Design and implementation of a Framework for remote experiments in education

Pavel Kuriščák  
Charles University, Faculty of Mathematics and Physics, Laboratory of General Physics Education  
Prague, Czech Republic  
kurispav@fjfi.cvut.cz

Pedro Rossa  
Universidade de Lisboa, Instituto Superior Técnico, Instituto de Plasmas e Fusão Nuclear  
Lisbon, Portugal  
pedro.rossa@tecnico.ulisboa.pt

Horácio Fernandes  
Universidade de Lisboa, Instituto Superior Técnico, INESC-ID Lisboa Nuclear  
Lisbon, Portugal  
hf@ipfn.ist.utl.pt

João Nuno Silva  
Universidade de Lisboa, Instituto Superior Técnico, INESC-ID Lisboa  
Lisbon, Portugal  
joao.n.silva@inesc-id.pt

Abstract——Remote Controlled laboratories are a teaching and learning tool that increasingly becomes fundamental in the teaching and learning processes at all levels. A study of available systems highlights a series of limitations on the used programming languages, overall architecture, and network communication patterns that hinder these systems to be further adopted. Current technologies and modern WEB architectures allow the resolution of such limitations.

Here we present the FREE (Framework for Remote Experiments in Education) platform, a novel system, that, using modern technologies, architectures, and programming practices, will be easier to integrate with external tools and services and new experiments. FREE was developed in Python, Django web framework, HTML, JavaScript, and web services to simplify the development of new functionalities. The designed architecture provides a loose coupling between the infrastructure and the remote experiments facilitating further developments and allowing new experiment integrations.

Currently FREE is already running in various countries providing access to about five types of integration with various Learning Management Systems and External Authentication. Using FREE the development and integration of new experiments (independently of their hardware and programming language) is now easier.

Keywords—remote laboratories, LMS, web applications

I. INTRODUCTION

Remote controlled laboratories (RCL) have been used in teaching various fields of engineering and sciences but COVID-19 and the requirement for remote learning have given a boost to their use, visibility for its need and usefulness.

With RCLs students can access laboratory apparatus and execute experiments deployed and configured in remote locations. Access to these remote-controlled laboratories is performed using a regular PC either using specific desktop applications or a simple web browser. The experimental apparatus can be configured before the start of the experiment execution or while the experiment is executed. On the physical location where the experiment is installed, a server allows and controls access to the physical apparatus.

The access to remote experiments further enriches the students’ learning, by providing access to resources and practical knowledge that otherwise was inaccessible. The use of a controlled interface for the setup of the experiments also facilitates the operation and the knowledge of the various experiment parameters.

This work was funded by the Erasmus+ program from the EU under the grant agreement 598929-EPP-1-2018-1-PT-EPPKA2-CBHE-JP and by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020.
The various supporting software systems presented here follow a generic architecture with 4 main components: User Interface (UI), Portal, the Global Control, and the specific Experiment Control, as illustrated in Figure 1. The Portal and Global Control usually run on a server and manage the various available experiments, user authentication and authorizations, and execution queues and reservations. Experiment Control is the low-level code that controls the physical experiment, sets it up, starts execution, retrieves the sensor data, and sends it to the server. The UI presents the user with the available experiments (provided by the Portal) and shows the Experiment Control and results.

The various supporting systems implement this architecture in diverse ways, and with different usability levels. The RELL implements one variation of this generic architecture, with no Global Control. Right after the user
selects one experiment on the Portal, all the interaction is done directly to the Experiment Control. This part is running in the microcontroller/raspberry pi directly attached to the physical experiment. The REMLABNET does not implement the Portal and Global Control has limited functionalities. Each experiment is a different implementation of the UI, Global Control and Experiment Control. The UI is implemented using WEB technologies and the experiment’s control runs on a different computer from Global Control. REMLABNET allows two different interactions between the user UI and the experiments.

In REMLABNET and RELLE, regular users’ browsers connect directly to the computer attached to the physical experiment. This solution is simpler than one with a Global Control module, but it is less secure and reduces the integration and services offered to the users. Only in isolated experiments [12, 13], this type of architecture can be practical.

Both WebLab-Deusto and elab implement the 4 components and distribute them in the best way. The UI executes on the user computer, both the Portal and Global Control execute on the server and each Experiment has its own Experiment Control. The Experiment Control can run on a different computer attached to the Experiments and communicate with the server through a specific API/message. elab implements the Portal using Web technologies, allowing its access using a regular browser, but to execute any experiment it is necessary to install a Java application. In the WebLab-Deusto Experiment Control can be hosted inside the whole infrastructure not requiring a different computer or processes.

Unilabs is supported by Moodle which offers some of the Portal and Global Control functionalities. The UI of Unilabs is web-based. Since there is no specification for the Experiment Control modules, these can be placed in remote computers local to the Moodle server, depending on the particularities and experiment deployment.

2) Communication protocols
To allow remote access to the physical experiments these components are distributed in diverse ways and communicated between them using different network protocols.

In RELLE and REMLABNET the communication between the user computer and the Experiment Control simply uses HTTP and HTML. The Experiment Control, the computer or raspberry Pi directly attached to the physical experiment, should implement a regular web server to supply the UI and answer the user requests.

Elab uses CORBA [15] to implement communication between the various components: UI, Global Control and Experiment Control. WebLab-Deusto uses regular HTTP/HTML to perform communication between the browser and the server. The interaction between the server and the Experiment Control is done using HTTP/REST methods. In these two systems, the Experiment Control (running close to the experiment) receives requests and commands from the server. Since it is the server that contacts the Experiment Control on a remote computer, the server needs to know the exact network location of the Experiment Control. This solution, as in the RELLE and REMLABNET, may require complex network configuration.

For Unilabs there is no specific protocol to be used, so the communication between the Server (Global Control) and the Experiment Control (if existing in a remote computer) can be implemented using any Internet base technology or protocol and be different for each experiment.

3) Programming language and environment
Elab uses JAVA as the main language for the Global Control, UI, and Experiment Control, and uses the language supporting features for the graphical elements. The communication between components can be programmed using CORBA allowing some sort of programming uniformity between components implemented in different languages. To implement a new experiment, Experiment Control can use any language if it supports CORBA.

REMLABNET uses either Java or PHP for the development of the Experiments control modules. Since there is no public documentation, it is hard to understand the complexity and effort to integrate or develop new components.

RELLE remote experiments can be implemented using any server-side language: the whole interaction with the user is directly done using the browser and no communication needs to be done with the Portal. Experiment registration is done via a web page on the Portal part, with the insertion of the experiment data (name, description, type, location, …). The Experiment Control is integrated into the server and can be implemented in python or any other programming language that supports XML-RPC. If implemented in python the local method calls are used to interact with the experiment, while in other languages, XML-RPC is used. Unmanaged are external processes that implemented a web service to receive commands from the server. These commands will be generated in the UI (JavaScript running in the browser) and sent to Experiment Control or generated by Global Control.

Although these commands can be specific for each experiment, there is a set of pre-defined ones: experiment start, experiment delete, send data to experiments, send command to experiments, and retrieve experiment status. To implement a new experiment, it is necessary to implement the Experiment Control and create specific configuration files before its registration on the server.

The module components for UNILAB should follow the Moodle specifications and be implemented in PHP. The user interface for the experiment’s control can be implemented in JavaScript [18]. All experiment data are stored in Moodle database, being available for later processing or use.

Only WebLab-Deusto is open source. Others’ documentation doesn’t give any information on how to access, use or install their source code.

4) External services
Some of these systems are supported by external authentication and Learning Management Systems. These two integrations allow ease of user management and facilitate the user’s access to the experiments. The integration with LMS further allows the development of customized exercises dependent on the execution of the experiments.

Concerning authentication, only WebLab-Deusto allows the use of remote authentication services (such as Google or Microsoft). In this system users do not need to create a new account; they just need to use their cloud service credentials. This is an advantage for the users that do not need to define and memorize a new username/password, but also for the remote-controlled laboratory administrators that do not need to manage and secure user’s passwords.

WebLab-Deusto, REMLABNET, and Unilabs are integrated into Moodle, allowing the development of courses that require the execution of remote experiments. WebLab-Deusto and REMLABNET [19] use LTI [20] to perform the integration with Moodle. Here these systems implement a set of endpoints that are called from the LMS. Users seamlessly access content locally to the LMS and the remote experiments. The other systems require the development of specific MOODLE activity modules. The programmer needs to implement a library and set of classes (following a MOODLE specification) using PHP and JavaScript. In small-scale systems such as Laborem [14] this solution may be adequate, but for large-scale systems is limiting.

C. Limitations

From our point of view, none of the available systems provides all the requirements and functionalities of a modern, easy-to-integrate, and extensible RCL. All of them present some characteristics that can be further improved. Since the first goal was to re-use or implement a new remote-controlled laboratory system, these limitations and deficiencies should be found. The existence of a Portal and Global Control that mediates all the interactions between the user and the remote experiment is a security requirement that must be implemented. RAMLABNET and RELLE require the client browsers to contact directly with Experiment Control. This poses a big security threat for every experiment and requires a complex network and firewall configuration.

With respect to programming languages, it is obvious that nowadays the use of Java (in elab) is a major limitation. Concerning the overall architecture, it is our understanding that Global Control and Experiment Control should be separated with the possibility to be executed on a different computer. WebLab-Deusto and elab clearly enforce this.

For communication protocols, the major drawback of the presented solutions refers to the direction of communication between the remote Global control (running on the server) and the Experiment control. In WebLab-Deusto and elab it is the server that initiates communication with the Experiment Control. This requires that an experiment installed in a different network or institution has a public address (so that the Experiment Control can contact it) or a VPN needs to be used. This approach not only requires complex network configuration but also decreases security.

The integration with the remote authentication mechanism should also be taken into consideration when supplying a Portal to access the experiments. WebLab-Deusto is the only one that offers such functionality: gateway4labs [21] (external system that supplies LMS interoperability functionalities by implementing LTI [20]), but at the expense of an added layer of complexity. The integration with Moodle using specific modules (such as in Unilabs and Laborem) limits the interoperability with other LMS.

III. THE FRAMEWORK FOR REMOTE EXPERIMENTS IN EDUCATION

This chapter describes the implementation of the FREE (Framework for Remote Experiments in Education), a novel remote-controlled laboratory management system, by first listing the global requirements that guided the technological decisions. After the presentation of the Architecture and Data Model of our solution, the technical implementation details and decisions are presented.

A. Requirements

The general functional requirements that FREE implements are those common to most of the Remote Controlled Laboratories management systems: (R1) it should allow users to control an experiment that is in a remote location; (R2) users should use a browser without installing any other software; (R3) users should have access to various experiments on the same portal; (R4) the user should be able to configure the selected experiment; (R5) users should see video and results of the experiment execution in real-time; and (R6) users should have access to past results.

With respect to the operation of the experiments system, the following requirements were set: (R7) experiment control (from Figure 1) should not have a public address nor be accessible from the Internet; (R8) one Apparatus can supply several types of related experiments for the user to execute; and (R9) replicas of the same experiment can be provided by multiple apparatus installed in multiple locations.

Another set of concerns and requirements is related to the integration with external components and programming of new features or experiments: (R10) FREE should allow the integration of multiple authentication methods; (R11) FREE should be interoperable with available LMS; (R12) Adding new experiments should not require the programming and changes on the FREE server; (R13) the code structure in the FREE server should be well defined and standardized; and (R14) the interface implemented by FREE server should be well defined and allow the interoperability with Experiment Controls implemented in multiple programming languages.

B. Architecture and data model

FREE follows the generic architecture presented in Figure 1, with its 4 main components presented in Figure 2: UI is presented to the user through the browser, and the server will implement a portal with a list and provide access to all the experiments. The server will also handle the global control of all the experiments (scheduling of execution, access control, results management, …) and the Experiment Control will be placed on a computer close to the Experiment apparatus.

![FREE detailed architecture](image)

**Figure 2 - FREE detailed architecture**

261
The first significant difference from the existing solutions is the capability of an Experiment Apparatus to be able to execute different Experiments (as illustrated with the Apparatus A that can execute Experiments 1 and 2).

The second fundamental difference (fundamental to fulfill R7) lies in the communication between Experiment Control and the FREE Server. In FREE, the server never starts communication with the remote Experiment Control, it is always the other way. This reduces the complexity of network configurations and increases the security of the experiments. This also simplifies the development of Experiment Control (it is a simple client) and its deployment without the need to previously defined the Experiment control network address.

All the user interaction is programmed using web technologies: a backend server and HTML/JavaScript/REST on the browser. The generic parts of the UI (portal, authentication, list of executions, and results) are common to all the experiments, but it is necessary to define specific forms (for the creation of experiments) and results visualization for each experiment. This is carried out with the development and installation of specific modules for each experiment. These modules have the specific HTML and JavaScript that are presented to the user.

In the interaction between the User (UI) and the Experiment Control/Apparatus) the FREE Server acts as a simple Proxy. It receives from the browser the data corresponding to the configuration of the Experiments and forwards it unprocessed to the Apparatus when asked. The flow of the results is similar, during the experiment Execution, the Apparatus sends periodically the results to the FREE Server that stores them unmodified in the database. The JavaScript in the browser requests those results, and formats and presents them to the user. The FREE Server is completely agnostic to the format of the Experiment configuration and Execution results. These can be JSON or XML that is produced and processed only by the JavaScript of the UI and by the Experiment Control.

FREE implements two more modules to allow its interaction and integration with the remote components and systems. The API (Application Program Interface) module will allow the Experiment Control processes to retrieve the experiments to be executed and supply status and results. The Integration Module will allow the use of external authentication methods (from Google, Microsoft, or others) and the integration of the existing experiences into LMSs.

Figure 3 presents the data model implemented by FREE. This diagram represents the database tables of all the necessary information: the various connected apparatus, the available experiments, the executions and the results. For simplicity, the specific attributes of each class are not presented here.

The top-level data types implemented by FREE are the Apparatus Types and Protocols. The apparatus type corresponds to the several classes of physical hardware that can be connected to FREE. For instance, in Figure 2 two distinct types of Apparatus exist: the one that implements the red and maroon experiments and the one that implements the green experiments. The second level datatype is the protocol that allows the representation of the available experiment types. In the previous examples, there are 3 different protocols, corresponding to three distinct types of experiments.

This data modeling decision complemented by the decoupling of the Experiment Control from the FREE Server allows each Apparatus (the physical hardware in the laboratory) to implement one or various Protocols and the existence of multiple similar apparatus that implement the same Protocol (requirements R8 and R9). Figure 2 shows 3 apparatuses that implement 3 distinct types of experiments.

Users can belong to groups that give them access to various permissions of administration or experiment accesses. When defining an experiment execution, the user selects one Apparatus, one of the Protocols implemented by such apparatus, and defines the initial execution parameters. After the execution is created, the Apparatus will retrieve its configuration, execute it and send to FREE the Results.

Each protocol record will store not only the specific Apparatus configuration but also the initial configuration that should be provided by the user. The Execution and Results records will store all the relevant information about each experiment execution: its type (Protocol) what Apparatus was executed at, the initial parameters, and all the results. Besides this data, FREE also associates to each Protocol and Apparatus the necessary HTML and JavaScript code to implement the UI on the user browsers.

C. Implementation

To satisfy the requirements R10, R11, R12 and R13, a suitable programming language and framework should be selected. From a study of the existing programming languages and corresponding frameworks, the selected ones were Python and Django. The use of this combination provides a set of advantages for the development, maintenance, and evolution of the FREE project.

Python is a widely used language with a large programming base. The number of proficient programmers in this language is continuously growing, guaranteeing the manpower to evolve the system in the future. The selection of Django instead of other options (such as Flask) was decided to take into consideration its well-defined project structure and the considerable number of available libraries. The use of Django also allows the choice of any database backed that best suits the infrastructures where FREE will be installed. Different FREE installations will be able to use different database systems (MySQL, Microsoft SQL Server, or any other supported by Django) with minimal FREE configuration. Django also offers simple and efficient mechanisms to upgrade and install new code in existing installations.

1) API

To fulfill requirements R7 and R14, it was decided to implement the FREE interface based on REST web services. In FREE, it is the Apparatus that initiates all interaction with the FREE server. Each Apparatus calls the provided endpoints to retrieve experiments and supply results and status. For the UI, besides static pages that show the Apparatus, and protocols static information and forms, the creation of
experiments and retrieving of results is done by JavaScript REST calls to the implemented web service.

The REST API endpoints for the interaction between the browser and FREE server are summarized as follows: creation and manipulation of experiments, retrieving a list of all results for the execution, retrieving the recent results for a given execution, starting a previously configured execution, and retrieving of the status of a given execution. For the interaction between Apparatus and the FREE server, the following REST endpoints are available: change a given execution status, notify the system that the Apparatus is alive, retrieve the next execution to be executed, send to the server execution results, retrieve configuration about an apparatus.

This is the minimum set of endpoints that allows the selected type of interaction: the users can configure, execute, and see the results of experiments, and the Experiment Control, can get specific configurations, and send to the server the status and results. These endpoints process, store, and read the information on the database.

To identify remote users and Apparatuses and to guarantee access control, modern techniques, such as security tokens and secrets, are implemented.

In the definition of an experiment configuration and processing of results, the FREE Server is completely transparent to each experiment data, it only stores and forwards this information. The processing of this information is only done in the JavaScript code in the UI and on the Experiment Control next to Apparatus. The use of REST endpoints further simplifies the development and integration of new experiments with no changes to the infrastructure and using different programming languages on the browser or on the Experiment Control (R14). Additional endpoints can be added to implement new functionalities, such as experiment reservations or control during the experiment execution.

2) Experiments

To create a new experiment and integrate it into FREE it is necessary to follow three steps: (i) implement the Experiment Control that will call the previously presented endpoints to interact with the FREE server, (ii) define the UI and (iii) insert the relevant data into the FREE server database.

In the creation of the UI, the programmer does not need to write any python server code, he only needs to define the HTML elements that will allow the insertion of the experiment configuration and write the JavaScript code to show the results on the web page. This is done in a template file that will hold all the supporting HTML and JavaScript code. The programmer needs to design a simple HTML form (specific for that experiment configuration), and a DIV with the tables and plots to be filled with the results. The programmer also needs to write the JavaScript code that should be executed whenever a result is received from the server. This code receives the results in tabular form as formatted by the Control Experiment, converts them and updates the plots previously designed. This file should be placed in a specific directory in the FREE Server.

Each type of Apparatus, type of experiment and specific physical apparatus requires the insertion of specific information on the database. This is done in the Django-provided Administration interface, as illustrated in Figure 4. The FREE administrator should create one new entry in the Apparatus Type (corresponding to the characteristic of the specific physical Apparatus) and an entry in the protocol table. After this, it is possible to create a new entry corresponding to the specific Apparatus.

The implemented API, the FREE Server organization and these implementation steps allow the creation of new experiments without changes to the FREE Server Code (R12). Furthermore, the code necessary for the integration of a new experiment is limited in the number of lines of code and complexity.

3) Services and libraries

Besides the easy integration of new experiments, FREE should also allow its easy integration with existing external services such as authentication services or Learning Management Systems (requirement R10 and R11). This integration was eased by the selection of programming language and framework. Django already offers libraries that completely hide the implementation details and implement all the interoperability mechanisms.

In FREE, the integration with existing authentication services (such as Microsoft, Google, or other specific university authentication) is carried out using the OAuth [22] protocol, an industry standard that with a suitable client library only requires a small configuration. In FREE the social- Django [23] library is used to activate the OAuth protocol, this library takes advantage of the user management embedded in Django, and with a simple configuration, the authentication for Microsoft or Google services can be activated.

Another integration that was considered fundamental and already implemented was with external Learning Management Systems (LMS) such as Moodle. To do this we selected the use of the industry standard LTI [20], by using the django-lti-provider [24] library it is now possible for Moodle (or any other LMS) to redirect users to FREE to execute any experiment. After being redirect to FREE, users will be automatically logged in without requiring any further credentials. The LTI protocol handles and transfers user credentials and all complementary user information.

FREE also supports the streaming of video from the apparatus to the user browser (requirements R5). This is carried out using WebRTC [25] protocol: a camera attached to the Apparatus sends a video stream to the FREE server that, when asked, forwards it to the user’s browser. This allows the use of a pre-existing video streaming server (such as Janus [26]) and relieves the installation of video visualization software on the user’s computer.

IV. VALIDATION

This section describes the functionalities and experiments that are now deployed and being used in a FREE installation.
A. Ported experiments

Currently, 6 different Protocols are ported to the FREE infrastructure. Two of these Protocols share the same Apparatus (as illustrated by Apparatus A in Figure 2). All the apparatuses were previously available through a different RCL infrastructure and provide experiments in physics. World Pendulum (mechanics) is an experiment devoted to studying the local gravity dependence with latitude but used for other physics studies requiring such experiment. The Inclined plane (mechanics) is used as the typical experiment for gravity determination with a small cart but equipped with a flap to increase the kinetic friction to model friction by numerical methods. The Photovoltaic panel (Energy) uses a dimmed tricolor LED panel to produce electrical energy in a small solar panel. The user can do several studies according to the impedance matching, including the efficiency according to the angle of incidence, irradiated power, and wavelength, for other physics studies requiring such experiment. The Langmuir Probe (Plasma Physics), an advanced experiment, allows for the characterization of plasma produced by RF. The user collects the I-V curve of a Langmuir probe from where we can extrapolate the electronic temperature and plasma density. The Cavity (Plasma Physics) is another advanced experiment used for studying the electromagnetic propagation in a resonant cavity and observing how the existence of a plasma shifts the resonant frequency by altering the speed of light as soon as a RF plasma is generated.

Currently, there are Pendulum apparatus connected to various FREE installations in Europe, Oceania, Africa, and Latin America.

The migration of Apparatuses and protocols to the FREE infrastructure required a new implementation of the Experiment Control to use the available endpoints and data generated in the UI. This development was accomplished in a short amount of time and was done by a MSc Student with intermediate programming knowledge. The effort to create the UI can be quantified by the number of lines of code necessary to define the whole UI. For an experiment whose configuration has 2 numeric values, and the results are presented in one table and 3 plots the programming effort is as follows: 20 HTML lines for the input of 2 values, 25 HTML lines for one table and 3 plots, and 180 lines of JavaScript for the update of the results. The effort to create the UI (HTM and JavaScript) for a new experiment (after the implementation of the corresponding Experiment Control) and the structure of the Execution configuration defined is, on average, lower than 5 hours of continuous work.

B. Remote Services

The FREE allows user authentication using external services such as Google, Microsoft or even University services, besides locally created users. Currently, the running installations integrate these external services differently. Figure 5.a shows the login page, where the user can either input the FREE username and password (for administrators, for instance) or select an external authentication service. In this FREE installation, three external services are configured that automatically appear on this page. After selecting one of these providers the user is redirected to the corresponding login page, and after successful authentication, the user can access FREE as any other user. Figure 5.b show the users that were authenticated using Google and the University services.

Before professors can provide access to FREE experiments in their courses, the LMS administrator needs to register for the FREE installation. The administrator only needs to insert the network address of the FREE installation (Tool URL), the Consumer Key and Shared Secret. This information is the responsibility of the FREE administrator to provide. After this configuration, professors can create activities in their courses that will redirect students to FREE. During the course, the student clicks on the provided and is redirected to FREE. Although only evaluated with Moodle, the use of LTI allows the integration of FREE into most LMS following similar steps. If using these authentication mechanisms, any FREE installation becomes public and with universal access. To guarantee fairness of the resources, experiment usage policies and mechanisms to delete old and inactive users were implemented for FREE. This way users will not be able to abuse the systems, and the number of database resources will be limited in order not to degrade other users’ experiences.

C. Functionalities

After a user enters FREE a home page with a simple menu is presented (Figure 6.a) allowing access to installed Apparatus and Protocols, Configure execution and results of past executions. FREE is multilingual with translations to English, Portuguese, and Spanish. The list of the installed apparatus (Figure 6.b), presents basic information about each apparatus, the available protocols and its state. From this page, users can create a new experiment by pressing the Run button.

The experiment configuration page presents three tabs, one with the description of the experiment, another with the configuration form and a final one with the results of the execution. A small window with the video stream from the experiment apparatus allows the user to see in real-time the execution of the experiment. One such example is illustrated in Figure 7.a). This corresponds to the configuration of the Pendulum Experiment, where the user must define the initial pendulum displacement and the number of samples to be measured. The three buttons allow the user to save the
configuration reset it or start the execution. After being saved, an experiment configuration can later be accessed and modified. Only when the Experiment configuration is submitted it may be executed by the apparatus. During the execution of the experiment the user can see the video in real-time, but also see the plots (Figure 7.b)). The table with all the relevant experiment measurements (not shown) can be downloaded in various formats (excel, or CSV).

The experiment programmer only had to define the two input fields and sliders. All the logic related to the buttons, configuration save, and the start of the experiment is generic and handled by the FREE server. Although the FREE user interface is minimal in the number of pages (home pages, list of apparatuses, list of user’s experiments and results, and Execution page) the fundamental functionalities available on currently existing RCL systems are already implemented in this first version, fulfilling the basic functional requirements R1 to R6.

V. CONCLUSION

FREE is a new RCL system that uses modern programming languages, techniques and software architectures, and allows the simple development of remote-controlled laboratories. FREE provides a portal to experiments that can be scattered in various institutions of networks and allows their integration and access through external authentication services and existing LMSs. The architecture, the explicit separation of components and the way they interact guarantee that the development or adaptation of new experiments into FREE is done in a fast and efficient way. The programming skills necessary to do these developments are also low. Only basic proficiency in python and web programming is enough.

FREE allows the development and integration of remote experiments developed in a programming language different from python because the remote interactions are well defined and based on REST web services. This explicit separation also allows the development of virtual experiments whose access can be mediated by FREE. In this case, the Experiment Controller of the virtual experiment would simulate the physical process and be integrated with FREE as any remote experiment.

FREE is being developed by a heterogeneous small programming team ranging from Electrical and Physics engineering students to one already graduated Informatics Engineers. In the future, this interdisciplinarity can be further leveraged in the development of new functionalities (such as Apparatus reservation or execution priorities), or new types of interactions, that require knowledge and expertise in algorithms, computer engineering and Informatics. The development of new experiments can be done in the future in the context of MSCs in specific study fields (Physics, EEC, Biology, Chemistry, or others). Another future possibility lies in the creation of a federation of RCL, where one installation of FREE in a certain institution, will provide seamless and transparent access to experiments in remote FREE installations. This approach will foster communication and collaboration in the development and deployment of experiments, but also increase the availability of resources (number and diversity of experiments) to users.

From the implementation efforts and attained results, we think that FREE successfully handles the limitations identified in current systems, but also provides a staring platform for the evolution, dissemination and increase of users of remote-controlled laboratories.

REFERENCES

[1] J. G Zubía, and G. R. Alves, eds. Using remote labs in education: two little ducks in remote experimentation. Universidad de Deusto, 2012.
[2] Labsland. “Labsland”. https://labsland.com/. Accessed July 20, 2022.
[3] Universidad de Santa Catarina. “Laboratórios | RELLE - Ambiente de Aprendizagem Com Experimentos Remotos”. http://relle.ufsc.br/labs . Accessed July 20, 2022.
[4] elaboratory Project. “SES - Internet School Experimental System: Crossroad.”. http://www.ises.info . Accessed July 20, 2022.
[5] Instituto Superior Técnico. “elab / Remote Laboratories.”. http://www.elab.ist.utl.pt . Accessed July 20, 2022.
[6] Universidad Nacional de Educación a Distancia. “University Network of Interactive Laboratories.”. https://unilabs.dia.uned.es/ . Accessed July 20, 2022.
[7] S. Rapuano, and F. Zoino. “A learning management system including laboratory experiments on measurement instrumentation.” IEEE Transactions on instrumentation and measurement 55, no. 5 (2006).
[8] Universidade Federal de Santa Catarina. “relle Docs - Documentação”. http://docs.relle.ufsc.br . Accessed July 20, 2022.
[9] University of Deusto. “WebLab-Deusto 5.0 documentation”. http://weblabdeusto.readthedocs.io . Accessed July 20, 2022.
[10] F. Schauer, et. al. “REMLABNET-open remote laboratory management system for e-experiments.” In 2014 11th Intl Conf. on Remote Engineering and Virtual Instrumentation. IEEE, 2014.
[11] K. E. Kharki, K. Berrada, and D. Burgos. "Design and implementation of a virtual laboratory for physics subjects in Moroccan universities." Sustainability 13, no. 7 (2021).
[12] Z. Laouina, et al. “Manufacturing and Developing Remote Labs in Physics for Practical Experiments in the University.” In Intl. Conf. on Remote Engineering and Virtual Instrumentation. Cham, 2020.
[13] F. Limpraptono, A. Ratina, and H. Suddibyo. "New architecture of remote laboratories multisuser based on embedded web server." Intl. Journal of online Engineering 9, no. 6 (2013): 4-11.
[14] C. Lavaysièrre, B. Larroque, and F. Luthon. “Laborem Box: A scalable and open source platform to design remote lab experiments in electronics.” HardwareX 11 (2022): e00301.
[15] S. Vinoski. "CORBA: Integrating diverse applications within distributed heterogeneous environments." IEEE Communications magazine 35, no. 2 (1997): 46-55.
[16] Pallets. "Welcome to Flask". https://flask.palletsprojects.com/ . Accessed July 20, 2022.
[17] SQLAlchemy authors. " SQLAlchemy - The Database Toolkit for Python". https://www.sqlalchemy.org/ . Accessed July 20, 2022.
[18] L. Torre, T. Andrade, P. Sousa, J. Sánchez, and M. T. Restivo. “Assisted Creation and Deployment of Javascript Remote Experiments.” Intl. Journal of online Engineering 12, no. 9 (2016).
[19] F. Schauer, et. al. "REMLABNET IV-LTI federated remote laboratory management system with embedded multiparameter simulations." International Journal of Online Engineering (2017).
[20] 1EdTech Consortium. " Learning Tools Interoperability | IMS Global Learning Consortium". https://www.imsglobal.org/activity/learning-tools-interoperability . Accessed July 20, 2022.
[21] P. Orduña, et al. "An extensible architecture for the integration of remote and virtual laboratories in public learning tools." IEEE Revista Iberoamericana de Tecnologías del Aprendizaje 10, no. 4 (2015).
[22] A. Parecki. " OAuth". https://oauth.net/ . Accessed July 20, 2022.
[23] Python Social Auth - Django authors and contributors. "Python Social Auth - Djanga ". https://github.com/python-social-auth/social-app-django . Accessed July 20, 2022.
[24] Django LTI Provider authors and contributors. "Django LTI Provider". https://github.com/cchmll/django-lti-provider . Accessed July 20, 2022.
[25] Google, Mozilla, W3C and IETF. "WebRTC Real-time communication for the web". webrtc.org/ . Accessed July 20, 2022.
[26] Meetecho. "Janus: the general purpose WebRTC server". https://janus.conf.meetecho.com/ . Accessed July 20, 2022.