An Open Source Multiplatform Virtual Laboratory for Engineering Education

http://dx.doi.org/10.3991/ijoe.v8iS3.2267

R.M. Fernández-Cantí, J. A. Lázaro-Villa, S. Zarza-Sánchez and A. Villar-Zafra
Signal Theory and Communications Department (TSC), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

***Abstract*—An autonomous and multiplatform virtual laboratory for educational purposes is presented. The implemented platform includes a server with a SSH (Secure SHell) connection and a separated repository containing the virtual experiments. The programming of the experiments is implemented in Java language based tool, the EJS (Easy Java Simulation) and uses an external computation engine, for example Matlab. The virtual laboratory provides control system experiments at University level. Two application examples are described, namely, a magnetic levitator and an inverted pendulum-cart system. The virtual laboratory has been successfully used for education and training of Electronics Engineering students. A discussion of the results of this e-learning experience is also presented.**

***Index Terms*—Virtual laboratory, open source software, control experiment, e-learning**

I. INTRODUCTION

Virtual laboratories constitute a useful teaching and learning environment where the physical system is virtualized by means of a series of simulations. The advantages and drawbacks of this educational tool have been studied, especially in the frame of control systems. See for instance [1-4] and the references therein.

The particular interest of virtual laboratories in the Control Engineering education is justified by the usually high cost of the purchase and maintenance of the equipment. Also, to give a good service to public education, there should be acquired several units of the hardware, which further increases the economic investment.

Another shortcoming, when performing the control experiments with the physical equipment, is the time necessary to carry out the practices, since many students may spend too much time in calculations and initial tests.

There are several approaches to overcome these two problems. A first solution is to operate remote laboratories [5], for instance by means of proprietary software as Labview. Remote laboratories require lower investment in hardware since only one or two units of the experimental equipment are required. This solution implies the programming of the front-end, and the development of a web application for the remote laboratory reservations. However, in this approach, the physical equipment has to be powered on during large periods of time, maybe including nights and holidays. Therefore, to ensure the availability and correct performance of the hardware, hard tasks of maintenance are needed.

A second solution is the development of virtual laboratories where a graphical environment is used to emulate the experiments to be performed with the physical hardware. This is a good choice even in the cases when the hardware is still acquired, since, this way, students can become familiar with the equipment before starting the experiment, thus saving significant training time in the physical laboratory [6]. Moreover, this approach is especially interesting in the cases when the hardware counterpart does not exist, because one can develop practices covering from classical benchmarking problems to complex experiments that would require expensive or simply unaffordable hardware [7].

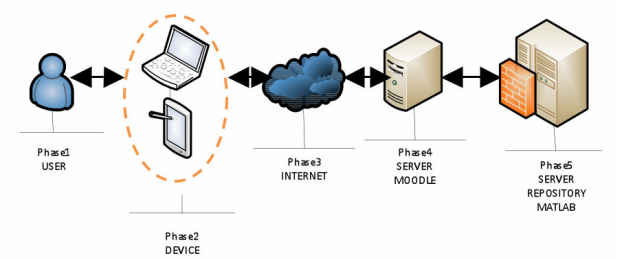

Figure 1. Phases of the multiplatform virtual laboratory

Focusing in virtual laboratories, an important issue is that the virtual laboratory must be available in Internet for all the students at any time. Another goal is that the virtual laboratory should be *multiplatform*, since students may use different work environments such as Unix and so on. In this paper, we take advantage of the adequacy of the Java programming language to achieve the required flexibility and the ability to integrate into any web platform.

Another important requirement is the *autonomy*, meaning that the virtual laboratory should not depend on any particular virtual campus software. This permits sharing or interchanging the virtual laboratory modules with other universities, even when they will be probably running a different virtual campus software. This *autonomy* has been accomplished by using separated servers for the experiments repository and the virtual campus.

The last relevant specification is the *modularity*. It means that the implementation must allow the communication between the different parts of the virtual laboratory, but in turn these parts must be completely independent one from each other.

We proceed to describe in detail the development of two examples of virtual laboratories fulfilling all those key specifications. This paper is organized as follows. In Section II, the different phases involved in the multiplatform virtual laboratory are described. Section III lists the system requirements. In Section IV the different blocks that implement the virtual laboratory are explained. Section V describes two of the available control experi-

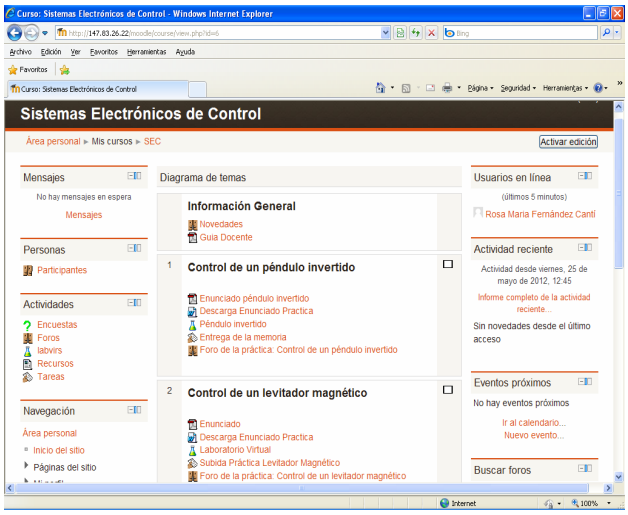

Figure 2. iLabVir Moodle

ments. In Section VI the results obtained with the students are presented. Finally, Section VII summarizes the conclusions and future research lines.

II. PHASES OF THE VIRTUAL LABORATORY

Figure 1 illustrates the implemented multiplatform virtual laboratory. Five different phases can be identified:

Phase 1: User

One of the main advantages for the user is the availability of the laboratory 24 hours a day and 365 days a year for accessing to the laboratory practices. It is also important that there is no limitation in the repeatability of the experiments (unless the teacher decides otherwise).

Phase 2: Device

Since the laboratory is programmed in Java, students can access it from any terminal and anywhere, provided that the internet browser has a Java client, which is the most common case.

Phase 3: The Internet

Nowadays, it is possible to access the cloud from almost everywhere, since the nearly total service is guaranteed.

Phase 4: Moodle server

The virtual laboratory presented in this paper is available from the iLabVir Moodle of the UPC (see Figure 2), but it was designed to allow access from any other virtual campus from any other University, thanks to the installation of a specific module developed by the (*Grup d'Interès Laboratoris Virtuals i Remots*) GilabVir group of the (*Universitat Politècnica de Catalunya*) UPC. The Moodle server is the one that provides the students the possibility to conduct remote experiments (real or simulated).

Phase 5: Repository server and computation engine

The repository server is separated from the Moodle server in order to give service to more than one virtual campus. This makes possible to have a computation engine in the same machine working as repository server. Our computation engine is Matlab but any other, including open source computation engines as e.g. Octave, can be used. Finally, a firewall is installed between the repository server and the Moodle server to get control of the different virtual campus that will aim to access the experiments.

III. SYSTEM REQUIREMENTS

On one hand, from the virtual laboratory users' viewpoint, the main requirements include issues such as: the availability, ease of usage, and repeatability of the experiments. On the other hand, from the designer viewpoint that we are considering here, the specifications of most interest are: autonomy, multiplatform, and modularity.

A. Autonomy

Implementing an autonomous system means that the virtual laboratory is totally independent of the particular virtual campus. In this way, the experiments can be accessed from any other University who could be interested in their exploitation. Even more, the system can be developed as an Open Educational Resource (OER) for benchmarking comparison among different working groups. Moreover, the autonomy requirement enhances the security of the system, which is an important issue when being accessed from different campus.

B. Cross-platform

Regarding the multiplatform specification, the first step is to develop the application in a programming language such that any web browser could successfully perform the experiment. Thus, any user with any web connection can access his particular virtual campus and run the experiment. A language that fully satisfies these requirements is Java.

C. Modularity

Finally, this specification leads to the implementation of a distributed system in separate blocks. This way new updates or modifications do not imply changing any of the other blocks.

IV. ARQUITECTURE OF THE VIRTUAL LABORATORY

In this section we describe the different blocks that constitute our virtual laboratory and explain how the communication between the different parts is carried out.

Figure 3 shows the different constituting blocks. These are the following:

Block 1. Virtual Campus Environment

The virtual laboratory presented in this paper is installed in the Moodle v2.0 running a one of the servers of the ETSETB (*Escola Tècnica d'Enginyers de Telecomunicació de Barcelona*) at UPC.

As mentioned before, to operate the virtual experiments it is necessary to install a module that has been developed by the GiLabVir group of the RIMA project (*Recerca i Innovació en Metodologies d'Aprenentatge*). This module is Open Source and it is available in [8].

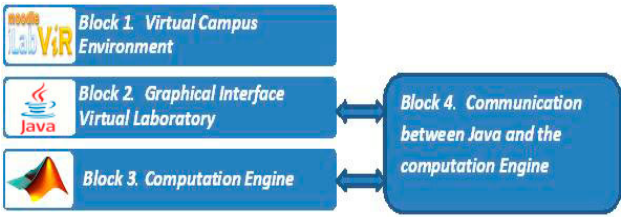

Figure 3. Block diagram of the virtual laboratory

The main feature of this module is that it allows having a repository of virtual experiments separated of the Moodle server. This allows the experiments not to be embedded in the virtual campus itself, and thus they can be accessed from any other virtual campus. The experiments are compiled in Java and embedded to HTML.

Also noteworthy is that we have this repository in a Xen virtual machine, in which it is also installed the software used as computation engine (Matlab in our case). The computation engine is the responsible of most of the computational consumption. Since it is installed in the same repository than the experiments, this means that the experiments do not spend virtual campus server resources.

Regarding the security issues, the communication between the Moodle server and the repository server is established by configuring the communication ports of both servers and the IP firewall between them two. This implies that the students do not have direct access to the experiment's repository. The only entity that can access the repository is the Moodle server, thus only playing the role of a frame for the experiment. By means of the editing tools of the Moodle virtual campus, the professor can establish which experiments will be available to the students and which will not.

Block 2. Graphical Interface Virtual Laboratory

Our application is developed in Java language; in particular the GUI (*Graphical User Interface*) of the experiments has been implemented using the Easy Java Simulation (EJS) package [9-10]. It is an open source tool specially designed for the implementation of Java applications that require a graphical environment to develop the simulation. As a result of the work with EJS, it is obtained a ".jar" file. This file is the compilation of the EJS project and it can be embedded in a HTML page, thus obtaining the typical structure of a Java Applet.

Block 3. Computation engine

However, depending on the complexity of the experiment to be virtualized, it might be interesting to use an external computation engine and/or mathematical software. Since the implemented experiments correspond to control systems we use several functions of the *Control Systems Toolbox* of Matlab [11]. A relevant advantage of running the computation engine in the repository server, is to avoid the use the Ethernet network to communicate the Java module with computation engine. This reduces considerably the delay between the call to the experiment from the student computer and the experiment execution.

However, even though in the configuration presented in this paper they are on the same machine, they also could be mounted in different machines, and even using remote supercomputers for extra complex calculations, if required.

Finally, and following the modularity requirement, EJS is used only to develop the graphical front-end (including the animation effects) while the Matlab m-file is the responsible of generating the different signals involved in the experiments. Hence, the m-file contains the dynamic equations that describe the behaviour of the system; while the EJS layout is the one that allows entering the experiment parameters (excitation signal, controller configuration and coefficients, initial conditions, etc.) and presents the results (animation effects, response plots).

This arrangement facilitates the development of new practices and experiments since, if the professor requires: changing the behaviour of the experiment; including more complex dynamic equations of the model; or generate new practices, it is only required to modify the m-files.

Block 4. Communication between Java and the computation engine

One important issue of the virtual laboratory is how to communicate Java with the computation engine in a simple and non-expensive way, independently of having both Java and computation engine running at the same machine or not. In our case we have solved this problem by establishing the communication between Java and the computation engine through SSH (Secure SHell). SSH is a network protocol for secure data communication among two networked entities, which guarantees that the information exchange remains completely safe. To carry out the SSH connection it is used the open source library of Java: JSch [12]. The use of JSch provides the necessary Java classes and objects to implement a SSH connection in Java language. This library is included in the ". jar" compilation, so that the connection between the repository and the computation engine is completely secure.

Finally, if the Java Applet uses an external resource (e.g. a connection to a database or to a hard disk), it is needed a digital signature of the ".jar" file.

Data flow

Figure 4 shows the different types of data flow between the different entities that constitute our virtual laboratory.

The arrow labelled as "1" represents the data flow between the virtual campus and the repository. It is unidirectional, i.e. data is transmitted only from first to the second element. The data flow corresponds to the user's connection to the list of experiments stored in the repository.

The second arrow, "2", represents the data flow between the repository and the virtual lab. Again, it is unidirectional, i.e., data is transmitted only from second to the third element, since these data represent the choice of the experiment to be simulated.

The arrow labelled as "3" represents the data flow between the Laboratory and the Java Applet. Data is transmitted among the third and fourth block bidirectionally. If the information travels from the Laboratory to the Applet Java, data flow will contain specific parameters of the simulation. Otherwise, if the information travels from the Java Applet to the Laboratory, data flow will contain information related to the graphic simulation.

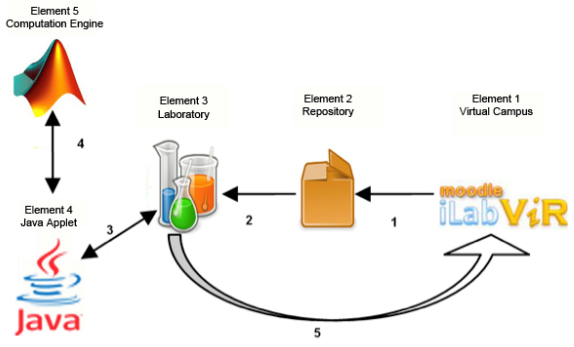

Figure 4. Data flow in the virtual laboratory

The fourth arrow, "4", represents the data flow between the Java Applet and the Computation Engine and it is bidirectional: data is shared between the third and fourth element, sending to the Laboratory (through Java Applet) each numerical value to be simulated.

Finally, the last arrow, "5", represents our scheme, the transmission between the data obtained from the simulation in the Java Applet and the Virtual Campus which is what is displayed to the students.

V. CONTROL EXPERIMENTS

To illustrate the experiments available from this multiplatform virtual laboratory, consider the layouts shown in Figure 6 and Figure 7.

A. *Magnetic Levitator*

The first example is a virtual magnetic levitator. In this case the virtual experiments emulate a real educational equipment, a Magnetic Levitator (MagLev) from Feedback Instruments Ltd., model 33-210 [13]. The physical parameters in the virtual model correspond to the real MagLev characteristics. Also, to ensure that the behaviour of the virtual levitator matches the real one, nonlinear effects have been included in the programming of the *m*-file.

Magnetic levitation is a very interesting process in the control engineering teaching since it is unstable and nonlinear, whereas the analytic model that describes its dynamical behaviour is relatively easy to obtain. For this reason this problem can be addressed to both undergraduate students and control researchers.

In the case of the magnetic levitator, different experiments can be performed and, in all of them, the student can see the ball movements in real-time. The objective of the control system is to maintain the ball in a specified distance from the electromagnet. If the control fails the ball either may fall or get stuck to the magnet. The students have to select a control strategy and design the controller. Controllers can be easily tested and the students can see if their designs are correct or not, and they can improve them if necessary.

The main concepts illustrated by the magnetic levitator virtual experiment include PID (Proportional-Integral-Derivative) controllers, root locus controller design, and ITAE (Integral Time Absolute Error) optimal controller design.

B. *Inverted pendulum-cart system*

The inverted pendulum-cart shown in Figure 7 is another classical experiment in control education. In our case, no physical counterpart for this experiment exists. For this reason we could select the parameters and control configuration in order to better illustrate the course concepts.

Again, students and researchers can select different control strategies (series controller, state space controller), adjust the parameters of the controller and test the design. In the students' case, several difficulty levels are posed, related to a limit time to stabilize the plant, so that the selection of fast poles and the behaviour trade-offs become important issues of the problem. As in the magnetic levitator case, the student can visualize the dynamical behaviour by both graphical results and the real time movements of the pendulum and cart.

Figure 5. Magnetic levitator from Feedback Instruments

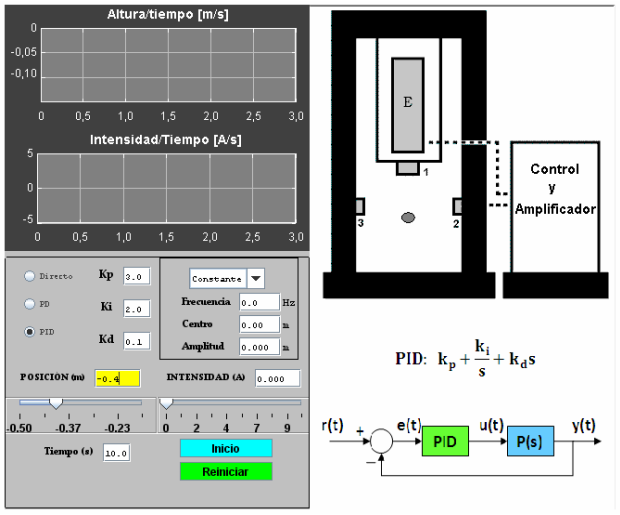

Figure 6. Magnetic levitator

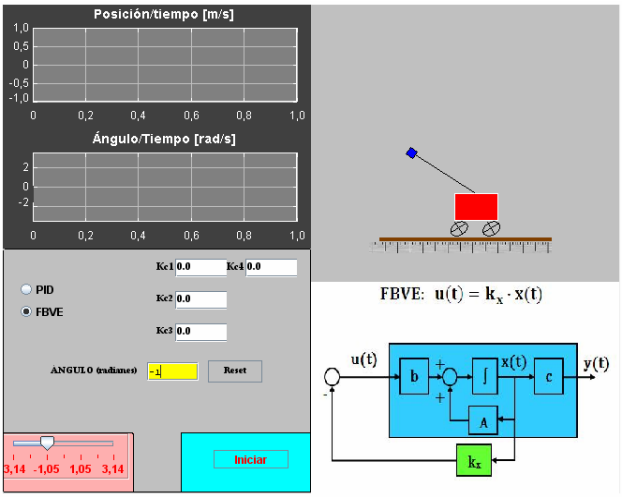

Figure 7. Inverted pendulum-cart system

All the control experiments allow the students to download files with the numerical results to further analysis. Additionally, they can capture the graphics and record video (animated GIF format) of the system behaviour.

VI. STUDENT ASSESSMENT

This two developed control experiments have been tested by different groups of students of the subject Electronic Control Systems, corresponding to the 7th semester of the Electronic Engineering degree, during the academic years 2010/2011 and 2011/2012.

One of the main achievements has been that the students were able to carry out the experiments from any terminal at any time. It has been also a very important achievement that once they were performing the physical laboratory practice, they were able to focus on checking the results obtained in the simulator (significantly reducing, about 80%, the time of occupation of the physical laboratory set-up).

The use of open source tools for the multiplatform virtual laboratory has provided several advantages. It can be counted among them, a significant reduction of the developing time of the laboratory, as many useful open source tools, for example the SSH connection between the graphical environment and the calculation engine, are thankfully, already available.

A questionnaire was distributed to the students in order to draw conclusions from the experience. The main conclusion is that students are very satisfied with the Virtual Laboratory. From the perspective of professors and lecturers, the experience has been also positive because the academic results of the students have improved.

Fig. 8 shows the averages obtained in the questionnaire for the most relevant questions, about their preferences for the most adequate way to perform the laboratory practices.

A. Strong points

Most students (85.71%) think that the virtual laboratory improves the learning process of the subject. Also, the same percentage of students (85.71%) thinks that the virtual practices have been useful to understand and consolidate concepts. And all the students think that the iLabVir platform is a useful resource.

B. Weak points

Nevertheless the majority of the students (71.43%) missed the presence of the professor to answer their questions. However, it is worth to mention that, even though a queries forum was available, none of the students made use of it. This is a point that needs much future work.

C. Practices typology

Regarding the practices typology (Fig. 8), physical practices at the laboratory are still preferred (average = 9). But virtual and remote laboratories are well qualified (average = 8), while simulation practices at the laboratory (average = 7.83) or at home (average = 6.5) are the least preferred by the students.

D. Comparison Magnetic Levitator/Inverted Pendulum

Both practices are well graded. The global grade for the magnetic levitator is 8.33 while for the inverted pendulum is 8.25 as shown in Fig. 9.

E. Comparison to real equipment

The students have done physical practices in the laboratory and virtual practices based on the same equipment. When asked to compare the performance of both practices types, the 66.67% thinks that the virtual practice is less accurate than the physical one, while the 33.33% considers that they are equivalent. In any case, the virtual practice was evaluated as a good first platform for better understanding the practice.

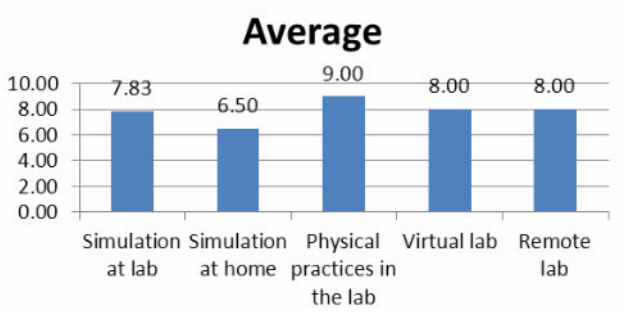

Figure 8. Average values of the prefered format of the pratice work for the students

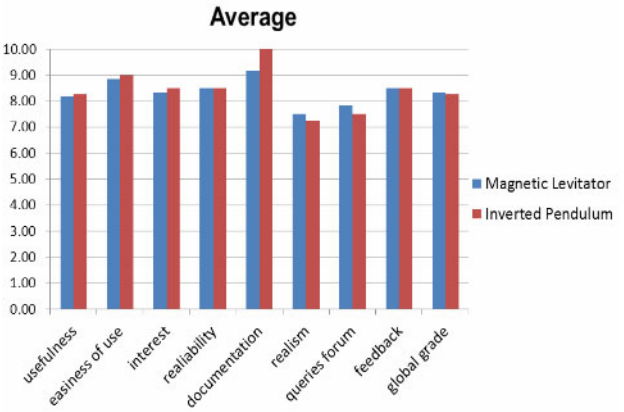

Figure 9. Practices average

F. Students' comments:

The most positive aspects are the flexibility (at home, any time, etc.), the unlimited time to do the practice, and the possibility to check the controller in a visual way (seeing the movements of the ball or the pendulum-cart).

VII. CONCLUSIONS AND FUTURE RESEARCH LINES

This paper describes an open source autonomous multiplatform modular virtual laboratory including two specific examples of control system experiments. The repository of experiments and the calculation engine run at in a different server from the one used as virtual campus' server. This allows: increasing the speed of response; overall safety; reduction of the use of server resources; and allows accessing to different virtual campuses and/or, allows accessing to different educational institutions to exploit our experiments.

The implementation presented here was performed using the EJS tool and Matlab. A fully open source implementation can be also done by just using an open source computational engine as for example Octave. The modularity of the platform facilitates the rapid development of new practices and applications, including experiments of other subjects.

The popularity of virtual laboratory among both students and professors encourages further research in for example: extending the capabilities of the display screen; virtualizing complex applications; editing a comprehensive manual of the virtual laboratory platform; promoting further experiments; and implementing a forum for consultation between professors and teamwork students.

ACKNOWLEDGMENT

The authors thank to the *Institut de Ciències de l'Educació* (ICE) and the *Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona* (ETSETB) of the *Universitat Politècnica de Catalunya* (UPC) for their support to this teaching innovation initiative.

This platform offers to the students the opportunity to conduct remote experiments (real or simulated) and is being developed by the GilabVir group (*Grup d'Interés en Laboratoris Virtuals i Remots*) which is part of the RIMA project (*Recerca i Innovació en Metodologies d'Aprenentatge*) of the UPC.

The authors thank also to the team of Informatics Systems of the Theory of Signal and Communication for their ideas and support in developing this project.

This work was partially supported by the Spanish MICINN/FEDER project TEC2008-01887.

REFERENCES

[1] S. Dormido, "Control Learning: Present and Future" *Annual Reviews in Control*, **28**(1): 115-136, 2004 http://dx.doi.org/10.1016/j.arcontrol.2003.12.002

[2] J.L. Guzman, M. Berenguel, F. Rodríguez , Laboratorio virtual y remoto para el control de clima en una maqueta de un invernadero, Universidad de Almería, http://greenhouse.ual.es

[3] F. A. Candelas, J. Sánchez, "Recursos didácticos basados en Internet para el apoyo a la enseñanza de materias del área de ingeniería de sistemas y automática" *Revista Iberoamericana de Automática e Informática Industrial*, **2**(2): 93–101, 2005.

[4] A. Cardoso, M. Vieira, and P. Gil, "A Remote and Virtual Lab with Experiments for Secondary Education, Engineering and Lifelong Learning Courses", *International Journal of Online Engineering (iJOE)*, **8**(2):49-54, 2012.

[5] I. Calvo, E. Zulueta, F. Oterino, J.M. Lopez-Guede , "A Remote Laboratory for a Basic Course on Control Engineering", *International Journal of Online Engineering (iJOE)*, **5**(3): 8-13 2009.

[6] http://www.aurova.ua.es/robolab/index.html

[7] A. García, N. Duro, R. Dormido and S. Dormido "The Reaction Whell Pendulum: An Interactive Virtual Laboratory for Control Education" , *International Journal of Online Engineering (iJOE)*, **6**(3): 54-58 2010.

[8] http://moodlelabs.svn.sourceforge.net/

[9] M. Domínguez, "Simulaciones interactivas en EASY JAVA de equipo Feedback MS-150", XXXI Jornadas de Automáticas, 2010.

[10] F. Esquembre, "Easy Java Simulations: a software tool to create scientific simulations in Java", Computer Physics Comm., 156(2): 199–204, 2004. http://dx.doi.org/10.1016/S0010-4655(03)00440-5

[11] *scientific simulations in Java*", Computer Physics Comm., **156**(2): 199–204, 2004.

[12] Control Systems Toolbox for Matlab. User's Guide, the Mathworks 2011

[13] http://www.jcraft.com/jsch/

[14] Magnetic Levitation Installation & Comissioning, 33-9421C, Feedback Instruments Ltd., 2006.

AUTHORS

R. M. Fernández Cantí is with the Department of Signal Theory and Communications, Campus Nord UPC, Edifici D4, 08034 Barcelona, Spain (e-mail: rfernandez@tsc.upc.edu).

J. A. Lázaro Villa is with the Department of Signal Theory and Communications, Campus Nord UPC, Edifici D4, 08034 Barcelona, Spain (e-mail: jose.lazaro@tsc.upc.edu).

S. Zarza Sánchez is with the Department of Signal Theory and Communications, Campus Nord UPC, Edifici D4, 08034 Barcelona, Spain (e-mail: szarza@tsc.upc.edu).

A. Villar Zafra is with the Department of Signal Theory and Communications, Campus Nord UPC, Edifici D4, 08034 Barcelona, Spain (e-mail: aitor.villar@gmail.com).

This article is an extended and modified version of a paper presented at the International Conference on Remote Engineering & Virtual Instrumentation (REV2012), held at University of Deusto, Bilbao, Spain, July 4-6, 2012. Received 13 September 2012. Published as resubmitted by the authors 14 November 2012.