IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 23, 2021, accepted March 8, 2021, date of publication March 12, 2021, date of current version March 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065742

Towards Reliable Remote Laboratory Experiences: A Model for Maximizing Availability Through Fault-Detection and Replication

AITOR VILLAR-MARTÍNEZ[1], JAVIER GARCÍA-ZUBÍA[1], (Senior Member, IEEE), IGNACIO ANGULO[1], AND LUIS RODRÍGUEZ-GIL[2], (Senior Member, IEEE)

[1]Faculty of Engineering, University of Deusto, 48007 Bilbao, Spain
[2]LabsLand, 48002 Bilbao, Spain

Corresponding author: Aitor Villar-Martínez (aitor.v@deusto.es)

ABSTRACT Educational remote laboratories allow users to access and control via the Internet remote equipment, that may be physically located anywhere in the world. They have a great potential for education, since they offer many advantages, such as cost reduction, high flexibility or the possibility of sharing equipment among institutions. However, their use is not yet as widespread as it could be. Freely available laboratories often present significant reliability issues. Instructors often plan their lessons in advance, and incorporating third-party tools that are prone to failure is risky, since they may be unavailable for the class and present errors to the students, which may make it necessary to suspend the session or find alternatives on the go. This contribution analyzes the reliability of a set of state-of-the-art laboratories with a view to categorizing and quantifying the most common failures. It then describes a multi-layered software model for detecting errors in laboratories. This model has two main goals. First, to ensure that the laboratory management system is aware of the state of the laboratory, so that users do not experience errors and laboratory maintenance personnel can repair it as soon as possible. Second, to serve as a base for a fault-tolerance model that leverages replicated instances of the laboratory. This fault-tolerance model is also described, and is designed to provide high availability and support for multiple concurrent users. The effectiveness of this model is validated by analyzing the reliability and availability of two remote laboratories in which it has been implemented.

INDEX TERMS Remote laboratory, reliability, availability, failure detection, replicability.

I. INTRODUCTION

Remote experimentation has been an active research field since 1995 [1]. The main difference between a remote and a traditional hands-on experiment lies in the student's proximity. In a hands-on laboratory, the student is physically in front of the experiment and is able to directly view and manipulate it. Meanwhile, in a remote experiment, the student accesses, controls and views the process via the Internet [2]. Remote laboratories can provide an educational experience that is similar to the hands-on experiment, and if properly designed they can achieve similar or even superior educational results [3]–[5]. In both cases, the experimentation is real and not a simulation. Thus, the main difference is in the means of interaction with the experiment. Instructors may have several different reasons for integrating a remote laboratory in their teaching and these are explained in more detail in [6], [7]. Remote laboratories can be used both as a complement to traditional hands-on laboratories or as a replacement [3], [4], [8], [9]. In both cases, they facilitate the work of the instructor and the students. Also, for example, in the COVID-19 pandemic situation, which has made online or distance training obligatory, the use of remote laboratories has allowed the students of some educational institutions to continue their laboratory practices almost normally, STEM experiments included. Different public bodies have included remote laboratories in their recommendations to instructors [10], making it possible to continue teaching remotely. Examples of remote laboratories used during the COVID-19 situation can be found in the USA [11], in Spain [12], [13], in Germany [14], or in Africa [15], [16].

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

There are many remote experiments, especially in the field of technology and experimental sciences [17]. However, despite their usefulness, their actual use in classrooms is not currently very widespread. Often the experiments are only used by the instructors who worked on their development as researchers, despite being offered to other instructors and institutions. One of the main advantages of a remote laboratory is that from a technical perspective, it can be shared by different instructors, students and educational centers, and most universities and researchers are willing to do this. However, in practice they are rarely used by third-parties, although in principle they are freely available. Both [18] and [19] analyzed the low utilization rate of remote laboratories and found two significant aspects. On the one hand, [18] remarked in his study with instructors that **reliability** was the fundamental characteristic expected in a remote laboratory, and on the other hand [19] stated ''However, while the number of remote laboratory initiatives is high, the overall impact of these laboratories is fairly limited beyond the scope of the host institution or the scope (and duration) of projects in which the host institution is involved. There are cases where the laboratories are regularly used by other institutions, but these are still exceptions and remote laboratories are not yet widely used. This is not the case for virtual laboratories (simulations), where the maintenance costs and work required once developed tend to be low''. In other words, instructors are not willing to plan and integrate within their class a resource that fails or may fail, especially when they cannot be sure of knowing this beforehand and cannot easily plan for an alternative should the lab fail. The reliability of freely-offered remote laboratories still raises too many uncertainties for them to be used as a dependable tool in classrooms. These uncertainties are more acceptable to and less of a concern for the original designers of the remote experiment, since the latter will have more information regarding the state of the laboratory, will probably be able to identify and understand problems, and may even have physical control over the equipment in order to fix it themselves should it fail. For example, an issue such as having a server accidentally turned off, or having a simple cable disconnected, can often be solved in real-time by the original designer. A third-party instructor, however, would have no control, and would have to cancel the planned activity, with the corresponding potential consequences. Considering the didactic usefulness and advantages of remote laboratories, there is therefore reason to believe that remote experimentation would have greater presence in classrooms if its reliability inspired more confidence in instructors. This contribution analyzes in detail the most common problems that occur when accessing remote experiments, and proposes strategies to mitigate these problems by verifying in real-time the availability and working order of the laboratories. The paper also describes a model for the replication and federation of those remote experiments in order to improve the availability of the service once failures have been detected by the aforementioned strategies. By having several copies of the same laboratory and by knowing whether a laboratory is in working order, it is possible to increase the reliability of the service from the students' perspective, automatically redirecting them to working instances.

The rest of the paper is organized as follows: Section II groups multiple relevant remote laboratories and analyzes their reliability. Section III introduces a set of two interlinked proposed solutions, including an integration example for each one. In Section IV, two remote experiments based on the proposed architecture are evaluated in terms of reliability. Section V summarizes the conclusions of this paper and Section V describes future lines of work that could be pursued.

II. LAB ANALYSIS FROM A RELIABILITY STANDPOINT

This section describes a preliminary study that has been conducted to analyze the state of remote experiments from a reliability perspective and to categorize the various failures observed into five different broad categories. The owners or maintenance personnel of the remote laboratories were not contacted by the authors and thus no particular measures were taken beforehand. The laboratories were chosen because they are well-known to researchers and users, are openly shared and are representative examples of remote laboratories. It is important to remark that the goal of the analysis is not to determine the reliability of specific laboratories or to compare them with one another. Reliability is not necessarily a goal for laboratories, which are not necessarily intended to offer high availability during the testing period. The goals of this preliminary study are instead to categorize the very broad diversity of observable failures into a manageable amount of categories, to provide a general overview of the remote laboratory ecosystem with regard to reliability, and to justify the importance of reliability for laboratories intended for broad third-party application, and thus the relevance of the subsequent effort to detect and mitigate remote experiment problems.

Several steps are performed by the user when accessing and controlling a remote experiment. Figure 1 shows some of these. From the user's point of view, the experience consists of the following steps:

- **Access to the laboratory:** First, users access the URL of the remote experiment. Usually a catalogue of the available remote experiments is displayed.
- **Access to the remote experiment:** Users then access the remote experiment itself by clicking on the appropriate icon or link. Sometimes if the remote laboratory platform provides access to a single experiment, this and the previous step are merged into one.
- **Setup of the remote experiment:** Once inside the remote experiment, users will normally have the chance to set the parameters of the experiment or to program the underlying device. Thus, users will normally have control over the laboratory setup.

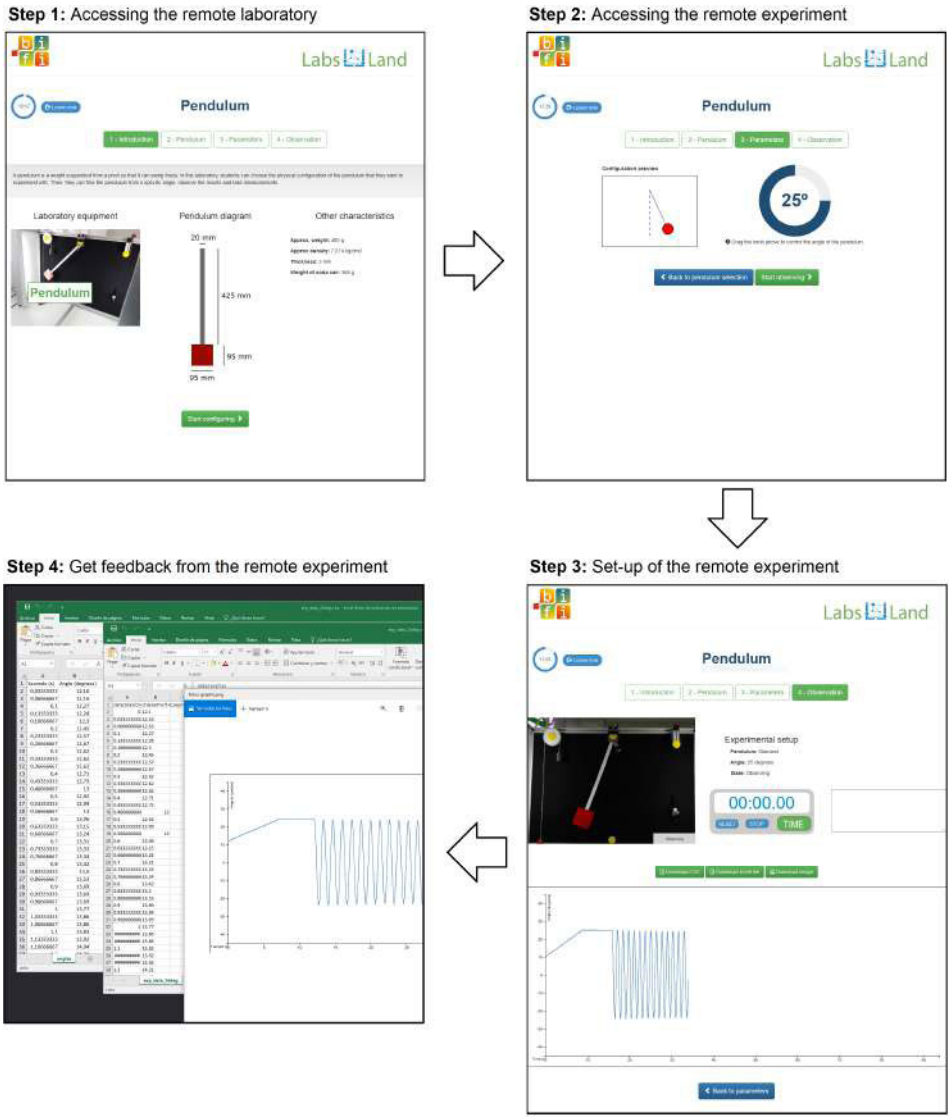


FIGURE 1. Steps followed by users during the interaction with a remote laboratory and/or experiment.

- **Feedback from the remote experiment:** Either during the experiment or upon completion, the laboratory interface shows the results via a real-time video stream (e.g., [20], [21]) tables, files, or similar means.

The design of a remote laboratory is highly complex since it integrates software, hardware, telecommunications, web design, and other fields. Even if the remote experiment was properly designed and worked and met all requirements when first deployed, over time, with a certain frequency, different errors can occur in any of the four previously described stages. These errors will be hard to predict: power outages, unavailable servers, hardware issues, unfocused or incorrectly positioned cameras and other problems. The goal of this work is not to detect the errors arising from the design phases of the remote experiment. It is, instead, to detect the faults that arise during its conventional use. The ultimate goal is to be able to know when a laboratory is not in working order before user experience is negatively affected, and to use that knowledge to solve and mitigate these failures, increasing laboratory reliability and user confidence.

A. ANALYSIS OF REMOTE EXPERIMENTS

Now that the remote experiment experience has been categorized into four different phases, this contribution will analyze a small but representative set of state-of-the-art remote experiments in terms of reliability, under the specific criteria detailed below. It should be noted that, as mentioned before, the goal of this analysis is not to determine the reliability of specific laboratories but instead to provide an overview of the reliability of examples of remote experiments and the relevance of subsequent research efforts aimed at improving that particular aspect of remote labs.

In order to select a set of remote laboratories to be analyzed the authors chose those that fit 4 main criteria:

TABLE 1. Table listing the state-of-the-art remote laboratories and experiments that were analyzed.

Laboratory	Experiment ID	Experiment name/description
Laboratory 1: ISES	Experiment 1	Diffraction of electromagnetic radiation on microobjects I
	Experiment 2	Temperature monitoring
	Experiment 3	Electromagnetic induction
	Experiment 4	Natural driven oscillations
	Experiment 5	Magnetic fields in a coil
	Experiment 6	Solar energy conversion
	Experiment 7	Diffraction of electromagnetic radiation on microobjects II
	Experiment 8	Photoelectric effect
	Experiment 9	Polarization
	Experiment 10	Protection against ionizing radiation by distance
	Experiment 11	Monitoring of the natural background radiation
	Experiment 12	Study of spectra
	Experiment 13	Faraday phenomenon
	Experiment 14	Water level control
	Experiment 15	Rectifier
	Experiment 16	Series RLC circuit
	Experiment 17	VA characteristics of a LED - Measurement of Planck constant
Laboratory 2: RemLabNet	Experiment 1	Induction
	Experiment 2	Solar energy
	Experiment 3	Rectifier
	Experiment 4	Series RLC circuit
	Experiment 5	Diffraction of electromagnetic radiation in microobjects
	Experiment 6	Heisenberg
Laboratory 3: RExLab	Experiment 1	Painel Elétrico CC (DC electric panel)
	Experiment 2	Painel Elétrico CA (AC electric panel)
	Experiment 3	Ambiente para Desenvolvimento em Arduino (Arduino development environment)
	Experiment 4	Meios de Propagação de Calor (Heat propagation media)
	Experiment 5	Microscópio Remoto (Remote microscope)
	Experiment 6	Plano Inclinado (Inclined plane)
	Experiment 7	Disco de Newton (Newton's disc)
	Experiment 8	Conversão de Energia Luminosa em Elétrica (Conversion of light to electric energy)
	Experiment 9	Condução de calor em barras metálicas (Heat conduction in metal bars)
	Experiment 10	Arduino/block.ino
	Experiment 11	VISIR
	Experiment 12	Lab Água (Water laboratory)
Laboratory 4: WebLab-Deusto (fault-tolerance schemes bypassed)	Experiment 1	Xilinx VHDL FPGA Laboratory (one specific instance)
	Experiment 2	Arduino Remote Laboratory (one specific instance)
	Experiment 3	Arduino Robot Remote Laboratory (one specific instance)
Laboratory 5: GOLDi	Experiment 1	MCU + 3-axis portal laboratory
	Experiment 2	CPLD + production cell laboratory
	Experiment 3	MCU + 3-axis portal laboratory
	Experiment 4	CPLD + production cell laboratory

1) The laboratory has a validated educational background.
2) The laboratory grants access to more than a single experiment.
3) The laboratory grants the educational community open access during the dates on which the analysis was conducted.
4) Access to the laboratory does not depend on proprietary or platform-dependent technologies (e.g., NI, LabView, Java).

As a result, five remote laboratories were chosen, which provide access to a total of 42 different remote experiments.

- **Laboratory 1** groups together seventeen remote experiments provided by iSES [22],[1] or Internet School Experimental System.
- **Laboratory 2** groups together six remote experiments provided by RemLabNet [23].[2]
- **Laboratory 3** groups together twelve remote experiments provided by RExLab [24],[3] a remote laboratory linked to *Universidade Federal de Santa Catarina*.
- **Laboratory 4** groups together three examples of remote experiment instances provided by WebLab-Deusto [25] and integrated into LabsLand [19].[4] These examples are associated with the authors of this paper. They were accessed directly through WebLab-Deusto rather than through LabsLand, so as to bypass LabsLand fault-detection and fault-tolerance mechanisms, some of which are described in this work, and which,

[1] https://www.ises.info/index.php/en/laboratory

[2] http://www.remlabnet.eu/

[3] https://rexlab.ufsc.br/

[4] https://labsland.com/en

as described in more detail in later sections, would have prevented the observation of faults.

- **Laboratory 5** groups together four remote experiments provided by GOLDi [26],[5] or Grid of Online Laboratory Devices Ilmenau. Two of them are located at *Ilmenau University of Technology* and two in Ukraine.

Most of the 42 remote experiments are focused on scientific experiments in the area of physics and biology. The remaining nine experiments, however, are focused on programmable devices.

The remote experiments were analyzed on four different dates, spread over almost a month: 2020/04/16, 2020/04/26, 2020/05/04 and 2020/05/12. Each experiment was accessed on each of the four aforementioned dates and analyzed in terms of five criteria to characterize its behavior.

- **Available:** The experiment is available to the user. There is a link, URL or other form of access that allows the user to access the experiment. An unavailable experiment would be one that is marked as offline or lacks a URL or access link.
- **Accessible:** The experiment is accessible by the user. After using the link, URL or other form of access, the system shows the user the remote experiment interface panel. An inaccessible experiment would be one that does not allow the user to access the experiment even when the user shows their willingness to access. A clear example of this is when the interface shows ''loading'' and freezes.
- **Visible:** The experiment is visible via webcam (if the experiment has a webcam, which is very common). The experiment can be considered not to be visible if the webcam does not work correctly (but should) and the user has no visual feedback of the experiment. In many cases, this feedback is critical, but in other cases it simply provides complementary information or immersion. For example, in a typical robotics experiment or in a microscope experiment it tends to be critical, but not so much in an electrical circuit experiment in which measurement values are provided through other means.
- **Controllable:** The user can set-up or modify the values of the experiment parameters via the interface or upload the program with the control algorithm that controls the hardware. Otherwise, manipulation of the experiment is not effective, or although the user activates controls such as switches and buttons, these do not work. Another scenario for an uncontrollable experiment is one in which the hardware to be controlled is broken.
- **Observable:** The experiment can be observed in its entirety by the user via interface, and the data feedback is correct and consistent. It can be considered not to be observable if the data displayed are not correct, or the data that are published on the interface are not consistent with what is shown in the webcam.

[5]https://www.goldi-labs.net/

Table 2 shows the summary of the behavior of the 42 remote experiments analyzed in the five remote laboratories. The experiments are grouped within the corresponding laboratory and for each criterion, the number of days that the experiment failed is indicated. For example, if Experiment 1 in Laboratory 1 has a value of two in the *Visible column*, it means that on two of the four days that the experiment was accessed, the webcam did not work. This criterion is designed to be objective and is explained below. Its analysis indicates the extent to which an experiment is reliable or not.

In some cases, it was not possible to fill all the boxes if certain failures made it impossible to obtain the required information. For example, Experiment 9 of Laboratory 1 was not accessible on all four days, so it was impossible to determine the rest of the criteria, which is why several boxes are empty. In the case of Experiment 10 from Laboratory 1, it was inaccessible on half of the days, and on one of the days when it was accessible, it was visible, but uncontrollable.

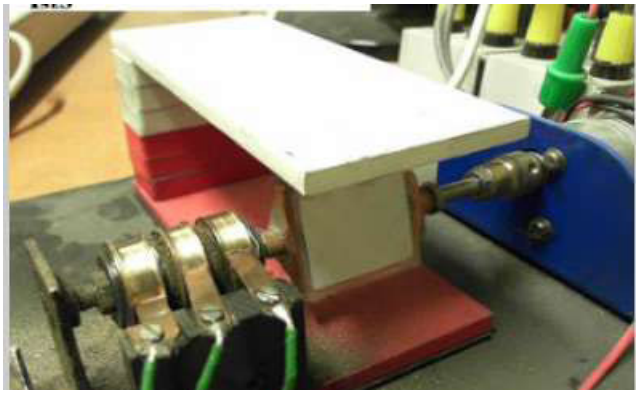


FIGURE 2. Webcam view of the experimentation bay of Laboratory 1, Experiment 3.

If the webcam of an experiment does not work (as is reflected in the *visible* column), the situation can have a different effect on students and their learning. If the *visible* cell is colored in light green, it means that the experiment is partially operational, although user experience is not complete. For example, Experiment 3 in Laboratory 1 is an inductive experiment that uses an electric motor, as can be seen in Figure 2. In this case, the result of the experiment is seen in the attached graph and therefore the webcam is only illustrative, although it clearly reinforces user experience and confidence in the experiment. In other cases, the camera is critical. For example in Experiment 14 in Laboratory 1, the camera shows the level of a liquid that is being controlled, and in Experiment 3 in Laboratory 4, the webcam shows the movement of the robot. In these two cases, the failure of the webcam hinders the performance of the experimentation process, so the cells are marked in red.

After evaluating the behavior of each of the experiments with respect to the previously set criteria, these were categorized according to their reliability. Table 2 lists four possible outcomes in its last column on the right. Those labs that did not experience errors are marked in green color, and were

TABLE 2. Summary of analysis of remote experiments over four days.

	Experiment	Available	Accessible	Visible	Controllable	Observable	Result
Laboratory 1	Experiment 1	0	0	2	0	0	Only camera non-critical failure
	Experiment 2	0	0	1	0	0	Only camera non-critical failure
	Experiment 3	0	0	3	0	0	Only camera non-critical failure
	Experiment 4	0	1	3	0	0	Single one-time critical error
	Experiment 5	4	-	-	-	-	Unreliable experiment
	Experiment 6	0	0	0	0	0	Fully reliable experiment
	Experiment 7	0	0	0	0	0	Fully reliable experiment
	Experiment 8	0	0	3	0	0	Only camera non-critical failure
	Experiment 9	0	4	-	-	-	Unreliable experiment
	Experiment 10	0	2	0	1	0	Unreliable experiment
	Experiment 11	0	2	0	0	0	Unreliable experiment
	Experiment 12	0	2	0	0	0	Unreliable experiment
	Experiment 13	4	-	-	-	-	Unreliable experiment
	Experiment 14	0	0	2	0	0	Unreliable experiment
	Experiment 15	0	0	1	0	0	Only camera non-critical failure
	Experiment 16	0	0	0	0	0	Fully reliable experiment
	Experiment 17	0	0	2	0	0	Only camera non-critical failure
Laboratory 2	Experiment 1	0	0	4	0	0	Only camera non-critical failure
	Experiment 2	0	3	0	0	0	Unreliable experiment
	Experiment 3	0	0	1	0	0	Only camera non-critical failure
	Experiment 4	0	0	0	0	0	Fully reliable experiment
	Experiment 5	0	4	-	-	-	Unreliable experiment
	Experiment 6	0	4	-	-	-	Unreliable experiment
Laboratory 3	Experiment 1	0	1	1	0	1	Unreliable experiment
	Experiment 2	0	1	0	0	0	Single one-time critical error
	Experiment 3	0	1	1	2	2	Unreliable experiment
	Experiment 4	0	4	0	-	-	Unreliable experiment
	Experiment 5	0	0	0	0	3	Only camera non-critical failure
	Experiment 6	0	0	0	0	0	Fully reliable experiment
	Experiment 7	0	0	0	0	0	Fully reliable experiment
	Experiment 8	0	0	0	0	0	Fully reliable experiment
	Experiment 9	0	3	0	0	0	Unreliable experiment
	Experiment 10	0	0	0	4	-	Unreliable experiment
	Experiment 11	0	0	0	0	0	Fully reliable experiment
	Experiment 12	0	0	0	4	-	Unreliable experiment
Laboratory 4	Experiment 1	0	0	0	2	-	Unreliable experiment
	Experiment 2	0	0	0	0	0	Fully reliable experiment
	Experiment 3	2	0	2	0	0	Only camera non-critical failure
Laboratory 5	Experiment 1	0	0	0	0	0	Fully reliable experiment
	Experiment 2	0	0	0	0	0	Fully reliable experiment
	Experiment 3	0	0	4	-	-	Unreliable experiment
	Experiment 4	0	0	2	0	0	Unreliable experiment

categorized as fully reliable experiments. Those labs that only had visibility failures but where the webcam was not critical have been marked in lighter green, and were categorized as reliable labs with non-critical failures. This category is special, since it includes experiments that, even presenting failures, would not have prevented their use in a class, for example. However, those laboratories where failures were only found during one of the four days of testing have been marked in yellow. During the testing period, it was not proven that they were especially unreliable, and having failed only for one day, it can be said that they could have been used in class without hindering the user experience too much. Finally, those experiments that presented critical errors for more than one day were categorized as unreliable experiments. This category includes those experiments whose webcam is critical and have had visibility failures.

B. QUANTITATIVE ANALYSIS

A total of 42 remote experiments from five remote laboratories were analyzed and their operation is reflected in Table 3.

TABLE 3. Operability of the analyzed remote laboratories.

Operability	Percentage
Fully operative experiments	26,1%
Semi-operative with non-critical errors	23,8%
Non-operative for at least one day	50,0%

Of these, 11 (26%) were fully operational for the four days and 21, (50%), presented a serious failure for at least one day. The rest of the experiments, 10 (24%), were partially operative. In the latter, the educational experience was correct but they presented some minor problems. For example, the webcam did not work but was not critical for learning since it only provided visual feedback to the user to increase the involvement in the experiment.

Remote experiments evolved differently over the testing period. Table 4, shows that nine (21%) experiments did not work at all during the analysis, similar to the number of experiments that worked on every occasion.

TABLE 4. Percentage of laboratories according to their operativity over a number of 4 days.

Operability per day	Percentage
Fully operative for 4 days	26,2%
Inoperative for 1 day	19,0%
Inoperative for 2 days	21,4%
Inoperative for 3 days	7,1%
Inoperative for 4 days	21,4%

TABLE 5. Relationship of types of errors and how they affected the analyzed remote experiments.

Type of error	Percentage
Experiment is offline	7,1%
Experiment is not accessible	30,9%
Experiment is not visible	35,7%
Experiment is not controllable	11,9%
Experiment is not observable	7,1%

Table 5 describes those errors, taking into account that the same experiment can present several errors simultaneously. The error distribution is not homogeneous. First, there are accessibility errors. In this case, the student, after accessing the remote laboratory interface and clicking on an experiment, sees that the experiment does not appear, does not load, or that the experiment interface cannot be controlled. This situation is very frustrating for the user. Errors in webcam functionality are present in 15 experiments (35%), and produce a very negative effect on users, since they do not see what is happening and experience a loss of control. In five of those cases (12% of the total), this failure is critical, since if the student does not see the controlled hardware, for example a robot, then there is neither experience nor learning.

Around 20% of the failures indicate that once students accesses the experiment, they are unable to control it or unable to extract the correct data from that experimentation session. For example, the student may not be able to program an Arduino device with the provided code, or once programmed, is unable to interact with the buttons or switches that control the programmed device. Other cases may show unrealistic parameters, such as zero milliamps of current flowing through a circuit that powers a light bulb, or a displayed mass of minus 200 kilograms when weighing a particular object.

Only three experiments, 7%, are registered as offline (not available). In these cases, users can know in advance that the experiment would not be available as supplementary material for their classes, and can take proactive action. This situation is therefore very different from the case of an inaccessible experiment, since in the latter case, both the experiment provider and the user may not be aware of the failure, and may be frustrated by this unexpected discovery.

Most experiments that presented failures more than once tended to present the same type of failure again. Of the 31 experiments that presented some type of failure, 26 presented the same failure during the testing period. Only five of the experiments presented multiple types of failure. The experiment that stands out in this respect is Experiment 3 of Laboratory 3. It was inaccessible on the first day, the webcam (a critical component in this experiment) failed on the second day, and it presented interface control errors on the two last days. It should also be noted that other types of problems may exist in those laboratories that are not accessible. Because they cannot be accessed, this cannot be verified, and therefore these cases are marked with dashes in Table 2.

C. QUALITATIVE ANALYSIS

From the previous analysis, the deduction is that the majority of the remote laboratories offer erroneous or unavailable experiments, which generates frustration and lack of confidence in remote experimentation for potential users, such as teachers and students who are looking to incorporate these technologies as in-class learning tools. In the following subsections, each of the error types are analyzed in a more detailed fashion.

1) EXPERIMENT NOT AVAILABLE OR OFFLINE

In this type of failure, the experiment is directly not available. This circumstance is normally communicated clearly and directly to the potential user, or at least the latter is inmediately made aware of the fact when trying to access. From a user experience perspective, this failure is the least negative one. In other types of failure, users invest time and effort before they realize that the experiment cannot be completed, or are even provided with incorrect results. In this case, such issues are avoided.

Unavailability can be due to different reasons, such as maintenance or repair tasks, or more serious failures automatically detected by the RLMS or *Remote Laboratory Management System*. RLMSs are tools that can facilitate the development of new laboratories via the integration of common features [25], [27]–[29].

It is noteworthy that, although less serious than other errors that have a more negative effect upon the user experience, unavailability is of course still negative. If users were expecting to use the resource in class, for example, and had planned this in advance, then they would have to suspend that session or work around it.

FIGURE 3. Three different remote experiment access panels that indicate availability failures.

Figure 3 shows three access panels for different experiments, showing availability failures. On the left, one can

see a message that appears when users attempt to access an experiment, indicating that it is offline. On the right, there are two access panels to different experiments, with red traffic lights, indicating that neither experiments are available.

2) EXPERIMENT NOT ACCESSIBLE

According to the results of the previous analysis, this is the second most common error in remote experiments, representing 30.9% of cases. In these cases, even though the experiment appears to be available, the control interface of the experiment cannot be accessed, or the experiment control panel does not load correctly. This can be due to a wide range of causes, including network problems, power-related problems, programming errors, or errors in the user/queue management system. Figure 4 shows an experiment control panel whose controls do not load correctly.

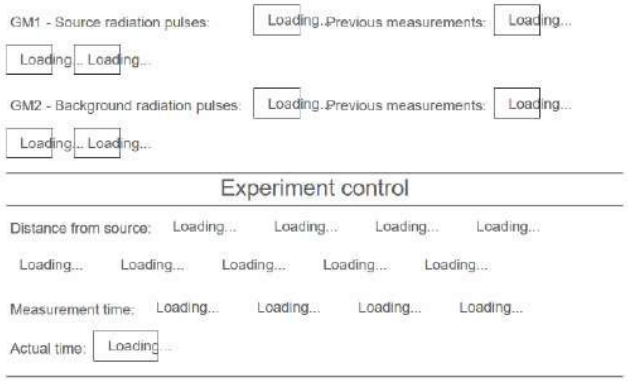


FIGURE 4. Example of loading and queue management errors in remote experiments.

3) EXPERIMENT NOT VISIBLE

The most common error found in the analysis is related to the visibility of the experimentation session. In 35.7% of cases, the experiment could not be seen correctly due to problems with the webcam image. In some cases the real-time image is not refreshed correctly or at all (so the results shown by the data do not match what is shown in the image) and in other cases the image is not available (due to network errors, laboratory errors, energy related errors, etc.).

In some of the experiments, the real-time view is not critical, because the experimentation data are also transmitted to the user by other means, but in the remaining experiments, the real-time streaming of the experimentation set-up is critical. In these cases, the remote experiment is rendered inoperative, as in cases 1 and 2.

Figure 5 shows an oscillation experiment where the webcam is not a critical component. In this case, the user cannot see the experimentation setup, but the measured data are still transmitted correctly to the user via a graph.

It is noteworthy that in general, this type of error was somewhat volatile. That is, some days it was present and others it was not. Specifically, out of the 15 experiments with webcam

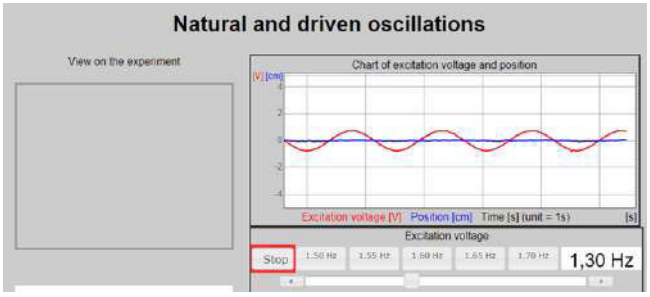


FIGURE 5. Example of a remote experiment that does not show the webcam view.

errors, only two displayed continuous errors throughout the testing period. In other words, the webcam brings instability to the experiment, probably due to network and configuration problems, and this instability generates insecurity for the user. If instructors are uncertain of whether or not students will be able to see the pendulum, it is likely that they will be very reluctant to incorporate the laboratory into their lesson plans.

4) UNCONTROLLABLE EXPERIMENT

Analysis has shown that although an experiment is available and accessible, it may not be controllable. In this case, when the experiment is accessed, it cannot be controlled for one or several reasons:

- Experiment inputs (switches, buttons, etc.) do not work.
- Experiment hardware (motors, servomotors, sensors, etc.) does not work.
- Experiment programmable devices (FPGA's, microcontrollers, etc.) are broken or switched off.
- Experiment programmable devices (FPGA's, microcontrollers, etc.) cannot be programmed.

These errors are usually hardware or network related, and in almost all cases do not allow users to conduct their experimentation sessions.

FIGURE 6. Example of a remote experiment that is not controllable by the user.

Figure 6 shows an experiment where a FPGA device should be programmed. In this case, it is impossible to upload the

FIGURE 7. Another example of a remote experiment that is not controllable by the user.

.bit file that programs the FPGA device, and therefore the experiment is uncontrollable. Figure 7 shows an experiment where water is pumped by an electrical pump into a tank. In this case, the electric pump is broken, and it is impossible to transfer the water into the tank. The errors that fall in this category are often difficult for laboratory maintenance personnel to detect, because a full experimentation session might be necessary in order to fully test all the hardware devices.

5) NON-OBSERVABLE EXPERIMENT

This is one of the least common experiment errors but it is particularly negative from a user experience perspective. In this case, the experiment is perfectly controllable by the user, but the feedback data transmitted to the user are incorrect or inconsistent. In many remote experiments, sensor data are captured and manipulated before being transmitted to the user by means of a control panel, and some errors can occur.

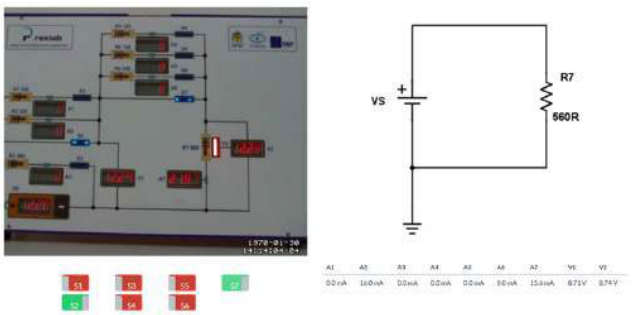


FIGURE 8. Example of a remote experiment that is not observable by the user. There is inconsistency between the results shown by the control panel and by the webcam view.

Figure 8 shows inconsistency in the experimentation results of an electrical measurement experiment. In this case, the data shown in the real-time video stream differ from the data projected on the control panel. The webcam shows a value of 12.23 V for measurement point V2, while the control panel indicates a voltage of 8.74 V. The same is true for measurement point A2, where a current of 0 mA is shown in the image and a current of 16 mA is shown on the control panel. In this case, the user may not know which set of values is correct, which renders the experimentation session useless. These errors are also difficult for laboratory maintenance personnel to detect, because a full experimentation session has to be carried out in order to fully test all of the signal-capturing devices.

In some cases, there might only be a single set of results reported to the user. If these results are wrong, users might in fact not realize and draw the wrong conclusions, or might take a long time to realize that they are obtaining incorrect results.

D. CONCLUSIONS OF ANALYSIS OF THE REMOTE EXPERIMENTS

A remote laboratory with its remote experiments is a complex design that combines software, hardware, graphic design and communications and therefore can fail for multiple reasons, creating frustration and lack of confidence in the end user, whether student or instructor. This is a significant obstacle that can prevent increased use of remote experimentation in classrooms.

The following sections describe a technological solution that mitigates the previously described issues. This solution integrates an array of automatic tests in the architecture of the remote laboratory, with the purpose of detecting failures and identifying those experiments that have failed as unavailable until they are repaired. Thus, the goal is for the system to know the state of a laboratory in real-time.

Once the state of each experiment is known, it is possible to employ this information in different ways. Firstly, the experiment maintenance personnel can be immediately aware of potential issues, and resolve these, if possible. For example, if a minor failure in a device such as a webcam is causing the experiment to fail, rebooting the webcam might be sufficient. Secondly, users can be made aware that the experiment is not available without having to experience failures once they are already performing the experiment and after they have already dedicated time and effort. Thirdly, this information can also be used as a basis for a more thorough fault-tolerance model. If there are multiple instances of the same experiment available, and one knows which are failing, it is possible to direct users only to those that are working. That way, users will not actually experience failures and the experiment will be available all the time, even if internally some instances of the experiment are experiencing issues. This model is discussed in the following sections.

III. PROPOSED SCHEME

As the analysis in previous sections shows, laboratories are often unreliable. While some could be made reliable by simply being particularly careful about software and hardware

and testing both thoroughly, there are others that contain many moving parts, or parts that wear out. These laboratories cannot be expected to work reliably for a long time unless specific measures are taken.

To achieve reliability, the proposed scheme works under one key assumption: all instances of individual experiments eventually break. To provide reliability to end users, it will therefore be necessary, first, to detect that a particular experiment instance is failing, and second, to have and be able to provide that user with a different, working instance of the experiment, transparently.

A. FAULT-DETECTION LAYERS

As previously explained, the key to the first part of this scheme is being able to detect that an experiment is failing. If this is not possible, the system will assume it is working, users will access it, and will potentially suffer a very bad user experience. Sometimes they will simply be unable to work on the laboratory as expected, and will have to reschedule their lessons, which might be a significant inconvenience. On other occasions, the experience might be even worse: they will not realize that the experiment is failing, but will instead be provided with incorrect experimentation data. They will invest their time in drawing incorrect conclusions, and will not trust the experiment again.

To detect failures, the proposed scheme relies mainly on four layers of tests:

- Exception catching
- Generic periodical health checks
- Experiment-specific periodical health checks
- Periodical test runs

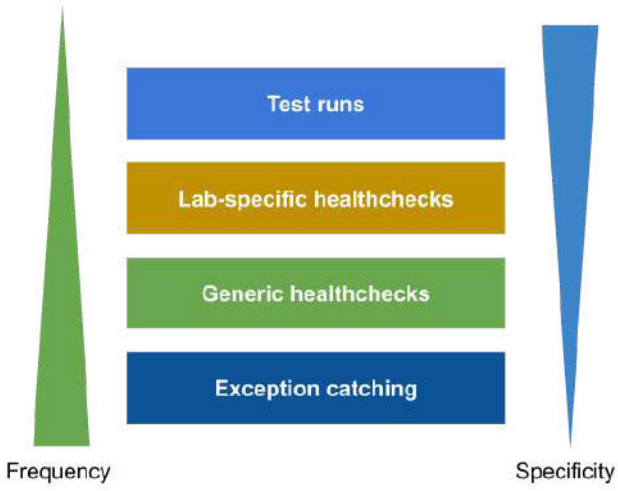


FIGURE 9. Diagram of the four fault-detection layers arranged in terms of frequency and specificity.

Figure 9 shows these four layers. The lower layers tend to be less experiment-specific, while the higher ones tend to be more experiment-specific. Likewise, the lower layers can be run more frequently, but the higher ones more rarely (because they tend to be more resource and time-intensive).

1) THE FIRST LAYER

relies on explicit exceptions that are captured and registered while a real user is using the experiment or when a test run is underway (test runs themselves will be described as the fourth layer, however). For example, if a student is using a remote pendulum experiment for physics and a servo handling system throws an exception, the laboratory system may assume that it is broken. This has at least two significant limitations. It only detects failures once they have occurred. Therefore, the experiment instance can be automatically turned off, but normally only after providing a poor experience for at least one user. Moreover, its effectiveness is limited: not all errors can be detected easily or throw exceptions. For example, in many cases, if a servomotor actually broke, the handling library would be unable to notice. The experiment would seemingly behave normally from a software perspective, but the pendulum (in this case) would not actually work.

2) THE SECOND LAYER

relies on generic periodical health checks that run against various components of the experiment. For example, an experiment could have components such as a device server to control remotized equipment, a web camera or an oscilloscope. Most generic health checks will involve actions such as sending HTTP GET or POST requests to particular endpoints, or simply *pinging* particular components. The complexity of a health check can vary. Normally, all health checks will expect a success response. For example, in the case of HTTP GET requests, they will most often expect a response with a 200 status code. In other cases, they may expect more specific data. For example, the web camera health checks expect to receive not only a 200 status code, but an apparently valid image. That is, binary data over a certain size that can be loaded as e.g. a JPEG. The checks in this layer are generic, in the sense that they are not experiment-specific. They will normally not check in detail that the specific experiment instance is working as expected, but rather ensure that certain key components are not offline or failing in an obvious way.

3) THE THIRD LAYER

is similar to the previous one, but includes experiment-specific checks. This kind of scheme resembles those used in other areas to help ensure reliability and failure detection.

The health checks in this layer will contain experiment-specific self-testing code. This will detect certain failures which would not be detected via high-level generic checks. The health check itself can be almost arbitrarily complex. For example, an experiment that involves sensors could try to read data received by the latter and evaluate its range, and report an error if it fails to read the data or if the values are not within a valid range. However, though powerful, these health checks also have certain practical limitations. One of these is that they are normally run periodically, while a user may be actively using the experiment. Thus, they need to be run relatively fast, and generally need to avoid interfering with

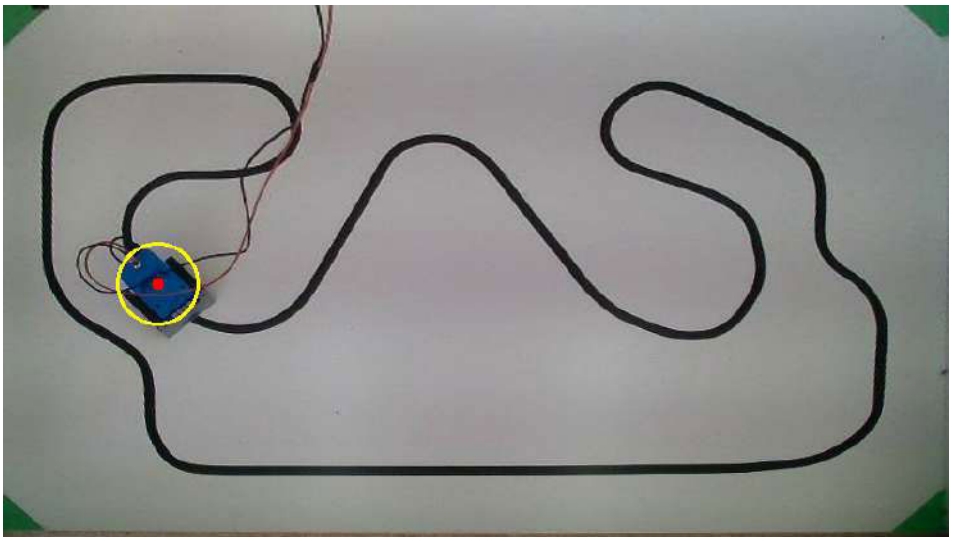


FIGURE 10. Frame of the robotics remote laboratory Arduino-based robot being tracked through computer vision techniques. The CV system detects a constellation of blue points over the white board.

user experience. Additionally, some hardware failures are still hard to detect.

4) THE FOURTH LAYER

relies on detailed periodical test runs. Those test runs will often simulate real usage of the experiment and will reserve it for a time, just as a real user would. The advantage of this is that more thorough tests can be run, and they can be results-oriented. Rather than checking isolated components, they can focus on the broader experience. As an example, in the case of the robotics experiment, the periodical tests will be run every few hours. They will then upload a line-following program into the robot, which will thus start moving round the circuit. A computer-vision based system tracks the robot's coordinates within the board. An image of a frame being processed and detecting the position of the robot can be seen in Figure 10. After a certain amount of time, the system evaluates whether or not the robot has followed the line correctly, and at what speed. If the robot has not moved, has not followed the line accurately enough, or has been moving too slowly, then the system assumes that the instance is broken. This can be due to a number of causes: a failing motor, infrared sensors covered with dust, etc. However, whatever the cause, it can be used to determine that the robot is not working as expected and thus mark the instance as failing, so that users can be redirected to different working instances.

B. FAULT-DETECTION EXAMPLE: LabsLand ROBOTICS LABORATORY

The LabsLand Arduino Robotics laboratory [30] features a real robot that is enclosed in a space and that students can program remotely. Figure 11 shows two instances of the Arduino Robotics laboratory deployed in the LabsLand offices in Bilbao. Other instances are deployed in universities around the world and are very similar (e.g., in the University of Deusto [31], in the University of Fort Hare [32], in UNAD [33] or in UNED in Costa Rica. This is a good example of the fault-detection scheme because it is a relatively complex laboratory with mechanical parts and several potential sources of failure. For example, the motors or the motor gears can break. This could be due to extensive usage, recurrent motor stalls due to students' programs, dust accumulation, or any other cause. Another observed fault source is dust accumulated over the infrared line sensors. There are also several others.

Figures 12 and 13 show the architecture of an instance of the laboratory. The online IDE that students use for programming is in the cloud. The *RLMS*, which can be shared with other laboratories, can be hosted anywhere (in the same space as the laboratory, or in the computer center of the institution that hosts it, or even in the cloud). The *laboratory server*, which is based on a Raspberry Pi 3 single-board computer, provides the Arduino robot-specific logic: it controls the robot hardware, manages programming, and in this case, even provides the UI through which students access the robot. The *experiment hardware* is the internal Arduino that is programmed by the student through the Raspberry Pi 3 and is directly connected to the sensors and actuators. The web camera provides a view of the robot, and the interactive live-streaming server is the component that offers that view to the users.

Some of the tests that are conducted to verify that particular instances are in working order are the following:

1) FIRST LAYER (EXCEPTION CATCHING)

Involves capturing all exceptions and generally assuming that unhandled ones or those that cannot be recovered imply that the particular instance is not working. Some failures that

FIGURE 11. LabsLand Arduino Robotics laboratory deployed at LabsLand, in Bilbao. The structure provides two separate experimentation instances.

generate exceptions and that involve marking the laboratory as failing are the following:

- Failing to program the Arduino.
- Failing to modify an emulated input peripheral value (e.g., the press of a button).

2) SECOND LAYER (GENERIC HEALTH CHECKS)

Those are run periodically (every 5-15 minutes). They can be configured but they do not have any experiment-specific logic. For example, the web camera check could test the web camera of this specific experiment, or the web camera of any other.

- Check that the web camera is responding.
- Check that the Raspberry Pi device is responding to pinging.
- Check that the HTTP server within the Raspberry Pi is responding with a 200-success code.
- Check that the web camera is providing seemingly valid images (valid JPEG files).

3) THIRD LAYER (EXPERIMENT-SPECIFIC HEALTH CHECKS)

These are also run periodically, but they are experiment-specific. For example:

- Verifying that the Arduino that is connected through USB to the Raspberry Pi is detected as a USB host with the correct ID.

4) FOURTH LAYER (TEST RUNS)

These are run more rarely (once every few hours). The instance of the laboratory is reserved, and a specific Arduino binary is programmed into the board, which should make the robot follow the line. Simultaneously, the CV-based system that was previously described (see Figure 10) keeps track of the position of the robot in real-time. The system thus verifies:

- That the line is followed as expected.
- That the circuit is completed in a certain amount of time.
- That the robot is stationary before the program begins, and stopped after the program ends.

The main disadvantage of this kind of test is that the experiment needs to be reserved and the robot has to execute it for a while. It also involves using the robot actively, so if it were to run thousands of times, this could lead to some extra wear of the robot. For this reason, it can be run less frequently than other tests. The advantage, however, is that it is very thorough. If the robot is not working as expected, the test will most likely be able to detect this, no matter which precise component is causing the failure. Figure 14 shows the UI that displays the result of a (successful) test run. The graphic above displays the robot's movements during the test, tracked automatically through computer vision.

We have included a supplementary MP4 format video file, which complements the explanation about the proposed solution and the integration of fault-detection layers in a multi-instance remote laboratory. The video shows the operation of a test-run of the aforementioned LabsLand Arduino Robotics laboratory, as well as the data collected for each test run. This material is 588.2 MB in size and will be available at http://ieeexplore.ieee.org.

C. MULTI-INSTANCE REMOTE LABORATORY ARCHITECTURE

The second step, after detecting that a remote laboratory is offline, is the capacity to redirect users seamlessly to another instance (a copy) of the same laboratory. Thus, users always access a working instance of the laboratory, do not experience any errors, and perceive that the laboratory is reliable. In order to make this model possible it is necessary to have various similar laboratory instances and a system that can keep track of which instances are working and direct incoming users to those that are working and available.

To achieve reliability based on replicability (multiple copies of the same experiment) it is necessary to use architectures that allow the control of a series of laboratory instances in an efficient way and with a limited cost. Maintaining a low cost makes it easier to have more instances. These instances

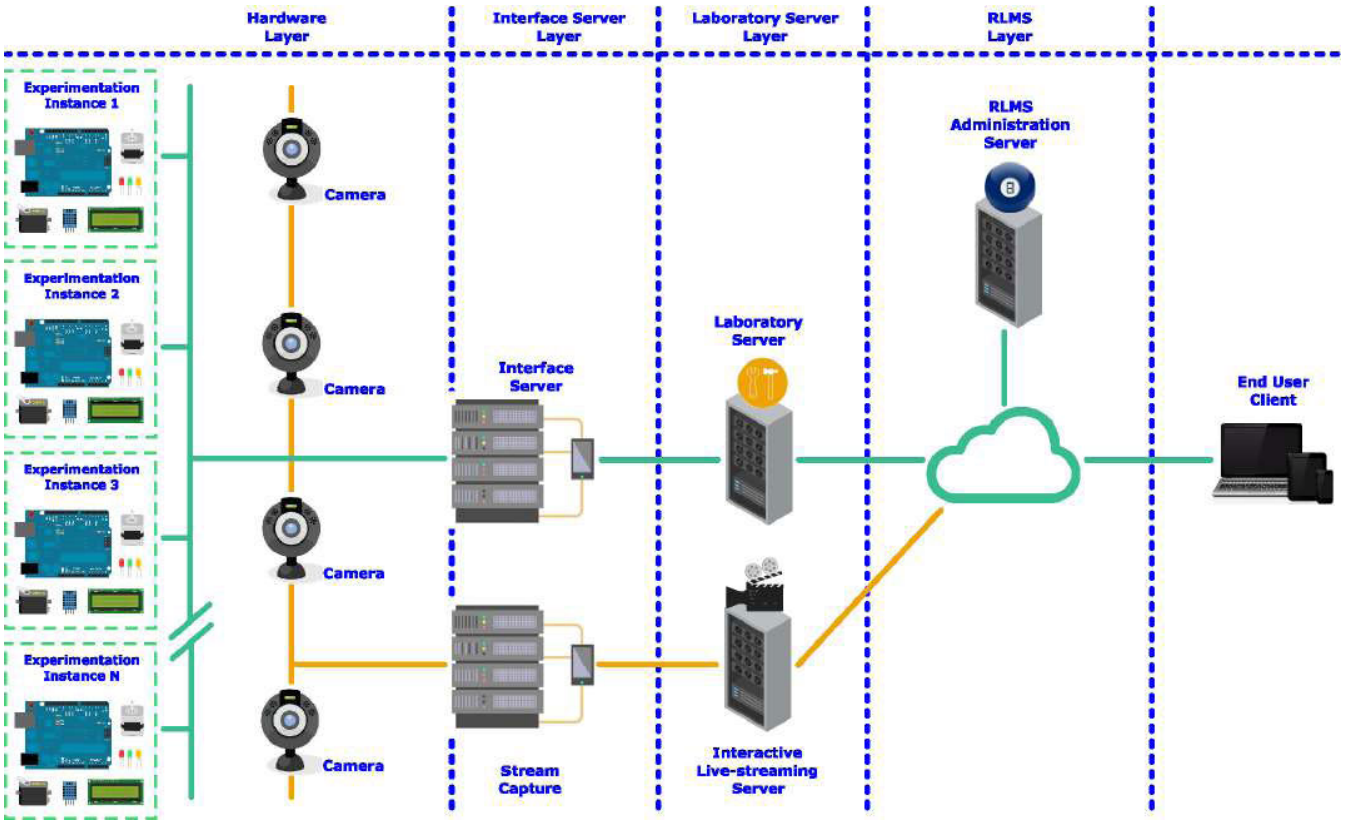


FIGURE 12. Overview of the multi-instance architecture split into several high-level layers. From [34].

can be used not only to provide greater reliability, but also in order to serve a larger number of simultaneous users. The architecture proposed in [34] allows for the creation of remote multi-instance laboratories based on the principles of high scalability, remotizable object flexibility and low cost. This architecture is partially-based on previous work from [35].

The aforementioned architecture is divided into four main layers to be able to control several experimentation instances of a remote laboratory. Figure 12 shows an overview of the architecture, with layer and component details. It is necessary to introduce the term *remotizable object*, which is the element or group of elements that are to be controlled remotely. Figure 12 shows an *Arduino UNO* microcontroller development board as the remotizable object, but this element can be replaced by any other element or experimentation setup. The layers that form the architecture are detailed below.

The lowest level layer in the architecture, which is the *hardware layer*, encompasses all of the hardware devices required to properly manage the remotizable object of the remote laboratory. Usually, a single-board computer coupled with a series of hardware components is capable of interfacing correctly with the remotizable object, but there are many different possibilities. In order to fit other laboratory setups, this layer could be based on a former PC or even a PXI chassis. The purpose of this layer is to replicate the physical actions that the user would perform in a hands-on laboratory. This remotizable object, which is the focus of the remote laboratory, is also part of the hardware layer.

Secondly, the *interface server layer* is an entity responsible for abstracting all of the control and hardware-related idiosyncrasies via a simplified REST API. It is a software layer that can be deployed in the aforementioned hardware device or in other specific devices. Its goal is to store and distribute all of the laboratory control tasks, which represent actions to be performed upon the remotizable object, to the existing hardware components. These tasks are received through a web server that exposes a REST API, via GET and POST requests, and are saved in a FIFO data structure. The tasks are then distributed sequentially to the available devices of the hardware layer in order to be performed. Both the storage and distribution of actions are based on Redis [36], an in-memory data store/message broker mixed component. As can be seen in Figure 13, the different objects of the *interface server layer* are connected via a TCP connection using HTTP protocol. This broadens the interconnection capabilities, enabling the use of different topologies and different numbers of devices in order to better adapt to deployment goals, such as capacity, modularity or cost.

Thirdly, the *laboratory server layer*, is a subsystem responsible for supporting the required interaction between user and experiment. This component supports the web-based client that is shown to the user, which includes a real-time video

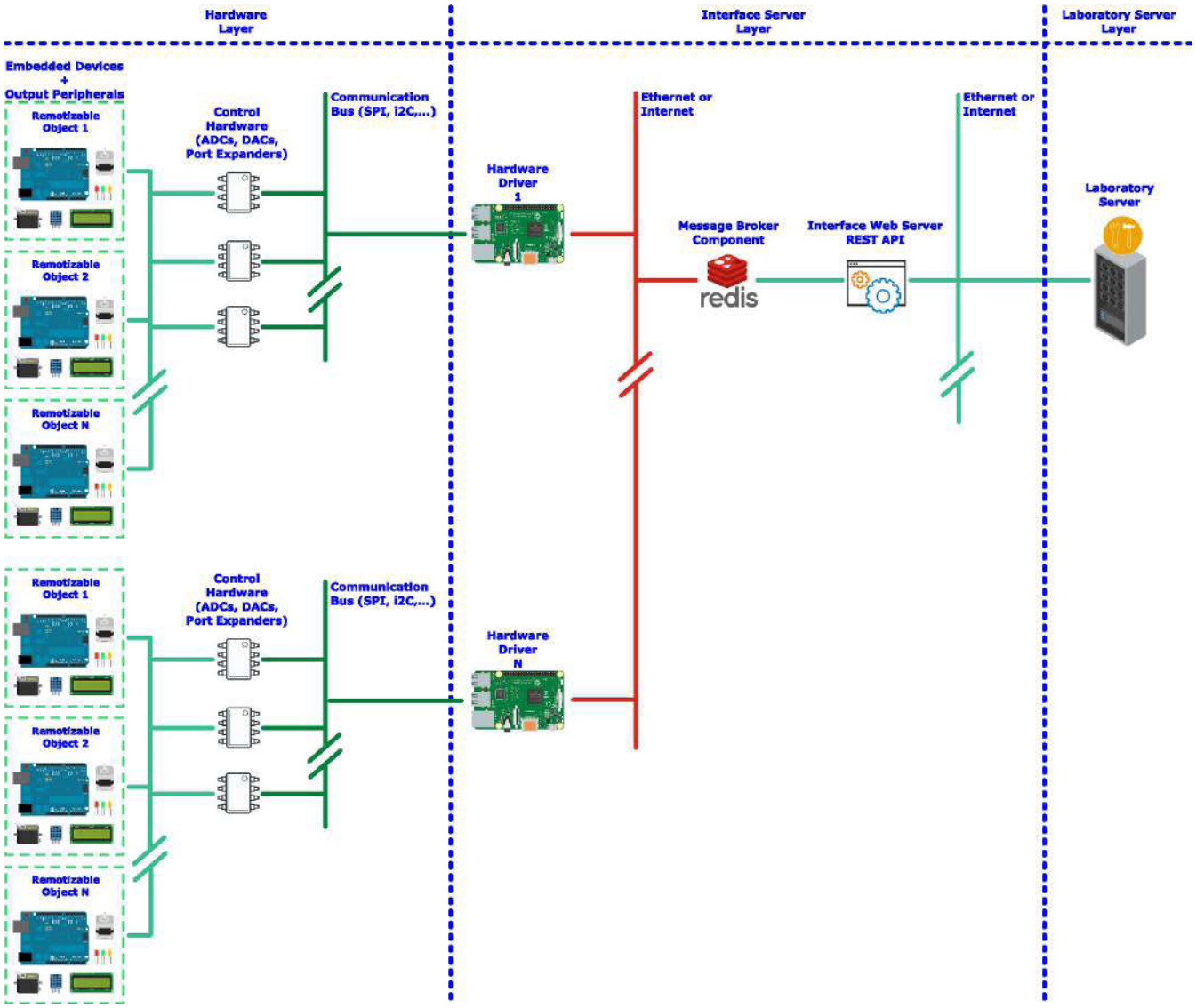


FIGURE 13. Detailed view of the multi-instance architecture split into several layers. In order to achieve scalability, the architecture supports one-to-many relationships (from right to left) between the components. From [34].

stream of the experiment and the different virtual laboratory controls. This entity also communicates with the *interface server layer* in order to propagate the user actions downstream and to retrieve the actual state of and information about the experiment.

Fourthly, the *remote laboratory management system* is an entity responsible for controlling all of the administrative tasks required to manage the different concurrent laboratory sessions that may arise. These tasks include user authorization, booking, queuing or load balancing, among others. It also serves as an integration tool, acting as a bridge between the remote laboratory and the different learning management systems or LMSs from which students gain access.

Finally, an *interactive live-streaming platform* is deployed in parallel to the other four subsystems. This component, which is also formed by various layers to ensure scalability and reliability, is responsible for of providing the user of the remote laboratory with a real-time video stream, which is captured on-site by a camera. This platform abstracts out specific camera peculiarities and reliably provides these functionalities.

The *remote laboratory management system* or *RLMS* layer mentioned above is responsible for controlling users and accessing laboratory instances. This layer has the information generated by the fault detection methods explained in Subsection III-A, and based on this is capable of redirecting users to the different available instances. If all the instances are available, the users will be distributed among them, in a transparent fashion. If any of the instances are not available, thanks to the active detection methods, users will only be distributed among the available instances, permitting a smooth experimentation experience. In the event that all available

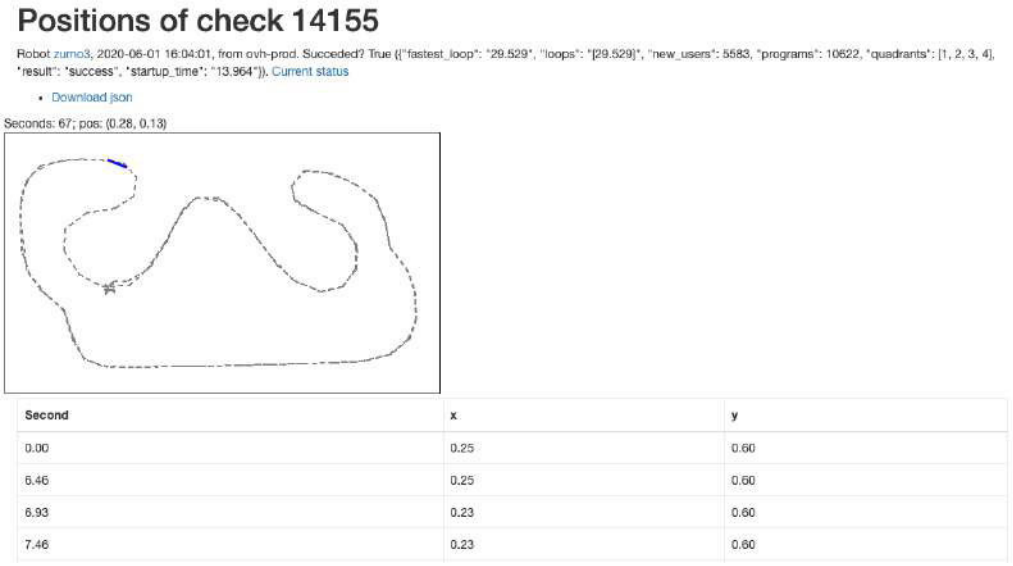


FIGURE 14. LabsLand Arduino Robot test run information panel. Shows the data acquired during the test run, along with a view of the points captured by the CV system that represent the robot's movement in the board.

instances fail, regardless of the severity of the failure, the user would be warned before accessing the laboratory.

This model therefore has thus two major advantages: firstly, it offers reliability, but also it makes it possible for multiple users to use the laboratory at the same time. The capacity will depend on the number of available working instances. With a high-enough number of instances, this makes it possible to use the laboratory in-class with a whole classroom, simultaneously. The number of instances that are required for this is usually significantly lower than the number of total students in the class, through the use of various strategies. Those strategies are nonetheless beyond the scope of this contribution.

When all of the instances are working correctly, the laboratory's capacity in terms of possible concurrent users is high. If any of the laboratory instances fail, the relationship between queued users and available instances is automatically recalculated. This keeps the laboratory available to all users, who also do not perceive that some instances of the laboratory may be defective.

D. MULTI-INSTANCE LABORATORY EXAMPLE: LabsLand ARDUINO BOARD LABORATORY

An example of a remote laboratory based on the aforementioned scalable laboratory architecture is the LabsLand Arduino Board laboratory. This remote laboratory is focused on providing users with the capability to program and interact with an Arduino UNO development board, which is based on an AVR ATmega328P microcontroller. This remote laboratory was developed as a proof of concept of the architecture, and at the time of writing this paper, sixteen laboratory instances have been deployed over the world. The total number of laboratory instances is expected to grow, as more remote laboratories similar to these are going to be deployed in different locations.

In order to mitigate the creation of single failure points, the existing twelve laboratory instances have been grouped into groups of four instances, which are located in different parts of the world. Two of these groups are in different locations in the city of Bilbao, Spain, another group is located in the University of Fort Hare, in Alice, Eastern Cape, South Africa, and the remaining group has recently been deployed in the UNED in Costa Rica.

Each experimentation instance is formed by the Arduino UNO Rev3 board and a series of peripherals, which include LED diodes, emulated potentiometers, a serial communications console, emulated switches, emulated buttons, an OLED screen and a servomotor. These peripherals are prearranged and already connected in a specific way, because users are unable to connect these components by themselves because the system is in a remote laboratory. A diagram, shown in Figure 15, is given to the users in advance, so they can program the Arduino board accordingly. Input peripherals, such as switches, buttons and potentiometers are emulated. The user controls virtual peripherals on the control panel, and via the use of hardware devices (e.g. digital-analog converters, port expanders, etc.), the equivalent electrical signals are transmitted to and from the Arduino board. These hardware devices are part of the *hardware layer* devices, which allow the user to replicate those electrical signals in-situ through the Internet.

To facilitate the deployment of the laboratory in each location, the four laboratory instances have been grouped into a single physical structure. As can be seen, Figure 16 shows the physical structure that contains the main PCB with the four Arduino UNO boards, its peripherals and the hardware parts needed to control each laboratory setup. The structure

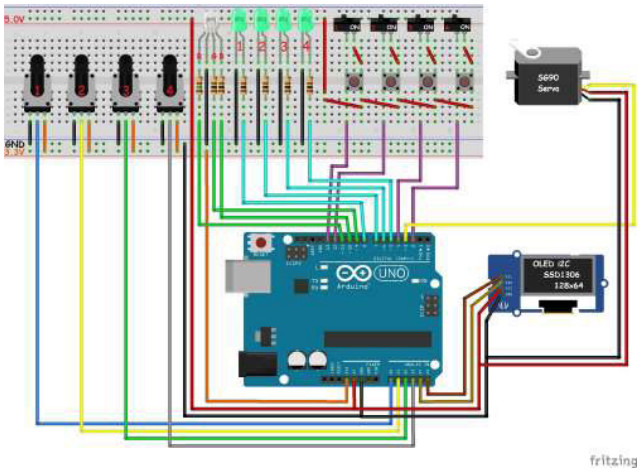


FIGURE 15. Fritzing schematic made available to the users of the ArduinoRL laboratory.

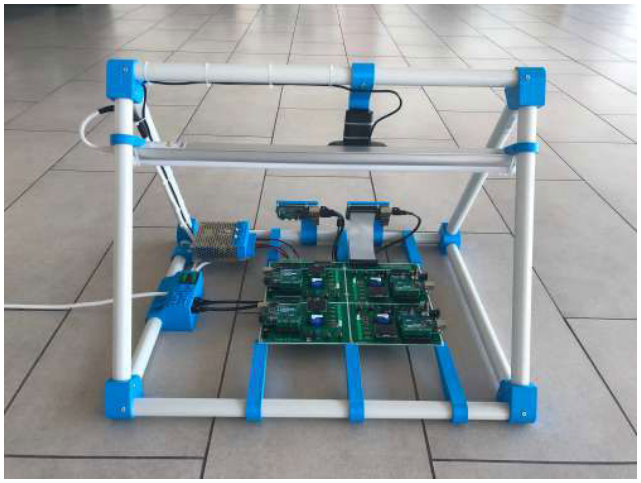


FIGURE 16. Overview of the Arduino Board lab structure, which is 3D printed and supports all the hardware components that are needed.

also serves as a support for two Raspberry PI3 single-board computers, one camera, one power supply, and a LED strip for illumination. The Raspberry PI3 single-board computer acts as a hardware platform to support the *interface server layer*. The single camera, in similar fashion, is capable of recording the four instances at the same time. The streaming platform is capable of slicing each laboratory setup, and providing each user with the correct video stream of each laboratory. As can be seen, the aforementioned remote laboratory architecture also provides flexibility and resource-sharing capabilities which allows for a lower deployment cost. Currently there are numerous remote laboratories similar to ArduinoRL in terms of experimentation capability devices used (Arduino-based development boards and Raspberry Pi-based control systems), but some of them [37]–[40] have a single experimentation instance, are not based on a resource-sharing capable architecture and/or do not use an RLMS as a decoupled management tool.

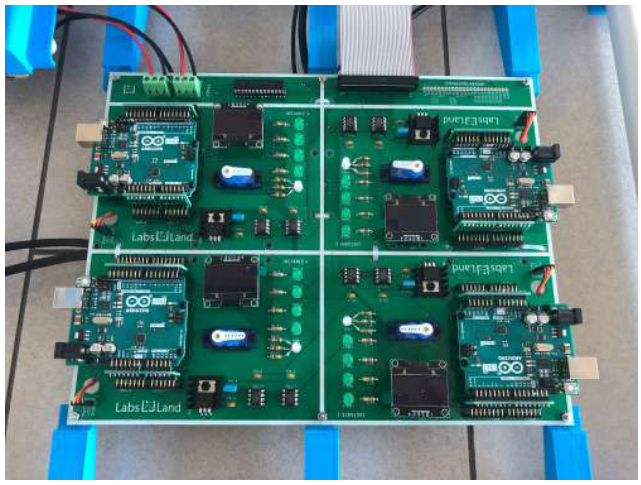


FIGURE 17. Detailed view of the Arduino Board lab PCB, which contains the Arduino UNO boards, as well as the required hardware devices.

In relation to the aforementioned structure layers, this remote laboratory can be described in the following manner:

- **Hardware layer:** It is formed both by the remotizable object and the control hardware. In this case, four remotizable objects (The Arduino UNO boards plus the LED diodes, servomotor, screen, etc.) coexist in a common PCB that provides the electrical interconnection between the devices, as well as physical support. The PCB can be seen in detail in Figure 17. The control hardware comprises a series of devices, such as *port expanders*, *digital-to-analog converters*, *a programmer* and other devices required to propagate the interactions between the user and the remotizable object and vice versa.
- **Interface Server layer:** This layer is limited hardware-wise, in a single Raspberry Pi 3 single-board computer. From a software point of view, it contains three major components, which are *the Hardware Driver*, the *Message Broker* and the *Interface Web Server*. This layer transforms the HTTP queries in the specific commands needed to control the hardware devices from the downstream layer. It also captures the experiment data and propagates them upstream. Redis [6] is used for the message broker and the interface server is based in Flask.[7]
- **Laboratory Server layer:** This software-only layer is responsible for providing the user with the laboratory-specific web-client, with all the tasks that this entails. It serves as a support for all the emulated controls (buttons, switches, sliding potentiometers), for the real-time video stream of the setup, for the serial communication console and for the Arduino code editor. Figure 18 shows the aforementioned control panel. As can be seen, the user is not aware that there are different instances of the same experiment, since the RLMS system in

[6]https://redis.io/
[7]https://flask.palletsprojects.com/en/1.1.x/

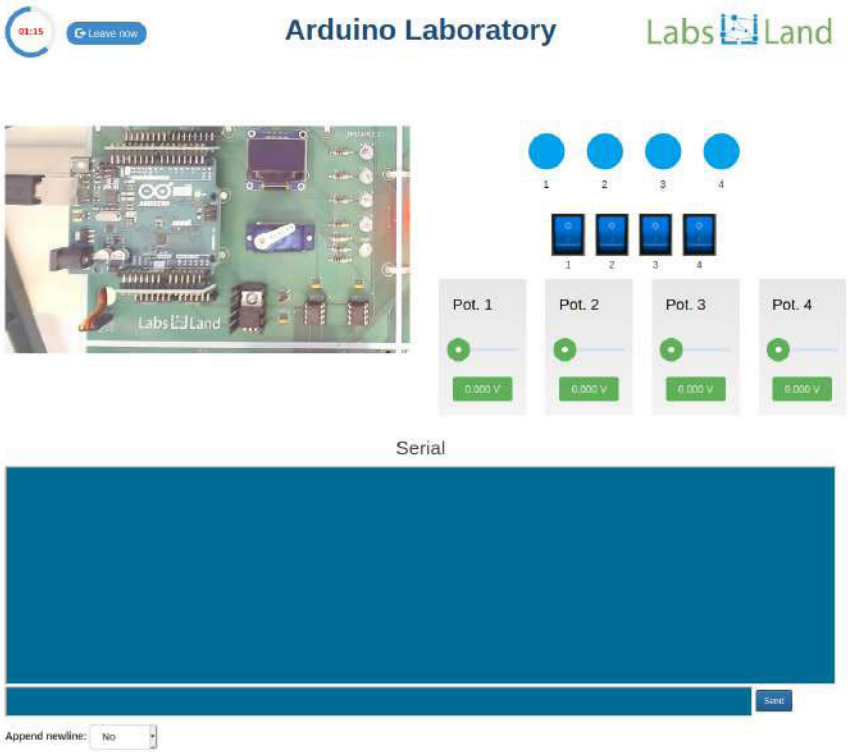


FIGURE 18. Control panel of the Arduino Board lab experiment. Contains a real-time video stream of the experimentation setup, virtualized switches, buttons and potentiometers, and the Serial console.

conjunction with this layer makes the control of each instance transparent.

- **Remote Laboratory Management System layer:** As in the previous Arduino Robot example, the RLMS for this laboratory is located in the cloud. Different locations use different RLMS components, since this component is deployed according to locations rather than laboratories.
- **Interactive live-streaming platform:** In parallel, this multi-layer component is also deployed together with the laboratory. From the hardware point of view, it uses a USB camera and a Raspberry Pi 3 as a capture element, as well as different services on cloud servers to process the images.

The goal of this layered distribution is to achieve a dual-use experimentation tool. Having multiple instances of the same experiment makes it possible to increase the capacity of the laboratory at a given moment, and provides the user with a satisfactory experimentation session, even when one or several instances are failing.

In order to perform the above, it is important to detect when a specific instance is failing. To do this, this laboratory also makes use of some of the fault detection layers explained in Section III-A. In this case, the first layer (*exception catching*), the second layer (*generic health checks*) and the third layer (*experiment-specific health checks*) are used.

IV. VALIDATION

So far, in our contribution we have analyzed five remote laboratories in the literature and their associated remote experiments, and determined that there are indeed general concerns regarding their reliability (Section II). We have then described a multi-layered architecture oriented towards detecting reliability issues and mitigating them through different approaches (Section III), the main one based on fault-tolerance through replication. To validate that approach, we have introduced those improvements into two specific remote experiments over which we have control. Our validation goal is thus to ensure that by means of these improvements we can achieve significantly improved reliability and availability for these two experiments, so that in the future other researchers and laboratory developers can also improve the reliability of their own remote experiments through similar approaches.

This section attempts to evaluate from a technical point of view whether the objective of improving the reliability of a remote laboratory using the solutions proposed in Section III has been met. The results obtained in the evaluation period of both solutions suggest that the objective has indeed been fulfilled. For this purpose, two additional remote experiments, out of the analysis included in Section II, have been designed in accordance with the methodology indicated in Section III. Both remote experiments were tested, over

a period of 16 days, from 07/12/2020 to 07/27/2020 in the case of the Arduino Robot Laboratory, and over a period of 5 months, from 05/01/2020 to 09/30/2020 in the case of the Arduino Board Laboratory.

A. THE LabsLand ARDUINO ROBOT LABORATORY

The first laboratory to use both proposed solutions is the LabsLand Arduino Robot Laboratory. As shown in section III, this lab allows the user to remotely control a robot based on the Arduino platform, which can be operated in a rectangular space. This laboratory currently has nine instances deployed globally, with four of them located in Bilbao, Spain, one in Colombia, two in South Africa and two in Costa Rica. The nine instances are physically deployed, though only five of them were operational at the time of conducting this study and writing this document. The two instances in South Africa can be seen in Figure 19.

FIGURE 19. LabsLand Arduino Robot Laboratory deployed at the University of Fort Hare, in South Africa. A single structure provides two different experimentation instances.

This laboratory uses both of the proposed solutions to ensure reliability for students, instructors and other potential users of the laboratory. Firstly, it uses the architecture mentioned in Section III to manage different equivalent instances in a transparent way, and secondly, it uses a series of failure detection tests in different layers to detect whether a specific instance fails. As mentioned in Section III, different types of tests are performed depending on the type of laboratory. In this case, due to its characteristics, it is possible to conduct periodic tests on the four layers described above. The most thorough test of those is the test run. In this test, a virtual computer-controlled user follows all the steps that a real user would perform. This virtual user makes a reservation in the laboratory manager, accesses it, programs the robot, executes the code, which makes the robot follow a certain circuit layout, captures a series of data, and leaves the laboratory. The information obtained allows the system to gather status information about the behavior of all the layers of the laboratory, from the queue management system to the operation of the specific laboratory hardware. In addition, when a critical failure is detected (the robot does not follow the circuit correctly, it does so but takes longer than it should, it is not possible to observe the robot or it is not possible to connect to the laboratory), it automatically disables that particular instance until the next successful test run. The result of each test is collected in an information panel (shown in Figure 14) that can be consulted by those who develop or maintain the laboratory. For each of the instances, three test runs are performed each day, at eight-hour intervals. This number of daily tests allows sufficient information to be obtained regarding the state of the robot, and at the same time is not too hard on the experiment hardware. A larger number of daily tests would accumulate too many uses, which could damage the robot prematurely.

During the sixteen days of evaluation, the five operational instances of the robot were tested daily. Table 6 shows the results of each test run (T1, T2 and T3) for each instance. The first significant conclusion that can be drawn is that all robots have failed at some point. The robots contain motors, gearboxes, and other mechanical components, and therefore failures are expected. The second significant lesson is that, despite the fact that all the robots have failed in the time frame in question, it has been possible to keep the service running, albeit with limited capacity. The worst day in this evaluation period was 07/20/2020, when three of the instances were completely inoperative during the whole day, and two other instances where inoperative for at least a third of the day. Even in this situation, the service was not inoperative from a users perspective, since at least one of the instances was working correctly. Thus, although in this situation users might have had to wait a longer time to access, even in this worst-case scenario, the service was online and working, and users with planned lessons using the laboratory would not have needed to suspend their activities. With a sufficiently high user-load, the service in that case could be slower, but the laboratory would still be reliable and would be perceived as such. As can be seen on the right-hand section of Table 6, the capacity of the laboratory was 80% or more during more than half of the 8-hour time frames, and it was never completely inoperative. This aspect highlights the importance of having several interchangeable instances of the laboratory. Usually, laboratory instances fail randomly, so by having them replicated, it is very unlikely that all instances will fail at once.

TABLE 6. Summary of the results of each test run for each of the operational instances of the LabsLand Arduino Robot Laboratory for each day in the evaluation period, as well as number of available instances and laboratory capacity.

Date	Test run	Instance 1	Instance 2	Instance 3	Instance 4	Instance 6	Available instances	Laboratory capacity	Laboratory online
07/12/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
07/13/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
07/14/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Fail	4 of 5	80.00%	Yes
07/15/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
07/16/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
07/17/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
07/18/2020	T1	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T2	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T3	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
07/19/2020	T1	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T2	Fail	Fail	Pass	Pass	Fail	2 of 5	40.00%	Yes
	T3	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
07/20/2020	T1	Fail	Fail	Pass	Pass	Fail	2 of 5	40.00%	Yes
	T2	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T3	Fail	Fail	Fail	Fail	Pass	1 of 5	20.00%	Yes
07/21/2020	T1	Fail	Fail	Pass	Pass	Fail	2 of 5	40.00%	Yes
	T2	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T3	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
07/22/2020	T1	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T2	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T3	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
07/23/2020	T1	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T2	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T3	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
07/24/2020	T1	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T2	Fail	Fail	Pass	Pass	Pass	3 of 5	60.00%	Yes
	T3	Fail	Pass	Pass	Pass	Pass	4 of 5	80.00%	Yes
07/25/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
07/26/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
07/27/2020	T1	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T2	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes
	T3	Pass	Pass	Pass	Pass	Pass	5 of 5	100.00%	Yes

This becomes more unlikely as the total number of deployed instances increases. In conclusion, it can be considered that the proposed solutions have improved the reliability of the laboratory service as a whole. Not having several instances would have led to unavailability for several days, which is unsustainable for daily use in class by instructors and students. Being able automatically to detect the different failures in the robot would have hindered the user experimentation sessions and provided a poor user experience, since users would have been exposed to the failures. Thanks to the use of these solutions, the laboratory has remained available and working correctly, and in addition, the laboratory maintenance team has been notified when the different failures have occurred, which minimizes the repair time of the laboratory instance.

B. THE LabsLand ARDUINO BOARD LABORATORY

As mentioned in Section III, LabsLand's Arduino Board remote laboratory bases its improved reliability on the combination of fault-detection techniques and the creation of several instances of the same experiment. Some of the instances share some of the hardware to reduce cost, and the hardware

TABLE 7. Summary of the incidents registered during the testing period of LabsLand's Arduino Board remote laboratory.

		University of Deusto				LabsLand Offices			
Date	**Fault type**	**Instance 1**	**Instance 2**	**Instance 3**	**Instance 4**	**Instance 1**	**Instance 2**	**Instance 3**	**Instance 4**
05/11/2020	Camera	1 min							
05/13/2020	Camera						1 min		
05/14/2020	Camera	2 min	2 min		2 min				
05/16/2020	Camera	1 min							
06/02/2020	Network	70 min	70 min	70 min	70 min				
06/08/2020	Network						1 min	1 min	
06/08/2020	Network						105 min	105 min	
06/12/2020	Network	1 min	1 min		1 min				
07/02/2020	Camera	2 min	2 min	2 min	2 min				
07/08/2020	Network					2 min	2 min	2 min	2 min
07/20/2020	Network						1 min	1 min	
07/21/2020	Development	3 min			3 min				
07/24/2020	Development	60 min	60 min	60 min	60 min				
07/26/2020	Network					45 min	45 min	45 min	45 min
08/14/2020	Network					1 min	1 min	1 min	1 min
08/18/2020	Network		2 min						
09/18/2020	Network		1 min	1 min					

to be shared is, by design, parts that are mostly reliable. The laboratory is currently deployed in four different locations (University of Deusto in Bilbao, Spain, LabsLand offices in Bilbao, Spain, University of Fort Hare in Alice, South Africa), and since recently, in UNED in Costa Rica, with four instances in each location. The data used for this analysis comes from the two deployments located in Bilbao, with a total of eight instances.

Regarding failure detection techniques, it is worth mentioning that in this laboratory only layers one, two and three of those mentioned in Subsection III-A are used. The experimentation set-up of this laboratory is simpler than the robot laboratory and contains fewer mechanical parts, which are the primary cause of failures in that laboratory. In addition, the hardware in this laboratory has been designed taking into account possible misuse by the user, which makes it more robust and less prone to failures.

Layers one, two and three tests, as they are software-based, can be performed with a high frequency, which minimizes the time required to detect possible failures. In the event of a failure being detected in a specific instance, it is taken offline automatically, and the users are redistributed among the remaining active instances in a transparent fashion. In addition, an email is sent to the maintenance personnel to ensure effective communication.

Due to the fact that this laboratory does not have hardware devices as delicate as in the case of the robot laboratory, the number of incidents detected is lower. Table 7 lists the incidents detected throughout the testing period. Each row of the table shows the date on which the incident occurred, the reason for the incident, the instances affected by the incident and the time in minutes during which the instance was unavailable.

As can be seen, a significant number of the incidents lasted three minutes or less. As a rule, these errors of such a short duration are usually glitches, specific communication errors or timeouts, and they tend to self-correct. Detecting these errors is possible thanks to the high frequency with which the tests are performed. If the tests had been conducted manually by the laboratory maintenance team, many of these errors would have gone unnoticed. During the testing period, other incidents also arose that took longer to resolve, close to an hour, but these were fewer in number.

The causes of the detected incidents were mostly related to the stability of the cameras, to the stability of the network and in a few cases, to the deployment of new code in some of the layers. Throughout the testing period, no errors were caused by the laboratory-specific hardware, and no electronic components had to be replaced. This is helped by the fact that it is a simple laboratory in terms of mechanical components (it only uses a servomotor, which only rotates to give visual feedback to the user) and its hardware was designed taking into account possible misuse by the user during the experimentation sessions.

In no case was the laboratory totally unavailable. On the worst days, the laboratory capacity was at least 50%, and only on four of the 17 days on which there were failures, did they last more than three minutes. On the remaining days of the testing period (136 of a total of 153), the laboratory maintained a full 100% capacity. Likewise, even in moments with a reduced capacity, the availability of the laboratory as a whole remained at 100%, since there was always at least one instance working correctly.

These results reinforce the idea that it is important to choose a laboratory architecture that allows the laboratory to scale at different levels. Thanks to this, the laboratory has been successfully deployed in different locations, which improves its resistance against points of failure that are difficult to control, such as instability of the network connection with the ISP or power outages.

V. CONCLUSION

Despite the important technological advances achieved in the field of remote experimentation and its proven validity as an educational tool, the adoption of remote laboratories in education has not led to a proportional progression. One of the main barriers to the adoption of remote laboratories by teachers is undoubtedly the lack of reliability of many of the remote experiments available. Very often, the providers of remote laboratories are the research teams in charge of their development, which usually include teachers who promote their use in courses of their own educational institution and, on many occasions, they share them with the community in a completely altruistic way. The complexity of remote laboratories, which include software and hardware components, requires constant maintenance to ensure proper use. Lack of control over experimental failures damages users' perception of the experiment by reducing the reliability of the technology. Apart from the cost, both in personnel and in equipment, it is necessary to provide techniques that allow early detection of any operating error. In order to identify the typology of the failures, this paper includes a study carried out over a set of 42 different experiments, hosted by six reference remote laboratories, quantitatively and qualitatively analyzing the errors sustained during multiple sessions of experimentation. This paper has identified the main types of failures deriving from the remote testing sessions, identifying the optimal action in response to each failure in order to guarantee the reliability of the remote laboratories. As a result, two compatible techniques are proposed to improve the reliability of remote laboratories: *Failure Detection* and *Scalability*. The first solution comprises an architecture that supports a set of fault detection levels, ranging from the lowest to the highest specificity, designed to validate the correct functioning of the different aspects of the experiment. Given the heterogeneity of the experiments analyzed, specific failures deriving from the very nature of each experiment have been detected, as well as other generic failures that arise globally with total independence of the particularities in the experiments. The less specific layers capture exceptions by automatically launching HTTP requests to verify that all laboratory resources are still online, while the more specific layers perform lab tests, simulating real user sessions and validating the results. These tests allow the self-detection of malfunctions in the laboratories without these needing to be reported by users, providing two improvements:

- Staff in charge of maintaining the remote laboratory are alerted early.
- The experimentation instance is marked as down for maintenance, preventing the user from experiencing failed experimental sessions.

The second proposed solution is the adoption of architectures for the deployment of remote laboratories focused on scalability and replicability. These architectures allow the deployment of multiple identical instances of experimentation, providing two fundamental advantages:

- The domain of students supported by the lab increases and concurrent users are allowed in the laboratory.
- In the event of one or more individual instances failing, load can be redirected to the available working instances, keeping the lab available in a transparent way for the user.

Both solutions were implemented in two remote laboratories to validate the improvements in the reliability of the experiments. In the case of the Arduino Robotics laboratory, it was observed that during a period of sixteen days, numerous faults were automatically detected, without compromising the availability of the laboratory thanks to the load balancing between the working instances available at each moment. At the same time, the implementation of the proposed techniques in the Arduino Board laboratory made it possible to keep the laboratory running continuously during a period of 5 months, during which time a total of 17 malfunctions were detected. The distributed disposition of multiple instances in different entities around the world has ensured that the availability of the laboratory has never fallen below 50% of its total capacity. Minimizing the downtime of a remote laboratory substantially improves the quality of service and the user experience. The proposed techniques have been shown substantially to improve the quality of remote laboratories by increasing laboratory reliability and user confidence. The proposed solution does not prevent or reduce the number of internal failures resulting from remote laboratories, but it does drastically reduce their effect on the availability of the remote experiment and it does reduce the number of failures to which users are exposed, thus improving the user experience.

FUTURE LINES

We are undergoing a global process of digitization that affects communication, industry, social relations and of course education. The distance education field is experiencing an enormous boom and its evolution depends on advances in educational technologies. Remote laboratories guarantee a process of interdisciplinary experimentation. For years, significant research efforts have been made in the development of remote laboratories. Powerful remote laboratory management systems (RLMS) [5], [41] have been developed such as iLab [42], LabShare [43], Weblab-Deusto [44], etc. From the social point of view, the educational efficiency of these tools has been demonstrated. However, the adoption of this technology depends on improving user experience and laboratory reliability. A bad experience with a remote laboratory undermines a user's confidence and prevents the adoption of other systems. This paper has laid a foundation in that direction but there is still a long way to go. From the educational point of view, a qualitative analysis of the user's perception must be conducted vis-à-vis users, both instructors and students, of remote laboratories designed according to the premises set out in this paper. Not all remote laboratories can provide 24/7 availability, but the deployment of fault

detection techniques makes it possible objectively to calculate the reliability of individual remote laboratories, providing users with a reliable indicator of a system's trustworthiness. Although there are very specific failures that require a custom development of the detection technology, most of the types of failures identified in the paper respond to generic malfunctions that usually affect remote laboratories, regardless of their nature, such as inability to access the server, failures in the authentication system, errors when accessing laboratory devices such as cameras, etc. Developing a standard that allows failure detection systems through Software as a Service (SaaS) methodology can facilitate the early detection of major errors and provide the availability ratio of each system, indicating to the user a quantitative parameter of the reliability of every system. This will make it possible to separate the reliability of a particular system from the reliability of remote laboratories in general. Detection of the specific defects of a remote laboratory also provides a line of research with many possibilities. Remote laboratories are systems that can be analyzed using the latest predictive maintenance technologies developed in the industry for the analysis of equipment used in production processes. The diagnostic capabilities of predictive maintenance technologies have increased in recent years with advances made in sensing technologies. Even in the most complex laboratories, with an infinite number of variables that complicate the repeatability of an experiment and make it difficult to detect errors in the provided data, the application of machine learning techniques allows for the automatic replication of experimental sessions and the evaluation of the validity of the results to detect an error in the calculations carried out by the laboratory. These techniques, combined with the capabilities of artificial vision, provide an open field of research with a view to developing failure detection systems suitable for remote laboratories that involve the use of any type of device. These future lines may constitute the cornerstone that will make possible the definitive take-off of remote experimentation in education.

ACKNOWLEDGMENT

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes one multimedia MP4 format video, which complements the explanation about the proposed solution and the integration of fault-detection layers in a multi-instance remote laboratory. The video shows the operation of a test-run in an Arduino-based robotics laboratory, as well as the data collected for each test run. This material is 588.2 MB in size.

REFERENCES

[1] C. Bohus, L. A. Crowl, B. Aktan, and M. H. Shor, "Running control engineering experiments over the Internet," *IFAC Proc. Volumes*, vol. 29, no. 1, pp. 2919–2927, Jun. 1996.

[2] S. D. Bencomo, "Control learning: Present and future," *IFAC Proc. Volumes*, vol. 35, no. 1, pp. 71–93, 2002.

[3] J. Ma and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review," *ACM Comput. Surv.*, vol. 38, no. 3, p. 7, Sep. 2006.

[4] J. R. Brinson, "Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research," *Comput. Educ.*, vol. 87, pp. 218–237, Sep. 2015.

[5] T. de Jong, M. C. Linn, and Z. C. Zacharia, "Physical and virtual laboratories in science and engineering education," *Science*, vol. 340, no. 6130, pp. 305–308, Apr. 2013.

[6] J. García-Zubía, U. Hernandez, I. Angulo, P. Orduña, and J. Irurzun, "Acceptance, usability and usefulness of Weblab-Deusto from the students point of view," *Int. J. Online Eng.*, vol. 5, no. 1, pp. 1–7, 2009.

[7] L. D. L. Torre, L. T. Neustock, G. K. Herring, J. Chacon, F. J. G. Clemente, and L. Hesselink, "Automatic generation and easy deployment of digitized laboratories," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7328–7337, Dec. 2020.

[8] J. E. Corter, J. V. Nickerson, S. K. Esche, C. Chassapis, S. Im, and J. Ma, "Constructing reality: A study of remote, hands-on, and simulated laboratories," *ACM Trans. Comput.-Hum. Interact.*, vol. 14, no. 2, p. 7, Aug. 2007, doi: 10.1145/1275511.1275513.

[9] M. Sauter, D. H. Uttal, D. N. Rapp, M. Downing, and K. Jona, "Getting real: The authenticity of remote labs and simulations for science learning," *Distance Educ.*, vol. 34, no. 1, pp. 37–47, May 2013, doi: 10.1080/01587919.2013.770431.

[10] P. de Vries, R. Klaasen, D. Ceulemans, and M. Ionnides, *Emerging Technologies in Engineering Education: Do We Need Them and Can We Make Them Work*. Delft, The Netherlands: Delft Univ. Technol., 2017.

[11] E. Pennisi. (Jul. 2020). *During the Pandemic, Students do Field and Lab Work Without Leaving Home*. Science Magazine. Accessed: Sep. 1, 2020. [Online]. Available: https://www.sciencemag.org/news/2020/07/during-pandemic-students-do-fie%ld-and-lab-work-without-leaving-home

[12] T. Flano. (Apr. 2020). *Una Startup Permite a Los Colegios Acceder Online y Gratis a Laboratorios Reales [A Startup Allows Schools to Access Real Laboratories Online for Free*. Diario Vasco. Accessed: Jul. 1, 2020. [Online]. Available: https://www.diariovasco.com/sociedad/educacion/startup-permite-colegios%-20200401184428-nt.html

[13] J. García-Zubía and U. Hernández-Jayo. (Jul. 2020). *Los Laboratorios Remotos Revolucionan el Aprendizaje Desde Casa [Remote Labs Revolutionize Learning From Home]*. The Conversation. Accessed: Sep. 1, 2020. [Online]. Available: https://theconversation.com/los-laboratorios-remotos-revolucionan-el-ap%rendizaje-desde-casa-137101

[14] K. Pretz, "German University opens up its hands-on remote FPGA lab during the coronavirus pandemic," *IEEE Spectr.*, May 2020. Accessed: Sep. 1, 2020. [Online]. Available: https://spectrum.ieee.org/news-from-around-ieee/the-institute/ieee-member-news/ger man-university-opens-up-its-handson-remote-fpga-lab-during-the-coronavirus-pandemic

[15] A. Linden. (May 2020). *UFH ReVEL Online Technology Prepares to Provide Support During Lockdown*. Univ. Fort Hare News. Accessed: Sep. 1, 2020. [Online]. Available: https://www.ufh.ac.za/news/News/UFHReVELONLINETECHNOLOGYPREPARESPROVIDE%SUPPORTDURINGLOCKDOWN

[16] W. Sawahel, "COVID-19 drives development of online laboratories," Higher Educ. Web Publishing, Jul. 2020. Accessed: Sep. 1, 2020. [Online]. Available: https://www.universityworldnews.com/post.php?story=20200715130543961

[17] J. García-Zubía and G. R. Alves, *Using Remote Labs Education: Two Little Ducks Remote Experimentation*, vol. 8. Bilbao, Spain: Univ. Deusto, 2012.

[18] H. A. Lahoud and J. P. Krichen, "Networking labs in the online environment: Indicators for success," *J. Technol. Stud.*, vol. 36, no. 2, pp. 31–40, May 2010.

[19] P. Orduña, L. Rodríguez-Gil, J. García-Zubía, I. Angulo, U. Hernandez, and E. Azcuenaga, "LabsLand: A sharing economy platform to promote educational remote laboratories maintainability, sustainability and adoption," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2016, pp. 1–6.

[20] N. Wang, X. Chen, G. Song, Q. Lan, and H. R. Parsaei, "Design of a new mobile-optimized remote laboratory application architecture for M-learning," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2382–2391, Mar. 2017.

[21] M. Schulz, F. Chen, and L. Payne, "Real-time animation of equipment in a remote laboratory," in *Proc. 11th Int. Conf. Remote Eng. Virtual Instrum. (REV)*, Feb. 2014, pp. 172–176.

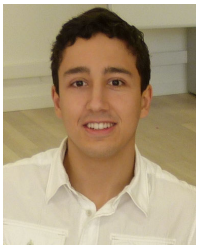
[22] P. Brom, F. Lustig, P. Kuriščák, and J. Dvorak, "DIY 'hands-on-remote'experiment in physics with Arduino," in *Proc. 15th Int. Conf. Remote Eng. Virtual Instrum. (REV)*, Düsseldorf, Germany: Univ. Applied Science Dusseldorf (HSD), Mar. 2018, pp. 617–624.

[23] M. Ožvoldová and F. Schauer, *Remote Laboratories: In Research-Based Education of Real World Phenomena*. Bern, Switzerland: Peter Lang Publishers, 2016.
[24] J. B. Da Silva, S. M. S. Bilessimo, and K. C. N. da Silva, "A estratégia de integração de tecnologia na educação do Grupo de Trabalho em experimentação remota Móvel (GTMRE) [mobile remote experimentation workgroup (GTMRE) strategy for integrating technology into education]," *Revista Tecnologias na Educação*, vol. 8, no. 17, pp. 1–14, 2016.
[25] P. Orduña, J. Irurzun, L. Rodríguez-Gil, J. García-Zubía, F. Gazzola, and D. López-de-Ipiña, "Adding new features to new and existing remote experiments through their integration in WebLab-Deusto," *Int. J. Online Eng.*, vol. 7, no. S2, pp. 33–39, 2011.
[26] K. Henke, H.-D. Wuttke, R. Hutschenreuter, and D. Streitferdt, "Interactive content objects as GOLDi-lab services: Interactive demonstration of ICOs within the hybrid online onlne lab GOLDi," in *Proc. 5th Exp. Int. Conf. (exp.at)*, Jun. 2019, pp. 229–230.
[27] F. Schauer, M. Krbecek, P. Beno, M. Gerza, L. Palka, and P. Spilakova, "REMLABNET—Open remote laboratory management system for e-experiments," in *Proc. 11th Int. Conf. Remote Eng. Virtual Instrum. (REV)*, Feb. 2014, pp. 268–273.
[28] A. Maiti, A. D. Maxwell, and A. A. Kist, "Features, trends and characteristics of remote access laboratory management systems," *Int. J. Online Eng.*, vol. 10, no. 2, pp. 30–37, 2014.
[29] D. Lowe, S. Murray, L. Weber, M. de la Villefromoy, A. Johnston, E. Lindsay, W. Nageswaran, and A. Nafalski, "LabShare: Towards a national approach to laboratory sharing," in *Proc. 20th Annu. Conf. Australas. Assoc. Eng. Educ.*, Adelaide SA, Australia: The School of Mechanical Engineering, Univ. Adelaide, 2009, pp. 458–463.
[30] I. Angulo, J. García-Zubía, U. Hernández-Jayo, I. Uriarte, L. Rodríguez-Gil, P. Orduña, and G. Martinez Pieper, "RoboBlock: A remote lab for robotics and visual programming," in *Proc. 4th Exp.@Int. Conf. (exp.at)*, Jun. 2017, pp. 109–110.
[31] J. García-Zubía, I. Angulo, G. Martínez-Pieper, P. Orduña, L. Rodríguez-Gil, and U. Hernández-Jayo, "Learning to program in K12 using a remote controlled robot: Roboblock," in *Online Engineering & Internet of Things*, M. E. Auer and D. G. Zutin, Eds. Cham, Switzerland: Springer, 2018, pp. 344–358, doi: 10.1007/978-3-319-64352-6_33.
[32] P. M. Kwinana, P. Nomnga, M. Rani, and M. L. Lekala, "Real laboratories available online: Establishment of ReVEL as a conceptual framework for implementing remote experimentation in South African higher education institutions and rural-based schools—A case study at the University of Fort Hare," in *Proc. Int. Conf. Remote Eng. Virtual Instrum.*, Springer, 2020, pp. 128–142.
[33] P. A. Buitrago, R. Camacho, H. E. Pérez, O. Jaramillo, A. Villar-Martínez, L. Rodríguez-Gil, and P. Orduña, "Mobile arduino robot programming using a remote laboratory in unad: Pedagogic and technical aspects," in *Cross Reality and Data Science in Engineering*, M. E. Auer and D. May, Eds. Cham, Switzerland: Springer, 2021, pp. 171–183, doi: 10.1007/978-3-030-52575-0_14.
[34] 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.
[35] I. Angulo, L. Rodríguez-Gil, and J. García-Zubía, "Scaling up the lab: An adaptable and scalable architecture for embedded systems remote labs," *IEEE Access*, vol. 6, pp. 16887–16900, 2018.
[36] J. L. Carlson, *Redis in Action*. New York, NY, USA: Manning Publications Co., 2013.
[37] P. Alexander and N. Radhakrishnan, "Remote lab implementation on an embedded Web server," in *Proc. Int. Conf. Circuits, Power Comput. Technol. [ICCPCT]*, Mar. 2015, pp. 1–5.
[38] H. Mostefaoui and A. Benachenhou, "Design of a remote electronic laboratory," in *Proc. Int. Conf. Interact. Mobile Commun. Technol. Learn. (IMCL)*, Nov. 2015, pp. 160–162.
[39] A. Fernández-Pacheco, S. Martin, and M. Castro, "Implementation of an Arduino remote laboratory with raspberry Pi," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Apr. 2019, pp. 1415–1418.
[40] R. Costa, F. Perola, and C. Felgueiras, "μLAB A remote laboratory to teach and learn the ATmega328p μC," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Apr. 2020, pp. 12–13.
[41] M. Tawfik, D. Lowe, S. Murray, M. de la Villefromoy, M. Diponio, E. Sancristobal, M. J. Albert, G. Diaz, and M. Castro, "Grid remote laboratory management system," in *Proc. 10th Int. Conf. Remote Eng. Virtual Instrum. (REV)*, Feb. 2013, pp. 1–9.
[42] V. J. Harward, J. A. D. Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, I. Jabbour, and P. D. Long, "The iLab shared architecture: A Web services infrastructure to build communities of Internet accessible laboratories," *Proc. IEEE*, vol. 96, no. 6, pp. 931–950, May 2008.
[43] D. Lowe, S. Murray, E. Lindsay, and D. Liu, "Evolving remote laboratory architectures to leverage emerging Internet technologies," *IEEE Trans. Learn. Technol.*, vol. 2, no. 4, pp. 289–294, Aug. 2009.
[44] P. Orduña, J. García-Zubía, L. Rodríguez-Gil, I. Angulo, U. Hernández-Jayo, O. Dziabenko, and D. López-de-Ipiña, *The WebLab-Deusto Remote Laboratory Management System Architecture: Achieving Scalability, Interoperability, and Federation of Remote Experimentation*. Cham, Switzerland: Springer, 2018, pp. 17–42, doi: 10.1007/978-3-319-76935-6_2.

AITOR VILLAR-MARTÍNEZ received the Bachelor of Science degree in telecommunication engineering and the Master of Science degree in telecommunication engineering from the University of Deusto, in 2016 and 2018, respectively. Since 2014, he has collaborated in various projects at the University, such as the 2015 and 2016 editions of the Smart Moto Challenge. Since 2017, he has been working as a Hardware/Software Developer with LabsLand (spin-off of the WebLab-Deusto project). He started his Ph.D. studies with the University of Deusto, in 2018, which involve work and research in relation to remote laboratories and the technologies employed in the latter. Since 2018, he has been collaborating with the MORElab Research Group, DeustoTech - Deusto Institute of Technology.

JAVIER GARCÍA-ZUBÍA (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Deusto, Spain. He is currently a Full Professor with the Faculty of Engineering, University of Deusto. He is also the Leader of the WebLab-Deusto Research Group. His research interest includes remote laboratory design, implementation, and evaluation.

IGNACIO ANGULO has been with the Department of Information Technology, Electronics and Communication, University of Deusto, since 2002. He has participated in over 25 research projects. He has collaborated in the writing of 32 scientific articles published in magazines of international impact. He presented his Ph.D. thesis on Open Architecture for the Deployment of Remote Laboratories on Embedded Systems, in 2015.



LUIS RODRÍGUEZ-GIL (Senior Member, IEEE) received the dual degree in computer engineering and industrial organization engineering, in 2013, the M.Sc. degree in information security, in 2014, and the Ph.D. degree in computer science from the University of Deusto, in 2017. During his Ph.D., he co-founded the LabsLand remote labs company, where he currently works as a full-time as its CTO. Since 2009, he has been a part of the WebLab-Deusto Research Group, collaborating in the development of the WebLab-Deusto RLMS. Throughout this time, he has authored various peer-reviewed publications and contributed to several open source projects.