Received December 21, 2017, accepted March 4, 2018, date of publication March 7, 2018, date of current version April 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2812925

Scaling up the Lab: An Adaptable and Scalable Architecture for Embedded Systems Remote Labs

IGNACIO ANGULO[1], LUIS RODRÌGUEZ-GIL[1,2], AND JAVIER GARCÌA-ZUBÌA[1], (Senior Member, IEEE)
[1]Faculty of Engineering, University of Deusto, 48007 Bilbao, Spain
[2]LabsLand, 48014 Bilbao, Spain

Corresponding author: Ignacio Angulo (ignacio.angulo@deusto.es)

The work of L. Rodriguez-Gil was supported by the Department of Education, Language Policy, and Culture of the Basque Government though a Predoctoral Scholarship. This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors, including a screencast that briefly shows the lab implemented under the proposed architecture.

ABSTRACT Upgraded embedded systems and technologies are continuously appearing. Remote laboratories are an ever more promising technology that educators can use to train their students. However, they are still costly to develop and maintain. Most software and hardware architectures are tailored to specific hardware boards, and little can be reused when a laboratory for a different board is to be created. This paper proposes a novel remote laboratory architecture. It is specifically designed to support a wide range of embedded systems, so that laboratories for new or upgraded ones can be implemented and deployed easily and efficiently. In order to propose an appropriate architecture, some of the most significant deployed remote laboratories for embedded systems education are first analyzed, and the key design requirements are determined. In order to evaluate it, a new microcontroller-oriented remote laboratory that implements it has been created. Results suggest that the proposed architecture does indeed meet the requirements and that it is a step towards more cost-efficient and reusable remote laboratory architectures.

INDEX TERMS Remote laboratory, embedded system, experimentation, online education.

I. INTRODUCTION

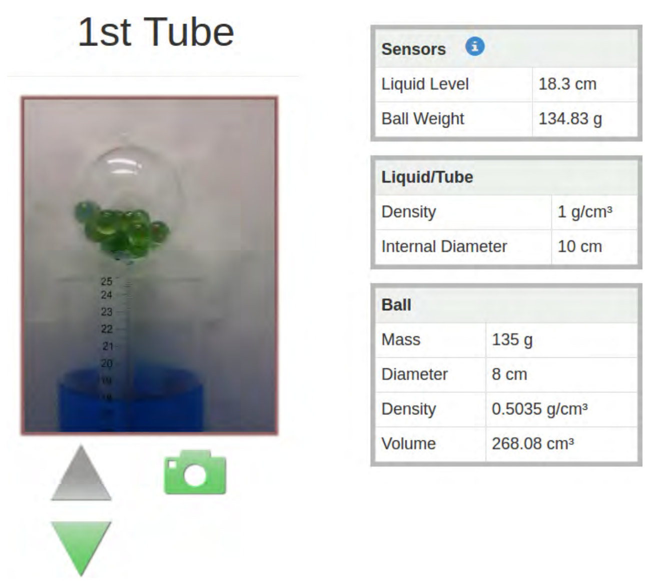
Today, most of the current remote labs for embedded systems have three significant constraints. Firstly, adapting the specific embedded system to new devices or models is not straightforward. This difficulty increases costs, limits the potential of the laboratories, and makes it more expensive to adapt to technological advancements. Secondly, they often do not allow the user to experience the full design-and-test workflow. This workflow should not only include programming and testing, but also debugging, which is a feature that is often lacking. Thirdly, this workflow is often not integrated into a single web-based platform, negatively affecting the user experience. Therefore, the main motivation and research challenge of this work is to design and evaluate a novel laboratory architecture that can overcome these three constraints.

The embedded systems field is always changing. New boards and upgraded devices are constantly being release. This demands from developers and educators continuous training in emerging technologies. Due to the wide catalog of vendors and device families, the election of the appropriate device is a very important step in the embedded systems design cycle. Educators have the responsibility to train students in skills for the most widely used technologies in the industry. For this, they need to train themselves in industrial informatics and to continuously update the labs and equipment they use during their courses. This requires a significant monetary and time investment that is often difficult to amortize.

Since 1996 [1], remote experimentation is part of Technology Enhanced Learning (TEL). This field can benefit from the emergence of various technologies that allow students to access lab resources remotely, including virtual and remote labs [2]. Remote experimentation provides users (e.g., students, teachers, researchers) with the means to experiment in a similar way as in a hands-on lab. Remote labs are suitable for teaching [3]–[6] and they can be considered a new trend in education [7]. Furthermore, a company can promote their new products through a remote lab. Instead of shipping samples and managing pre-stocking, they can offer them remotely to prospective buyers.

For all these reasons, there are many possibilities for remote experimentation in the field of embedded systems.

There are a high number of deployed remote labs for embedded system devices [8], [9].


1st Tube

Sensors	
Liquid Level	18.3 cm
Ball Weight	134.83 g

Liquid/Tube	
Density	1 g/cm³
Internal Diameter	10 cm

Ball	
Mass	135 g
Diameter	8 cm
Density	0.5035 g/cm³
Volume	268.08 cm³

FIGURE 1. **Archimedes remote lab at https://labsland.com.**

Generally, remote labs provide both input and output peripherals. The inputs (e.g., switches) can be controlled through the lab client's GUI. The outputs can be observed through a live webcam. Fig.1 shows an example of such a lab. Users can see several liquid-filled tubes. They can calculate whether certain objects would sink or float if they were dropped into the liquid. Then, they can drop them for real and see whether their calculations were correct. They can also monitor several variables such as ball weight or liquid height.

In the case of an embedded system, users would often write a program or a logic definition (e.g., in C, Python, Assembler, or VHDL). They would then upload that program into the remote device (e.g., microcontroller, DSP, FPGA). Being able to program the device could in fact be considered as the main functionality of conventional embedded systems remote laboratories. Nonetheless, in further sections, this and other important requirements will be described and analyzed in detail.

This work presents a novel remote laboratory architecture for embedded systems. It has three main goals. The first is to facilitate the implementation, deployment and upgrading of remote laboratories for different embedded platforms. The second is to assist remote laboratory developers in providing access to new embedded devices, through several common and reusable architectural components. The third is to ensure that students can acquire the skills in embedded systems development that they require.

In order to evaluate the proposed architecture, this work presents a new remote laboratory that implements it. It provides access to different ARM Cortex M microcontrollers. It has been designed to leverage all the advantages that the architecture offers. It is thus highly scalable, and it offers students not only conventional board programming features, but also debugging ones. It is described in detail. The requirements for the architecture are compared against that implementation, and user studies are conducted to evaluate its effectiveness.

The remainder of this paper organized in five sections. Section II describes the main requirements and characteristics of any remote lab for embedded systems, and analyzes five advanced existing remote laboratories. Section III describes the novel proposed architecture. Section IV describes the ARM microcontroller remote laboratory that implements the proposed architecture. Section V reports the results of the user study using the remote laboratory in the classroom. Section VI outlines conclusions and future work.

II. ANALYSIS OF DEPLOYED SOLUTIONS

Remote experimentation in embedded systems has been considered to be one of the major shifts in education [7]. Previous research works have determined the main requirements for interdisciplinary remote laboratories [10], [11]. Nonetheless, an additional, specific analysis should be done according to the particularities of embedded systems.

A. EDUCATIONAL REQUIREMENTS

The conventional development cycle of an embedded system design consists of writing the code, uploading it to a hardware platform, and testing and debugging it. Throughout this process the user requires an IDE (Integrated Development Environment) and a testing board.

Embedded systems require a codesign: developers must deal with both hardware and software components. Most embedded systems courses are based on development kits that integrate hardware peripherals on the system itself. Therefore, students can learn about both the hardware and software related tasks without needing to physically integrate the peripherals themselves. Remote laboratories can provide a similar experience, integrating a full set of peripherals [12]. Fig. 2 depicts the classical waterfall design system [13]. Firstly, the experiment requirements are analyzed. Secondly, the system is designed in terms of block diagrams or data flows. Then, students recursively repeat a cycle: coding the design, compiling or synthesizing it, and testing and debugging it. Eventually, this cycle leads to a validated system, and it is directly programmed into the final device [14].

In a "hands-on" laboratory the appropriate vendor-specific IDE provides all the required tools to properly progress through the design cycle. Similarly, remote laboratories should include the tools to offer students a similar experience. Nowadays, most embedded systems remote labs, however, are limited at that respect. They allow users to interact with the hardware through a live-stream and virtualized controls. However, they do not offer an IDE: they only offer an interface to upload an already-compiled file. This binary file needs to have been compiled or synthesized locally by the laboratory users. For this, they require their own IDE. This is

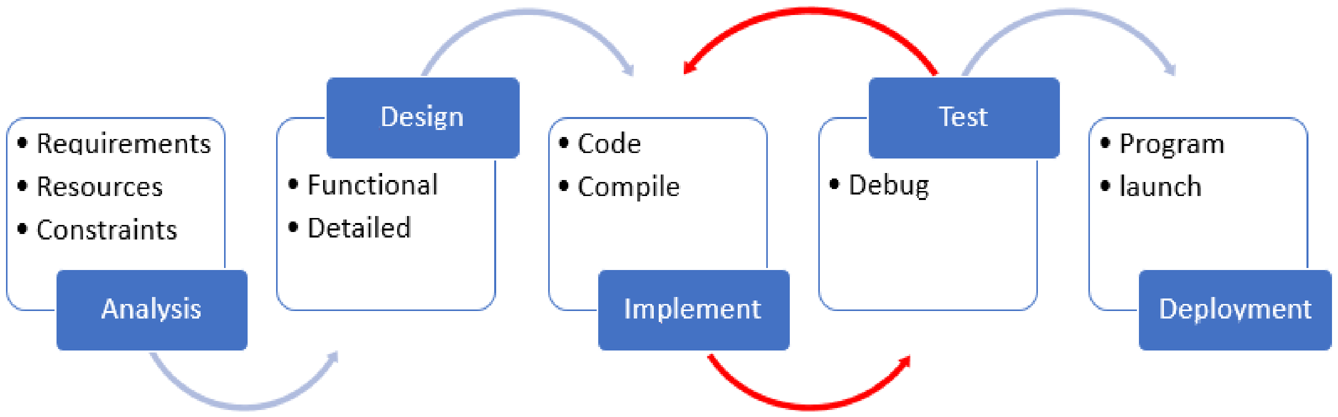


FIGURE 2. **Embedded systems education design cycle stages and the tasks that require most time.**

often very inconvenient for them, because such IDEs tend to be platform-dependent, not web-based, and often require administrator privileges to deploy.

The results of the annual embedded development survey conducted by UBM in 2017 [15] among more than two thousand embedded developers, remark that the most time-consuming stage in an embedded system development cycle consists in the debugging task. This is done during the testing stage, consuming a 29% of the full embedded design time.

The specific software such as IDEs and debugging tools are sometimes expensive and are often platform-specific. If alternatives are not part of the remote laboratory, users need a computer with the software installed. This is often significantly inconvenient, especially for students.

A successful remote laboratory for embedded systems should thus include the tools that are necessary to progress through all the stages in the embedded design development cycle. It should provide, by itself, all the required hardware and software tools.

B. TECHNICAL REQUIREMENTS

When including remote laboratory based instruction in regular courses, some technical requirements must be considered. Otherwise, the remote lab could be unsuccessful [10]:

1) Access to the laboratory should comply with the security policies of the institutional IT services without requiring specific configurations (e.g., open ports, firewalls, deployments of specific software). This promotes *deployability*.
2) *Universality.* This implies that the remote laboratory may be accessed from any device (e.g., laptop, tablet, smartphone) with any operating system in any web browser.
3) Usually, the remote laboratory is integrated in an educational platform. Examples are Moodle and Google classroom. *Integrability* is therefore a fundamental feature for a remote laboratory.

Therefore, if a company were to implement a remote laboratory to promote its new SoC (System on a Chip), this platform should be accessible from any device, without restrictions. It should support the full user experience. Otherwise, the remote lab may be unsuccessful and may contribute to a poor adoption of the proposed technology.

When a user is accessing, programming and testing a programmable device in a remote laboratory, other users cannot access it. They must wait in a queue until the remote experiment is released. *Scalability* for multiple users can be achieved by adding multiple experimentation instances.

The short life cycle of embedded device products requires professionals and educators to continuously adopt new device families and technologies. Therefore, the only way to ensure the sustainability of a remote laboratory is to facilitate this upgrading. *Adaptability* must be taken into account in the design stage of the remote laboratory to avoid having to redo the design of the remote laboratory from scratch. Therefore, the cost of annexing a new instance or adapting an existing one must be taken into account.

C. MAIN EMBEDDED SYSTEMS REMOTE LABS

In [9] more than 20 remote laboratories that are deployed to conduct experimentation with different embedded technologies are analyzed in depth, attending to the previous requirements.

Some remote laboratories for other fields are designed to allow concurrent access to the device. Nonetheless, for this type of remote laboratory, it is needed to physically program the device with the logic the user provides. Therefore, a single instance can only serve a single user. The only way to scale the laboratory is replicating the number of experimentation instances.

Most embedded systems laboratories [16]–[27] are designed under a centralized architecture. The laboratory server directly controls the experimentation platform through a standard protocol (e.g., USB, UART). This limits the number of possible experimentation instances and greatly hinders the scalability of the whole system. Other more advanced systems have been designed with a distributed architecture. In those, the main laboratory server can control several experiment servers. These experiment servers are connected to the local area network and handle the hardware platform over which the experimentation is conducted. This approach can provide access to big domains of students [28]–[35].

Throughout the remainder of this section, the five most representative embedded systems remote laboratories are going to be presented. This analysis describes the fulfillment of the embedded systems features that were previously observed for remote laboratories.

1) REMOTE LABORATORY FOR FPGA FROM THE UNIVERSITY OF ERLANGEN-NUREMBERG

This laboratory [28] was developed as the main tool to conduct an online course on FPGA design as part of the Virtual University of Bavaria (VHB). The design corresponds to a hierarchical architecture where one unique resources server is responsible of all administration tasks (user authentication, booking, analytics) but also of programming of hardware platform, interfacing the inputs and outputs, and webcam streaming. This system provides significant scalability: up to 10 instances can be connected to the server. The programming of the embedded device is performed through UrJTAG [29]. This is an open software package which enables working with JTAG-aware (IEEE 1149.1) hardware devices. Due to this, it is easily adaptable to other devices as long as they are JTAG programmable.

The user experience differs deeply from that conducted in a regular lab. Firstly, students need a computer with the appropriate FPGA design suite installed. Secondly, connection to the server requires an SSH connection, a web browser and a RTSP streaming client. Thirdly, HTTP sockets to non-standard ports are mandatory, making it difficult to access from other institutions [11].

2) ViciLab

This remote laboratory provides experimentation over digital circuits and FPGA devices. It was developed by the Reconfigurable Computation Research Team of the National University of Ireland, Galway [30]. It is nowadays hosted by a spinoff company called Vicilogic [31]. This system presents a very particular design. The main components of the laboratory are implemented within the experimentation platform. An IP core [31] programs the users' binary into the experimentation platform. It also supports the management of inputs and outputs. The system provides full scalability but adaptability is limited to FPGA-based systems that can implement the RTL core. The user interface is managed by a proprietary software that provides access to a system based on a Microsoft Windows platform. It provides a user experience with high educational possibilities. Nonetheless, it is very different to the experience in a conventional laboratory.

3) DSP-BASED REMOTE CONTROL LABORATORY FROM THE UNIVERSITY OF MARIBOR

This laboratory allows students to perform several experiments on automatics control [32]. The system follows a distributed architecture where a unique administration server manages different laboratory servers, guaranteeing scalability. All the laboratory is based on a specific DSP educational development board so no adaptability is possible. The user experience is the same as the one conducted on a conventional laboratory. However, students need a computer with MATLAB/Simulink and a browser that is compatible with the LabView plugin.

4) MICROLAB

This remote laboratory is hosted by the Faculty of Technical Education in the Suleyman Demirel University [33]. It was developed in 2004 and it has been continuously updated. It is used in the present day. The main feature of this system is its communication between multiple instances through the CAN protocol. This provides a distributed architecture that is managed by the experiment server. The programming of the embedded system is conducted through a custom bootloader developed for 8051 MCU. This limits adaptability. Access to the laboratory is conducted through the Microclient proprietary software [33]. It needs to establish several TCP connections through non-standard TCP ports. This discourages access from other institutions. Users require a Windows platform to run the Microclient software, though no other software is required to conduct the experimentation.

5) GOLDi

This laboratory [34], [35] allows students to control a group of didactical electromechanical models from a set of different embedded platforms. The architecture of the system is based on a unique laboratory bus. This bus is able to multiplex the signals of the selected embedded device and model. The addition of new components (embedded platforms and electromechanical models) can be done by designing a custom adaptor that hinders the replicability of the system by third parties. Although the system provides great adaptability over different devices, the only way to provide access to multiple users is duplicating the whole system. The cost of the system, due to the complex custom-designed components and the use of very expensive electromechanical models, hinders affordability.

D. ANALYSIS OF EMBEDDED SYSTEMS REMOTE LABS

Table 1 shows the degree of compliance for each feature by the systems highlighted in this chapter.

Beyond the aforementioned features, it is essential that a remote laboratory allows the completion of all the steps that are typically conducted in a conventional laboratory while experimenting with embedded systems. This process consists basically in a cyclic progression through four main steps: coding, preparing a binary file, testing and observing how the system runs the program with the real inputs and outputs injected by the user or sensors.

Table 2 indicates which experimentation steps are supported directly in each analyzed laboratory. A significant limitation of the analyzed systems [16]–[35] is that none of

TABLE 1. Comparison between remote laboratories designed under a distributed architecture.

System	Scalability	Adaptability	Deployability	Universality	Integrability
FPGA FAU (1)	✓	✗	~	✓	~
ViciLab (2)	✓	✗	✗	~	✗
DSP UM (3)	✓	~	✓	~	✗
MicroLab (4)	✓	~	✓	✓	✗
GOLDi (5)	✗	✓	✗	✓	✗

✓ Completely achieved ~ Partially achieved ✗ Not achieved

TABLE 2. Experimentation steps supported by the analyzed remote laboratories

System	Code	Binary generation	Debug	Run
FPGA FAU (1)	~	~	✗	✓
ViciLab (2)	✓	~	✗	✓
DSP UM (3)	~	~	✗	✓
MicroLab (4)	~	~	✗	✓
GOLDi (5)	✓	~	✗	✓

✓ Inside the laboratory ~ Using external tools ✗ Not supported

them provides real debugging capabilities during the experimentation stages. This step is the most time-consuming one according to [15]. The novel proposed architecture solves this problem and includes web-based debugging support.

III. NEW ARCHITECTURE TO DEPLOY REMOTE LABORATORIES FOR EXPERIMENTATION OVER EMBEDDED SYSTEMS

The main goal of most remote laboratories is to be able to provide an experimentation process that is similar to that performed in a hands-on laboratory. Thus, the proposed architecture has been built around a set of components that are oriented to fulfill the described embedded systems experimentation steps. The main subsystems proposed by the architecture are the following (Fig. 3):

1) **Experiment Platform**. Real experimentation requires that the program users provide is really programmed into the target embedded device. The Experiment Platform holds the embedded device and provides similar resources (e.g., inputs, outputs, communication buses) to those that the development systems used in hands-on laboratories provide.
2) **Interface Server**. In a real laboratory, students can interact directly with the peripherals. However, using a remote laboratory, a connected system is required to transform the client commands into physical signals. These signals monitor and process the outputs generated by the embedded device.
3) **Experiment Server**. This subsystem manages the interaction with the experiment platform. It manages all the tasks that students perform over it through any IDE. This includes not only programming but also erasing the devices' memory. It also includes executing the different debugging commands, which have to be available from the Experiment Server.
4) **Binary Generator**. In order to free the laboratory from any specific tool, this subsystem receives the source code, generates the appropriate binary file, and returns it. The binary file can then be programmed into the Experiment Platform through the Experiment Server. This system can be implemented in the cloud through virtualized machines or deployed locally.
5) **Client**. This subsystem provides the student with all the graphical user interfaces required to perform the experimentation process. In order to fulfill the previously described requirements of accessibility, it has to consist of a web interface accessible from multiple platforms. That includes supporting access from mobile devices, which is ever more demanded [36].
6) **Administration Server**. This component is in charge of the administration tasks required to properly manage a session of a remote laboratory (authentication, booking, queuing, load balance between different instances, user tracking). Every analyzed system includes

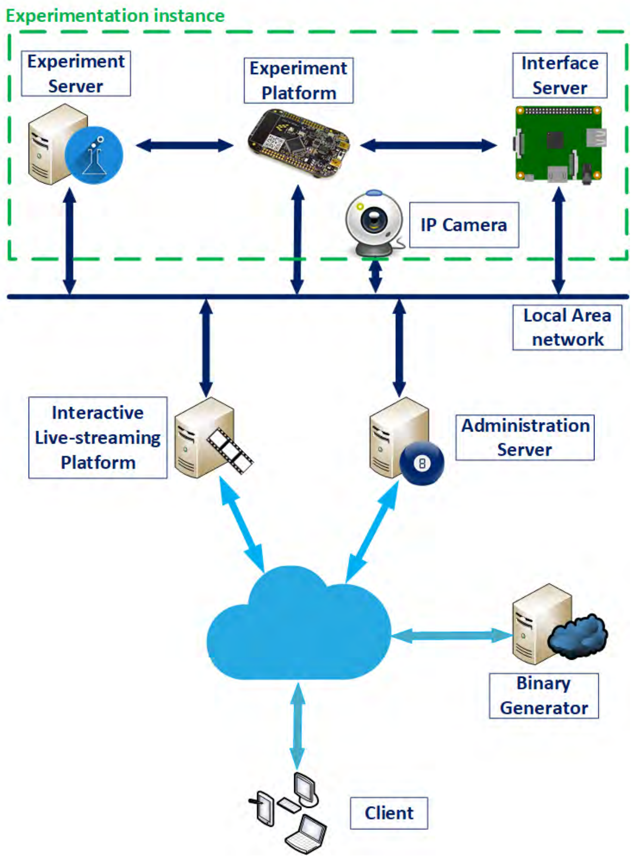


FIGURE 3. Diagram of the proposed architecture for the development of embedded systems remote laboratories.

a custom experiment server. However, by integrating it in a Remote Laboratory Management System (RLMS) [37], the architecture also makes it easier to cover the requirements and guarantee sustainability.

7) **Interactive live-streaming platform:** A key component of most remote laboratories is the live-stream through which users can monitor the device and see the results of their actions. In an interactive remote lab, this live-stream needs to provide a very low capture-render delay, so that interaction is not hindered. This platform abstracts out specific camera idiosyncrasies and reliably provides these functionalities.

Just as important as the components, is their interconnection. The dependence between the components that support the experiment (IP Camera, Experiment Server, Interface Server, Experiment Platform) is critical. These components comprise the Experimentation Instance. They are independent from the other components. This makes it easier to replicate, thus providing higher scalability. Interactive experimentation over embedded systems requires that the particular user's binary be programmed into the board. Therefore, several experimentation instances are required for multiple simultaneous users. The RLMS can balance the load among different instances. Through this scheme, it can provide simultaneous access to as many users as instances are available.

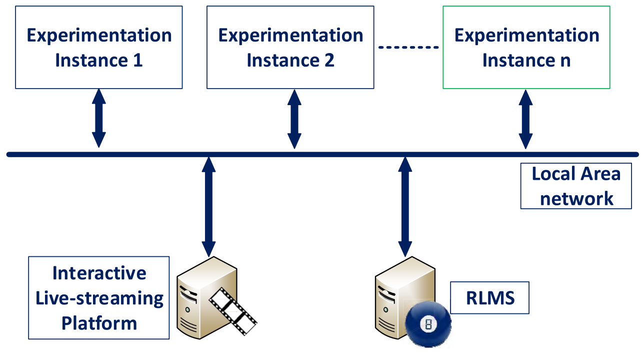

Experimentation Instance 1
Experimentation Instance 2
Experimentation Instance n
Local Area network
Interactive Live-streaming Platform
RLMS

FIGURE 4. Embedded laboratory with multiple instances under the proposed architecture.

Nonetheless, there can also be a queue. Users' time in it will be lowered with a higher number of instances (Fig. 4). The Experimentation Instance can only be accessible from the RLMS. In the same line, the independence of the Experimentation Instance provides adaptability. It becomes easier to exchange the experimentation platform, by simply updating the Experiment Server and adapting the Interface Server to the new embedded system. Decoupling the Binary Generator from the Experiment Server makes it easier to replace. Moreover, it allows highly time-consuming tasks, such as compilation or synthesis, before accessing the laboratory. This way, a user can experiment while other user is synthesizing a source code, minimizing inactivity time. The Binary Generator component consists of a system capable of queuing compile or synthetization requests and generating the resulting binaries. Generating certain kinds of binaries takes a significant amount of computational resources. The Binary Generator is designed as a Cloud service, so that it can scale horizontally when new experimentation instances need to be served.

The Client component provides the web-based Graphical User Interface (GUI). It provides the student with a full IDE to experiment with, similarly to how it would be done in a hands-on laboratory. In a conventional laboratory, this would normally be a desktop-based IDE that is physically connected to the embedded system and that offers debugging features. Therefore, in the case of a remote lab, the Client should provide similar features. In this case, the Client provides a web-based source code editor, and different view perspectives to either run the experiment or debug it, always remotely.

IV. THE WEBLAB-ARM LABORATORY

This system has been developed as a proof of concept implementing the proposed architecture. Weblab-ARM provides experimentation with NXP Kinetis KL25-48 MHz, Ultra-Low-Power Microcontrollers (MCUs) based on ARM® Cortex®-M0+ Core. This choice has been made due to its extensive use in different universities, and the fact that the board is recommended by the ARM University Program for conducting its basic embedded systems course. The RLMS used in this work, and the library developed for facilitating the development of new labs under this architecture are released as open source.[1]

In the next subsections, the implementation of each component is presented according to the proposed architecture.

A. EXPERIMENT PLATFORM

A new trainer board has been designed to allow performing most typical embedded systems exercises. This board is connected to the same FRDM KL25Z [38] development system that students are using in the hands-on laboratory:

- Digital Input/Outputs (GPIO). The student can experiment with GPIOs managing the LED diodes included in the platform. The web client features, also, the virtual representation of several switches and buttons that are converted into physical signals through the Interface Server.
- Motor control. 2 servo motors, 1 stepper motor and a DC motor have been included in the experiment platform. They allow students to experiment with different motor control approaches.
- Communication Protocols. Several peripherals that are made available through the experiment platform (e.g., RTC, Port Expander, EEPROM Memory, Serial Terminal) can be managed through the I2C, SPI or UART protocols.

[1]The RLMS is available at https://github.com/weblabdeusto/weblabdeusto and the library at https://github.com/weblabdeusto/weblablib

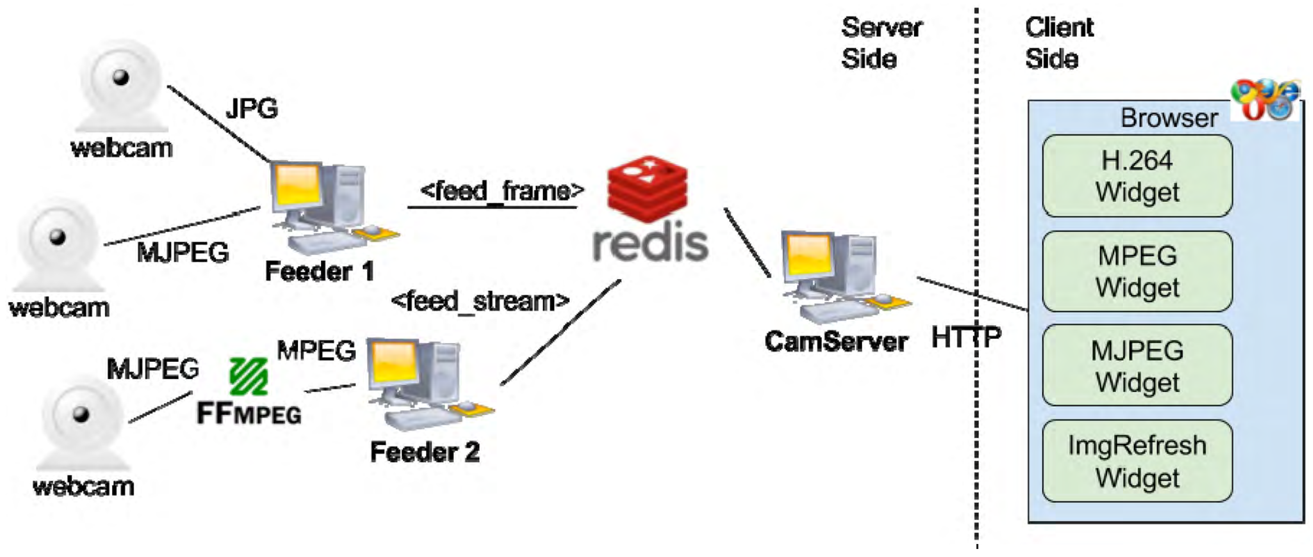


FIGURE 5. Architectural overview of the Interactive Live-Streaming Platform. From [39].

- Alphanumeric LCD. A 2 × 16 characters alphanumeric LCD is integrated in the experiment platform and visible through the displayed live-stream.

B. INTERACTIVE LIVE-STREAMING PLATFORM

The main characteristic of remote laboratories is that they allow users to access real, remote equipment, attempting to resemble hands-on laboratories [39]. In a hands-on laboratory, users view the equipment with their own eyes, and rely on their sight to interact with it. To achieve the same purpose, most remote laboratories rely on a live-stream from a webcam (e.g. [40]–[42]). Through that live-stream, remote users can view the equipment, and use what they see for their interaction with the equipment.

A proper design and implementation of this feature is critical for a remote laboratory to provide a satisfying user experience. The live-stream is the main link between the laboratory user and the remote equipment. It is important to remark that the live-streaming system of a remote lab needs to be *interactive* [43]–[45]. That is, it needs to guarantee a particularly low delay between the moment a frame is captured and the moment it is rendered (the capture-render delay). This is not the case for standard live-streaming systems. Popular live-streaming platforms such as Youtube Live[2] or TwitchTV,[3] in fact, often allow for a several seconds delay [46]. For standard live-streaming applications (e.g., live sports events streaming) this is an adequate choice because that delay makes the platform more scalable, allowing it to rely on buffering and high-compression transcoding techniques.

The architecture that is proposed in this work relies on an Interactive Live-Streaming Platform that has been purposefully designed to meet the requirements of remote laboratories [44]. The main requirement is, as previously described, a minimal capture-render delay. Other important requirements, however, are being fully web-based, supporting many source webcams, and being scalable to many clients. Being web-based is important because most educational remote laboratories today are (and certainly are expected) to be accessible through a simple web-browser. In the past, laboratories often relied on desktop-based software or in vendor-specific custom browser plugins (such as Adobe Flash[4] or Java Applets) [11]. This hinders universality, which is ever more important, especially with the growing usage of mobile devices in education [36], [47], [48]. Supporting many webcams and being scalable to many clients is also important, because a key feature of the remote lab architecture that is proposed in this work is, precisely, to be highly scalable.

Fig. 5 shows an architectural overview of the interactive live-streaming platform and its main components. The webcams (depicted to the left) are normally IP-based. They tend to contain a very simple web server to provide streams (often in M-JPEG format), but their hardware and software is limited, they are not meant to support a significant number of issues, and they are prone to security issues [49], [50]. Thus, they are secured within the LAN and feed their image into a *Feeder* component. The Feeder component takes the stream as input (supporting several formats) and forwards it into a Redis[5] instance, transcoding it if necessary using FFmpeg.[6] A *CamServer* component reads the stream from Redis, and serves it to the end-user, in various formats which are web-based and appropriate for remote laboratories [39]. This architecture is highly scalable. There can be any number of webcams, any number of Feeders and any number of

[2]http://www.youtube.com
[3]http://www.twitch.tv
[4]http://www.adobe.com/products/flashplayer.html
[5]https://redis.io
[6]https://ffmpeg.org

CamServers. Redis, which is a general-purpose in-memory data engine, designed precisely for scalability, is at the core.

By integrating the interactive live-streaming platform into the remote laboratory architecture that is proposed in this work, the previously defined constraints are appropriately satisfied. Live-streams are supported and can scale for any number of experiment instances and of users. The laboratories can still be fully web-based and thus easy to deploy and mobile-friendly. And the capture-render delay is minimized to provide a satisfactory user experience.

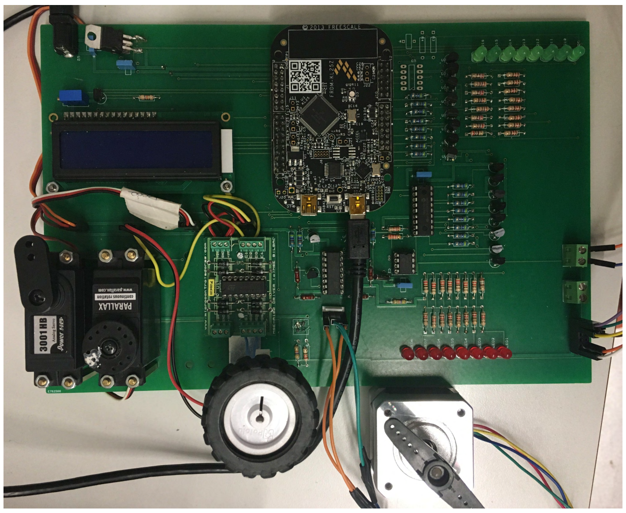


FIGURE 6. Hardware platform for the ARM remote lab.

C. INTERFACE SERVER

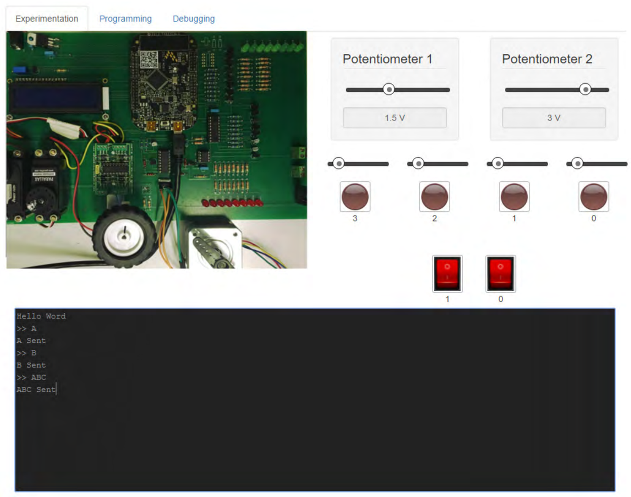
The output peripherals can be monitored through the provided live-stream (see Fig. 6). Controlling the input peripherals is always a key factor in the design of an embedded systems remote laboratory.

Users must be able to act on the inputs just as they would in a conventional laboratory. Because the interaction with the student is performed through the web client, this must be done through virtualized controls.

Their appearance and UX is designed to be similar to the physical ones they are intended to replace. Their effect is mirrored physically in the hardware, thus being highly realistic.

The Interface Server that has been developed for the proposed remote laboratory provides several input peripherals through which students can interact with the MCU. These include 2 potentiometers, 4 buttons and 2 switches (see Fig. 7).

The virtual representation in the client of these peripherals is intended to provide a physical behavior similar to that of fully real peripherals. The implementation of the system is based on a REST API. It provides several HTTP methods to be consumed by the client. These methods trigger the corresponding physical signals. The API is directly deployed in an embedded platform (Raspberry Pi 2) wired to the Experiment Platform through the SPI and UART interfaces and 6 GPIO.



```
Hello Word
>> A
A Sent
>> B
B Sent
>> ABC
ABC Sent
```

FIGURE 7. Web Client of the "Experimentation perspective".

Client-side, potentiometer management is assisted by two slide widgets that allow the student to select the voltage that the MCU receives through two ADC channels.

The Interface Server includes a SPI interfaced Digital-Analog converter (MCP4902) that adjusts the signal connected to each ADC channel every time the slide is changed in the client.

Buttons are controlled from the client through a graphical button widget and a slide widget that sets the pulse duration.

Switch widgets in the client allow students to set and clear their respective GPIOs in the MCU.

Finally, the client also includes a serial terminal widget that allows sending and receiving ASCII characters to and from the MCU. It is backed by the Interface Server through the UART included in the Raspberry Pi. The main HTTP methods for the REST service, implemented in Python with the Flask framework, are shown in Table 3.

TABLE 3. Main HTTP commands included supported by the interface server.

Command	Method	Function
potentiometers	GET	Returns the current value of potentiometers.
potentiometers/post_id/value	POST	Sets a new value in a potentiometers.
buttons/but_id_time	POST	Generates a pulse in a buton.
switches/switch_id/value	POST	Sets or clear a switch.
serial	GET	Retuns received buffer from UART.
serial	POST	Sends a string to UART.
actuators	GET	Returns the current values of all the input peripherals.

D. EXPERIMENT SERVER

The implementation of the previous components follows the traditional mandates of embedded remote laboratory development. The experiment server is the key component that provides new demanded capabilities for remote experimentation over embedded systems. The goals are:

TABLE 4. Main HTTP commands included supported by the ExperimentServer.

Command	Method	Function	GDB Command
start	POST	Connects the hw. target, loads the file and receives the binary file to be loaded	target remote/monitor reset/file/load
end	POST	Just disconnects the in the GDB session	quit
stepinto	POST	Executes an instruction	step
stepover	POST	Executes an instruction/function	next
run	POST	Executes the program until Breakpoint	continue
interrupt	POST	Interrupts the execution of the program	ctrl+c
breakpoint	POST	set/delete a breakpoint	b | clear
status	GET	Returns the state of the GDB session	frame

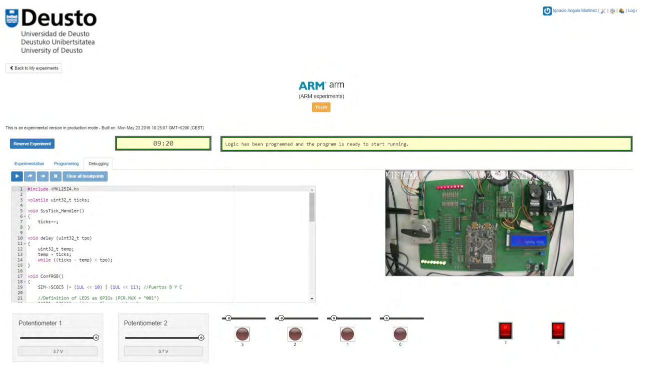
- To support every stage of the embedded systems design cycle. The server is not only responsible for receiving the user's binary file and programming it into the MCU, but also for providing all the required functions that are necessary for a complete debugging experience.
- To facilitate portability to new platforms or embedded devices. The laboratory supports microcontrollers based on ARM Cortex M, which are highly popular. Just performing minor changes, the remote laboratory could support up to 3887 different devices (This is the MDK5 device list number on March 2017).

The implementation of this component requires a Linux machine running OpenOCD (Open On-Chip Debugger) [51] and the GDB (GNU Project Debugger) [52]. A Python REST API provides HTTP methods that handle debugging actions that users can request on the experiment platform. The server keeps a GDB session opened and automatically injects GDB commands as needed. A list of the main GDB commands that are supported through the implemented REST API are shown in Table 4.

E. CLIENT

The ARM laboratory's client is a key component for the proposed embedded remote laboratory architecture. The GUI that it provides is fully web-based. Nonetheless, it has been designed to support the basic features of the traditional desktop-based software development kits for embedded systems. Particularly, the web-based interface has been designed to support editing and debugging code on-line. The Weblab-ARM laboratory features three different perspectives:

- Experimentation: To interact with the hardware system once it has been programmed. It provides control over all the virtual interaction devices, such as the switches or the potentiometers. It also provides a live-stream through a webcam to monitor the device's behavior (Fig. 7).
- Programming: It provides a syntax-highlighting editor to code for the ARM device. Currently it supports "C" and "C++" programming.
- Debugging: This view shows, simultaneously, the code that is programmed into the device, the device itself through a live-stream, and the interaction components. With those tools, users can debug their program, relying on standard debugging features such as breakpoints and execution controls (Fig. 8).



This is an experimental version in production mode - Built on: Mon May 23 2016 18:25:07 GMT+0200 (CEST)

Logic has been programmed and the program is ready to start running.

```
#include <MKL25Z4.h>

volatile uint32_t ticks;

void SysTick_Handler()
{
    ticks++;
}

void delay (uint32_t tpo)
{
    uint32_t temp;
    temp = ticks;
    while ((ticks - temp) < tpo);
}

void ConfRGB()
{
    SIM->SCGC5 |= (1UL << 10) | (1UL << 11); //Puertos B Y C

    //Definition of LEDS as GPIOs (PCR.MUX = "001")
```

FIGURE 8. Screenshot of the "Debugging" perspective.

As described above, one of the remarkable aspects of the proposed GUI is that users view and interact with the laboratory hardware through a live-stream. To do so, the architecture relies on an interactive live-streaming system [42]. Unlike standard live-streaming systems, interactive ones focus on minimizing the capture-render delay: the time that elapses between the moment a frame is taken by the camera, and the moment it is rendered in the screen. This can be achieved by relying on certain techniques such as image-refreshing, M-JPEG, or even small video buffers H.264 [43], and avoiding others which require large buffers or heavy transcoding. Maintaining such a low, interactive-level delay is of utmost importance for this kind of system, because users expect to see an immediate response in the remote equipment whenever they hit a button or interact with other components.

Binary Generator: As discussed, the cloud-based design of the Binary Generator that generates binary files from the source code is important for scalability. It is also useful to facilitate the adaptation of the implementation to new or upgraded hardware boards.

The binary generator in Weblab-ARM is implemented as a cloud service based on Azure services. One or several virtual machines (VM) run the ARM GCC compiler. A message brokering service based on the Azure Service Bus (ASB) guarantees the possibility of load balancing under intensive demand. Virtual machines continuously monitor the queue, waiting for unassigned tasks. When a new task is available, the Cloud Compiler Tool receives the source code and updates the broker's task object, publishing the corresponding binary file and setting the task to "finished" (Fig. 9). The compiling process for the ARM Cortex M microcontrollers are very fast tasks. A single VM can handle many compilations. However, the system is designed to be easily portable to other hardware platforms where binary file generation is more complex and takes higher amounts of time and resources, such as FPGAs.

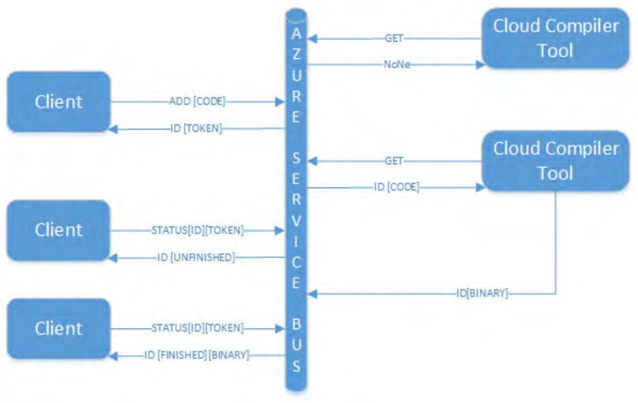


FIGURE 9. Different stages in the cloud compiling process.

F. REMOTE LABORATORY MANAGEMENT SYSTEM

Weblab-ARM is hosted under the WebLab-Deusto RLMS. Its main goal is to provide basic common features such as user management, learning analytics, LMS integration, or laboratory federation among different institutions.

V. EVALUATION

A. TECHNICAL EVALUATION

The main objective of the proposed architecture is to satisfy two key requirements identified during the analysis of remote experimentation for embedded systems: scalability and adaptability. Most efforts are therefore oriented towards facilitating the addition of multiple instances and the portability to new embedded devices. Section IV: a new remote laboratory for ARM-based boards, describes an implementation of the architecture. This implementation has been developed to validate the architecture attending to the previously listed requirements.

To test the adaptability capabilities that it provides, we have analyzed the effort that is required to adapt the developed remote laboratory to other embedded development kits. The "ARM University Program" recommends the following development boards for teaching Embedded Systems/MCUs from different vendors:

1. Freescale Freedom FRDM-KL25Z
2. ST STM32F4 Discovery Board
3. NXP LPC1115 LPCXPRESSO Board
4. NXP LPC4088 Experiment Base Board
5. Cypress PSoC4 Pioneer Board
6. ST Nucleo F401RE Board.

Furthermore, beyond the ARM MCU architecture, other two development boards are considered:

7. Arduino Uno
8. PIC18 Explorer Board.

TABLE 5. Level of redesign required for different embedded systems development boards.

Development board	RLMS	Binary Generator	Client	Experiment Server	Experimentation platform	Interface Server
Freescale Freedom FRDM-KL25Z	0	0	0	0	0	0
ST STM32F4 Discovery Board	0	1	0	1	2	0
NXP LPC1115 LPCXPRESSO Board	0	1	0	1	2	0
NXP LPC4088 Experiment Base Board	0	1	0	1	1	0
Cypress PSoC4 Pioneer Board	0	1	0	1	1	0
ST Nucleo F401RE Board	0	1	0	1	1	0
Arduino Uno	0	3	0	4	0	0
PIC18 Explorer Board	0	3	0	4	2	0

TABLE 6. Efforts in the Weblab ARM remote laboratory development.

Task	Nº Hours
Weblab-Deusto RLMS deployment and Configuration	96
Interactive Live-Streamming Platform deployment and Configuration	72
Binary generator development	120
Experiment server development	160
Experimentation platform implementation	60
Interface server development	80
Remote laboratory integration and testing	160
Total	748

Table 5 shows a rating from 0 (no changes required) to 5 (complete re-design required). It describes the approximate effort that it would take to modify the described Weblab-ARM remote laboratory to support the listed devices.

As the table indicates, several components of the remote laboratory are completely independent from the embedded device (the RLMS, the Client and the Interface Server components). Only three components must be re-developed to adapt the remote laboratory to a different development device:

- **Binary generator:** All the boards based on the ARM Cortex M architecture are compatible with the Keil development tools. Only minor changes (mainly, reconfiguration of the device package) are required for systems 1 to 6 (see Table 7). Adaptation to different MCU architectures (7 and 8 in Table 7) requires deeper

TABLE 7. Required updating efforts for the considered embedded platforms.

	Freescale Freedom FRDM-KL25Z	ST STM32F4 Discovery Board	NXP LPC1115 LPCXPRESSO Board	NXP LPC4088 Experiment Base Board	Cypress PSoC4 Pioneer Board	ST Nucleo F401RE Board	Arduino Uno	PIC18 Explorer Board
Weblab-Deusto RLMS deployment and Configuration	96	0	0	0	0	0	0	0
interactive Live-Streamming Platform deployment and Configuration	72	0	0	0	0	0	0	0
Binary generator development	120	24	24	24	24	24	72	72
Experiment server development	160	32	32	32	32	32	128	128
Experimentation platform implementation	60	24	24	12	12	12	0	48
Interface server development	80	0	0	0	0	0	0	0
Remote laboratory integration and testing	160	42	42	36	72	36	108	134
Total	748	122	122	104	140	104	308	382
Adaptation efforts required		16%	16%	14%	19%	14%	41%	51%

changes to this component. The compiler would need to be switched for a different one, and integrated with the developed web service.

- **Experiment Server:** The design of this component in the Weblab-ARM remote laboratory is based on the OpenOCD on-chip debugger. The boards that are compatible with this system (1-6) would only require to change the OpenOCD configuration file. However, in order to adapt the system to a new architecture (7 and 8), this component would need to be redesigned. The handlers for each command in Table 4 would need to be re-developed.
- **Experimentation platform:** The Weblab-ARM laboratory is based on the FRDM KL25Z development board. It provides an Arduino UNO compatible interface. Therefore, all the boards that offer the same interface (1, 4, 5, 6 and 7) can be plugged directly to the experimentation board. For those with a different connector, an adaptation board must be implemented.

Table 6 shows the total resulting staff hours required for the Weblab ARM remote laboratory development. Considering these times, and the required updating effort for every component (Table 5), Table 7 estimates the time it would take to adapt the system for new embedded platforms.

As described in previous sections, the architecture is designed so that adding new experiment instances requires little effort.

After setting up the hardware for the new instance and cloning the systems, only their registered IP addresses would need to be updated. Then, the new instance would need to be registered in the WebLab-Deusto RLMS configuration. Currently, registering new instances is relatively simple. The Experiment Servers for these instances can be deployed anywhere within the local network, and then registered into the system. Once properly set up, load balancing is automatic. Users are automatically and transparently redirected to a free instance when one is available, or asked to wait in a queue if all instances are busy. Therefore, the sum of the proposed architecture and of the native load-balancing capabilities of the RLMS lead to a very adaptable and highly scalable framework.

B. DIDACTICAL EVALUATION

The ARM-based remote laboratory was used in the 2016-2017 course for the Degree in Industrial Electronics at the University of Deusto. The Microprocessors subject is part of the first semester of the third school year. Its main goal is to teach the principles and applications of microprocessors and microcontrollers. The teaching and learning strategy is mostly practical. Students can design and test their programs using both a hands-on laboratory and a remote laboratory. The latter relies on the WebLab-Deusto RLMS. Therefore, every practical session can be done by the student using either the classical or the remote laboratory.

During the 2016-2017 course, 27 students were enrolled in the subject and 25 followed it. They accessed the remote lab 1581 times. The average was 63 accesses per student, the maximum number of accesses was 153, and the minimum was 16.

Every week the students have to write and test a program that is associated to a challenge. The students can use either of the laboratories; it is up to them based on their preferences or on their location during the session. Using the remote laboratory was therefore not mandatory, it was offered as a complementary resource to help them with programming tasks. To access the remote lab, students access the weblab.deusto.es web platform, introduce their user and password, and access the ARM remote lab. After accessing it they upload the produced HEX file to see how their program behaves when running under real hardware. They have 5 minutes per session to complete the testing process. After 5 minutes, the session is automatically closed. When students do not finish in time, they can reserve a new session and repeat the process. The maximum amount of time per session can be easily changed by the teacher, but if the time is too high it can lead to a long access queue. Therefore, the teacher needs to choose the appropriate time, so that students have enough time but at the same time every student gets the chance to access fast enough.

When the subject was over, the students were asked to fill a survey about their opinion on the remote lab and its usage. The original survey was created by [53] and [54] and the final version includes the result of the eMerge project [55]. It has 11 items with a 5-point Likert scale (1: completely disagree, 2: disagree, 3: neutral, 4: agree, 5: completely agree). 25 of the students filled the survey, and 2 did not. The results are shown in the Table 8.

TABLE 8. Survey's results in microprocessors subject in 2016-2017.

	Average	Deviation
1. The WebLab has helped me in the subject: concepts, practice, the project, etc.	4.72	0.54
2. When I am using the remote lab I feel that it is real and that it is not a simulation.	4.32	1.22
3. It is a good idea to extend the use of the remote lab to other students.	4.76	0.44
4. The WebLab is easy to use	4.64	0.91
5. The quality of the webcam-provided visual feedback is good.	4.30	0.9
6. The time assigned to each session is	3.44	1.47
appropriate.		
7. Even being far from the remote lab I felt that I had control over it.	4.08	1.38
8. I would like to use the WebLab in other subjects.	3.60	1.61
9. In general, I am satisfied with the remote lab.	4.48	0.96
11. The WebLab-Deusto RLMS is good enough and it has helped me during the session.	4.52	0.96

The experience with students shows that the designed remote laboratory is well-regarded as a tool for programming and debugging code in ARM microcontrollers. Most of the values are higher than 4, and in general the students are satisfied with the remote lab (4.48) because it is easy to use (4.64), they feel that there is a real hardware that they control (4.08 and 4.32) and the RLMS that manages the remote lab is good (4.52).

VI. CONCLUSION AND FUTURE WORK

The analysis of the state of the art shows that the field of remote laboratory development for embedded systems is a very active field. However, it also reveals some existing problems and challenges. This work extracts a set of technical and educative requirements, and proposes a novel remote laboratory architecture for embedded systems.

The proposed architecture is designed to provide certain key capabilities. Firstly, it is designed to provide scalable remote laboratories, that support multiple instances of the same equipment to support multiple users. Secondly, it is designed to avoid deployment and networking limitations. It relies only on HTTP and standard ports to avoid firewall traversal limitations, which obstruct the accessibility of other architectures. Thus, it can be deployed on any web platform, LMS environment, or institution. Thirdly, it is designed to be adaptable. The laboratories that implement the architecture require very few modifications to support new embedded system families, or to be upgraded to their latest versions. This feature is of particular interest for industry, because it makes it possible to swiftly deploy various devices for them to be tested remotely by prospective users. It should be possible, for instance, to support various FPGA devices and various microcontroller platforms.

Beyond the aforementioned characteristics, the proposed architecture includes a particularly important one which is unique: full access through the Web. Users do not need to have any programming or testing environment on their machines. An Internet connection and a web browser are enough. A complete embedded system programming and testing experience is possible. Thus, the laboratories also become available for tablets and smartphones. This was not possible through most previous architectures, and it used to be a very significant restriction, since the popularity and ubiquity of these devices is certainly growing. This is also interesting for industry.

Additionally, this work complements the proposed architecture with a new interactive live-streaming platform for remote laboratories. This augments the effectiveness and sustainability of the system in terms of image and video quality, bandwidth and user immersion.

Finally, the architecture and its implementations have been evaluated from a mainly technical perspective, but also considering some didactical issues.

This work has presented, in detail, a remote laboratory that implements the architecture, for an ARM-controlled robot. Additionally, it has presented another laboratory for a FPGA device. Thus, the scalability and adaptability of the architecture have been validated. The ARM deployment has been used in a classroom environment, and the experience has been positive, both in objective terms (access number and lack of significant issues) and subjective ones (satisfactory for students and teacher).

The developed laboratory is currently being used by real students, and in the future we aim to conduct further pedagogically-oriented research using the resulting data. Additionally, in the future, the next steps will be oriented towards deploying implementations of the proposed architecture in other educative and research centers, and also towards being used and proving its value for the embedded systems industry.

REFERENCES

[1] C. Bohus, B. Aktan, M. H. Shor, and L. A. Crowl, "Running control engineering experiments over the Internet," Dept. Comput. Sci., Oregon State Univ., Corvallis, OR, USA, Tech. Rep. 95-60-07, Aug. 1995.

[2] R. Bose, "Virtual labs project: A paradigm shift in Internet-based remote experimentation," *IEEE Access*, vol. 1, pp. 718–725, 2013, doi: 10.1109/ACCESS.2013.2286202.

[3] J. R. Brinson, ''Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research,'' *Comput. Edu.*, vol. 87, pp. 218–237, Sep. 2015, doi: 10.1016/j.compedu.2015.07.003.

[4] L. D. Feisel, G. D. Peterson, O. Arnas, L. Carter, A. Rosa, and W. Worek, ''Learning objectives for engineering education laboratories,'' in *Proc. 32nd Annu. Frontiers Edu. (FIE)*, vol. 2. 2002, p. F1D, doi: 10.1109/FIE.2002.1158127.

[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, 2013, doi: 10.1126/science.1230579.

[6] J. Sáenz, J. Chacón, L. Da La Torre, A. Visioli, and S. Dormido, ''Open and low-cost virtual and remote labs on control engineering,'' *IEEE Access*, vol. 3, no. , pp. 805–814, 2015, doi: 10.1109/ACCESS.2015.2442613.

[7] J. E. Froyd, P. C. Wankat, and K. A. Smith, ''Five major shifts in 100 years of engineering education,'' in *Proc. IEEE*, vol. 100, Special Centennial Issue, pp. 1344–1360, May 2012, doi: 10.1109/JPROC.2012.2190167.

[8] L. Gomes and S. Bogosyan, ''Current trends in remote laboratories,'' *IEEE Trans. Ind. Electron.*, vol. 56, no. 12, pp. 4744–4756, Dec. 2009, doi: 10.1109/TIE.2009.2033293.

[9] I. Angulo, ''Arquitectura abierta para el despliegue de laboratorios remotos sobre tecnologías de desarrollo de sistemas embebidos,'' Ph.D. dissertation, área de Electricidad, Electrónica, Automática y Comunicaciones, Univ. Deusto, Bilbao, Spain, 2015.

[10] 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, Oct. 2009, doi: 10.1109/TLT.2009.33.

[11] J. García-Zubía, P. Orduña, D. López-de-Ipiña, and G. R. Alves, ''Addressing software impact in the design of remote laboratories,'' *IEEE Trans. Ind. Electron.*, vol. 56, no. 12, pp. 4757–4767, Dec. 2009, doi: 10.1109/TIE.2009.2026368.

[12] C. Ebert and C. Jones, ''Embedded software: Facts, figures, and future,'' *Computer*, vol. 42, no. 4, pp. 42–52, Apr. 2009, doi: 10.1109/MC.2009.118.

[13] T. Myers, R. G. Dromey, and P. Fritzson, ''Comodeling: From requirements to an integrated software/hardware model,'' *Computer*, vol. 44, no. 4, pp. 62–70, Apr. 2011, doi: 10.1109/MC.2010.270.

[14] H. Mitsui, H. Kambe, and H. Koizumi, ''Use of student experiments for teaching embedded software development including HW/SW codesign,'' *IEEE Trans. Educ.*, vol. 52, no. 3, pp. 436–443, Aug. 2009, doi: 10.1109/TE.2008.930096.

[15] *Embedded Market Study, Produced by Embedded Systems Design magazine and the Embedded Systems Conference*, UBM Tech, Denver, CO, USA, 2014.

[16] F. Y. Limpraptono, H. Sudibyo, A. A. P. Ratna, and A. S. Arifin, ''The design of embedded Web server for remote laboratories microcontroller system experiment,'' in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2011, pp. 1198–1202, doi: 10.1109/TENCON.2011.6129302.

[17] T. Hoang and H. N. Quang, ''A low-cost remote laboratory of field programmable gate arrays,'' in *Proc. 12th Int. Conf. Remote Eng. Virtual Instrum.*, Feb. 2015, pp. 172–176, doi: 10.1109/REV.2015.7087286.

[18] H. Zhao and T.-J. Xiao, ''An innovative remote experiment system for FPGA-based curriculum,'' in *Proc. IEEE Int. Symp. IT Med. Edu.*, Dec. 2008, pp. 870–875, doi: 10.1109/ITME.2008.4743991.

[19] J. Butime, R. Besiga, A. Bwonyo, V. Nakanwagi, T. Togboa, and A. Katumba, ''Design of online digital electronics laboratories based on the NI ELVIS II platform,'' in *Proc. 9th Int. Conf. REV Remote Eng. Virtual Instrum.*, Jul. 2012, pp. 1–3, doi: 10.1109/REV.2012.6293098.

[20] S. Karthik, P. Shreya, P. Srihari, and N. M. Viswanath, ''Remote field-programmable gate array (FPGA) lab,'' *IJRET, Int. J. Res. Eng. Technol.*, vol. 3, no. 4, pp. 842–845, 2014, doi: 10.15623/ijret.2014.0304149.

[21] M. Gilibert, J. Picazo, M. E. Auer, A. Pester, J. A. Cusidó, and J. A. Ortega, ''80C537 microcontroller remote lab for E-learning teaching,'' *iJOE, Int. J. Online Eng.*, vol. 2, no. 4, pp. 1–3, 2006.

[22] A. G. de Moraes and A. K. M. de Sales, ''Aplicacao de laboratorios remotos em microcontroladores PIC,'' in *Proc. 19th Congr. Brasileiro Autom. (CBA)*, 2012, pp. 3634–3641.

[23] P. Zenzerović and V. Sučić, ''Remote laboratory for microcontroller systems design,'' in *Proc. 34th Int. Conv. MIPRO*, May 2011, pp. 1685–1688.

[24] V. Fotopoulos, I. S. Anastasios, and F. Anastasios, ''Preparing a remote conducted course for microcontrollers based on Arduino,'' in *Proc. 7th Int. Conf. Open Distance Learn. (ICODL)*, 2013, pp. 1–6.

[25] J. Ferreira, Z. Nedić, J. Machotka, A. Nafalski, and Ö. Göl, ''International collaborative learning using remote workbenches,'' in *Proc. Annu. Conf. Eng. Technol. Edu.*, 2010, pp. 47–51.

[26] M. Tawfik, E. Sancristobal, S. Martin, G. Diaz, J. Peire, and M. Castro, ''Expanding the boundaries of the classroom: Implementation of remote laboratories for industrial electronics disciplines,'' *IEEE Ind. Electron. Mag.*, vol. 7, no. 1, pp. 41–49, Mar. 2013, doi: 10.1109/MIE.2012.2206872.

[27] M. Tawfik, E. Sancristobal, S. Martin, G. Diaz, and M. Castro, ''State-of-the-art remote laboratories for industrial electronics applications,'' in *Proc. Technol. Appl. Electron. Teach. Conf. (TAEE)*, Jun. 2012, pp. 359–364, doi: 10.1109/TAEE.2012.6235465.

[28] M. Reichenbach, M. Schmidt, B. Pfundt, and D. Fey, ''A new virtual hardware laboratory for remote FPGA experiments on real hardware,'' in *Proc. 2011 Int. Conf. E-Learn., E-Bus., Enterprise Inf. Syst. E-Government (EEE)*, 2011, pp. 17–23.

[29] K. Waschk. (2008). *Universal JTAG Library, Server and Tools*. [Online]. Available: http://urjtag.org/book

[30] F. Morgan, S. Cawley, and D. Newell, ''Remote FPGA lab for enhancing learning of digital systems,'' *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 3, Oct. 2012, Art. no. 18. doi: 10.1145/2362374.2362382.

[31] F. Morgan *et al.*, ''ViciLogic: Online learning and prototyping platform for digital logic and computer architecture,'' in *Proc. eChallenges (e)*, Oct. 2014, pp. 1–9, 2014.

[32] D. Hercog, B. Gergic, S. Uran, and K. Jezernik, ''A DSP-based remote control laboratory,'' *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3057–3068, Dec. 2007, doi: 10.1109/TIE.2007.907009.

[33] A. Kutlu and K. Taşdelen, ''Remote electronic experiments using LabVIEW over controller area network,'' *Sci. Res. Essays*, vol. 5, no. 13, pp. 1754–1758, Jul. 2010.

[34] K. Henke, T. Vietzke, H.-D. Wuttke, and S. Ostendorff, ''GOLDi—Grid of online lab devices Ilmenau,'' *Int. J. Online Eng.*, vol. 12, no. 4, pp. 11–13, 2016.

[35] K. Henke, T. Vietzke, H.-D. Wuttke, and S. Ostendorff, ''GOLDi—Grid of online lab devices Ilmenau: Demonstration of online experimentation,'' in *Proc. 3rd Experim. Int. Conf. (EXP.AT)*, Jun. 2015, pp. 109–110, doi: 10.1109/EXPAT.2015.7463230.

[36] J. Garcia-Zubia, D. Lopez-de-Ipina, and P. Orduna, ''Mobile devices and remote labs in engineering education,'' in *Proc. 8th IEEE Int. Conf. Adv. Learn. Technol.*, Jul. 2008, pp. 620–622, doi: 10.1109/ICALT.2008.303.

[37] P. Orduña *et al.*, ''An extensible architecture for the integration of remote and virtual laboratories in public learning tools,'' *IEEE Rev. Iberoamericana Tecnol. Aprendizaje*, vol. 10, no. 4, pp. 223–233, Nov. 2015, doi: 10.1109/RITA.2015.2486338.

[38] Freescale Semiconductor, Inc. (2013). *FRDM-KL25Z User's Manual Revision 2.0*, [Online]. Available: http://www.nxp.com/

[39] J. E. Corter, J. V. Nickerson, S. K. Esche, and C. Chassapis, ''Remote versus hands-on labs: A comparative study,'' in *Proc. 34th Annu. Frontiers Edu. FIE)*, 2004, pp. 1–5.

[40] C. A. Jara, F. A. Candelas, and F. Torres, ''Virtual and remote laboratory for robotics e-learning,'' *Comput. Aided Chem. Eng.*, vol. 25, pp. 1193–1198, Jun. 2008, doi: 10.1016/S1570-7946(08)80205-2.

[41] A. Yazidi, H. Henao, G.-A. Capolino, F. Betin, and F. Filippetti, ''A Web-based remote laboratory for monitoring and diagnosis of AC electrical machines,'' *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4950–4959, Oct. 2011, doi: 10.1109/TIE.2011.2109331.

[42] H. Vargas, G. Farias, J. Sanchez, S. Dormido, and F. Esquembre, ''Using augmented reality in remote laboratories,'' *Int. J. Comput. Commun. Control*, vol. 8, no. 4, pp. 622–634, 2013.

[43] L. Rodriguez-Gil, P. Orduña, J. García-Zubia, and D. López-de-Ipiña, ''Interactive live-streaming technologies and approaches for web-based applications,'' in *Multimedia Tools and Applications*. Cham, Switzerland: Springer, Mar. 2017, pp. 1–32, doi: 10.1007/s11042-017-4556-6.

[44] L. Rodríguez-Gil, J. García-Zubia, P. Orduña, and D. López-de-Ipiña, ''An open and scalable Web-based interactive live-streaming architecture: The WILSP platform,'' *IEEE Access*, vol. 5, pp. 9842–9856, 2017, doi: 10.1109/ACCESS.2017.2710328.

[45] G. Paravati, C. Celozzi, A. Sanna, and F. Lamberti, ''A feedback-based control technique for interactive live streaming systems to mobile devices,'' *IEEE Trans. Consum. Electron.*, vol. 56, no. 1, pp. 190–197, Feb. 2010, doi: 10.1109/TCE.2010.5439144.

[46] C. Zhang and J. Liu, ''On crowdsourced interactive live streaming: A Twitch.tv-based measurement study,'' In *Proc. 25th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, 2015, pp. 55–60.

[47] S. N. Şad and Ö. Göktaş, "Preservice teachers' perceptions about using mobile phones and laptops in education as mobile learning tools," *Brit. J. Edu. Technol.*, vol. 45, no. 4, pp. 606–618, 2014.
[48] D. G. de la Iglesia, J. F. Calderón, D. Weyns, M. Milrad, and M. Nussbaum, "A self-adaptive multi-agent system approach for collaborative mobile learning," *IEEE Trans. Learn. technol.*, vol. 8, no. 2, pp. 158–172, Apr. 2015, doi: 10.1109/TLT.2014.2367493.
[49] H. Crompton, D. Burke, K. H. Gregory, and C. Gräbe, "The use of mobile learning in science: A systematic review," *J. Sci. Edu. Technol.*, vol. 25, no. 2, pp. 149–160, 2016, doi: 10.1007/s10956-015-9597-x.
[50] Z. K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: Ongoing challenges and research opportunities," in *Proc. IEEE 7th Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2014, pp. 230–234, doi: 10.1109/SOCA.2014.58.
[51] *OpenOCD*. Accessed: Sep. 2017. [Online]. Available: http://openocd.sourceforge.net/
[52] R. M. Stallman, R. H. Pesch, and S. Shebs, *Debugging With GDB: The GNU Source-Level Debugger for GDB (GDB)*, 10th ed. Boston, MA, USA: GNU Press, 2017.
[53] J. Garcia-Zubia, D. Lopez-de-Ipina, P. Orduna, U. Hernandez, I. Angulo, and J. Irurzun, "Acceptance, usability and usefulness of WebLab–Deusto from students point of view," in *Proc. 3rd Int. Conf. Digit. Inf. Manage. (ICDIM)*, Nov. 2008, pp. 899–904, doi: 10.1109/ICDIM.2008.4746846.
[54] I. Gustavsson *et al.*, "On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories," *IEEE Trans. Learn. Technol.*, vol. 2, no. 4, pp. 263–274, Oct./Dec. 2009, doi: 10.1109/TLT.2009.42.
[55] D. Lang, C. Mengelkamp, R. S. Jäger, D. Geoffroy, M. Billaud, and T. Zimmer, "Pedagogical evaluation of remote laboratories in eMerge project," *Eur. J. Eng. Edu.*, vol. 32, no. 1, pp. 57–72, 2007, doi: 10.1080/03043790601055626.

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

LUIS RODRÌGUEZ-GIL received the dual degree in computer engineering and industrial organisation engineering in 2013, the M.Sc. degree in information security in 2014, and the Ph.D. degree in computer science from University of Deusto. He is currently a co-founder of the LabsLand remote labs company. Since 2009, he has been with the WebLab-Deusto Research Group, collaborating in the development of the WebLab-Deusto RLMS. He has authored several peer-reviewed publications and contributed to some open source projects.

JAVIER GARCÌA-ZUBÌA (M'08–SM'11) received the Ph.D. degree in computer science from University of Deusto, Spain. He is currently a Full Professor with the Faculty of Engineering, University of Deusto. He is the Leader of the WebLab-Deusto Research Group. His research interest is focused on embedded systems and remote laboratory design, implementation and evaluation.