Laboratory as a Service (LaaS): A Novel Paradigm for Developing and Implementing Modular Remote Laboratories

http://dx.doi.org/10.3991/ijoe.v10i4.3654

Mohamed Tawfik[1], Christophe Salzmann[2], Denis Gillet[2], David Lowe[3], Hamadou Saliah-Hassane[4], Elio Sancristobal[1], and Manuel Castro[1]
[1]Spanish University for Distance Education, (UNED), Madrid, Spain
[2] Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland
[3] University of Sydney, Sydney, Australia
[4] University of Quebec, Montreal, Canada

***Abstract*—The increasing adoption of remote laboratories in education along with the shift from eLearning 2.0 towards eLearning 3.0, have demanded several considerations in their implementation and delivery format. In response to these needs, this contribution introduces a novel model, Laboratory as a Service (LaaS), for developing remote laboratories as independent component modules and implementing them as a set of loosely-coupled services to be consumed with a high level of abstraction and virtualization. LaaS aims to tackle the common concurrent challenges in remote laboratories developing and implementation such as inter-institutional sharing, interoperability with other heterogeneous systems, coupling with heterogeneous services and learning objects, difficulty of developing, and standardization. Beyond the academic context, LaaS will facilitate the incorporation of remote laboratories in the ecosystem of the ubiquitous smart things surrounding us, which increases everyday with the approaching Web of Things (WoT) and artificial intelligence era. This, in turn, will create a breeding ground for online control, experimentation, and discovery—in either formal or informal context and with neither temporal nor geographical constraints.**

***Index Terms*—component-based remote laboratories, LaaS, modular remote laboratories.**

I. BACKGROUND

In the recent years, remote laboratories [1-4] have got a widespread acceptance among universities, particularly in engineering education and it's related. This is essentially owing to the significant benefits they provide compared to their traditional counterparts such as: improved student access and associated increases in utilization; support for resource sharing between institutions to offset costs; availability of a more diverse range of experimentation; mitigation of safety issues; and managed access which limits either intentional or unintentional misuse. Advocates of online laboratories outline the above mentioned conveniences associated with their use, whereas advocates of traditional laboratories argue that students should be exposed to real environments, which is consistent with the constructivism learning theory—though proponents of remote laboratories might argue that nowadays industrial processes are commonly automated and controlled remotely. In a similar fashion, several questions, debates, and empirical comparative studies on whether remote format is as effective as the traditional one have been generated [5, 6]. The general conclusion from these studies [6, 7] was that learning outcomes depends on the exact instructions given to the group and the different patterns of work and collaboration regardless of the delivery format.

For this reason, current array of concerns is primarily focused on issues related to remote laboratories delivery format and their pedagogical impact. These issues encompass their integration with heterogeneous educational systems and coupling with other services and learning objects in order to yield a rich scaffold educational environment and hence better learning experience and outcomes. In addition, such integration will promote sharing laboratories across institutions and hence more availability and cost offset. As a result, earlier efforts endeavored to integrate remote laboratories into educational systems in order to address a set of needs can be categorized and summarized as shown in Table I.

In a narrow sense, each approach within a particular scenario might have its own exclusive features such as queuing in Sahara and distribution through service brokers in iLab. In a abroad sense, integration with open source LMSs is more likely to be the ideal candidate, to build upon, in order to approach a complete educational platform because they typically allow: creating and realizing formal online courses with all the ancillary tools (e.g., administration tools, communication tools, evaluation, and tracking); support eLearning standards such as Shareable Courseware Object Reference Model (SCORM); and support metadata standards such as Learning Object Metadata (LOM) and Dublin Core.

However, unlike PLEs, LMSs are still abide by a monolithic design and lack interoperability with external services (e.g., learning objects, remote laboratories, etc.) or other LMSs [13]. Once these services are imported, student could easily mash up them to build his own learning environment either in a formal or informal context. For instance, services could be mashed up to provide scaffolding and other theoretical resources beside remote laboratories sessions (i.e., in the same portal) and hence, more student engagement and better distance education experience. For these reasons, several initiatives such as E-Learning Framework (ELF), Open Knowledge Initiative (OKI), IMS Abstract Framework (IMS AF), and IMS

Tool Interoperability (IMS TI) have defined initial steps towards customizable service-oriented LMSs. Service-oriented LMSs extend the capability of the ordinary LMSs and embrace the PLE approach in order to support a wider range of interoperability for both formal and informal learning. This shift is in line with the shift from Web 2.0 to Web 3.0, and consequently from eLearning 2.0 to eLearning 3.0. Even though Web 3.0 is still purely hypothetical, it is anticipated that future Web will embrace: the semantic Web and artificial intelligence; the Web of Things (WoT) and the omnipresence of everyday connected objects, and the mashup of loosely coupled services. These three speculated mainstreams could give us insights about the characteristics of next generation online learning environments and accordingly a delivery format of modern remote laboratories suited for this kind of environments can be determined.

Thus, delivering remote laboratories as a fixed GUI should be avoided as it confines consumers to a certain Web technology, as well as, impedes teachers and students from customizing their courses from their own point of view and opposes the spirit of Web 3.0 and next generation learning environments. Efforts done in order to convert fixed GUIs of provided laboratories into customizable GUIs of different Web technologies can be summarized as shown in Table II.

Nevertheless, the aforementioned solutions in Table II partially address the fixed GUI issue since they are all confined to Java technology. In attempt to address such issue, this article proposes a novel paradigm, Remote Laboratory as a Service (LaaS), for developing modern remote laboratories—as independent component modules—and implementing them as a set of loosely-coupled services to be consumed with a high level of abstraction and virtualization.

The rest of the article is structured as follows: *Section II* discusses the previous efforts and the related works, and outlines the novelty of the proposed solution. Prior delving into the description of the LaaS model, *Section III* describes the modular remote laboratories concept. *Section IV* describes the proposed LaaS model which builds on top of the modular remote laboratory concept. As a Proof-of-Concept (POC), in *Section V* the proposed theory is applied to the real-world by developing the first modular remote prototype. Finally, a conclusion is drawn in Section VI along with the future works.

II. RELATED WORKS

Earlier attempts to deliver remote laboratories as a service can be found in [17-21]. In [17], the functions of commercial instruments based on Virtual Instrument Software Architecture (VISA) and Interchangeable Virtual Instruments (IVI) were listed in Web Services Description Language (WSDL) files and registered in a Universal Description, Discovery and Integration (UDDI) to be allocated and consumed by Simple Object Access Protocol (SOAP) Web service. Asimilar approach for controlling instruments online using SOAP Web service is found in[18, 19]. In [20], a simple experiment was developed in

TABLE I.
A COMPARISON BETWEEN THE THREE COMMON EDUCATIONAL SYSTEM TYPES AND INTEGRATION PATTERNS

Integration Pattern (Educational System)	Objectives	Example	Indexing (metadata)	Standard Compatibility	Creating courses	Customization /Importing External Services
Remote Laboratory Management System (RLMS)	to exclusively allow accessing and administrating a wide pool of heterogeneous remote laboratory systems, which might be distributed across different institutions, through a common portal	iLab [8], Sahara [9], and Weblab Deusto [10]	×	×	×	×
Personal Learning Environment (PLE)	to allow experimentation in a customized and informal learning environment	Graaasp [11]	×	×	×	✓
Learning Management Systems (LMS)	to allow delivering remote laboratories within formal online courses making use of the services provided by LMSs	Library of Labs (LiLa) [12]	✓	✓	✓	×

TABLE II.
SOLUTIONS THAT ADDRESSES FIXED PROVIDED GUIS AND CONVERTS THEM INTO CUSTOMIZABLE GUIS OF DIFFERENT WEB TECHNOLOGIES

Ref.	Conversion	Description
[14]	LabVIEW→Java Applets	a middleware that publishes LabVIEW VIS's on the Internet providing a TCP/IP wrapping to their control loops. The server performs an automatic scan of all the VI's controls and indicators, initializes the network input port, and waits for an incoming connection from a remote Java clients. A Java library file is added to the final Java application IDE to allow it to automatically connect to the VIs and retrieves the list of controls and indicators URLs published by the middleware server. Afterwards, the developer links the URLs to the variables declared in the Java model in the IDE
[15]	Simulink→Java Applets	a middleware that enables connecting the Java model's variables to the block variables of the Simulink model without requiring any modification of it. It is direct if both software are in the same computer and if MATLAB and the Java model are located in different computers a stand-alone Java server application is used to allow the Java model to use a remote MATLAB server.
[16]	Simulink→Java Applets	a middleware for bridging MATLAB and JAVA applications through COM technology but only under Windows operating system

LabVIEW and delivered as Representational State Transfer (REST) Web service, to be consumed using Asynchronous JavaScript and XML (AJAX) calls. A similar approach based on REST Web service and AJAX is found in [21].

A distinctive approach was adopted in [22], where a further effort have been realized in order to add intelligence to remote laboratories at the server side and to make little or no assumption about the client. The underlying communications in this approach were realized using Websockets owing to its efficiency and high transmission rate. On the other hand, to promote compatibility with different client applications.

It is also worth noting that the acronym LaaS has been pronounced in the literature for almost three years few times, with two different interpretations. The first interpretation refers to the cloud computing and the Anything as a Service (XaaS) concepts as described in [23-27]. In these approaches, however, the difference between cloud computing and remote access is still blurred. Yet, there is no clear application of the cloud computing principles on real physical laboratories that might be distributed at various universities globally.

The second interpretation—the interpretation assumed in this paper—simply refers to the delivery of remote laboratory as a service that can interoperate with heterogeneous systems and services. The second interpretation is more generic and can be implemented many ways. For instance, in [28], Web service was adopted in conjunction with a proprietary Lab Description Language (LDL) developed by the author in order to achieve interoperability.

Even though, the solution proposed in this paper (LaaS) builds on top of these efforts, it has four main distinctive aspects. The first aspect is that Web service in LaaS is a method and not a solution itself and its adoption is not necessary. For instance, for data streaming (e.g., video and measurement streaming) low level protocol communications are implemented instead. The second and most important aspect is that LaaS goes further beyond abstracting the functions of the laboratories; it implies their development as a set of independent component modules in order to allow interchangeability of components between providers and consumers—seamlessly and programmatically—insofar as consumer could contribute with one or more component instead of the fully-reliance on the provider's equipment and facilities. The third aspect is that LaaS contemplates the future Web and the next generation learning environment in terms of: (1) seamlessly allocating and importing services; (2) bringing objects to the Web; and (3) mashing up and coupling services together—which was possible, in part, thanks to the modular remote laboratories concept. The fourth and last aspect is that LaaS is meant to be a model that addresses the development of remote laboratories, as well as, their implementation process broadly—which entails the relation between consumers, providers, and service broker, as well as, the format of exchanging information and resources between them. As a Proof-of-Concept, a modular remote laboratory was developed successfully and implemented according to the LaaS model.

III. MODULAR REMOTE LABORATORIES

Prior delving into the description of the LaaS paradigm, let's explain first the modular remote laboratory concept.

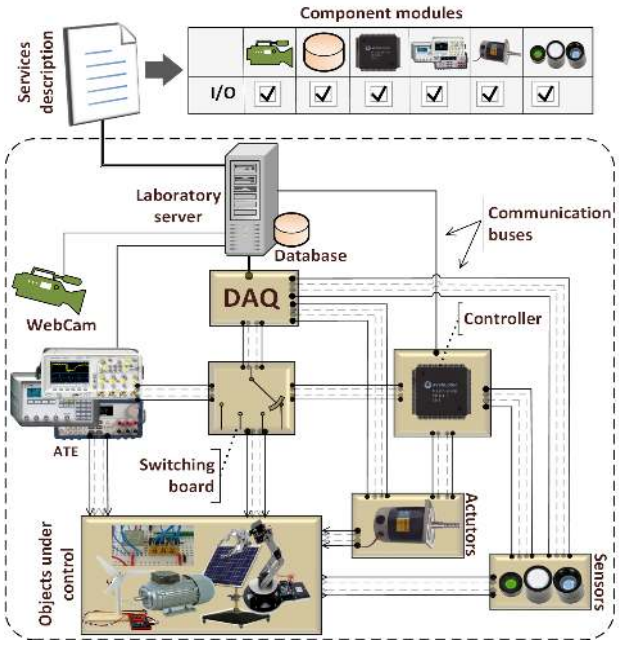

Figure 1. Modular remote laboratories architecture

Consider the generic and common remote laboratory architecture shown in Figure 1. Typically a remote laboratory consists of a laboratory server—which is connected to all the equipment and hosts the control software—in addition to any combination of the on-demand components shown in Figure 1. The control software can be developed either from scratch using a multipurpose programming language (e.g., Java, C#, or C/C ++) or using a commercial solution, commonly LabVIEW or MATLAB. The Data Acquisition (DAQ) board acts as an interface between the laboratory server and the equipment that don't support direct interface to the computer. A Webcam is used for video live streaming. Commercial Automatic Test Equipment (ATE) is used for specific signal generation or acquiring tasks. Standard connectors are used for connecting components directly to the laboratory server without intermediaries and they encompasses Universal Serial Bus (USB), LAN eXtensions for Instrumentation (LXI), PCI eXtensions for Instrumentation (PXI), and AdvancedTCA Extensions for Instrumentation and Test (AXIe). Sensors and actuators are used to convert physical parameters from the objects under control to electrical signals, and vice versa, respectively. A switching board is used for remote switching or wiring any terminals either from the objects under control or the ATE. Some applications might require a controller—in addition to the laboratory server—for a specific task. Commonly used controllers are either microcontrollers or Field Programmable Logic Arrays (FPGAs).

Modular remote laboratories are based on interchangeable component modules that expose their I/O terminals or their I/O connectors (i.e., if they physically don't exist or unavailable) in an independent and an abstracted way. Some components can be modularized and some are fixed and cannot be modularized or interchanged programmatically (e.g., laboratory server and DAQ board). The idea beyond modularizing remote laboratories is to facilitate maintenance, reusability, and interchangeability of components seamlessly and programmatically. In this sense, if the I/O terminals and connectors of all the component

modules of a remote laboratory are provided in a "service description file" in order to allow consumers to get clues on them as shown in Figure 1, the consumer would be able to consume them separately. Furthermore, if one of the component modules is not available and the appropriate I/O connectors are provided instead, the consumer could replace this module with his/her own one instead of the fully-reliance on the provider's equipment. For instance, a remote laboratory for image processing may expose an Application Programming Interface (API) to allow user to connect his/her camera capture. The image will be transmitted to the laboratory for processing and then return back to the user.

IV. LABORATORY AS A SERVICE (LAAS) PARADIGM

LaaS is a paradigm for developing and implementing modular remote laboratories with a high level of abstraction and virtualization. It builds upon the modular remote laboratory concept and implies the delivery of the entire laboratory functions and components in the "service description file" as a set of abstracted services. LaaS follows the Service Oriented Architecture (SOA) and fulfills its essential requirements in terms of interoperability, service description, and exchanging messages. It defines the relation between laboratory providers (i.e., providers of the "service description files"), service broker repository (i.e., Web portal in which "service description files" are indexed), and laboratory consumers (i.e., who build an end-user application upon the provided services).

All the laboratory functions are implemented using—but not limited to—resource-oriented Web service, REST, owing to its simplicity and high performance, as well as, its homogeneity with Web applications in general and mashup applications in particular. Activity-oriented Web service, SOAP, is another alternative for advanced applications with high level Quality of Service (QoS) requirements. Other middleware technologies such as Common Object Request Broker Architecture (CORBA), .NET Remoting, JAVA Remote Method Invocation (RMI), and Data Distribution Service (DDS) were excluded—despite their higher performance—for any of these reasons: (1) firewall restrictions; (2) complexity; and (3) platform dependency. For server pushing and persistent connections like data streaming (e.g., Webcam video streaming or measurement streaming), encoding over low level protocols such as TCP and UDP is provided as a URI with public IP address instead of Web service. The LaaS paradigm can be resumed in the following demonstrative case studies.

A. Case Study 1

Consider a modular remote laboratory for implementing a control strategy on an electric motor, where the user first uploads his/her Proportional-Integral-Derivative (PID) control program to the controller. Afterwards, he/she changes the PID parameters (e.g., speed and position) through the control software, and monitors the feedback effect of different control loops. The component modules of this laboratory are: a controller, a power supply, a Webcam, a database, and an electric motor. The LaaS paradigm implementation is depicted in Figure 2.

First, the lab provider prepares the "service description file" of his/her laboratory. The "service description file" contains all the services provided by the laboratory in either Web service format (i.e., for all transactions) and/or URIs (i.e., for data streaming). In addition, the file includes all the ancillary information or policies (e.g., metadata ontologies, days and hours of availability, providers contact details, cost if applicable, experiment description, additional URLs, etc.). The provider deposits the service description file into the service broker Web portal and indexes it using the associated metadata ontologies (e.g., control, electrical machines, and engineering) in order to be easily allocated by the consumer. The consumer allocates his/her desired laboratory—as a "service description file"—at the service broker Web portal and based upon the provided services, the consumer builds his/her own end-user application container using any technology he/she might prefer. Native-Web technologies such as AJAX and HTML5 are recommended in order to facilitate mobile access through Web apps, but consumer can also use Rich Internet Applications (RIAs). Assume using the native-Web based OpenSocial widget (www.opensocial.org). Once the widget is created, it can be rendered in any widget engine (e.g., PLE or LMS). The provided services can be consumed by more than one widget in conjunction. For instance, the consumer might want to introduce the PID parameters through his own widget but monitor the results in an external widget from different server that is specialized in building charts as shown in Figure 2.

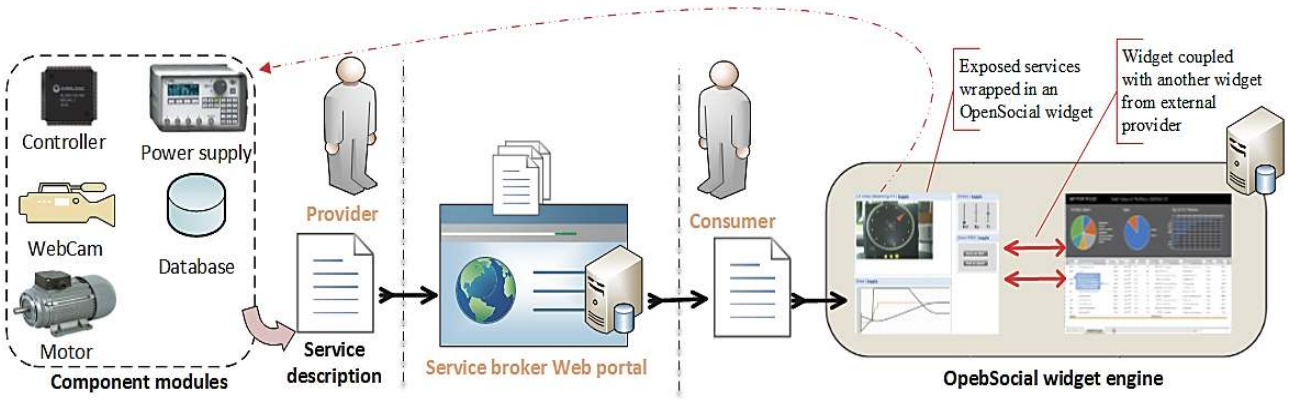

Figure 2. Case study 1: topology on LaaS paradigm implementation.

B. Case Study 2

Now consider another scenario similar to the previous but with the following modifications: the provider doesn't wish to share some of his laboratory component modules (i.e., the database and the power supply) in order to reduce the load on his/her own equipment and facilities and instead he/she leaves it to the consumer to connect his/her own component modules through I/O connectors, as depicted on Figure 3. In order to facilitate interchangeability, it is recommended to rely on well-known standard connectors such as Open Database Connectivity (ODBC) for the database and IVI or VISA for the power supply instrument. Thus, if the ODBC database I/O connectors are provided to the consumer in the "service description file", he/she would be able to connect his/her own ODBC-compliant database independently of its model or manufacturer as long as it adheres to the standard, and likewise for his/her IVI/VISA-complaint power supply.

If the laboratory is made available, it should be accessible unless another session is currently running by another user. A mechanism for checking "whether it is currently available or not" should be included in its design (e.g., using a Web service call). Else, the consumer should contact the provider for enquiries. The consumer can also build his/her own scheduling system for a large scale deployment with numerous groups and students. Scheduling system is out of the scope of the LaaS paradigm as it focuses on the lowest level side in order to maintain the "service description file" as abstracted as possible.

V. MODULAR REMOTE LABORATORY PROTOTYPE

In this section we apply the proposed theory to the real world by developing the first modular remote laboratory prototype to be delivered according to the LaaS paradigm. The developed laboratory is a motor-tacho laboratory shown in Figure 4, which consists of a NI USB-6009 DAQ card from National Instruments (www.ni.com), a 28GD11-222E/404E motor-tacho from Portescap (www.portescap.com), and an integrated Webcam. The software was entirely developed using LabVIEW and a numerical control code was developed using MATLAB and imported as an ".m file" into the LabVIEW code using the "LabVIEW MathScript Node".

A. Experiment Description

The idea of its experiment is very simple as it aims to emphasize the theory and prove its reliability rather than delving into the technical details of the experiment per se. In the experiment, user feeds the motor with a voltage range from 0V to 5V and monitors the corresponding voltage value measured by the tachometer. A control strategy is implement—using MATLAB—so that if the applied voltage is greater than 5V it will be automatically modified and introduced as only 5V. Likewise, if the applied voltage is lower than 0V, it will be introduced as 0V.

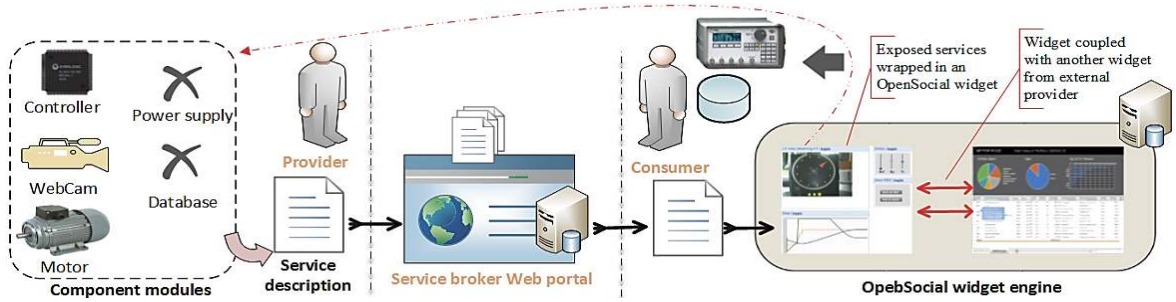

Figure 3. Case study 2: topology on RLaaS paradigm implementation.

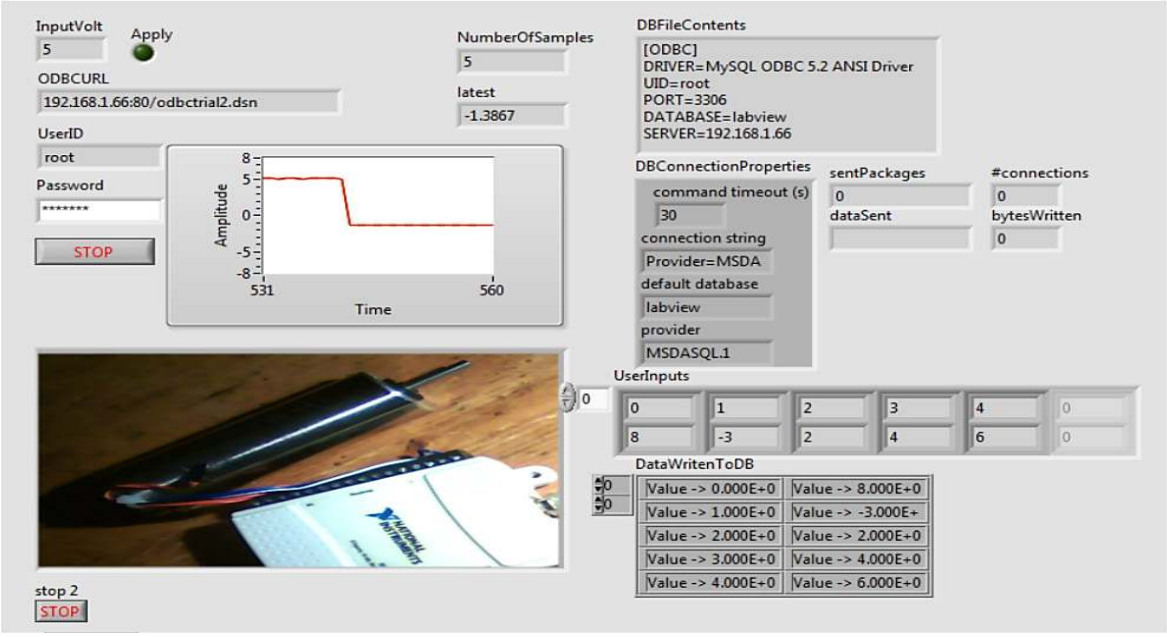

Figure 4. Motor-tacho laboratory.

The tachometer measurement is streamed continuously until the user either stops the experiment or introduces a new input voltage value. Each time the user inputs a value, it is automatically recorded and stored temporarily. Finally, when the user stops the experiment, all the introduced input voltage values—previously stored—are retrieved and copied to his/her database (i.e., not the database of the provider).

B. LaaS implementation

The motor-tacho laboratory has three component modules as shown in Figure 5, and one of these modules, the database, allows interchangeability using a standard connector, ODBC. This laboratory requires that the consumer connects his/her own ODBC-compliant database to the laboratory server software by sending its Data Source Name (dsn) file in order to be identified. The file is transferred as follows: first, the consumer hosts the ".dsn file" in a Web server and provides the URL of the file to the laboratory server software through a Web service call; then, the laboratory server software copies the information in the ".dsn file" and use it to communicate remotely with the consumer's database.

Web service were created using the "LabVIEW REST-Ful Web Service Tool" and through proxy VIs—not the main laboratory software VI—because Web service in LabVIEW cannot run with loops owing to the inherent HTTP latency compared to the loops speed. Thus, in order to keep the main laboratory software VI running and accepting sequential calls, Web service should be implemented using proxy VIs that don't contain loops so that Web service calls would be handled by the proxy VIs and the proxy VIs would accordingly communicate with the running laboratory software VI. This will, in turn, allows the provider to visually monitor the main laboratory software VI running and its associated bugs. The communication between the proxy VIs and the laboratory software VI cannot be local even if both run on the same machine because the proxy VIs will be: uploaded to memory, hosted by the "LabVIEW Application Web Server", and deployed independently of the LabVIEW environment. Thus, the communication will be realized on network using "LabVIEW Shared Variables" as illustrated in Figure 6. In the current example, two proxy VIs will be needed, method1 and method1, as shown in Table III. Each proxy VI acts as a single Web service method. The default TCP port of the "LabVIEW Application Web Server" is 800.

TABLE III.
URL MAPPINGS OF THE TWO HTTP METHODS

	Web Method VI	HTTP Method	Default Override
/method2/:UserID/:Password/* ODBCURL	method1.vi	GET	Apply (value=1)
/method1/:InputVolt	method2.vi	GET	stop (value=1)

The tachometer streams the encoded reading on the TCP port 89 once an incoming connection is detected. The Webcam server was developed in LabVIEW using the ActiveX control distribution of VideoCapX (www.videocapx.com). LabVIEW act s as an ActiveX container and sequential methods and properties were created using the "Property Node" and the "Invoke Node" functions in order to allow streaming encoded captures over the TCP port 88. Table IV shows the content of the "service description file" in details.

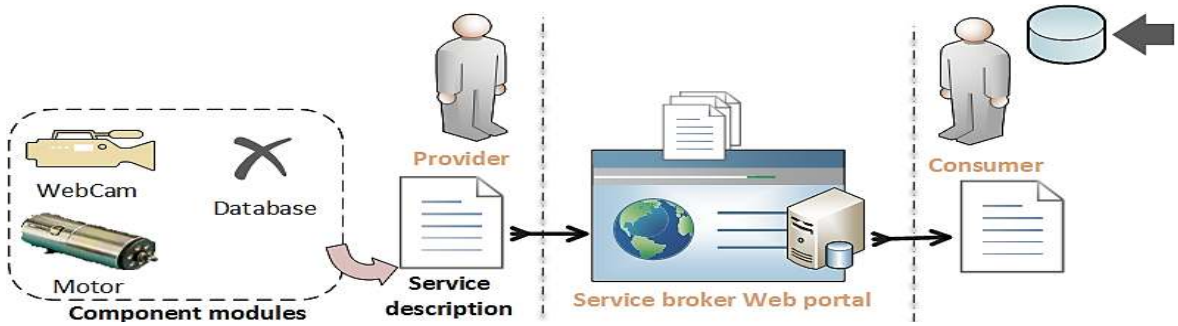

Figure 5. LaaS implementation of the motor-tacho laboratory prototype.

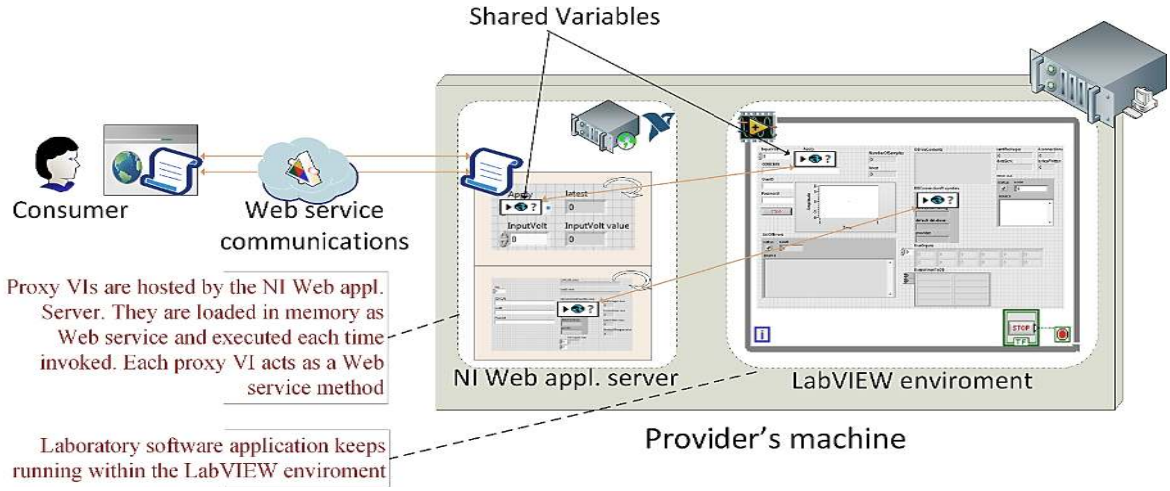

Figure 6. Web service implementation in LabVIEW.

TABLE IV.
SERVICE DESCRIPTION FILE OF THE TACHO-MOTOR REMOTE LABORATORY

Description	A remote laboratory for switching on a permanent magnet DC motor and reading its speed using a tachometer.
Keywords	Control theory, electrical machines, DC motor, and electrical engineering.
Provider	Spanish University for Distance Education (UNED)
Contact	mtawfik@ieec.uned.es
Operation days/hours	Open for public on Friday of each week, 9am to 9pm, otherwise contact for enquiry.
Additional resources	http://...../Motortacho_User_Manual.pdf , http://youtube.com/watch?V=6.....MotortachoRemoteLab.
Modular components	WebCam (provider), motor-tacho (provider), database (user, standard=ODBC).
Provided services	1. http://...lab server IP...:88 *i.e., for Webcam streaming>>encapsulation ASF/WMV, video codec (VP8), audio codec: MPEG audio, caching 600ms.* 2. http://... lab server IP...:89 *i.e., for tachometer reading streaming>>data is sent as a String and at 400 ms sampling rate.* 3. http://...app Web server IP...:80/webServicemethod1/:InputVolt *i.e., allows user to start a new session, insert the input voltage value, apply the change, and read the tachometer latest value.* 4. http://... app Web server IP...:80/webservice/method2/:UserID/:Password/*ODBCURL *i.e., allows user to: stop the running experiment session, introduce the address of his/her ODBC-compliant database's ".dsn file", and introduce the database credentials (i.e., username and password). In response, the user gets back: the introduced parameters by him/her except the password value; a list of all voltage input values introduced (i.e., this list is also copied to the user's database in a new created columns); database connection information; number of TCP connections for tachometer reading streaming and the sent packages; and number of samples written by the virtual DAQ channel.*
Access management	If the laboratory is available, a single user can start a session using method1. Once a session is started the laboratory will be occupied and not responding any calls until the current session is terminated using method2.

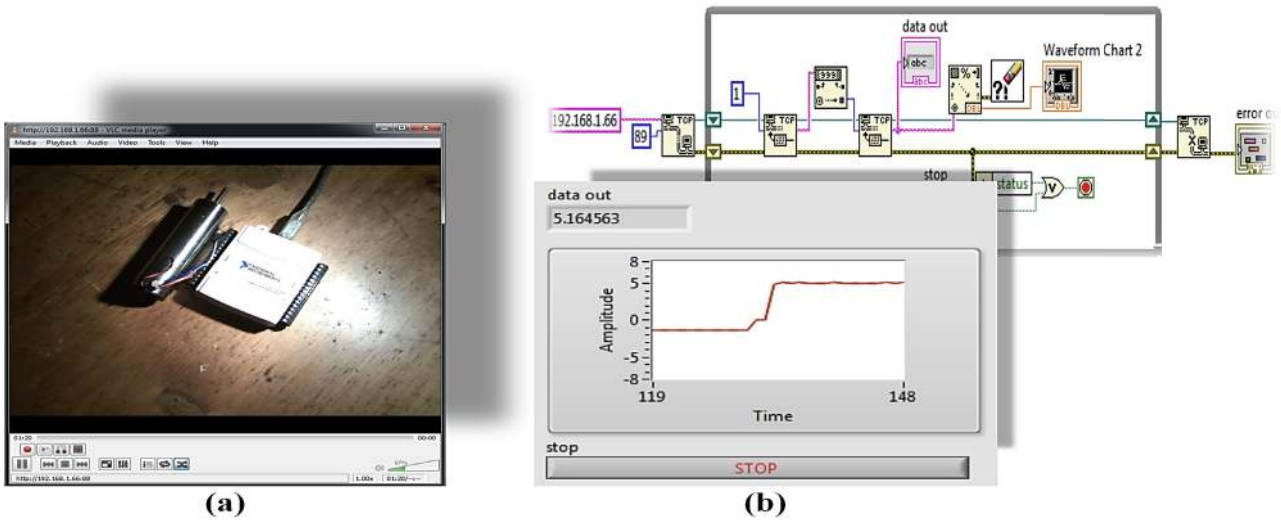

Figure 7. Data streaming: (a) video streaming using VLC and (b) tachometer reading streaming using LabVIEW.

C. Consumption

Assuming the "service description file" was deposited and indexed in a service broker Web portal, now let's consider the scenario from the consumer's perspective. After allocating and reading the "service description file", and having understood the laboratory functions and components, the consumption can be realized as shown in the following sequences:

1. We start a new session and introduce a voltage input value of 8V via the Web service method1 call, "http://…app-Web server-IP…:800/webservice/method1/8". As a result, only 5V is applied due to the implemented control strategy, the tachometer latest reading is retrieved, and the motor keeps rotating. The response is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Response>
    <Terminal>
        <Name>Inputvolt value</Name>
        <Value>8.00</Value>
    </Terminal>
    <Terminal>
        <Name>latest</Name>
        <Value>4.44</Value>
    </Terminal>
</Response>
```

2. Camera capturing can be streamed by decoding the data received over the URL, "http://...lab-server-IP...:88", either programmatically or using a decoding client such as VLC from VideoLAN (www.videolan.org/vlc), as shown in Figure 7(a). Similarly, tachometer reading can be streamed during the execution of the motor, by decoding the data received from the URL, http://...lab-server-IP...:89, either programmatically or using any decoding client such as LabVIEW, as shown in Figure 7(b).

3. Step 1 is repeated with different input values: -3, 2, 4, and 6V. Notice that the -3V value will apply only 0V.
4. We prepare the URL of the ".dsn file" of our ODBC-compliant database, "http://...consumer's-Web-server-IP...:80/odbctrial2.dsn". Afterwards, we send the file along with the database credentials (i.e., username = root and password = labview) to the laboratory server software via the Web service method2 call, "http://...app Web server IP..:800/webservice/method2/root/labview/…consumer's-Web-server IP…:80/odbctrial2.dsn". As a result, the session is ended and the following parameters are shown: the database connection properties; the 2D array of the 5 introduced voltage values (i.e., this array is copied to our database); the TCP connections for tachometer reading streaming and the sent packages; and the number of samples written by the virtual DAQ channel. The response is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Response>
  <Terminal>
      <Name>UserID value</Name>
      <Value>root</Value>
  </Terminal>
  <Terminal>
      <Name>sentPackages value</Name>
      <Value>79</Value>
  </Terminal>
  <Terminal>
      <Name>#connections value</Name>
      <Value>15</Value>
  </Terminal>
  <Terminal>
      <Name>NumberOfSamples value</Name>
      <Value>5</Value>
  </Terminal>
  <Terminal>
    <Name>User Inputs values</Name>
    <Value>
     <DimSize>5</DimSize>
     <DimSize>2</DimSize>
     <Name><Value>0.00</Value></Name>
     <Name><Value>8.00</Value></Name>
     <Name><Value>1.00</Value></Name>
     <Name><Value>-3.00</Value></Name>
     <Name><Value>2.00</Value></Name>
     <Name><Value>2.00</Value></Name>
     <Name><Value>3.00</Value></Name>
     <Name><Value>4.00</Value></Name>
     <Name><Value>4.00</Value></Name>
     <Name><Value>6.00</Value></Name>
    </Value>
  </Terminal>
  <Terminal>
    <Name>ODBCURL value</Name>
    <value>192.168.1.66:80/odbctrial2.dsn</value>
  </Terminal>
  <Terminal>
    <Name>DBConnectionProperties value</Name>
    <Value>
      <Name>command timeout (s)</Name>
      <Value>30</Value>
      <Name>connection string</Name>
      <Value>Provider=MSDASQL.1;User ID=root;</Value>
      <Name>default database</Name>
      <Value>labview</Value>
      <Name>provider</Name>
      <Value>MSDASQL.1</Value>
    </Value>
  </Terminal>
</Response>
```

5. Finally, we check the new created columns and the data written to our database.

Upon these provided services, consumer can build the end-user application using any technology or programming languages he/she prefers. Consumer can also build customizable applications suited for service-oriented LMSs and can couple and mash up the laboratory with other interoperable learning objects across the Web.

VI. CONCLUSION AND FUTURE WORKS

In this contribution, two novel concepts were introduced: (1) modular remote laboratories, which aims to convert laboratories into modular components in order to facilitate maintenance, reusability, and interchangeability of components seamlessly and programmatically; and (2) LaaS paradigm: which aims to convert modular remote laboratories into a set of services to be consumed by users with a high level of abstraction and virtualization. It defines, as well, the broader implementation mechanism of these laboratories. A broad case-study example that resumes both concepts was provided. Afterwards, a practical implementation of both concepts was provided, where a simple modular remote laboratory prototype was successfully developed and consumption results were provided.

From the low level perspective, future works will be focused on expanding the application range and modularizing different kinds of remote laboratories with different components and operation scenarios in order to investigate further issues and discover further solutions. As well, future works will be focused on implementing a scheduling mechanism using extra layers while maintaining the service description file as abstracted as possible in accordance with the premise of the LaaS paradigm.

From the high level perspective, the final goal is to set bases towards an acceptable standard model to which developers and laboratory providers could adhere to. For this purpose, further collaboration will be realized with the IEEE P1876™ Standard for Networked Smart Learning Objects for Online Laboratories Working Group and the Global Online Laboratory Consortium (GOLC) consortium.

ACKNOWLEDGMENT

Authors would like to acknowledge the support of the following projects: e-Madrid (S2009/TIC-1650), RIPLECS (517836-LLP-1-2011-1-ES-ERASMUS-ESMO), PAC (517742-LLP-1-2011-1-BG-ERASMUS-ECUE), and MUREE (530332-TEMPUS-1-2012-1-JO-TEMPUS-JPCR). As well, Authors would like to acknowledge the support of the VISIR Community, the GOLC consortium, and the IEEE P1876™ Standard for Networked Smart Learning Objects for Online Laboratories Working Group.

Last but not least, authors would like to acknowledge the project s-Labs (TIN2008-06083-C03-01) for financially supporting the research visit of Mr. Tawfik at UTS and EPFL, which resulted in developing the theory and the prototype.

REFERENCES

[1] M. Tawfik, E. Sancristobal, S. Martin, G. Diaz, and M. Castro, "State-of-the-art Remote Laboratories for Industrial Electronics Applications," presented at the Technologies Applied to Electronics Teaching (TAEE), 2012. http://dx.doi.org/10.1109/TAEE.2012.6235465

[2] 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," *Industrial Electronics Magazine, IEEE,* vol. 7, p. 9, March 19 2013. http://dx.doi.org/10.1109/MIE.2012.2206872

[3] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, J. Peire, *et al.*, "On the Design of Remote Laboratories," presented at the

IEEE Global Engineering Education Conference (EDUCON), Marrakesh, Morocco, 2012.

[4] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, J. Peire, *et al.*, "Virtual Instrument Systems in Reality (VISIR) for Remote Wiring and Measurement of Electronic Circuits on Breadboard," *Learning Technologies, IEEE Transactions on,* vol. 6, p. 13, March 12 (First Quarter) 2013.

[5] E. D. Lindsay and M. C. Good, "Effects of laboratory access modes upon learning outcomes," *Education, IEEE Transactions on,* vol. 48, p. 13, November 2005. http://dx.doi.org/10.1109/TE.2005.852591

[6] J. E. Corter, S. K. Esche, C. Chassapis, J. Ma, and J. V. Nickerson, "Process and learning outcomes from remotely-operated, simulated, and hands-on student laboratories," *Computers & Education,* vol. 57, p. 14, November 2011. http://dx.doi.org/10.1016/j.compedu.2011.04.009

[7] I. E. Allen and J. Seaman, "The ninth annual survey of Solan-C: Going the Distance: Online Education in the United States, 2011," The Sloan Consortium, Newburyport 2011.

[8] V. J. Harward, J. A. del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, *et al.*, "The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proceedings of the IEEE,* vol. 96, p. 20, June 2008. http://dx.doi.org/10.1109/JPROC.2008.921607

[9] D. Lowe, T. Machet, and T. Kostulski, "UTS Remote Labs, Labshare, and the Sahara Architecture," in *Using Remote Labs in Education: Two Little Ducks in Remote Experimentation*, J. G. Zubía and G. R. Alves, Eds., ed University of Deustoo, 2011, p. 22.

[10] P. Orduna, "Transitive and Scalable Federation Model for Remote Laboratories," Doctoral Thesis, University of Deusto, Bilbao, 2013.

[11] B. Evgeny, S. Christophe, and D. Gillet, "Widget-Based Approach for Remote Control Labs," presented at the 9th IFAC Symposium on Advances in Control Education, Resort Automobilist, Russia, 2012.

[12] V. Mateos, A. Gallardo, T. Richter, L. Bellido, P. Debicki, and V. A. Villagrá, "LiLa Booking System: Architecture and Conceptual Model of a Rig Booking System for On-Line Laboratories," *International Journal of Online Engineering (iJOE),* vol. 7, p. 10, 2011.

[13] D. Dagger, A. O'Connor, S. Lawless, E. Walsh, and V. P. Wade, "Service-Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services," *Internet Computing, IEEE,* vol. 11, p. 8, May-June 2007. http://dx.doi.org/10.1109/MIC.2007.70

[14] H. Vargas, J. Sánchez-Moreno, S. Dormido, C. Salzmann, D. Gillet, and F. Esquembre, "Web-Enabled Remote Scientific Environments " *Computing in Science & Engineering,* vol. 11, p. 11, May-June 2009. http://dx.doi.org/10.1109/MCSE.2009.61

[15] G. Farias, R. D. Keyser, S. Dormido, and F. Esquembre, "Developing Networked Control Labs: A Matlab and Easy Java Simulations Approach," *IEEE Transactions on Industrial Electronics,* vol. 57, p. 10, October 2010. http://dx.doi.org/10.1109/TIE.2010.2041130

[16] P. Bisták and P. Folvar, "Remote Laboratory Java Server Based on JACOB Project," *International Journal of Online Engineering (iJOE),* vol. 7, p. 4, February 2011.

[17] H. Saliah-Hassane, D. Benslimane, I. D. L. Teja, B. F. L.K., G. Paquette, M. Saad, *et al.*, "A General Framework for Web Services and Grid-Based Technologies for Online Laboratories," presented at the The International Conference on Engineering Education and Research (iCEER) Tainan, Taiwan, 2005.

[18] C. D. Capua, A. Liccardo, and R. Morello, "On the Web Service-Based Remote Didactical Laboratory: Further Developments and Improvements," presented at the Instrumentation and Measurement Technology Conference (IMTC), Proceedings of the IEEE (Volume:3) Ottawa, Ont. , 2005.

[19] A. Baccigalupi, C. D. Capua, and A. Liccardo, "Overview on Development of Remote Teaching Laboratories: from LabVIEW to Web Services," presented at the Instrumentation and Measurement Technology Conference (IMTC), Sorrento, Italy, 2006.

[20] M. Ngolo, L. B. Palma, F. Coito, L. Gomes, and A. Costa, "Architecture for remote laboratories based on REST web services," presented at the E-Learning in Industrial Electronics. ICELIE '09. 3rd IEEE International Conference on Porto 2009.

[21] S. Dutta, S. Prakash, D. Estrada, and E. Pop, "A Web Service and Interface for Remote Electronic Device Characterization," *Education, IEEE Transactions on,* vol. 54, p. 6, November 2011.

[22] C. Salzmann and D. Gillet, "Smart device paradigm, Standardization for online labs," presented at the Global Engineering Education Conference (EDUCON), IEEE, Berlin, 2013.

[23] V. Cheruku, "Integrating Physical Laboratories into a Cloud Environment," Master Thesis, Computer Science, North Carolina State University, Raleigh, North Carolina, 2013.

[24] R. I. Dinita, G. Wilson, A. Winckles, M. Cirstea, and A. Jones, "A Cloud-based Virtual Computing Laboratory for Teaching Computer Networks," presented at the Optimization of Electrical and Electronic Equipment (OPTIM), 13th International Conference on, Brasov, 2013.

[25] C. R. S. d. Oliveira and I. N. d. Oliveira, "Uma proposta para a disponibilidade de Laborató-rios de Física como serviços da Computação em Nuvem," presented at the 9th International Information and Telecommunication Technologies Symposium (I2TS'2010) Rio de Janeiro, Brazil, 2010.

[26] J. Rugelj, M. Ciglarič, A. Krevl, M. Pančur, and A. Brodnik, "Constructivist Learning Environment in a Cloud," in *Workshop on Learning Technology for Education in Cloud (LTEC'12)*. vol. 173, L. Uden, E. S. Corchado Rodríguez, J. F. De Paz Santana, and F. De la Prieta, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 193-204. http://dx.doi.org/10.1007/978-3-642-30859-8_18

[27] Y. Luo and X. Q. Zhang, "Cloud-Based Platform Building Research of Teaching Resources," *Applied Mechanics and Materials,* vol. 347-350, p. 4, August 2013. http://dx.doi.org/10.4028/www.scientific.net/AMM.347-350.3347

[28] S. Seiler, "Laboratory as a Service A Holistic Framework for Remote and Virtual Labs," Doctoral Thesis, Faculty of Mechanical Engineering-Department of Mechatronics, Tallinn University of Technology, 2012.

AUTHORS

M. Tawfik, E. Sancristobal, and M. Castro are with the Electrical & Computer Engineering Department, Spanish University for Distance Education (UNED), Madrid, Spain, (email: mtawfik@ieec.uned,es, elio@ieec.uned.es, and mcastro@ieec.uned.es).

C. Salzmann and D. Gillet are with the School of Engineering, Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland (email: Christophe.salzmann@epfl.ch and denis.gillet@epfl.ch).

D. Lowe is with the Faculty of Engineering and Information Technologies, University of Sydney, Sydney, Australia (email: david.lowe@sydney.edu.au).

H. Saliah-Hassane is with TELUQ, University of Quebec, Montreal, Canada (email: saliah@teluq.ca).

This research was partially funded by the European Union in the context of the Go-Lab project (Grant Agreement no. 317601) under the Information and Communication Technologies (ICT) theme of the 7th Framework Programme for R&D (FP7. This document does not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of its content." Submitted, January, 17, 2009. Published as resubmitted by the authors on May, 16, 2009.