources to the outside world [1]. The lab server commonly has three responsibilities: being connected to the lab equipment and controlling it through software [23], running the experiment logic [25], and communicating the lab resources outside [1].

The Laboratory as a Service (LaaS) model [23] proposed to deliver the lab resources as well-defined services that enable interoperability [1], [2]. These services are exposed on the lab server, and the client can access the experiments to consume these services. The IEEE 1876-2019 standard defined an online lab following LaaS, with the help of metadata for standardization and interoperability.

Although approaches other than LaaS have also been taken, such as the already named Experiment Dispatcher and NCSLab, a further approach is to completely prescind from the laboratory server and use another mechanism, instead of the client-server approach, for the communication of the lab station. A widely used alternative model is the publish–subscribe communication paradigm, which allows distributed, asynchronous, and fully decoupled communication between communicating entities [26], [27]. In publish–subscribe communication, message consumers subscribe to events that interest them, and message producers publish events anonymously and asynchronously [28]. Subscribers are notified of events that match their interests and receive the data associated with the event [26], [28]. A dispatcher, commonly called a broker [29], is responsible for message forwarding. When a lab server is responsible for lab communication, it forms a part of the communication layer. On the other hand, if the responsibility for the lab station communication is passed to an external entity, such as to the broker in the publish-subscribe approach, the communication layer is completely independent of the lab layer.

C. APPLICATION LAYER

The user accesses the laboratory from the application layer. In a simple scenario, the application layer can consist uniquely of the experiment application that users interact with to perform the experiment [1]. This application could be a local installable one or, more commonly, a web application accessed from a web browser [2]. Users interact with the lab station through the experiment user interface (UI), which is a graphical interface with visual elements with which the user can interact to monitor and control the laboratory equipment. These interactions are transmitted to the lab station thanks to the Lab Client, which provides data to users as they require it, and sends the user control commands to the lab station [24], [30], [31].

With respect to the deployment of the experiment app, three main scenarios are possible, as depicted in Fig. 2. In Fig. 2(a), the lab client program and experiment UI are in the same program, which runs in the browser. This is the least common and scalable scenario. In Fig. 2(b), the lab client

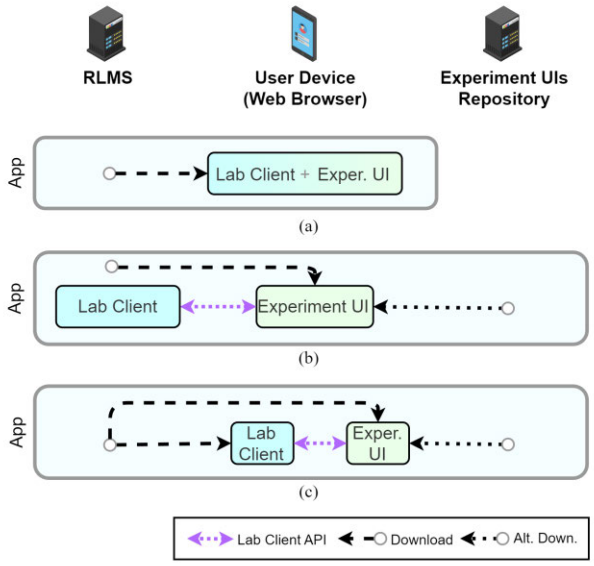

FIGURE 2. Deployment cases of the experiment app.

and the experiment UI are separated. Only the experiment UI runs on the web browser and communicates with the lab client through an API provided by the latter. In Fig. 2(c), they remain separated, as in (b), but now both run in the browser. Because of the decoupling of the lab client and the experiment UI, both Fig. 2(b) and (c) have more flexibility. In both cases, the experiment can be provided by the same server that also provides the lab client but could also be provided by an external entity, such as an experiment repository. In Fig. 2(a), as the lab client forms part of the same code as the Experiment UI, the creator of the latter must also implement the required code of the former. In Fig. 2(b) and 2(c), lab client becomes completely independent from any particular experiment UI, all types of experiment UIs can be adapted to communicate with the same client through an API.

Another key element in the application layer of modern remote laboratory systems is the remote laboratory management system (RLMS), also known as Remote Laboratory Broker, or Online Laboratory Management System (OLMS) [6], [21]. The RLMS was born of the necessity of centralizing the management and communication of remote laboratory stations. Every lab station requires a group of management tasks to be used properly, such as the administration of lab resources, the time management of the resources being used, or the authentication and authorization of lab users [32]. The RLMS brings a common framework to lab stations [8], [33], in which they can delegate responsibility for the development and deployment of these features and tools to the RLMS. This reduces the effort of development [8], [32] because every lab station integrated with an RLMS automatically has the management features and tools provided by it [32] and only concentrates on the development of experiment-related features [8]. The RLMS concentrates, manages, and shares lab stations [4] and their experiments [21]. This allows the formation of repositories [1] or hubs of experiments (as shown in Fig. 2) from different institutions around the globe from which users can choose and practice.

The main role of RLMS, which gives it its name, is the management of lab stations and experiments. However, RLMS also hosts the experiment app [22], as illustrated in Fig. 3. For lab station management, the administrator can access the manager module through the UI. In order for the lab station to be integrated into and managed by the RLMS, the lab administrator uses the API provided by the RLMS [8]. RLMSs can also provide development tools to facilitate the integration process [32]. It is common that the management communication uses the HTTP protocol while the experiment communication, performed by the experiment app, uses a near real-time communication protocol, typically transported over WebSocket as recommended by [1] and as used by [34] and [35].

Remote educational experiments are carried out within a learning context. Experiments is the name given to the learning activity made with labs acting as learning objects [1]. A learning object is defined as an entity that can be used for educative purposes. Reference [1] provides recommendations for integrating online laboratories, which act as learning objects, in learning environments. These learning environments are provided neither by the lab station nor by the RLMS, since learning is outside their scope [33], but by specialized systems for these purposes, called virtual learning environments (VLE). Because remote labs do not provide an environment for learning, it is necessary to integrate them into VLEs [33] to provide pedagogical value [1]. This integration between the learning environment and remote laboratory experiments is a typical feature assigned to RLMSs [33]. Once integrated into a VLE, RLMS can benefit from the educational context to improve students' and teachers' learning experiences. Users that come from the VLE can be identified without needing any extra registration because VLEs provide this information, which can also be used to personalize the experiment UI [30]. One of the most used technologies for RLMS-VLE integration is the Learning Tools Interoperability (LTI) specification [36], which enables the integration of educational applications, called tools, into VLEs. With LTI, students can also submit and be graded on assignments, all from within the RLMS. Once integrated the experiment and used from a VLE, Student's behavior and actions are invaluable sources of feedback for technical and pedagogical purposes. The Experience API (xAPI) [37] is the technology recommended by [1] for tracking learning activities. Using xAPI, RLMS can monitor student behavior and actions during the experiment activity. These records are used for two main purposes: Learning analytics (for educational purposes), and user experience feedback (for more technical purposes).

For the creation of experiments and their content two kinds of tools are used: *programming tools* and *authoring tools*. Programming tools deal with code and commonly require technical knowledge for their usage.

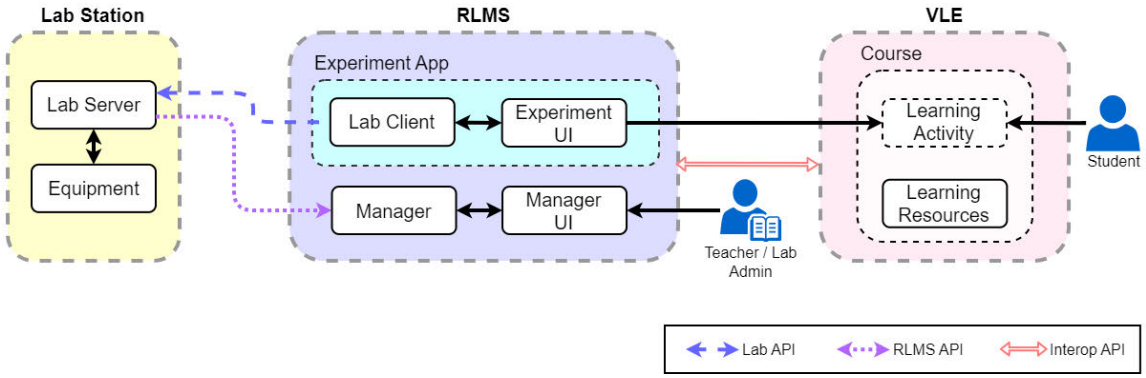

FIGURE 3. General overview of an RLMS integrated into a VLE. The student accesses the experiment through a learning activity added by the course teacher.

An example of a widely used programming tool is Node-RED, which allows the creation of programs using drag-and-drop nodes and has also been used to create experiment UIs [38]. On the other hand, authoring tools allow for the creation of resources in environments that commonly require less technical knowledge than programming tools. One common approach is to use programming tools for the creation of the experiment UI and authoring tools for the creation of the learning content of the experiment UI. There are also authoring tools specifically designed for composing experiments, that is, experiment authoring tools. These tools can be optional features of RLMSs [21], and their capabilities vary between tools. An important example of an experiment authoring tool is Easy Java/JavaScript Simulations (EjsS) [39], [40], which was developed as a modeling and authoring tool for science simulations and evolved as a tool capable of creating experiment UIs for remote laboratories using web technologies. However, there is a type of tool that has the capacity to merge in the same environment the capabilities of both programming and authoring tools, as well as a high potential to be used as learning objects; we are referring to *computational notebooks*.

D. NOTEBOOKS

Computational Notebooks [15] or "Interactive Notebooks" [41], are interactive, literate programming documents [42] designed to integrate code, visualizations, text, and other rich media to support the construction and sharing of computational and interactive narratives [16]. Despite having been born in a scientific computing context [16], [43], [44], notebooks expanded outwards this scientific root, becoming interdisciplinary and ubiquitous [14], and experiencing a renaissance in recent years [15], [45]. Computational notebooks are used in many areas, such as industry, education, data science, computer science, and machine learning [14], [46]. Notebooks are executed and created using software applications that take on the role of notebook authoring tools. It is common to refer not only to the notebooks themselves, but also to these software applications as computational notebooks [46], [47]. Examples of these applications include Mathematica,[1] Apache Zepellin,[2] and Jupyter Notebook.[3]

Interactivity is the key feature of notebooks. Notebooks are considered interactive applications rather than just conventional documents [14], [48], [49]. Furthermore, rather than just authoring tools, notebook environments are also programming tools and environments [50]. The type of software built with notebooks is meant to be shared with other users rather than deployed in the traditional software lifecycle [14]. Notebooks encourage interactive computation, in which the user interacts iteratively with the program [14], [47] in an explorative conversation, making hypotheses and analysis, thinking, modifying, and repeating [15]. Notebooks present a sandbox for exploration and experimentation, where goals are not required to be predefined, as they are discovered while iteratively gaining understanding [14].

Computational Notebooks are becoming a standard not only for research, but also for the teaching community. Every named aspect of notebooks can also be leveraged in education. The interactive nature of notebooks has the potential to engage students [15], allowing them to enhance the learning process and their experimental skills [17], creating a student-centered, technology-rich learning environment [51]–an approach that shifts the traditional paradigm of education, which often focuses on the teacher as the primary source of knowledge. Instead, it places the student at the center of the learning process, allowing, and encouraging active participation, exploration, and autonomy in learning. Computational notebooks are used to create interactive guides, tutorials, manuals, textbooks, and worksheets [15], allowing teachers to illustrate abstract concepts using virtual demonstrations and interactive resources.

Notebook interactivity is powered by code, which students can execute and modify in the same environment. Code is required for many processes in engineering learning; functions, formulas, and models can be explored through code,

[1] https://www.wolfram.com/mathematica/
[2] https://zeppelin.apache.org/
[3] https://jupyter.org/

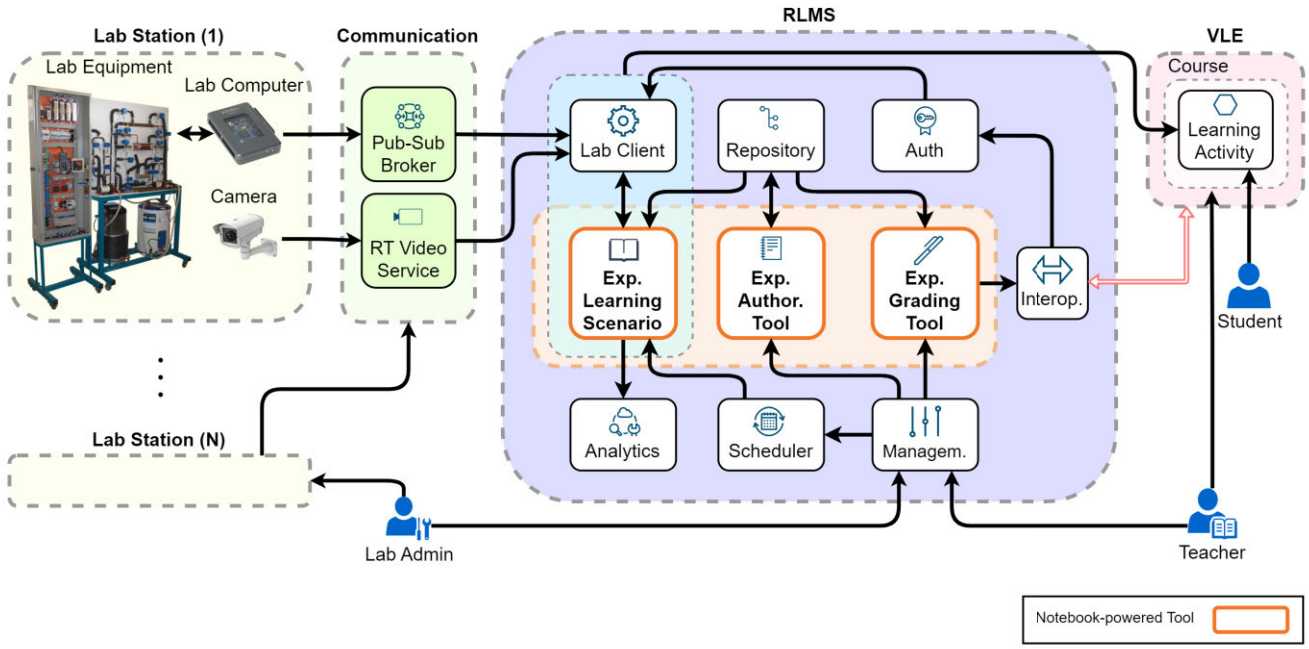

FIGURE 4. General overview of the RemoteLabo architecture, integrated into a VLE. The three main tools for experiments communicate with the pub-sub broker and are powered by computational notebooks.

generating new insights with data and visuals. All academic fields necessitate code to collect and analyze data [16]. Computational notebooks are one of the main tools used to learn how to solve complex problems with a programming language, thus reinforcing computational thinking [52], the thought processes to solve problems as computer science does [53]. Computational thinking is a relevant skill for professionals in all domains [52] and is no longer exclusive to engineers and computer science students [53]. Teachers can also require students not only to apply code to solve specific problems but also to explain that code, with the help of the rich content available in notebooks [50]. These notebooks with explanations can become computational essays that incorporate code to support their theses [50]. Notebooks are used in various pedagogical scenarios and fields [48], [53], with different applications and systems involving student and teacher collaboration [54], used to create educational web apps [48], and research web apps [49].

Since its inception, there has been a relationship between computational notebooks and laboratories. Computational notebooks appeared as digital laboratory notebooks to support the laboratory workflow [15], [16], [50]. In terms of remote labs, although scarce, there are studies implementing computational notebooks for remote equipment or as support for this remote working. In [17] and [18], remote laboratories used notebooks to interact with remote equipment. In [55], notebooks were used as auxiliary tools in a digital twin laboratory. ''IoT Notebooks'' are proposed in [47], whose back-end part can detect, identify, and connect to physical IoT devices.

Jupyter [16] is an open-source project that develops an environment of products around notebooks, with Jupyter Notebook as the core product [14]. Jupyter Notebook is Jupyter's document format and notebook environment, which is considered the standard and most commonly used computational notebook for education and research [15], [46], [54]. The open nature of Jupyter allows the possibility of extending the software base capabilities [15] but also manipulating the notebooks themselves with custom tools, given the open nature of the document format [14], which is stored in plain JSON format. JupyterLab[4] is a new-generation notebook user interface based on the classic Jupyter Notebook [14]. JupyterLite is a ready-to-be-used distribution of JupyterLab that runs entirely in the web browser [56]. This distribution provides a lightweight computing environment that can be accessed quickly without the need to set up any server or install dependencies on the user's device. This is possible owing to WebAssembly,[5] an open technology that makes it possible to run in the browser code of programming languages not made for the browser. To execute code in a conventional Jupyter environment, the code of the user is passed to the server, which executes the code using specialized processes called kernels [14] and then returns the response to the user [15]. In JupyterLite, this process is performed entirely within the user's browser.

[4]https://github.com/jupyterlab/jupyterlab
[5]https://webassembly.github.io/spec/core/

III. RemoteLabo ARCHITECTURE

The architecture of RemoteLabo is the cornerstone on which the functionalities of the system rest. Its main goals are the conception of an environment in which experiments are treated as first–class citizens for the system, and to provide a mechanism to optimally communicate these experiments. RemoteLabo's architecture aims to achieve these goals without neglecting the commonly desired remote lab quality attributes, such as accessibility, shareability, security, maintainability, scalability, interoperability, and optimization of resources and costs. A remote laboratory system requires the synergic work of various subsystems developed by different entities, which are integrated to effectively communicate the lab stations with students from different VLEs. To fulfill these goals, we established the following group of requirements to meet:

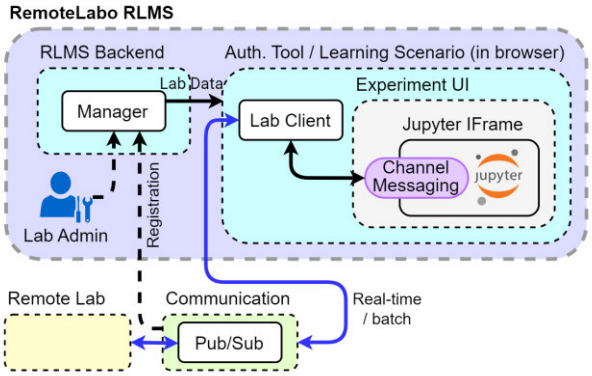

FIGURE 5. Experiment communication scheme: Signals are sent to the RLMS for registration. Once registered, the corresponding signal metadata is relayed to the experiment tools, allowing the lab client to manage signals in real-time communication.

A. RemoteLabo's ARCHITECTURE OVERVIEW

- *Lab stations without servers*: Lab administrators should be able to integrate and communicate with the lab station without building or operating infrastructure. For this, the role of communication commonly assigned to the lab server must be assigned to an external entity.
- *Experiment authoring Tool*: Teachers and lab administrators should be able to create and modify experiment user interfaces from a familiar computational notebook environment, in a collaborative way, integrated into the same remote lab system.
- *Experiment capabilities*: Created experiments should be able to connect in real-time with different lab stations and enrich the experiment activities with the required pedagogical content within the experiment.
- *Experiment Learning Scenario*: Students should be able to perform all the experiment activities without leaving the learning scenario or changing the context.
- *Experiment Grading Tool*: Teachers should be able to establish assignments for experiment sessions according to the experiment goals, and review and provide feedback on these student assignments. Students should also be able to review both the grade and the teacher's feedback from the virtual learning environment.

To address these requirements, RemoteLabo's architecture, illustrated in Fig. 4, implements two key solutions: the implementation of the publish-subscribe communication model for decoupling communication from the lab station, and the integration of computational notebooks to allow an entire collaborative lifecycle of experiments in which admin, teachers, and students are involved. Both the solutions are detailed below.

B. PUBLISH–SUBSCRIBE COMMUNICATION FOR REMOTELABS

To facilitate the communication of lab experiments, RemoteLabo proposes to separate the communication responsibility from the lab station using the publish–subscribe communication paradigm. The publish–subscribe paradigm is used for near real-time communication with minimal latencies [13]. Unlike the HTTP protocol used in web services, which is synchronous and used for 1-to-1 communication, the asynchronous publish–subscribe model is optimal for the one-to-many communication [34] required for remote lab systems, as one lab station publishes data to many users, and users can also publish data to the lab station and receive immediate feedback on their interactions with lab equipment and procedures. In contrast to the request/response approach of the client–server architecture, in the event–driven approach of publish–subscribe [57], the lab station can transmit data without waiting for a request by users [29], whereas they can receive data without continually requesting it. Lab stations can also react to data from users only when they arrive, without the need to continually check for it.

The main effect achieved by implementing the publish-subscribe model is the elimination of the interdependence between the lab station and client [13], and there is no direct communication between the lab and RLMS. In other words, the lab station controller is a black-box system for the RLMS [22]. With this approach, RemoteLabo follows a basic principle for RMLS interoperability: The RLMS should know only the minimum of the lab station design [58]. This separation removes the responsibility of implementing communication logic for the lab station and reduces the complexity of the code and the requirements required for interconnecting the lab. This allows for the independent development of both the components. Because both are connected to the broker, they do not need to know each other to communicate [29]. Every part publishes or subscribes according to its role and permissions, and without requiring knowledge of the mechanism of the other part. A lab station does not need to develop servers or manage services, which relieves the lab equipment of the computational work required to run and expose computational services [58]. To transmit the lab station data, administrators only need to use a publish-subscribe client with a security certificate

obtained from RemoteLabo, which will grant the communication. Lab administrators do not need to manage security at this level because only certificate-authorized entities have access to the broker. The lab administrator is free to choose where and how to deploy the client program that connects to the RemoteLabo communication layer, along with the corresponding internal validations and security measures for their equipment. As with any other approach, the cyber and physical security of the lab station and its equipment is the responsibility of the lab administrator. Aspects such as the validation of the signal and rules according to the input are expected to be implemented in the lab station. RemoteLabo is agnostic to the logical location of computing that powers these services, which can run in a self-managed infrastructure or in the cloud.

The RLMS centralizes lab stations, their experiments, signals, and functionalities. Lab station signals can be automatically or manually registered from the RLMS lab admin panel. To monitor them, all the signals that the RLMS receives from the lab station are automatically registered along with their corresponding identification and data type. To control signals, the lab administrator can indicate to the system which of the registered signals from the registered ones are of control or register a new one. From the RLMS lab admin panel, lab administrators can establish settings related to signals such as limits or alarms. Optionally, lab admins can register lab station functionalities that the user can activate using the same communication mechanism. All registered signals and functionalities will be available for the experiment authors to create experiment UIs connected to lab stations.

C. INTEGRATED COMPUTATIONAL NOTEBOOKS

RemoteLabo uses computational notebooks, specifically Jupyter Notebooks, as the fundamental tool to develop and use experiment UIs. RemoteLabo implements three main tools to manage experiments: for experiment creation, the experiment authoring tool; for experiment usage by students, the experiment learning scenario; and for experiment grading by the teacher, the experiment grading tool. The former two tools communicate with the remote lab station; experiment creators set up and test the communication, and students use that communication, whether in real-time or batch mode. Fig. 5 illustrates the communication process for both tools. Owing to the decoupling of the experiment UI from the lab client, these tools can easily communicate with lab stations. The experiment creator connects the created experiment to the lab client, without technical procedures, using the experiment authoring tool. In doing this, communication is set up for when the students use the newly created experiment in the learning scenario. Note that while using these tools, the messages from the lab stations containing the signal data do not pass by the RLMS back-end but are communicated directly from the communication layer to the experiment UI running inside the user device.

Both tools form part of the RemoteLabo RLMS and feature a customized Jupyter Lab embedded in an IFrame. Web browsers offer the Channel Messaging API,[6] which allows the communication of, among other scenarios, a webpage and its embedded IFrame, assuming that both are previously programmed for this. This API allows both parts of the tool (IFrame and the rest of the tool) to communicate immediately without leaving the browser. The lab client is abstracted and exposed in a user-friendly way to the experiment UI authors through the Channel Messaging, which takes the role of the API that intercommunicates the decoupled lab client and UIs. The authoring tool offers the author this API to allow visual and interactive elements to effectively monitor and control the lab station. It enables the creation of pedagogically self-contained experiments without the implementation of the lab client or knowledge of the lab client's operation, which opens the possibility that users without technical knowledge can create new experiment UIs and easily access the lab client to communicate the experiment UI with the lab station.

D. VLE INTEGRATION AND USERS ACCESS

The RemoteLabo RLMS integrates into VLEs, using LTI to receive teachers and students with their learning context without requiring any registration step, and using xAPI to support learning analytics. RLMS uses a group of services to offer all its functionalities. RemoteLabo uses a role-based access control [59]. The user authentication process follows a different flow for each user role: lab administrators, teachers, and students. RemoteLabo also adds the role of the system administrator, which is the user with the highest permissions and the responsible for the system itself. System administrators and lab administrators are not affiliated with an institution from the system perspective and have higher permissions in the system than teachers and students. Because the experiment users interact with real equipment, mere authentication is not sufficient. It is needed a system of authorization that considers several factors to allow users to access experiments, such as the time the lab station is available, the role and schedule of the user, and permissions to control or just monitor lab equipment. From the data that LTI provides from the VLE, the authentication, and authorization of teachers and students are automated in a completely transparent way for the users, which are automatically signed up for the first time they visit RemoteLabo from their virtual learning environment and are directly logged in on subsequent visits.

IV. NOTEBOOKS POWERED REMOTE EXPERIMENTS

Along with its architecture, the main contribution of RemoteLabo is a workflow for the entire lifecycle of remote educational experiments. Every step of this workflow allows the experiment to evolve into collaborative environment where lab administrators and teachers can actively participate, reuse, and improve them with the help of Jupyter

[6]https://html.spec.whatwg.org/multipage/web-messaging.html#channel-messaging

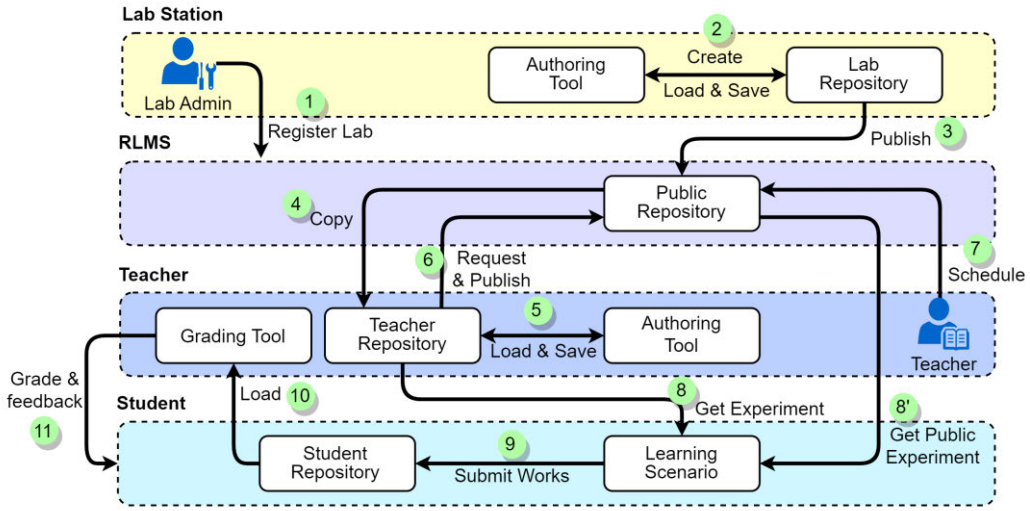

FIGURE 6. The workflow of RemoteLabo experiments, from the registration of the remote lab and creation of the experiment, through the execution of the experiment by students, to the grading of the assignment by the teacher.

Notebooks and all the capabilities it offers for teaching. Two modules are the main elements of this workflow: the experiment authoring tool and the experiment learning scenario, which were already mentioned. While the architectural aspects of these tools were yet detailed from a global system perspective, here we detail ethe reasoning behind them, their role in this workflow, and how this complete workflow, along with the already presented architecture fulfills the functional requirements for RemoteLabo.

A. EXPERIMENT WORKFLOW

The entire workflow of an experiment, depicted in Fig. 6, begins prior to its actual creation, when the lab station to which it belongs is registered in the RemoteLabo RLMS by the lab administrator. Lab registration requires essential lab station data, such as name and operating time, as shown in Fig. 7(a). The lab admin can also optionally modify the inferred data type and set the limits of the signals that the RLMS receives from the lab station. After registering and integrating a lab station, with the help of the experiment authoring tool, the lab administrator creates an experiment for it, which is stored as regular Jupyter Notebook files in a private repository of the lab station. The experiment becomes publicly available just once the lab administrator publishes it, which makes a copy of the experiment files in a public repository where all the experiments are made available for use by teachers, as depicted in Fig. 7(b). As lab stations, every teacher of the system has a personal private repository for storing private versions of the public experiments. Teachers can freely obtain a personal copy of any public experiment and modify it as desired, thus saving these versions in their private repository. These modifications are only available for the teacher and do not affect the public experiment or private experiment versions of other teachers. A teacher can have a private version of every public experiment and can make modifications to it. Teachers can collaborate with lab administrators proposing for any experiment their private modifications to be integrated into the public experiment. Lab administrators manage the private lab station repository and decide how to update the public experiments belonging to their lab station. Lab admins can make modifications without having to publish them, as these modifications are stored in the private repository of the lab station and copied to the public repository just once explicitly done by the lab administrator. A public experiment can receive a proposal from a teacher and the lab administrator decides whether to accept or reject the changes. If accepted, the updated experiment becomes the new public version that all teachers receive when copying it to their personal repository. All private changes that teachers make in their repositories persist unless they explicitly delete their private experiment or revert their modifications by copying again from the public experiment. By doing the latter, teachers can obtain the last public version of the experiment but lose their private version. However, lab administrators and teachers can always download a copy of their experiments. This collaborative environment facilitates fixing errors in the experiment code or text, improving aspects of the experiment, or customizing them for specific courses or uses.

Teachers can use any experiment from their personal repository in any course. To do this, the teachers first create a RemoteLabo activity in the virtual learning environment where they can select any experiment in the public repository. Here, teachers can choose between using the public version of the experiment or their private version if it exists. They can also choose to modify the experiment in place when selecting one. When modifying experiments, teachers can freely access them and run tests at any time, disregarding that the lab station is unavailable because the transmitted data are emulated when teachers modify experiments. This allows teachers to work on their experiments without interrupting the availability of the

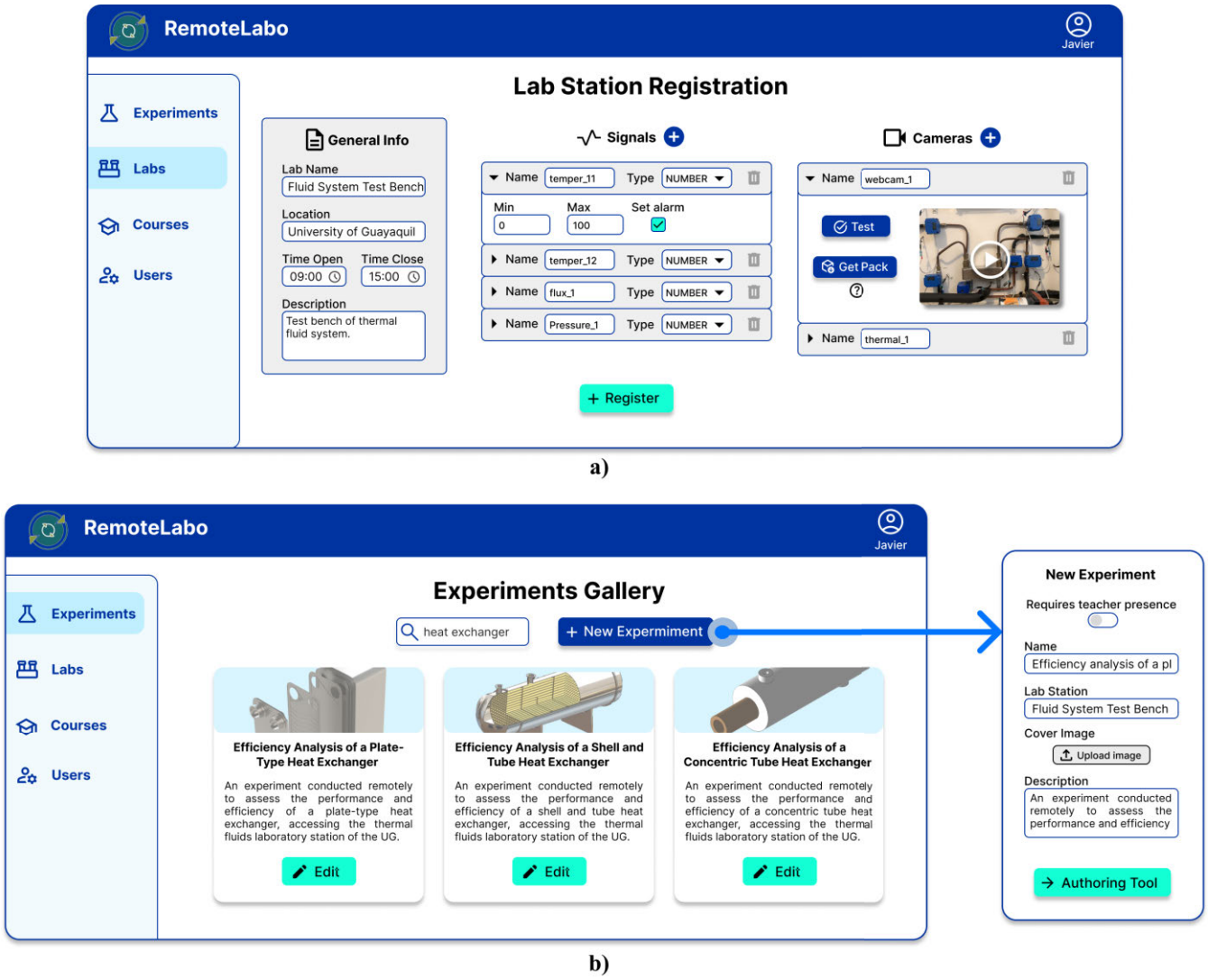

FIGURE 7. Registration of a lab station in the RLMS. On the registration page (a), the lab administrator adds the signals and sets up the cameras. Once registered, the lab is ready to host a new experiment (b).

lab station equipment. However, before the course can use the experiment, the teacher must schedule the corresponding lab station for the required period. When students enter the environment to perform the experiment, the corresponding version will be downloaded to the experiment learning scenario, within which the student will perform the experiment interacting with actual real-time data coming from the lab equipment. If the experiment requires it, at the end of the session, students can upload their sandbox notebook for the teacher to review it, grade it, and commenting it using RemoteLabo's experiment grading tool.

B. JUPYTERLITE INTEGRATION

RemoteLabo not only embeds a Jupyter Notebook in these tools but also performs seamless integration at both the UI and functional levels. To achieve this integration RemoteLabo uses two resources: the previously mentioned Channel Messaging API and the system of extensions of Jupyter. The messaging API can receive all the actions that the user performs in the UI and communicate them to the JupyterLite extensions of RemoteLabo. The extensions can communicate with the internal APIs of JupyterLite and control the internal behavior and functionality of the notebooks, such as running code, running a cell, opening a new notebook, or modifying the appearance of the notebook UI. This also works the other way around: JupyterLite can send any type of message from the extensions to the external UI that contains it. This bidirectional communication allows the experiment learning environment and the experiment authoring tool to take advantage of the JupyterLite capabilities, in an efficient way, without leaving the browser.

The main effects of this integration are the customization of the notebook UI, management of the notebook content, and communication of the notebook with the lab station. The regular notebook user interface is customized for integration

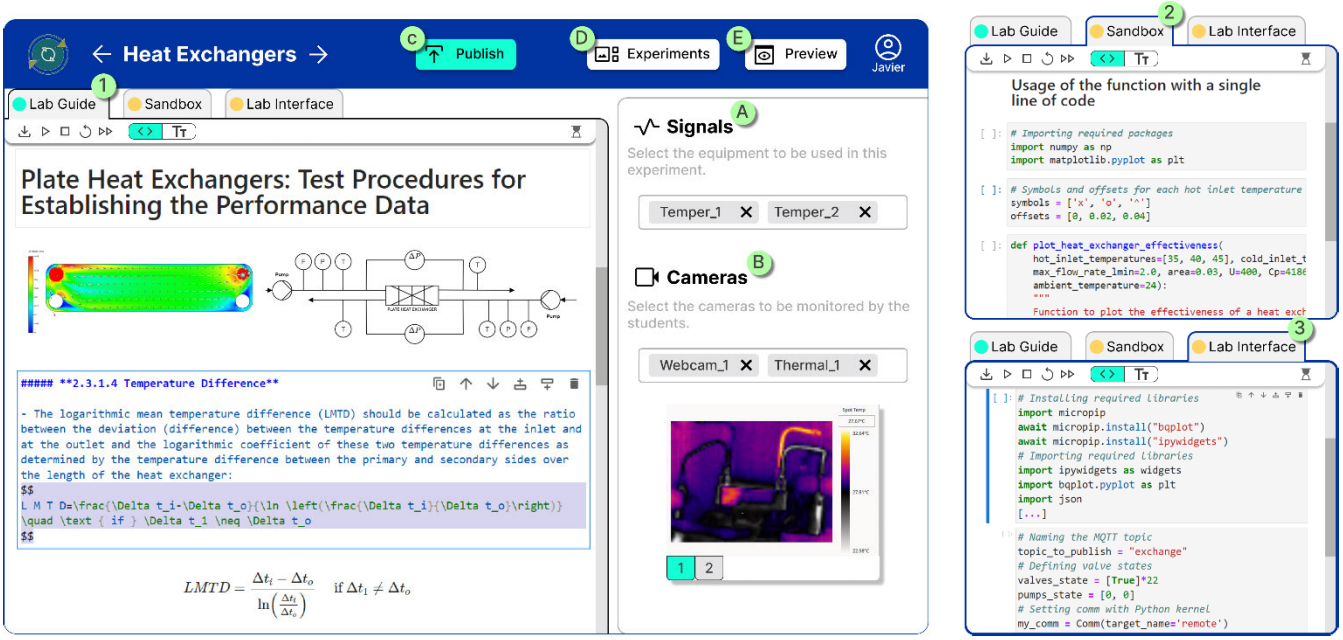

FIGURE 8. UI of the authoring tool. Here, the lab guide (1), the sandbox (2), and the lab interface (3) are created in their respective tabs. The sidebar allows the experiment author to select the required signals (A) from the remote lab for the experiment, and the required cameras (B). The top bar allows the author to publish the experiment (C), go to the experiment gallery (D), or preview how students will see the current experiment (E).

with the rest of the RemoteLabo interface and to restrict some of the default notebook functionalities that are incompatible with the RemoteLabo workflow. This is achieved through an extension that adapts the user interface to the intended notebook usage. Related to notebook content management, the tool is responsible for saving and retrieving notebook content from the corresponding repositories. When saving, the content of the notebook is sent from the JupyterLite notebook to the tool that stores the content in the repository. During loading, the required content is retrieved from the repository and passed to JupyterLite. By default, JupterLite stores notebook files locally in the browser, but RemoteLabo stores all files online in the repositories using the described mechanism. Each time a user accesses a notebook, a temporary file is created and populated with the contents of the appropriate notebook files from the correct notebook repository. This is an automated and transparent process needed every time a teacher or lab administrator creates or edits an experiment and every time a student enters the experiment learning scenario. Consequently, although the notebooks run entirely on the user's device, the user does not depend on any device to access their experiments; they can start editing an experiment on a device and later continue on another device. However, the key feature of the JupyterLite integration in RemoteLabo is the ability to communicate the JupyterLite notebook with lab stations in real-time and securely. The architecture of this communication is described in the architecture section. A corresponding extension is in charge of making this data available to notebook users. As the data arrives it is instantly made available to the notebook to be used with any library or function that the notebook can run.

C. AUTHORING TOOL

Experiments are created using the experiment authoring tool, as illustrated in Fig. 8. The RemoteLabo experiments are made of three main elements whose content is created here: The lab guide, the sandbox, and the lab interface. These three elements are created within JupyterLite Notebooks, which are integrated into the tool flow. The roles of these three elements are as follows:

Lab Guide: Provides students with the theoretical context they need and guides them through the steps to successfully complete the different activities of the experiment. Here, lab administrators and teachers can create and compose a great variety of rich content thanks to Jupyter Notebooks: text formatted with the convenient Markdown format; show code inline or in blocks in any programming language; LaTeX for mathematical notation; embedded multimedia such as image, video, audio, and even content in HTML and CSS, which opens many possibilities for the format and style of the content. As it is a regular Jupyter Notebook, it can include the result of a code snippet as content, for example, a table, plot, or graphics. The author can test any code in the notebook and verify the results before publishing the guide. The cells of Jupyter Notebooks allow authors to segment the content as they consider it appropriate, easing the creation of different parts of the guide.

Lab Interface: Responsible for displaying the interactive visual elements with which students will interact with the lab station through remote monitoring and control of the lab equipment. Here, authors can use the entire notebook possibilities to create multiple types of interactive elements for the representation of the lab equipment and their signals, such as

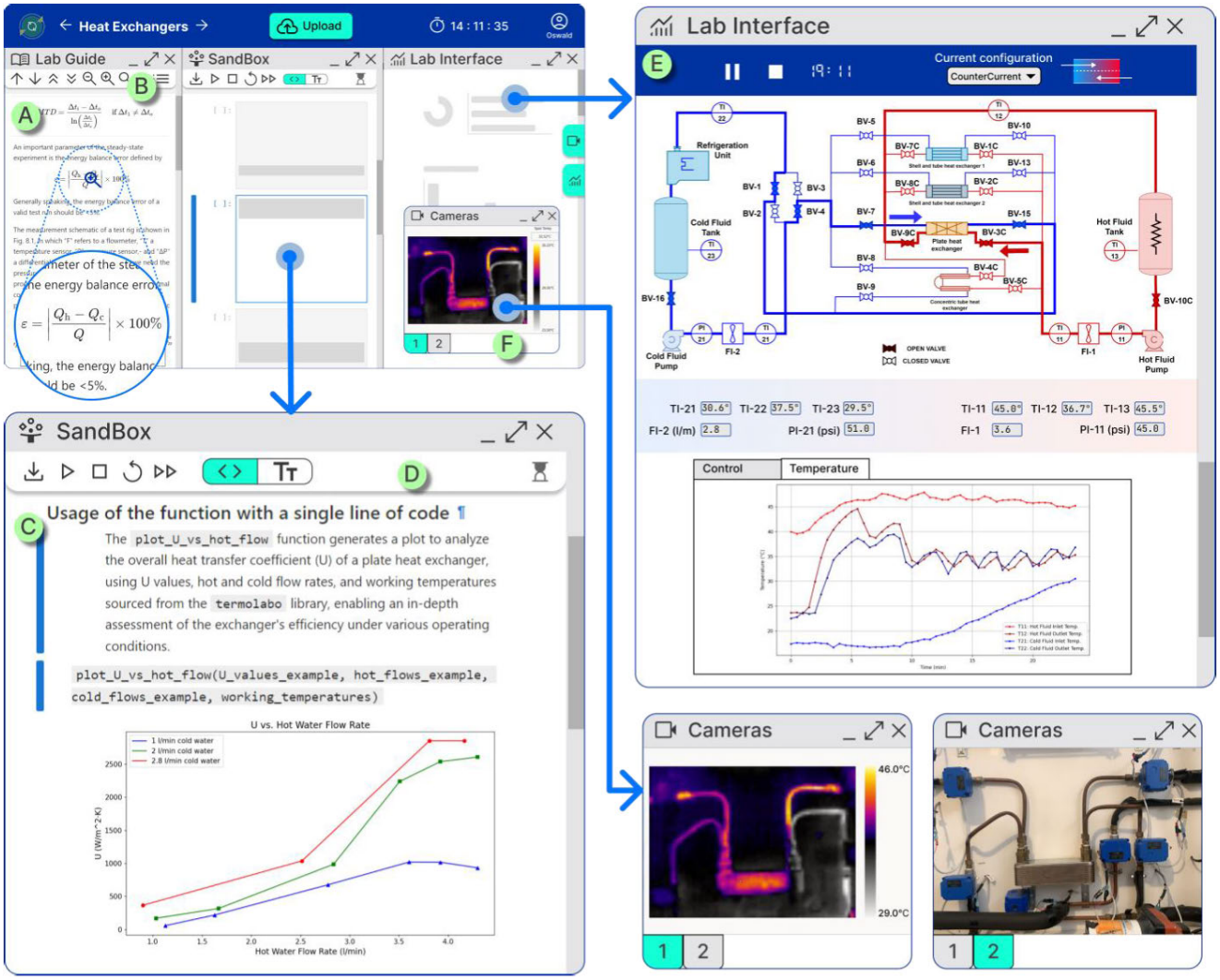

FIGURE 9. The learning scenario UI, consisting of the lab guide (A), the sandbox notebook (C), the lab interface (E), and the lab cameras (F). Each panel has a bar that allows the user to resize and move it according to their preferences. The lab guide has a navigation bar (B) that allows the user to move between the different sections of the guide, search for text, and control the size of the text. The sandbox notebook has a control bar (D) inspired by the vanilla JupyterLab control bar.

plots, interactive diagrams, buttons, sliders, and so on. These interactive elements are created through languages supported by JupyterLite, the most used being the Python language. JupyterLite supports a series of libraries that facilitate the creation of graphical and visualization libraries with which student can interact, such as IPyWidget, a very common Python library for the creation of interactive control in Jupyter Notebooks. Teachers can link the data coming from the lab station with these interactive elements and send the control actions of students to the lab station because of the previously described communication possibilities of these RemoteLabo tools.

Sandbox: As its name suggests, it is a space for students to practice on their own. With the help of the lab guide and the lab station monitor and control possibilities provided by the lab interface, students can experiment at their own pace and be guided by the teacher through predefined steps in the same space. This is the only element shown to students as a Jupyter Notebook. Here lab administrators and teachers can prefill cells with text and code to assist student activities. In this notebook, students can retrieve data from the lab station and use it for analysis, extraction of information, or plotting, all within the same notebook to perform the activities required by the experiment.

The lab administrator or teacher is free to create or modify the content of the experiment in the preferred order. Every element of the experiment is created in its corresponding tab, which deploy an independent notebook for every case. At the sidebar, the author can add the desired signals of the lab station and the desired cameras to the experiment. The resources are only those available from the specific lab station where the experiment will run. The cameras can be previewed by the author in real-time. The selected signals will be available for use in the sandbox and in the lab interface

to feed the interactive elements of the GUI. These signals are retrieved simply by using their names, as with any other common Python variable, owing to the work made for the integration depicted previously. The signal variables contain a data structure with telemetry packed into arrays. In the experiment shown in Fig. 8, given the signals added, the teacher will be able to access two temperature signals from the lab interface code, and the students will be able to use them to experiment within their sandbox. At the top bar, authors can return to the gallery of their private experiments or create a new one. From the top bar, the author can also publish their work or previews at any time on how students will view the experiment in their learning scenario. The next section describes this scenario, where the experiment elements are displayed to students.

D. LEARNING SCENARIO

All elements created in the authoring tool are displayed in a unified environment called the learning scenario, as shown in Fig. 9. This is an environment designed to engage students in performing experiments through an active approach that encourages discovery and experimentation. All the content created and added for the experiment in the authoring tool is displayed here, each one in a panel, forming a layout that can be customized by the user. Despite all three elements of the experiment being created and stored as Jupyter Notebooks, they are displayed in different ways, according to the intended goal of every element.

Lab Guide Panel: The Learning Scenario takes the rich content of the lab guide notebook and converts it to HTML to be rendered as Web-native content. It also automatically performs tasks to improve the student experience, such as generating a table of contents that the student can easily navigate. This table of contents and other tools are available in a toolbar that also allows the user to navigate through the guide, increase or decrease the font size, and search for text.

Sandbox Panel: As indicated, the sandbox panel is the only panel that displays a Jupyter Notebook to students. This is in keeping with its purpose as a computational notebook is an effective tool for exploration, discovery, and experimentation. Rather than a customized toolbar, this panel is a regular Jupyter Notebook that also has access to the signals coming from the lab station.

Lab Interface Panel: The lab interface panel along with the cameras panel is the most direct contact that the user has with the lab station. This panel runs the lab interface notebook created in the authoring tool and displays it as an interactive web application. This is accomplished by automatically executing the code in the notebook and hiding everything in the notebook except the results of the code, leaving only the programmed interactive user interface visible for monitoring and control. Monitoring is available to all participants, but control is limited to the teacher and students who have received this permission.

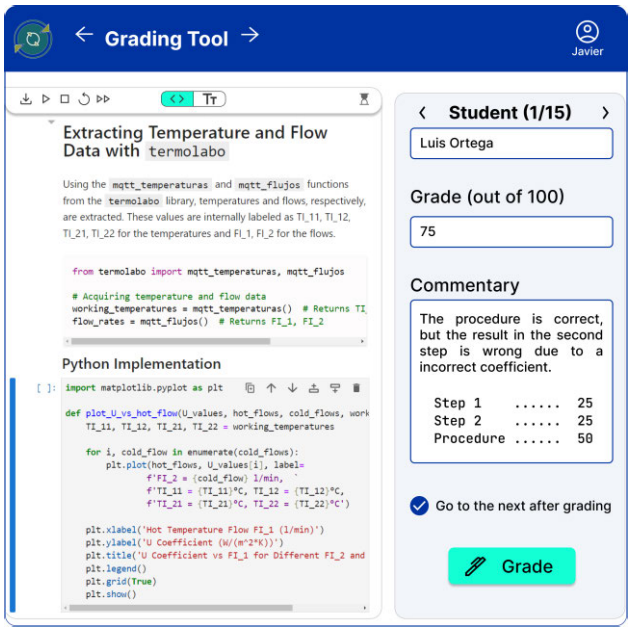

FIGURE 10. The grading tool user interface.

When multiple cameras are available, the camera panel offers controls to switch between them. By default, this panel is floating over the other panels. All panels are contained within a movable and resizable window that the student can reorder at desire. The controls from the windows allow the user to maximize a panel to occupy the entire screen, restore it to its previous size, and hide and unhide every panel. When a panel is hidden, a side tab indicates this and allows the user to click on it to reappear the panel. At the top bar, students can monitor the remaining time of the experiment and manage the time format and time notification. Before ending the session, the students, if it is the case, must upload their sandbox notebook for the teacher to evaluate. If the session ends before the work is uploaded, it is automatically executed.

E. GRADING TOOL

Teachers can set an assignment for each experiment, in which case, a button will be available to upload the work made in the sandbox panel from the experiment learning scenario. Teachers have access to the grading tool, an environment for grading, and providing feedback on the students' experiment work, as illustrated in Fig. 10. Once students upload their work, the teacher can review each assignment using the grading tool, where all the works from an experiment session are listed. Every work is shown as an executable notebook along with fields to grade and provide feedback to the student, such as comments or the rubric of the experiment. Teachgers can freely run notebook cells but cannot modify the original student work. This comment and grade are automatically synchronized with the students' grade books in the virtual learning environment thanks to LTI. Any subsequent

modification of this feedback will also be synchronized with the VLE's grade book.

V. IMPLEMENTATION AND STUDY CASE

A. RemoteLabo IMPLEMENTATION

To implement RemoteLabo, the architecture proposed in Section III and the experimental workflow from Section IV were used. The system consists of a thermal fluid system test bench, an RLMS, and a VLE, where remote experiments were deployed to obtain the characteristics and performance of various types of heat exchangers for courses related to thermal and fluid engineering. AWS services are used to deploy back-end services for time optimization and robustness.

1) COMMUNICATION LAYER

The Message Queuing Telemetry Transport (MQTT) [60] protocol was chosen for the pub-sub communication implementation. Web Real-Time Communication (WebRTC) [61] was used for video streaming from the lab station. The MQTT protocol is one of the most widely used publish-subscribe protocols [27], [29], [57] and is also used for remote laboratories [6], [35], [38]. It is a lightweight and easy-to-implement [57] real-time messaging protocol that is suitable for resource-constrained situations. MQTT manages publications and subscriptions over channels called topics, which follow a hierarchical structure. When the lab administrator registers the lab station, RemoteLabo reserves an MQTT topic for exclusive use of this lab station. A minimal and flexible structure is required for the message that the lab station and the lab client send and receive in JSON format. The pub-sub communication of MQTT is implemented using the AWS IoT service, which allows secure connections with security credentials, security policies, and custom rules. The lab station is connected to this MQTT server through an endpoint provided by AWS IoT. Both the lab computer and the experiment client run an MQTT client to connect to the MQTT server, which dispatches messages.

While MQTT is used for data communication, WebRTC is used to transmit lab station cameras in real-time. WebRTC aims to standardize real-time communication for web browsers and applications and enables peer-to-peer transmission of video, audio, and data to multiple viewers. The Kinesis Video service from AWS manages WebRTC communication through Kinesis Video channels. In this implementation, every camera in a lab station has a corresponding reserved Kinesis video channel.

2) APPLICATION LAYER

All applications of the remote lab system are included in RLMS, for which modern web technologies are used. Both the back-end and front-end software is written in TypeScript. For front-end, Svelte[7] was used as the component framework, and TailwindCSS[8] was used as the CSS framework. For the back-end, the SvelteKit[9] app framework oversees other services and functions that do not run on users' devices. Amazon DynamoDB was used as the database using the GraphQL query language.[10] The different AWS services were managed on the front-end through the Amplify library from Amazon. AWS Amazon Cognito was used for authentication and authorization. The RLMS was deployed on the Vercel[11] platform.

JupyterLite was also hosted in Vercel, which compiles the extensions and publishes the RemoteLabo JupyterLite. The GitHub[12] repositories store the different experiments that are stored and retrieved through the GitHub API. The JavaScript library LTIJS[13] was used for LTI, and the service was hosted as a Docker[14] container. The course was developed in a Moodle[15] VLE, which was deployed as a Docker container.

The lab interface was developed in the experiment authoring tool using the Python programming language and IPWidget[16] for interactive controls, such as buttons and sliders. Bqplot[17] was used to plot the signals. Students can observe the lab station directly through the camera panel, which transmits in real time the lab station cameras that monitor the installation operations. A notable feature of this system is its ability to monitor thermography videos, providing a detailed view of the status of the components involved in the experiment.

Once an experiment is started, the experiment authoring tool automatically runs an MQTT client and a WebRTC client. The MQTT client obtains permissions to connect to the MQTT server and subscribe to the lab station topic because of the permissions assigned to the user within the frame time of the experiment. This permission, by default, allows only the monitoring of the lab station signal; hence, all the students can monitor the lab station of their experiments. The permission to control the lab station is by default given only to the teacher, and it can explicitly transfer the control to a student. In the same way, WebRTC acquires permissions to receive streaming from the corresponding Kinesis channels assigned to the lab station.

3) THERMAL FLUID SYSTEMS LAB STATION

The lab station is structured into two functional blocks: the Control Unit, which is responsible for the management of controls and communications, and the Service Unit, which comprises laboratory equipment, as illustrated in Fig. 11. These components are detailed below.

The Service Unit is an industrial educational facility for remote teaching of thermal fluids. It includes a closed-loop

[7] https://svelte.dev
[8] https://tailwindcss.com
[9] https://kit.svelte.dev
[10] https://graphql.org
[11] https://vercel.com
[12] https://github.com
[13] https://cvmcosta.me/ltijs
[14] https://www.docker.com
[15] https://moodle.org
[16] https://ipywidgets.readthedocs.io
[17] https://bqplot.github.io/bqplot

piping system, pumps for mobilizing working fluids, electrovalves for flow direction, storage tanks, and heating and cooling systems for the working fluid. In addition, it has sensors to monitor the system variables, such as flow sensors, RTD temperature sensors, pressure sensors, and thermographic cameras. Its multifunctional design, as depicted in Fig. 9, provides instructors with the possibility of conducting various educational experiments related to industrial applications.

The Control Unit consists of a real-time embedded system based on NI-MyRio®, a lab controller that runs the experiments logic and controller software. In addition, it incorporates signal conditioners for the sensors and transducers of the service unit, controllers for the water-cooling and heating system, and pump controllers for the working fluids.

The instrumentation and control software are based on NI-LabVIEW®. The implemented architecture is illustrated in Fig. 12. For communication with lab users, the unit has an MQTT client that facilitates the subscription and publication of various topics related to the developed remote experiments. The topic information is transmitted in JSON format, which is processed by the MQTT server. This part of the system provides the data necessary for the embedded controller to determine the experiment to be performed remotely. Owing to the NI-RIO® architecture, equipped with a real-time processor and FPGA, it is possible to carry out multiple experiments or processes concurrently. This functionality optimizes the use of resources, thereby allowing the system to execute several operations simultaneously.

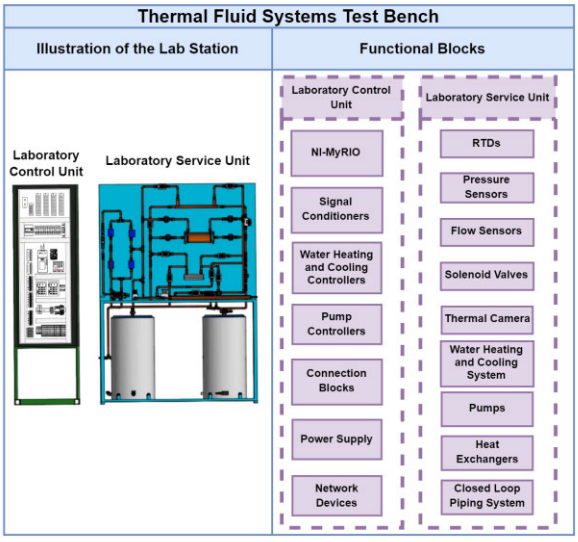

FIGURE 11. Functional blocks of the thermal fluid systems test bench.

As illustrated in Fig. 12, the lab station transmits real-time video through a single-board computer, Raspberry Pi, which runs the RemoteLabo video streamer, a program to stream the video captured from the cameras connected to the Raspberry. The program captures the camera images, encode the video using the multimedia library GStreamer[18] and transmits it to the corresponding Kinesis channel with a C WebRTC client for Kinesis[19] able to run in low-resource embedded devices.

In terms of security, both the lab camera WebRTC client the lab communication MQTT client require credentials to use these services. These credentials are generated when the lab administrator registers the lab station in the RLMS. The lab administrator can obtain the security credentials of any of their registered lab stations for use with appropriate clients.

B. THERMAL FLUID LAB CASE STUDY

Fig. 9 shows the user interface of the learning scenario for the experiment of characterization and performance of a heat exchanger. The objective of this experiment is for students to perform a comprehensive analysis of the behavior of heat exchangers in a thermal fluid system. This is achieved by observing and characterizing their transient and steady-state behaviors and evaluating their efficiency and overall heat transfer coefficient, in contrast to their theoretical and practical performance.

This remote laboratory experiment allows students to interactively read, from the Lab Guide, the instructions and theory related to thermal fluid systems using rich content, as shown in Fig. 9(A). For the development of the practice, the student had the Sandbox Panel shown in Fig. 9(C). Here, the students can see the calculations and graphs obtained with real data provided by the lab client through the developed Jupyter extension. In this space, from the temperature and flow data of the installation, the heat transfer coefficients are obtained, in a plate heat exchanger in two different configurations (co-current and counter-current). These results allow students to understand how the flow rates of the working fluids affect the heat transfer efficiency in each configuration. In the Lab Interface Panel, shown in Fig. 9(E), students have access to an interactive dashboard that allows them to monitor and manage the lab station. This panel shows the Piping and Instrumentation Diagram (P&ID) of the installation, as shown in Fig. 9(E). The various electrovalves of the installation can be activated or deactivated using Boolean buttons. In addition, there are other types of indicators and graphic controllers for the management and control of pumps, heating or cooling systems of the working fluids, flow rate, and pressure at various locations within the installation.

VI. EVALUATION AND ANALYSIS

The following section analyzes the proposed remote laboratory system architecture using the study presented in Section V as a reference. This analysis involved the collaboration between the two universities. The evaluation was carried out during the first semester of the year 2023, with the participation of students from two institutions and programs, among them: Second semester students of the ''Heat Trans-

[18]https://gstreamer.freedesktop.org/

[19]https://github.com/awslabs/amazon-kinesis-video-streams-webrtc-sdk-c

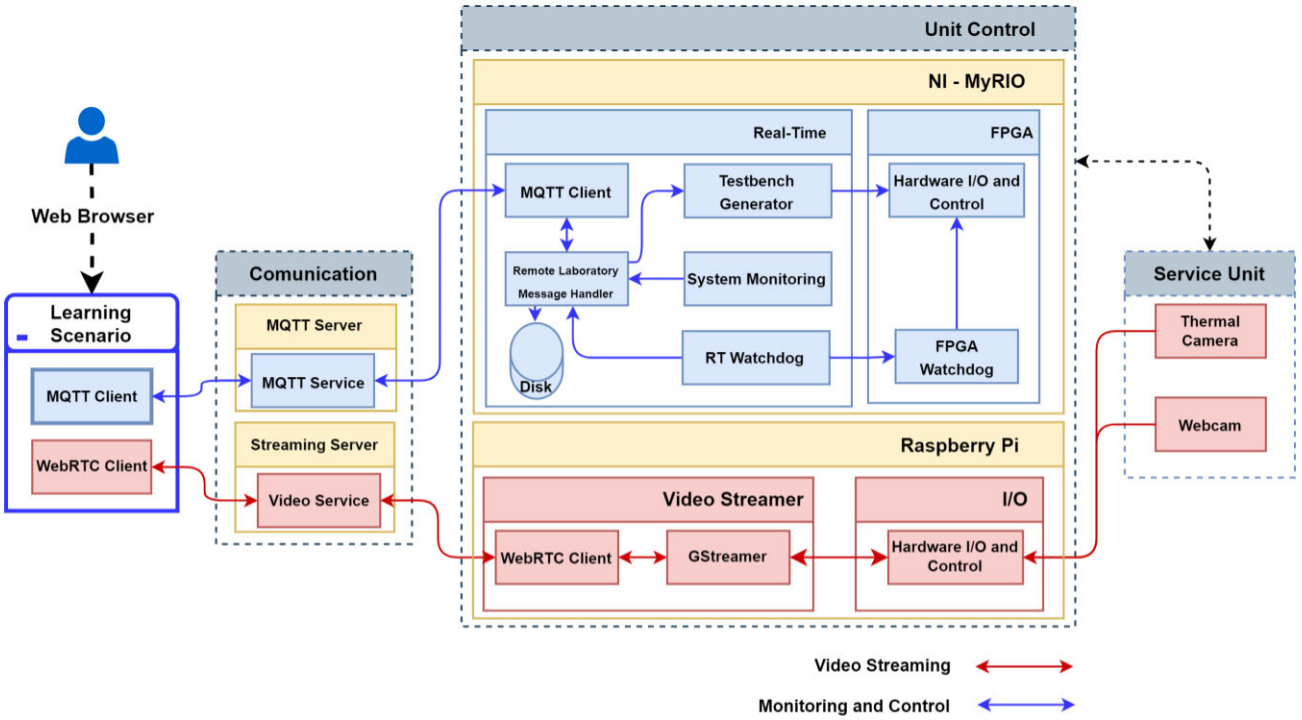

FIGURE 12. Embedded control, monitoring and video software architecture of thermal fluid systems test bench.

fer'' course of the Civil Engineering course at the University of Guayaquil ''UG,'' and of the eighth semester of the ''Process Simulation'' course of the Industrial Engineering course at the Salesian Polytechnic University ''UPS.''

To evaluate RemoteLabo, the UNE 71362:2020 standard [20] was adapted, which provides a model and tools to evaluate the quality of Digital Educational Materials (DEM) created and used in e-learning and teaching environments. The adapted tool is composed of ten evaluation criteria grouped into three categories for better understanding: educational, technological and accessibility Table 1. The educational criteria evaluate the learning scenario in terms of: the quality of the content presented and whether it is suited to the learner's level of knowledge; the adaptability criterion evaluates the application's ability to adjust to different types of learners and teachers, and the possibility of easily modifying its content; and the interactivity criterion, which analyzes how the application encourages learner participation during reading, whether it contains interactive activities and whether it allows the learner to control his or her learning. The technological criteria evaluate: the format and design to determine the quality of the presentation of the remote practices, whether they are clear and facilitate understanding of the contents and learning; reusability, which evaluates whether the system can be used multiple times or some of its components in different educational experiences; the portability criterion, which evaluates whether the system can be used in multiple environments and computer systems; and the technological criterion of technical robustness or stability, which analyzes whether the system presents technical failures and whether its use is not affected by erroneous user interactions or by changing the device or technology. The group of accessibility criteria evaluates whether the structure of the learning scenario facilitates access, understanding and progress through the content; whether navigation within learning scenarios is correct, clear, and coherent; and the operability criterion, which evaluates the functionality of the system with standard input devices such as keyboards and mice.

TABLE 1. Evaluation criteria of the quality of digital educational materials.

	Groups		Criteria
1		Educational	Content Quality Adaptability Interactivity
2		Technological	Format and Design Reusability Portability Robustness
3		Accessibility	Learning Scenario Structure Navigation Operability

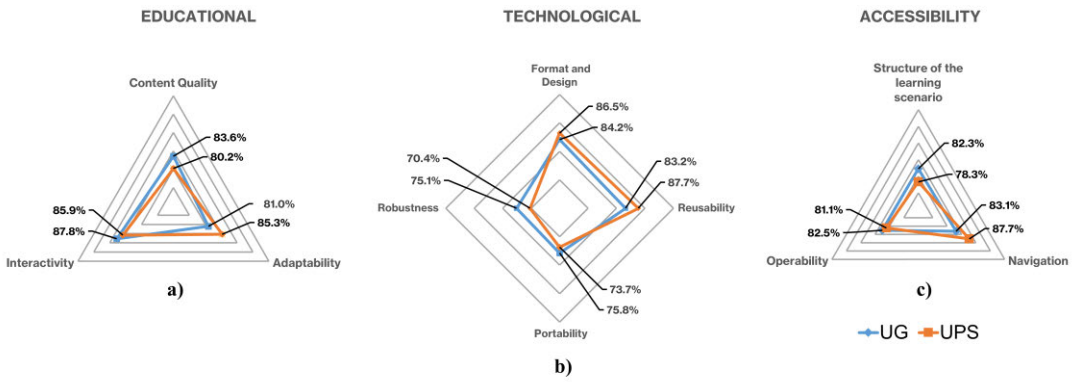

FIGURE 13. Evaluation results grouped by category: educational (a), technological (b), and accessibility (c).

Students accessed the experiment from their respective courses in the Moodle LMS, where teachers previously added it as a learning activity and scheduled the Thermal Fluids Lab Station for their students' usage through the RemoteLabo RLMS. The students entered asynchronously in the experiment of Section V which lasted 30 minutes, and at the end of it, they accessed a web form for evaluation.

To calculate the score of the experiment implemented in Remotelabo, the participants checked each of the criteria, verifying the degree of compliance with the items that comprised it. Compliance with each item was scored using a Likert scale, where a value of 1 means that the laboratory does not comply with the item, and a value of 5 means that it fully complies. The degree of compliance with the criteria is the sum of the degrees of compliance of the items that compose the criteria. The quantitative quality rating assigned to the remote laboratory system is the arithmetic mean of the item scores.

Finally, 40 participants from the UG (36 students and four professors) and 30 from the UPS (28 students and two professors) participated in the Remotelabo evaluation. The results obtained from an educational perspective are shown in Fig. 13(a). Both institutions highlighted the high quality of the contents, with scores of 83.6% by UG and 80.2% by UPS, emphasizing that the contents are presented in a clear and understandable manner. In addition, the application demonstrated adaptability (UG: 81.0%, UPS: 85.3%) because the use of notebooks allows for easy integration with different students and professors, in addition to offering an interactive and enriching user experience that fosters participation in the learning process, encouraging student exploration and autonomy (UG: 87.8%, UPS: 85.9%). Regarding the technological aspects, Fig. 13(b) show that both universities positively valued the format and design (UG: 84.2%, UPS: 86.5%), pointing out a high-quality visual and structural presentation that shows the structure of the learning scenario: a lab guide, a sandbox for experimentation, the lab interface where remote equipment can be observed and controlled, and the real-time video of the installation. Its reusability potential was also highlighted (UG: 83.2%, UPS: 87.7%) because by using Python, parts of the learning scenario and the experiment can be used multiple times. However, an improvement was suggested in terms of portability (UG: 75.8%, UPS: 73.3%) because JupyterLite and the Python Kernels have drawbacks when deployed in outdated browsers, and in robustness (UG: 75.1%, UPS: 70.4%) because some students experienced prolonged times to deploy the learning scenario. In the accessibility criteria the learning scenario structure (UG: 82.3%, UPS: 78.3%) was highlighted for its design that facilitates access, understanding and progress of remote practice; easy and consistent navigation (UG: 83.1%, UPS: 87.7%); and high operability with standard devices such as a mice and keyboards (UG: 82.5%, UPS: 81.1%), ensuring accessibility and ease of use on different devices, as shown in Fig. 13(c).

VII. CONCLUSION

This study presents an architecture for a remote laboratory system that includes an RLMS equipped with authoring, learning scenarios, and grading tools, powered by computational notebooks. The RemoteLabo architecture implements a publish-subscribe model that decouples the remote lab communication mechanism, removing direct interaction and interdependency between the lab station and the RLMS. As an effect, it significantly simplifies the configuration, maintenance, implementation of changes or updates, and security of the system, avoiding coordinating adjustments between different parts of the system, unlike client-server approaches. This approach contrasts with other methods, such as LaaS, where the interaction between the client and lab station is more direct and requires the implementation of more infrastructure on the part of the lab station, such as servers, and the need to develop or manage services on them.

The integration of JupyterLite-powered computational notebooks into the authoring (authoring tool), development (learning scenario), and grading (grading tool) cycle in the RLMS allowed the integration of code, visualizations, and other multimedia, facilitating the creation of thermal fluid systems experiments with quality content, interactivity, and adaptability for both students and teachers using the tool. The use of JupyterLite, powered by WebAssembly technology, allowed the learning scenario, authoring, and grading tool to run entirely in the user's browser, eliminating the need

for new infrastructure and the installation and configuration of a Jupyter server, resulting in an ideal environment for remote labs, where students can securely access real-time lab equipment from any device with a modern browser. The use of computer notebooks in Remotelabo promoted the creation of student-centered experiments, where the main mode of interaction was user writing, executing Python code with access to the lab station and observing the result in real time; this approach allowed students to have autonomy in their learning.

This approach also introduces challenges that need to be overcome. The architecture of RemoteLabo is new compared to other widely accepted, standardized and time-tested solutions. Shifting the responsibility for communication from each lab station to a centralized entity also brings with it the computational requirements and scalability appropriate for this entity or entities running the publish-subscribe broker. However, when using WebAssembly, and because it is still under active development, JupyterLite may present lower performance compared to native executions of other Jupyter Notebook distributions, with limitations in Python packages, as not all of them are available for Pyodide, the default kernel used. Browser resource consumption is also a limiting factor for application portability in computationally constrained devices owing to WebAssembly's use of memory and CPU resources.

The evaluation and analysis of the proposed solution revealed its advantages and challenges, highlighting key aspects of interaction, scalability, reusability, interoperability, and accessibility. The power of the computational notebooks was confirmed in the remote experiments phases, with specific data supporting their effectiveness and adaptability. These findings indicate an enhanced approach that not only optimizes technical management but also supports the learning process by encouraging greater student participation, exploration, and autonomy. This suggests that the proposed architecture can be useful for the development lifecycle of remote educational laboratories in science and engineering.

VIII. FUTURE WORK

RemoteLabo's architecture, which incorporates computational notebooks into the life cycle of a remote laboratory, raises several lines of future work that are focused on development. An analysis of the technical requirements and behavior of the publish-subscribe communication infrastructure in function of the number of lab stations is required to maintain key architecture aspects, such as scalability. Although many users can now concurrently observe the lab station and monitor its signals, the control is limited to one user at once per session. The security challenges that imply the coordination of the lab station control from multiple users, is a pending issue to be addressed. To increase the robustness of the system and the portability of the proposal, new kernels can be developed or customized for use with JupyterLite, adapting it to the requirements of the new remote labs to be developed. Likewise, new JupyterLite extensions can be designed to add new features or enhance existing ones, for example, extensions that allow the grading tool to comment and grade more atomically the work within the computer notebook. RemoteLabo tools can potentially be improved by integrating AI. An AI chatbot to assist the user through the learning process, the use of AI in the process of learning analytics, and experiments auto adaptable to individual student skills and needs are good examples of this integration. The incorporation of new types of online laboratories at different levels of higher education in science and engineering is proposed. Based on the proposed architecture, these laboratories could manifest themselves as simulations, hybrid, or fully remote environments, which presents new scenarios for educational research in online laboratories.

REFERENCES

[1] (May 2019). *1876-2019—IEEE Standard for Networked Smart Learning Objects for Online Laboratories*. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.2019.8723446

[2] C. Salzmann, W. Halimi, D. Gillet, and S. Govaerts, "Deploying large-scale online labs with smart devices," in *Cyber-Physical Laboratories in Engineering and Science Education*, M. E. Auer, A. K. M. Azad, A. Edwards, and T. de Jong, Eds. Cham, Switzerland: Springer, 2018, pp. 43–78, doi: 10.1007/978-3-319-76935-6_3.

[3] Z. Lei, H. Zhou, W. Hu, and G.-P. Liu, "Controller effect in online laboratories—An overview," *IEEE Trans. Learn. Technol.*, early access, Apr. 17, 2023, doi: 10.1109/TLT.2023.3267491.

[4] V. Kammerlohr, D. Paradice, and D. Uckelmann, "A maturity model for the effective digital transformation of laboratories," *J. Manuf. Technol. Manage.*, vol. 34, no. 4, pp. 621–643, May 2023, doi: 10.1108/jmtm-01-2022-0050.

[5] M. Hernández-de-Menéndez, A. Vallejo Guevara, and R. Morales-Menendez, "Virtual reality laboratories: A review of experiences," *Int. J. Interact. Design Manuf. (IJIDeM)*, vol. 13, no. 3, pp. 947–966, Sep. 2019, doi: 10.1007/s12008-019-00558-7.

[6] J. A. Sanchez-Viloria, L. F. Zapata-Rivera, C. Aranzazu-Suescun, A. E. Molina-Pena, and M. M. Larrondo-Petrie, "Online laboratory communication using MQTT IoT standard," in *Proc. World Eng. Educ. Forum/Global Eng. Deans Council (WEEF/GEDC)*, 2021, pp. 550–555, doi: 10.1109/WEEF/GEDC53299.2021.9657292.

[7] H. Adineh, M. Galli, B. Heinemann, N. Höhner, D. Mezzogori, M. Ehlenz, "Challenges and solutions to integrate remote laboratories in a cross-university network," in *Proc. Int. Conf. Remote Eng. Virtual Instrum.*, in Lecture Notes in Networks and Systems, vol. 298, 2022, pp. 189–202, doi: 10.1007/978-3-030-82529-4_19.

[8] D. G. Zutin, M. Auer, P. Orduña, and C. Kreiter, "Online lab infrastructure as a service: A new paradigm to simplify the development and deployment of online labs," in *Proc. 13th Int. Conf. Remote Eng. Virtual Instrum. (REV)*, Feb. 2016, pp. 208–214, doi: 10.1109/REV.2016.7444467.

[9] Z. Lei, H. Zhou, W. Hu, and G.-P. Liu, "Unified and flexible online experimental framework for control engineering education," *IEEE Trans. Ind. Electron.*, vol. 69, no. 1, pp. 835–844, Jan. 2022, doi: 10.1109/TIE.2021.3053903.

[10] Z. Lei, H. Zhou, W. Hu, and G.-P. Liu, "Concurrent experimentation in NCSLab: A scalable approach for online laboratories," *Future Gener. Comput. Syst.*, vol. 148, pp. 139–149, Nov. 2023, doi: 10.1016/j.future.2023.05.014.

[11] L. Xue, W. Hu, and G. Liu, "Learning with remote laboratories: Designing control algorithms with both block diagrams and customized C code schemes," *Comput. Appl. Eng. Educ.*, vol. 30, no. 5, pp. 1561–1576, Sep. 2022, doi: 10.1002/cae.22544.

[12] J. Broisin, R. Venant, and P. Vidal, "Lab4CE: A remote laboratory for computer education," *Int. J. Artif. Intell. Educ.*, vol. 27, no. 1, pp. 154–180, Mar. 2017, doi: 10.1007/s40593-015-0079-3.

[13] M. Pau, M. Mirz, J. Dinkelbach, P. Mckeever, F. Ponci, and A. Monti, "A service oriented architecture for the digitalization and automation of distribution grids," *IEEE Access*, vol. 10, pp. 37050–37063, 2022, doi: 10.1109/ACCESS.2022.3164393.

[14] B. E. Granger and F. Pérez, "Jupyter: Thinking and storytelling with code and data," *Comput. Sci. Eng.*, vol. 23, no. 2, pp. 7–14, Mar. 2021, doi: 10.1109/MCSE.2021.3059263.

[15] J. M. Perkel, "Why jupyter is data scientists' computational notebook of choice," *Nature*, vol. 563, no. 7729, pp. 145–146, Nov. 2018, doi: 10.1038/d41586-018-07196-1.

[16] T. Kluyver, "Jupyter notebooks—A publishing format for reproducible computational workflows," in *Proc. 20th Int. Conf. Electron. Publishing (ELPUB)*, 2016, pp. 87–90, doi: 10.3233/978-1-61499-649-1-87.

[17] A. Cardoso, J. Leitão, and C. Teixeira, "Using the Jupyter notebook as a tool to support the teaching and learning processes in engineering courses," in *The Challenges of the Digital Transformation in Education* (Advances in Intelligent Systems and Computing), M. E. Auer and T. Tsiatsos, Eds. Cham, Switzerland: Springer, 2019, pp. 227–236, doi: 10.1007/978-3-030-11935-5_22.

[18] A. Cardoso, J. Leitão, P. Gil, A. S. Marques, and N. E. Simões, "Demonstration: Using IPython to demonstrate the usage of remote labs in engineering courses—A case study using a remote rain gauge," in *Smart Industry & Smart Education* (Lecture Notes in Networks and Systems), M. E. Auer and R. Langmann, Eds. Cham, Switzerland: Springer, 2019, pp. 714–720, doi: 10.1007/978-3-319-95678-7_79.

[19] O. A. V. Guillén, J. M. Antón, J. B. Maldonado, and J. Z. Gamboa, "Termolabo project: Design and implementation of thermo-fluids systems online laboratory," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Apr. 2021, pp. 1066–1072, doi: 10.1109/EDUCON46332.2021.9454066.

[20] (Feb. 5, 2020). *UNE 71362:2022 Quality of Digital Educational Materials*. [Online]. Available: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0063263

[21] L. F. Z. Rivera, "Models and implementations of online laboratories; A definition of a standard architecture to integrate distributed remote experiments," Ph.D. dissertation, Florida Atlantic University, Boca Raton, FL, USA, 2019. Accessed: Mar. 24, 2023. [Online]. Available: https://www.proquest.com/docview/2287470698/abstract/B55FF287C5CF4F2DPQ/1

[22] A. Maiti, D. G. Zutin, H.-D. Wuttke, K. Henke, A. D. Maxwell, and A. A. Kist, "A framework for analyzing and evaluating architectures and control strategies in distributed remote laboratories," *IEEE Trans. Learn. Technol.*, vol. 11, no. 4, pp. 441–455, Oct. 2018, doi: 10.1109/TLT.2017.2787758.

[23] M. Tawfik, C. Salzmann, D. Gillet, D. Lowe, H. Saliah-Hassane, E. Sancristobal, and M. Castro, "Laboratory as a service (LaaS): A novel paradigm for developing and implementing modular remote laboratories," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 10, no. 4, p. 13, Jun. 2014, doi: 10.3991/ijoe.v10i4.3654.

[24] A. Villar-Martínez, J. García-Zubía, I. Angulo, and L. Rodríguez-Gil, "Towards reliable remote laboratory experiences: A model for maximizing availability through fault-detection and replication," *IEEE Access*, vol. 9, pp. 45032–45054, 2021, doi: 10.1109/ACCESS.2021.3065742.

[25] M. Soll, J. Haase, P. Helbing, and J. Nau, "What are we missing for effective remote laboratories?" in *Proc. IEEE German Educ. Conf. (GeCon)*, Aug. 2022, pp. 1–6, doi: 10.1109/GeCon55699.2022.9942771.

[26] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003, doi: 10.1145/857076.857078.

[27] P. Pierleoni, R. Concetti, A. Belli, and L. Palma, "Amazon, Google and Microsoft solutions for IoT: Architectures and a performance comparison," *IEEE Access*, vol. 8, pp. 5455–5470, 2020, doi: 10.1109/ACCESS.2019.2961511.

[28] P. Eugster, "Type-based publish/subscribe: Concepts and experiences," *ACM Trans. Program. Lang. Syst.*, vol. 29, no. 1, p. 6, Jan. 2007, doi: 10.1145/1180475.1180481.

[29] N. Ferraz Junior, A. A. A. Silva, A. E. Guelfi, and S. T. Kofuji, "Performance evaluation of publish-subscribe systems in IoT using energy-efficient and context-aware secure messages," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–17, Jan. 2022, doi: 10.1186/s13677-022-00278-6.

[30] W. Halimi, C. Salzmann, D. Gillet, and H. Saliah-Hassane, "Standardization layers for remote laboratories as services and open educational resources," in *Online Engineering & Internet of Things*, M. E. Auer and D. G. Zutin, Eds. Cham, Switzerland: Springer, 2018, pp. 874–884, doi: 10.1007/978-3-319-64352-6_81.

[31] A. Villar-Martínez, L. Rodríguez-Gil, I. Angulo, P. Orduña, J. García-Zubía, and D. López-De-Ipiña, "Improving the scalability and replicability of embedded systems remote laboratories through a cost-effective architecture," *IEEE Access*, vol. 7, pp. 164164–164185, 2019, doi: 10.1109/ACCESS.2019.2952321.

[32] P. Orduña, L. Rodriguez-Gil, I. Angulo, U. Hernandez, A. Villar, and J. Garcia-Zubia, "WebLabLib: New approach for creating remote laboratories," in *Cyber-physical Systems and Digital Twins* (Lecture Notes in Networks and Systems), M. E. Auer and K. B. Ram, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 477–488, doi: 10.1007/978-3-030-23162-0_43.

[33] D. G. Zutin, "Online laboratory architectures and technical considerations," in *Cyber-Physical Laboratories in Engineering and Science Education*, M. E. Auer, A. K. M. Azad, A. Edwards, and T. de Jong, Eds. Cham, Switzerland: Springer, 2018, pp. 5–16, doi: 10.1007/978-3-319-76935-6_1.

[34] M. Orlando, A. Estebsari, E. Pons, M. Pau, S. Quer, M. Poncino, L. Bottaccioli, and E. Patti, "A smart meter infrastructure for smart grid IoT applications," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12529–12541, Jul. 2022, doi: 10.1109/JIOT.2021.3137596.

[35] T. S. Uhlmann, H. D. Lima, A. L. Luppi, and L. A. Mendes, "ELSA-SP—Through-the-cloud subscribe-publish scheme for interactive remote experimentation under iLab shared architecture and its application to an educational PID control plant," presented at the *Proc. 5th Exp. Int. Conf. (EXPAT)*, Jun. 2019, pp. 58–62, doi: 10.1109/EXPAT.2019.8876567.

[36] IMS Global Learning Consortium (LTI). *Learning Tools Interoperability Core Specification 1.3*. Accessed: Mar. 14, 2023. [Online]. Available: https://www.imsglobal.org/spec/lti/v1p3/

[37] J. M. Kevan and P. R. Ryan, "Experience API: Flexible, decentralized and activity-centric data collection," *Technol., Knowl. Learn.*, vol. 21, no. 1, pp. 143–149, Apr. 2016, doi: 10.1007/s10758-015-9260-x.

[38] R. Schiano Lo Moriello, A. Liccardo, F. Bonavolonta, E. Caputo, A. Gloria, and G. De Alteriis, "On the suitability of augmented reality for safe experiments on radioactive materials in physics educational applications," *IEEE Access*, vol. 10, pp. 54185–54196, 2022, doi: 10.1109/ACCESS.2022.3175869.

[39] F. Esquembre, "Easy Java simulations: A software tool to create scientific simulations in Java," *Comput. Phys. Commun.*, vol. 156, no. 2, pp. 199–204, Jan. 2004, doi: 10.1016/S0010-4655(03)00440-5.

[40] F. Esquembre, "Facilitating the creation of virtual and remote laboratories for science and engineering education," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 49–58, Jan. 2015, doi: 10.1016/j.ifacol.2015.11.212.

[41] H. Shen, "Interactive notebooks: Sharing the code," *Nature*, vol. 515, no. 7525, p. 152, Nov. 2014, doi: 10.1038/515151ax.

[42] M. B. Kery, M. Radensky, M. Arya, B. E. John, and B. A. Myers, "The story in the notebook: Exploratory data science using a literate programming tool," in *Proc. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–11, doi: 10.1145/3173574.3173748.

[43] D. E. Knuth, "Literate programming," *Comput. J.*, vol. 27, no. 2, pp. 97–111, Feb. 1984, doi: 10.1093/comjnl/27.2.97.

[44] F. Perez and B. E. Granger, "IPython: A system for interactive scientific computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, May 2007, doi: 10.1109/mcse.2007.53.

[45] A. Rule, A. Tabard, and J. D. Hollan, "Exploration and explanation in computational notebooks," presented at the *Proc. CHI Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–12, doi: 10.1145/3173574.3173606.

[46] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, "A large-scale study about quality and reproducibility of jupyter notebooks," in *Proc. IEEE/ACM 16th Int. Conf. Mining Softw. Repositories (MSR)*. Montreal, QC, Canada: IEEE, May 2019, pp. 507–517, doi: 10.1109/MSR.2019.00077.

[47] F. Corno, L. De Russis, and J. P. Sáenz, "Computational notebooks to support developers in prototyping IoT systems," *Int. J. Hum.-Comput. Stud.*, vol. 165, Sep. 2022, Art. no. 102850, doi: 10.1016/j.ijhcs.2022.102850.

[48] D. Du, T. J. Baird, S. Bonella, and G. Pizzi, "OSSCAR, an open platform for collaborative development of computational tools for education in science," *Comput. Phys. Commun.*, vol. 282, Jan. 2023, Art. no. 108546, doi: 10.1016/j.cpc.2022.108546.

[49] A. V. Yakutovich, K. Eimre, O. Schütt, L. Talirz, C. S. Adorf, C. W. Andersen, E. Ditler, D. Du, D. Passerone, B. Smit, N. Marzari, G. Pizzi, and C. A. Pignedoli, "AiiDAlab—An ecosystem for developing, executing, and sharing scientific workflows," *Comput. Mater. Sci.*, vol. 188, Feb. 2021, Art. no. 110165, doi: 10.1016/j.commatsci.2020.110165.

[50] T. O. B. Odden, E. Lockwood, and M. D. Caballero, "Physics computational literacy: An exploratory case study using computational essays," *Phys. Rev. Phys. Educ. Res.*, vol. 15, no. 2, Dec. 2019, Art. no. 020152, doi: 10.1103/physrevphyseducres.15.020152.

[51] C. J. Fitzgerald, S. Laurian-Fitzgerald, and C. Popa, *Handbook of Research on Student-Centered Strategies in Online Adult Learning Environments*. Hershey, PA, USA: IGI Global, 2008, doi: 10.4018/978-1-5225-5085-3.ch021.

[52] A. De Santo, J. C. Farah, M. L. Martínez, A. Moro, K. Bergram, A. K. Purohit, P. Felber, D. Gillet, and A. Holzer, "Promoting computational thinking skills in non-computer-science students: Gamifying computational notebooks to increase student engagement," *IEEE Trans. Learn. Technol.*, vol. 15, no. 3, pp. 392–405, Jun. 2022, doi: 10.1109/TLT.2022.3180588.

[53] J. C. Farah, A. Moro, K. Bergram, A. Purohit, D. Gillet, and A. Holzer. (Sep. 2020). *Bringing Computational Thinking to non-STEM Undergraduates through an Integrated Notebook Application*. [Online]. Available: https://ceur-ws.org/Vol-2676/paper2.pdf

[54] R. Castilla and M. Peña, "Jupyter notebooks for the study of advanced topics in fluid mechanics," *Comput. Appl. Eng. Educ.*, vol. 31, no. 4, pp. 1001–1013, Jul. 2023, doi: 10.1002/cae.22619.

[55] A. S. Behr, L. M. Neuendorf, P. Sakthithasan, K. E. R. Boettcher, and N. Kockmann, "Process control using AI on a digital twin of an extraction column in VR," in *Proc. IEEE German Educ. Conf. (GeCon)*, Aug. 2022, pp. 1–6, doi: 10.1109/GeCon55699.2022.9942788.

[56] V. F. Ochkov, A. Stevens, and A. I. Tikhonov, "Jupyter notebook, JupyterLab—Integrated environment for STEM education," in *Proc. VI Int. Conf. Inf. Technol. Eng. Educ. (Inforino)*, Apr. 2022, pp. 1–5, doi: 10.1109/Inforino53888.2022.9782924.

[57] J. P. Dias, A. Restivo, and H. S. Ferreira, "Designing and constructing Internet-of-Things systems: An overview of the ecosystem," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100529, doi: 10.1016/j.iot.2022.100529.

[58] J. Sáenz, L. de la Torre, J. Chacón, and S. Dormido, "A study of strategies for developing online laboratories," *IEEE Trans. Learn. Technol.*, vol. 14, no. 6, pp. 777–787, Dec. 2021, doi: 10.1109/TLT.2022.3145807.

[59] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996, doi: 10.1109/2.485845.

[60] *Information Technology—Message Queuing Telemetry Transport (MQTT) V3.1.1*, Standard ISO/IEC 20922:2016, International Organization for Standardization, 2016. [Online]. Available: https://www.iso.org/standard/69466.html

[61] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Congestion control for web real-time communication," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2629–2642, Oct. 2017, doi: 10.1109/TNET.2017.2703615.

OSWALDO VANEGAS-GUILLÉN (Member, IEEE) received the B.S. degree in electronic engineering from Universidad del Azuay, Ecuador, and the master's degree in energy engineering from the School of Industrial Engineering (ETSII), Technical University of Madrid (UPM), Spain, where he is currently pursuing the Ph.D. degree. He is an Associate Professor with the Department of Information Technologies, Universidad de Guayaquil, Ecuador. His research interests include online laboratories, educational innovation, engineering education, virtual instrumentation, information security, and renewable energy systems.

PABLO PARRA-ROSERO received the degree in engineering, with a mention in electrical engineering, from Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, in 1997, the M.Sc. degree in automatic production and robotics from Universidad Politécnica de Catalunya, Barcelona, Spain, in 2003, and the Ph.D. degree in automation, control, and optimization from the University of Piura, Peru, in 2017. He is currently the Coordinator of Academic Development and the Industrial Processes Research Group (GIPI), Universidad Politécnica Salesiana (UPS), Guayaquil Campus. His research interests include modeling, simulation, control of industrial processes, and educational innovation.

JAVIER MUÑOZ-ANTÓN received the master's and Ph.D. degrees in mechanical engineering from the Technical University of Madrid (UPM), Spain. He is currently a Professor and a Researcher with the Energy Engineering Department, School of Industrial Engineering (ETSII), UPM. His research interests include thermal engineering, renewable energy, concentrating solar power, and educational innovation.

JOHANNA ZUMBA-GAMBOA received the master's degree in business administration with a focus on enterprise information systems from Universidad de Guayaquil, Ecuador. She is currently an Associate Professor with the Department of Computational Systems, Faculty of Mathematical and Physical Sciences, Universidad de Guayaquil. Her research interests include information and communication technologies in education.

CARLOS DILLON received the degree in system engineering from Universidad Politécnica Salesiana (UPS). He worked in the IT sector and as a Software Developer. He is currently a Researcher with the Department of Development of Applications for Education, SignalView Ecuador. His research interests include online laboratories and cybersecurity.