

Automatic Management Services for Remote/Virtual Laboratories

<http://dx.doi.org/10.3991/ijoe.v10i6.3992>

R. Pastor-Vargas , Ll. Tobarra , S. Ros , R. Hernández , A. Robles and M. Castro
Spanish University for Distance Education, Madrid, Spain

Abstract—Development of virtual/remote laboratories is a common task involved in the design of course’s assessments (in engineering disciplines). The deployment of these laboratories must manage several aspects related to the real use of them, as user’s authorization and access, tracking of usage information, loading/saving data from experiments and so on. These features must be implemented by developers in a particular way for each laboratory. This paper shows how these services can be used automatically, with no development, using the RELATED management services. The deployment task consists of making a connector to the RELATED framework in order to consume these services. A full example is shown, describing the steps followed to integrate a previously developed low cost laboratory (based on a Lego system).

Index Terms—component; management services; software integration; remote and virtual laboratories

I. INTRODUCTION

Basic development of virtual and remote laboratories [1] [2] consists in designing a software system with Internet connection capabilities. This system must allow direct interaction with such laboratory [3]. Once this is done, the next step consists in the laboratory’s deployment. This deployment permits managers to control the laboratory’s visibility. Also, users (usually students) can get access to the laboratory in order to run the defined experiments. In a real environment it is very important to take into account some aspects related to the laboratory’s management:

- Users associated to the laboratory.
- Authorization system for users (role based)
- Analytics about laboratory use (sessions based).
- Booking system.
- Access to experiment’s saved data.

The laboratory’s developer must take into account the previous aspects, so he/she must provide an implementation of these services (more development effort is needed). The final objective is to get a reliable and efficient system related to the use/consumption of laboratory’s resources. Obviously, the implementation can be specific for the lab’s implementation, but it’s more efficient to reuse tools and services developed with this objective in mind. This is the RELATED case [4]. RELATED allows the building of remote and virtual laboratories in a methodological and structured way. But it also provides a set of services which implement the aspects to consider in the deployment of online laboratories.

The RELATED’s services are implemented using a reliable architecture based on Service Oriented Architectures (SOA). Every aspect previously considered is implemented by SOAP (Simple Object Access Protocol) services. These services are orchestrated to get a fully functional system which combines different scenarios of services usage. Also, these SOAP services can be consumed by web/mobile applications. In this particular case, RELATED has a functional web which uses these services [5].

The services architecture is based on a modular approach and every laboratory integrated in RELATED uses them automatically. This way, any RELATED laboratory uses this service to check user’s authorization, bookings, etc.

In Figure 1, the RELATED service’s architecture is described. The main features in the deployment of remote/virtual labs are defined as SOAP services running in a Web services container (Axis2/Tomcat). Every feature (users, booking, data, sessions, etc.) is implemented as an individual web service with a subset of methods providing/setting information associated to the environment of a real laboratory (users, lab’s experiments, lab’s data, lab’s bookings, etc.). Every laboratory can be integrated by means of a connector (called RLAB system) in order to consume the management services. All these services are integrated in a transparent way, configuring the connector using properties defined in a declarative way (XML based).

Additionally, the connector defines its own services in order to be consumed from/to RELATED infrastructure. These services are based on RMI (Remote Method Invocation) and REST (Representational State Transfer) technologies. In particular, the REST service associated to the connector can be used to build web/mobile labs in a simple way. RELATED provides a full Javascript library to use the management services from HTML5/CSS/Javascript applications. These applications can be used as hybrid applications in a mobile environment, using specific frameworks to build native applications (for example, PhoneGap [6]). The RMI services are consumed from Java applications in desktop environments, using the RELATED client for experiments.

In this paper is shown how a simple online laboratory (with no management services) can be integrated into RELATED. This integration provides automatic access to the full set of services provided by RELATED. The automatic use of services has many advantages, but the most important is the development’s time saving.

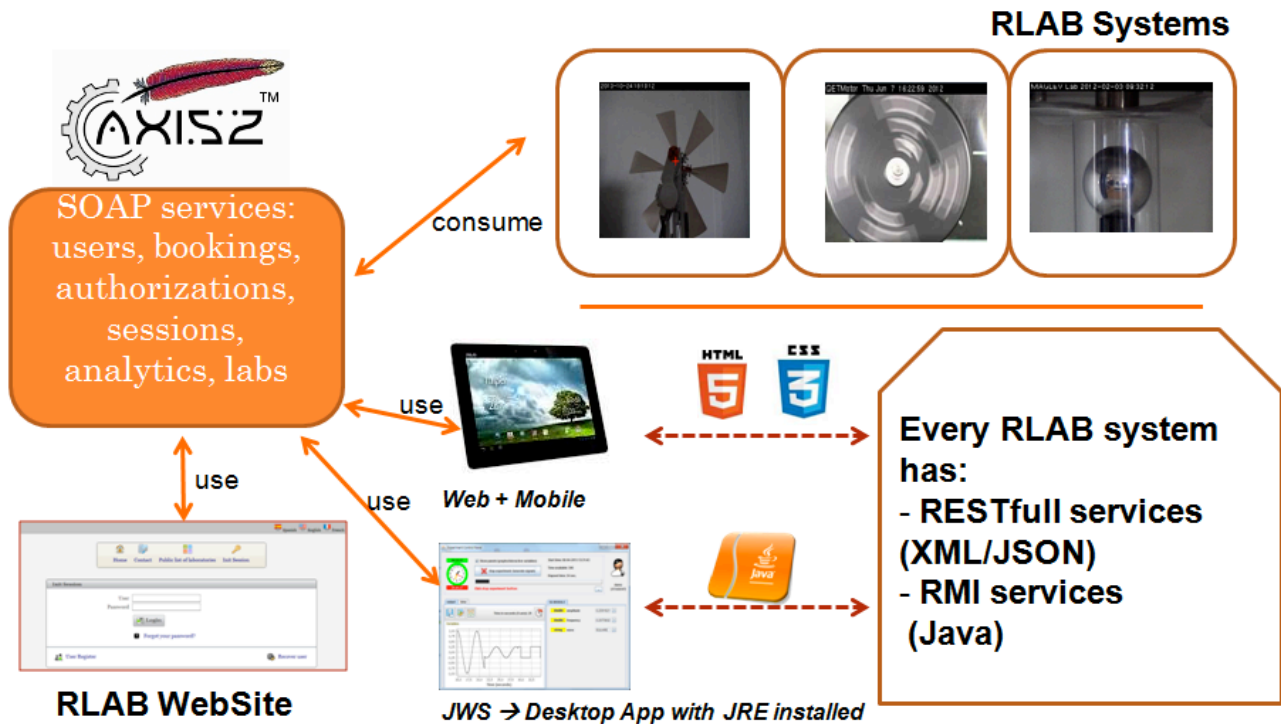


Figure 1. RELATED framework architecture.

The paper is structured as follows. In section II, it is shown the initial development of a low cost online laboratory. This laboratory uses Lego components to build a wind turbine system. The integration with the RELATED framework and services is presented in section III. Section IV shows how managers and users can use the services by interacting with the RELATED web site. Also, the running of laboratory's experiment, from the RELATED's point of view, it will be presented. Finally, conclusions will be presented in the last section.

II. LABORATORY WITH NO MANAGEMENT SERVICES

The developed laboratory is based on low cost Lego components. The laboratory is a renewable energy system based on the Lego Mindstorms NXT v2 robotic toolkit [7], extended with the renewable energy kit [8] which uses a sensor called Energy Meter. The Energy Meter sensor is able to measure a set of parameters related to the power production (such as voltage, electric current, joules stored in the battery, and watt), as well as storing the electrical power. This way, a remote lab on aeolian energy is implemented. This aeolian laboratory focuses on helping students to understand how wind is directly related to the generation of energy. For this reason, students are asked to register and observe the generation of voltage for different wind power values. A picture of the aeolian laboratory can be seen in Figure 2.

Such laboratory is implemented with specific technology provided by LabView. This LabView's technology allows the programming of several Lego's components, in order to get/set values from these components (for example, energy produced by the wind turbine).

The entire laboratory is represented as a set of REST (Representational State Transfer) services. These services allow the experiment's control of wind power generation. Also, it's possible to get/set the laboratory's data (motor's

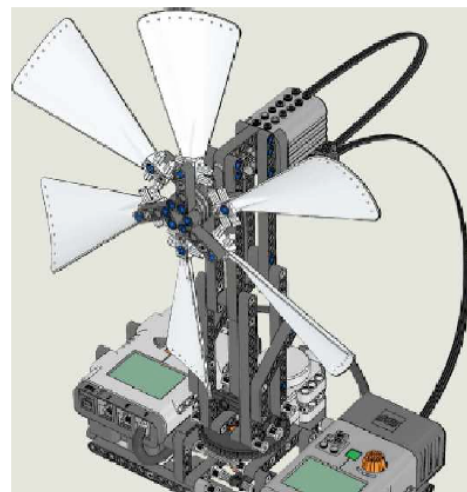
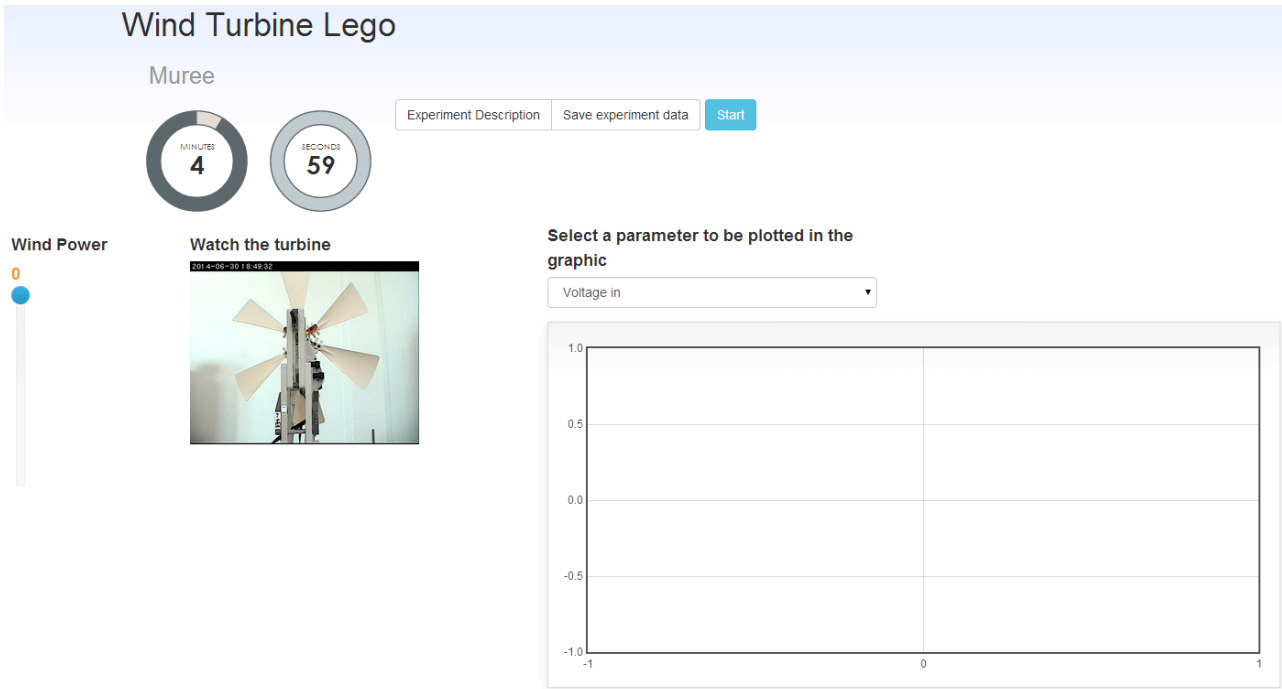


Figure 2. The aeolian laboratory.

power). These services are consumed by a web interface (simple web page with Javascript support) in order to carry out the experiment. The web interface is shown in Figure 3.

This REST API provides a full set of web methods which can be integrated in several development environments, but also has the advantage to be managed by "web commands". For example, it's possible to get the lab's status using a simple web command (a HTTP request like this: <http://62.204.199.219:8080/WindTurbineWS3/status>) which provides status information (JSON format):

```
{ "Status":
  ["Ok", "NXT", "8655", "No Error",
   "Idle", "30/06/2014
  18:54:01"]
}
```



Copyright © 2013 - SCC @ UNED

Figure 3. Web interface for the aeolian laboratory.

In this case, the Lego lab is ready (“Ok”) and available to be used (“Idle”). There are several web commands (HTTP request) available to get/set the information related to the lab, but these commands are only functional if there is an open experiment session.

This laboratory has no management services associated, so probably it is not ready for a real environment with several users accessing at the same time. To provide these management services, a full integration with the RELATED framework has been done. This integration is described in section III.

III. RELATED’S INTEGRATION

In order to get the laboratory’s integration into RELATED, the methodology of development defined in RELATED has been used [9]. This methodology defines how a lab is considered from the RELATED perspective and how a connector must be built using the ‘RLAB system’ concept. A laboratory (from the RELATED perspective, see Figure 4) is a set of experiments defining the different behaviours of the lab.

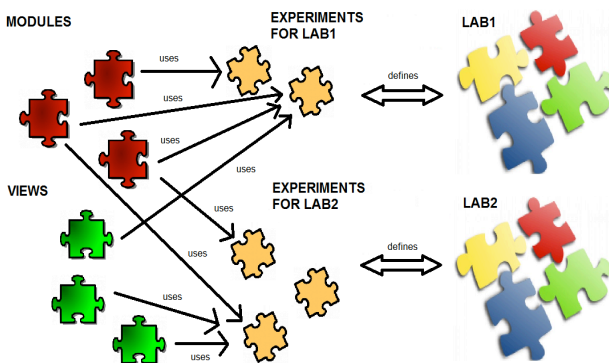


Figure 4. Laboratory components form RELATED’s perspective.

Every experiment is composed by two types of components: modules (runnable entities with a controlled lifecycle) and views (corresponding with graphical interfaces used in the lab’s interaction). At least one module is needed to build a RELATED laboratory. The module lifecycle defines the running of the experiment (in the case of only one module). In this case, we have the aeolian laboratory which exposed a set of REST service, so the developed RELATED module is really a simple REST client consuming the LabView services. Otherwise, the user’s interaction is very limited because it is only possible to change/set the power applied to the wind turbine motor (0-100, in percentage). This way, the specific view for the aeolian laboratory has been developed to provide visual feedback of main variables. In order to get a real video streaming of the camera which focuses to the wind turbine (Aeolian lab), a previously developed view has been added. The modular approach of RELATED allows the reuse of components (modules and views), only supplying the module/view definition and its implementation.

Once the components are defined, experiments are declared in the configuration’s lab file (XML based). The experiment tag includes so many modules and views tags as needed. While the experiment is running, every module defined in the experiment is executed as well. Also, every view included is shown to the user running the experiment. The configuration’s lab file is presented in Figure 5.

Next, details about the module and the view development are presented.

A. Module implementing laboratory proxy to RELATED

A module is implemented by means of a Java interface called IRLABModule. Interface’s methods describe the way the lifecycle of the module is executed (init, start, stop, exit) and how module information is get/set (module information is based in the variable’s paradigm). In this

```

<?xml version="1.0" encoding="UTF-8"?>
<system description="Lego Laboratory representing a Wind Turbine" type="0" name="Wind Turbine">
- <module description="Communication with REST implemented in the module" name="WTModule">
  <param type="string" name="host" value="62.204.199.219">Host running LabView model</param>
  <param type="int" name="port" value="8080">Port</param>
  <var type="double" name="Voltage In" units="V" min="0" max="0.9" initial="0">Voltage In</var>
  <var type="double" name="Current In" units="Amp" min="0" max="0.3" initial="0">Current In</var>
  <var type="double" name="Voltage Out" units="V" min="0" max="0.9" initial="0">Voltage Out</var>
  <var type="double" name="Current Out" units="Amp" min="0" max="0.5" initial="0">Current Out</var>
  <var type="double" name="Joules" units="J" min="0" max="100" initial="0">Joules</var>
  <var type="double" name="Watt In" units="watt" min="0" max="3.0" initial="0">Watt In</var>
  <var type="double" name="Watt Out" units="watt" min="0" max="4.5" initial="0">Watt Out</var>
  <var type="double" name="Wind Power" units="%" min="0" max="100" initial="0">Wind Power</var>
  <implementation type="JAVA" classname="es.uned.scc.rlab.modules.windturbine.WTModule" jarfile="..\\examples\\wind_turbine\\code\\WindTurbineModule.jar">REST communication</implementation>
</module>
+ <module name="VIDEO SERVER module">
- <view description="Show image" name="ImageViewer" classname="es.uned.scc.rlab.views.imageviewer.ImageViewAxis" jarfile="..\\examples\\servidor_video\\code\\ImageViewerAxis.jar"
  helperUrls="..\\examples\\servidor_video\\code\\CameraViewer.jar,..\\examples\\servidor_video\\code\\VideoRecordingsClassLibrary.jar">
  <param type="boolean" name="allowPTZ" value="false">Allow PTZ operations on Axis cam</param>
  <param type="boolean" name="allowRecord" value="false">Allow Recording...</param>
  <use name="image" as="image" module="VIDEO SERVER module"/>
  <use name="command" as="command" module="VIDEO SERVER module"/>
  <use name="recs" as="recs" module="VIDEO SERVER module"/>
  <use name="file_chunk" as="file_chunk" module="VIDEO SERVER module"/>
  <use name="error" as="error" module="VIDEO SERVER module"/>
</view>
- <experiment description="Manual control of Wind Power" name="Simple experiment" concurrentUsers="1" slotTime="20" logging="no" sampleTime="100">
  <duration type="time" time="300"/>
  - <run module="WTModule">
    <interactives show="true" names="Wind Power"/>
    <paint names="Voltage In,Voltage Out" colors="black,red"/>
    <paint names="Current In,Current Out" colors="black,red"/>
    <paint names="Watt In,Watt Out" colors="black,red"/>
    <paint names="Joules" colors="blue"/>
  </run>
  - <run module="VIDEO SERVER module">
    This module is necessary for image viewing
    <interactives show="false,false,false,false" names="command,recs,file_chunk,error"/>
  </run>
  <open view="ImageViewer"/>
</experiment>
<manager name="rpastor"/>
<student name="demo"/>
</system>

```

Figure 5. Configuration file for a RELATED laboratory.

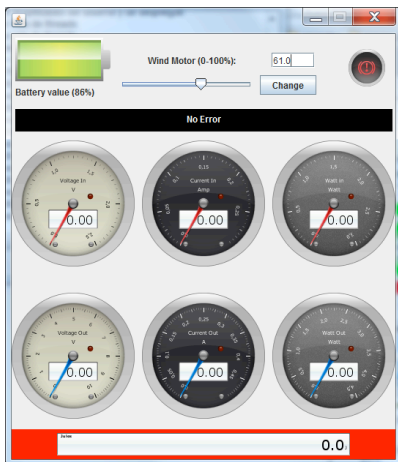


Figure 6. Interaction view for aeolian laboratory.

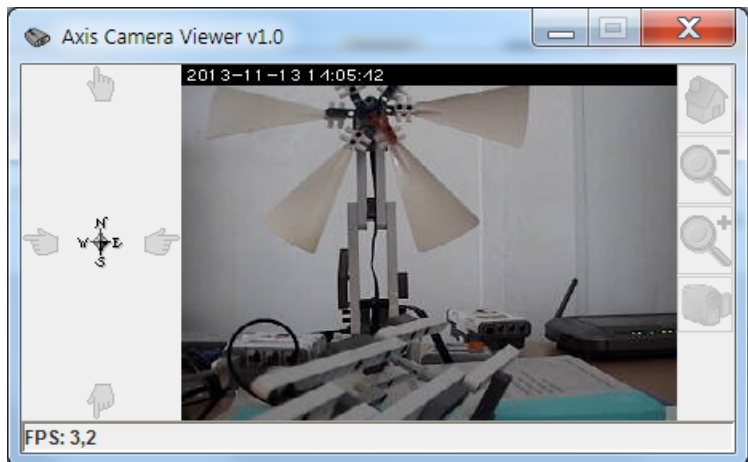


Figure 7. Real time video streaming view for aeolian laboratory.

case, it is implemented as a REST client which gets/sets the module information (wind turbine data) using the REST services defined in LabView. The configuration's lab file showing the module definition is presented in Figure 5.

B. View implementing visual feedback

A view is implemented in a similar way than a module. The interface for a view is called IRLABView. This interface has methods to control the view's visualization and also methods related to get/set data from the experiment running inside the RELATED framework. That is, these methods define the model of information exchange between the views and running experiments. In this case, a reusable view has been declared in the laboratory's configuration file (tag <view>) as it can be seen in Figure 2.

In this case, one specific view is developed to provide the interaction with the laboratory (see Figure 6) and another one (see Figure 7) is reused from previous developments with other laboratories. A RELATED component can be reused in other laboratories in a simple way, being this an additional feature of using the development framework.



IV. AUTOMATIC MANAGEMENT SERVICES

Once the laboratory and its components are defined and developed, it is the time to run the publishing tool provided by RELATED. This tool reads the lab's configuration file (which defines the connector) and consumes the management services to deploy the laboratory. Also, the tool generates the RMI and REST services to provide a multi-device application access. Once this is done, the laboratory can be managed and used in the RELATED Web site.

This way, the integrated laboratory has access to the full services previously commented. The ‘manager role’ can be used to access to the web application in order to get/set information about management services and laboratory (see Figures 8 and 9). It is mandatory to have a <manager> tag in the configuration file. This tag indicates the user name of the laboratory’s manager, so this user will be the main laboratory’s manager.


One of the interesting features provided by the publishing tool is the checking of lab’s availability. The publishing tool sends a “signal” to declare the lab “alive” (using the framework services at a fixed rate time, usually 60 minutes). This way, users can check if the lab is “alive” or not. Also, the last updated signal time is shown to manager, in order they can review this information to check any problem in the laboratory. This information is only available for the lab’s managers (an also the rate in minutes). In Figure 8 is shown the manager view for the status of laboratory (located in the Description tab).

The experiment’s list is located in the “Experiments” tab (see Figure 9). This listing is public to all the users with authorization to access the lab. For every experiment in the list, the name of the experiment is shown and also the role associated to current user viewing the listing. Additionally, a slot time is assigned to the running of every experiment (this is done in the XML definition file for every experiment with a special attribute called ‘slot-Time’, in minutes) which defines the maximum time in minutes available for a lab’s session. So, any user can check how much time is available for using the lab.

Finally, the experiment’s list allows to users gets access to the experiment’s client using the corresponding icons:  for the java client and  for the HTML5 client. These icons are available only if there is no an open session for another user. The Open Sessions label, located near of the top right of the content window, reports about the number of open sessions (in Figure 9 are, of course, zero).

One feature available for all the users is the session’s accounting. The feature allows to users to get information about their work sessions, and also play again their experimental sessions (generated during the work session with the lab). The session’s accounting is located in the “Sessions” tab, as it’s shown in Figure 10.

The user gets information about the total number of sessions (453 in this case) and a paged list of these total sessions. For every session, the start and finish time is presented and also the total amount of time for the session (Work) and the medium time for the experimental sessions carried out during the work session. In the case of only one experimental session, the time showed corresponds with the time used in this experimental session. All the times are in minutes.

Another service provided by RELATED is the experimental data utility (using the  icon at the right part of the particular session information). This tool allows users to view/download data from their experimental sessions and reproduce the experimental session offline. In Figure 11 is shown the main screen for the data utility.

In order to view/download data from an experimental session, the first step is select the particular experimental data session in the main screen. This step start the downloading of data from the RELATED’s data services and

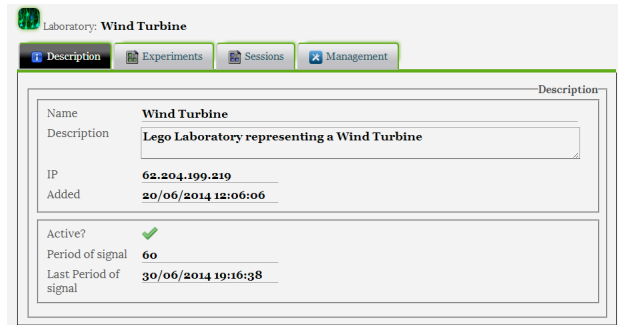


Figure 8. Web interface used to get information about the laboratory.

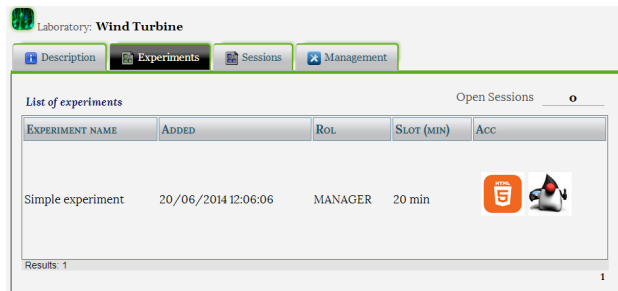


Figure 9. Web interface used to get information about the experiments defined in the laboratory (configuration file of the connector).

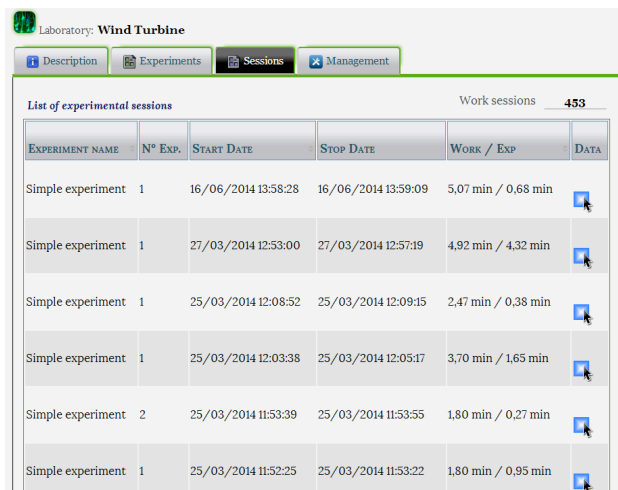


Figure 10. Work sessions with the lab for the current user.

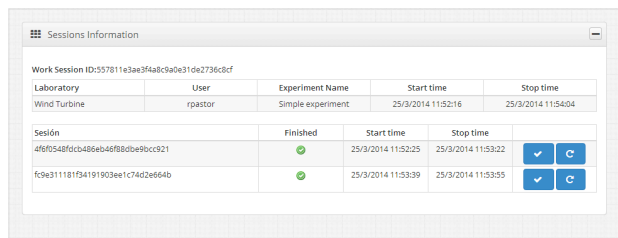


Figure 11. Data utility for experimental session data.

makes available to the data utility. There are two options available as icons once the data is downloaded:

Graphics. It allows to users to paint the data information, grouping this data in a single trend graphic. The user can select any group of variables to be represented and also, download the data using the Save button (actually, the data is available in CSV format). In Figure 12 is

shown the battery charge of the Lego lab during all the experimental session carried out by the user.

Live. It allows the playing of the experimental session. The tool has a play/pause/stop button to control the playing of the session and also, it's possible to use the slider to go to a particular experiment time. The live option can be configured in real time to show graphics (the same defined in the experiment component for the XML definition file) or any view defined in the laboratory. In Figure 13 is presented an example of "live" playing of an experiment for the Lego lab, showing the video view while the experiment is being playing offline.

The before features are available for all the users, but additionally the managers have others features related to the management services described in the first section.

The manager can use the 'Management' tab associated to a lab in order to configure the services. These services include:

Users management. The lab's manager can add/remove registered users to have access to the lab. Also, the rol associated to users in the laboratory can be change to role 'student' (laboratory access) or 'manager' (laboratory management). In Figure 14, a list of actual users (with roles) who have access to the lab is shown. A manager can as well manage the request to the labs. The requests are made by registered users and are included in a list of pending request (every request is sent by email to the manager in order to report him/her). The requests can be approved or denied by a manager, adding a text comment for this action.

Session data management. Using the 'Sessions' tab (see Figure 15), the manager can get global information about work sessions in the lab for authorized users. This information presents the global number of sessions per user, as a rating of use. It is possible to get detailed information about every session for every user (see Figure 10) in the same way as it was described before. Managers have access to all the experimental data for every user of lab, so the managers (for example, teachers) could use this data to evaluate the performance of the user's experiments. The data utility can be a ver usefull tool to check the objectives defined for the experiments carried by the students.

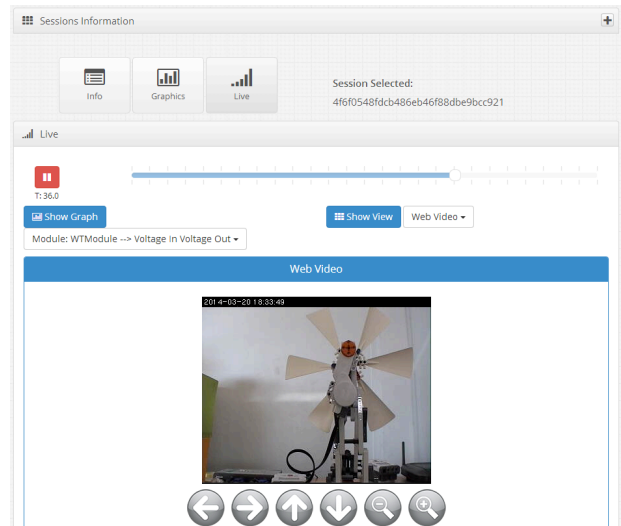


Figure 13. Offline playing of an experimental session data.

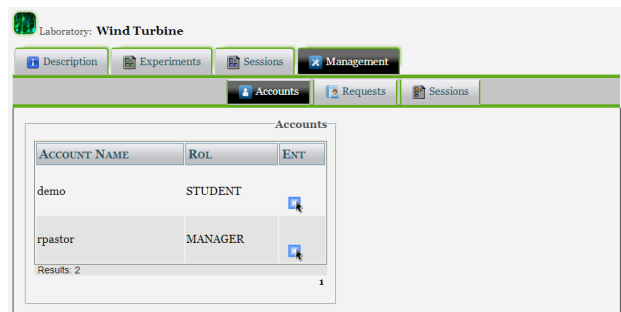


Figure 14. Web interface used to access to user's management services.

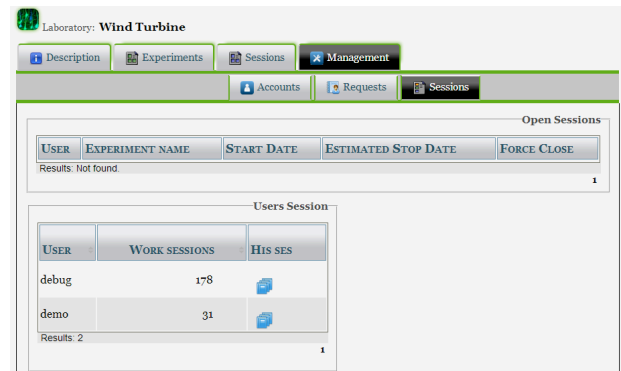


Figure 15. Web interface used to access to session's management services.

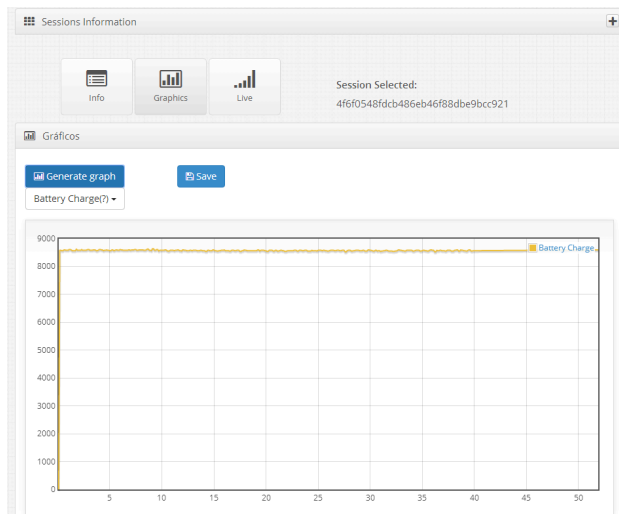


Figure 12. Graphics options for an experimental session data.

Booking services. In order to book a slot time for using a lab, the web application provides a service to authorized users in order to do the bookings. Managers can get statistical information about the total number of bookings (used and not used) and work sessions associated to used bookings. Also, a manager can modify/delete bookings of authorized users.

All these management services can be used in other environments (not only by using a web application based on browser access) integrating them in the corresponding interface.

V. CONCLUSIONS AND FUTURE WORK

The deployment process of a developed laboratory involves not only the development process but the definition of a real interaction environment. This environment must provide some basic services as user's access and authorization, booking procedures, data analytics and usage information. Usually, a laboratory developer is focused on his/her lab, not on the environment, so all these services must be provide in a transparent way by means of standard services.

RELATED's framework provides a scalable architecture which provides a real interaction environment for laboratory users (students and managers). The development of a connector (following the framework guidelines based on reusable components) provides a simple way to integrate any laboratory into the RELATED services, as it is shown in the paper with a detailed example. A lot of functionality is added automatically, so the laboratory's developer only has to focus on what they better know: his/her laboratory.

REFERENCES

- [1] L. Gomes and S. Bogosyan, "Current Trends in Remote Laboratories," *Industrial Electronics, IEEE Transactions on*, vol.56, pp. 4744-4756, 2009. <http://dx.doi.org/10.1109/TIE.2009.2033293>
- [2] Sergio Martin, Gabriel Diaz, Elio Sancristobal, Rosario Gil, Manuel Castro, Juan Peire, "New technology trends in education: Seven years of forecasts and convergence", *Computers & Education*, Volume 57, Issue 3, November 2011, Pages 1893-1906. <http://dx.doi.org/10.1016/j.compedu.2011.04.003>
- [3] A. Nourdine, R. Pastor, G. Vivas, "Limitations of remote laboratories in control engineering education". *International Journal of Online Engineering*, on vol. 6, pp. 31-33, 2010.
- [4] R. Pastor, R. Hernández, S. Ros y M. Castro, "Methodological specification of implementation and development of experimental environments", *Latin-american Learning Technologies Journal, RITA*, on vol. 1, pp. 27-35, 2006.
- [5] <http://lab-app.scc.uned.es:8080/RLABWebSite/>, online. Last access on 11/13/2013.
- [6] <http://phonegap.com/>, online. Last access on 11/13/2013.
- [7] Lego Mindstorms, Web page at <http://mindstorms.lego.com/en-us/default.aspx>. Last access on 11/13/2013.
- [8] Lego Education: Renewable Kit, Web page at http://www.legoeducation.us/eng/product/renewable_energy_add_on_set/2101. Last access on 11/13/2013.
- [9] R. Pastor-Vargas, D. Sánchez, S. Ros, R. Hernández, A. Caminero, L. Tobarra, A. Robles-Gomez and M. Castro, E. San-Cristóbal, G. Diaz, M. Tawfik, "An XML Modular Approach in the Building of Remote Labs By students: a Way to Improve Learning". *International Journal of Online Engineering*, on vol. 9, pp. 5-12, 2013.

AUTHORS

R. Pastor-Vargas , Ll. Tobarra , S. Ros , R. Hernández, and **A. Robles** are with the Communications and Control System Department, Spanish University for Distance Education, Madrid, Spain.

M. Castro is with the Electrical, Electronic and Control Engineering Department, Spanish University for Distance Education, Madrid, Spain.

The Authors would like to acknowledge the support of the following European Union projects: RIPLECS (517836-LLP-1-2011-1-ES-ERASMUS-ESMO), PAC (517742-LLP-1-2011- 1-BG-ERASMUS-ECUE), EMTM (2011-1-PL1-LEO05- 19883), MUREE (530332-TEMPUS-1-2012-1-JO-TEMPUSJPCR), and Go-Lab (FP7-ICT-2011-8/317601). Furthermore, we thank Spanish Ministry of Science and Innovation for the Project TIN2008-06083-C03/TSI and the Region of Madrid for the support of E-Madrid Network of Excellence (S2009/TIC-1650). This article is an extended and modified version of a paper presented at the EDUCON2014 conference held at the Military Museum and Cultural Center, Harbiye, Istanbul, Turkey, 3-5 April 2014. Submitted 30 June 2014. Published as resubmitted by the authors 25 October 2014.