Towards a Canonical Software Architecture for Multi-Device WebLabs

Javier García-Zubía
Faculty of Engineering
University of Deusto
Apdo. 1, 48080 Bilbao, Spain
zubia@eside.deusto.es

Diego López-de-Ipiña
Faculty of Engineering
University of Deusto
Apdo. 1, 48080 Bilbao, Spain
dipina@eside.deusto.es

Pablo Orduña
Faculty of Engineering
University of Deusto
Apdo. 1, 48080 Bilbao, Spain
pablo@ordunya.com

***Abstract* – Traditionally the focus on WebLab design has been placed on the hardware side, i.e. enabling data and program transfer between a PC remotely accessible through TCP/IP and its attached controllable/programmable device. Little attention has been paid to the other communication segment going from the controlling PC (WebLab server) to the remote users' PCs, since this has been regarded as a "solved software problem". Consequently, aspects such as security, scalability, accessibility, user friendliness, or the possibility of collaborative work in WebLabs have often been disregarded. This situation may be resolved if a serious effort is placed on the definition of a proper distributed software architecture for WebLabs. In this paper, we describe such ideal software architecture, resulted from an iterative process seeking a web-based, secure, scalable, multi-user, multi-device WebLab.**

I. INTRODUCTION

The concept of WebLab has been around since the early nineties. Its development is widespread in laboratories of analog [1] and digital [2] electronics, programmable logic [3] or process control [4]. We can encounter good examples of WebLabs in different countries: USA [5], Colombia [6], Spain [3,7,8], Italy [9], Corea [4] and so forth.

A WebLab can be studied from different points of view:

- *Didactical*: didactic goals, quality and suitability of the WebLab, didactic platform integration, etc. [9,10,11,12]
- *Hardware technology*: cards, electronic prototypes, data acquisition, etc. [6,10]
- *Software technology*: client/server design, security, integration, etc. [10]
- *Software development platforms*: Web-Services [4], LabView [2,13], C applications [14], JAVA [15], Matlab [9], etc.
- *Communication*: through RS-232[16], TCP/IP[12], XML[17], etc.
- *Social*: international solidarity, disabled people adaptation, etc. [3].

The recent popularity of the WebLab concept, its different approaches and the abundant existing bibliography only prove the great activity on a field which is called to represent a cornerstone of worldwide engineering education.

WebLabs are traditionally designed by electronic and control engineers who naturally tend to place a major attention on the hardware side of the system. They usually follow a three step process: (1) choose a programmable device, (2) attach it to a server, accessible through the web or simply a TCP/IP socket, and (3) design a simple protocol to record programs in the remote device, send inputs and receive outputs. Unfortunately, the software side involved in the last two steps is often paid too little attention and hence a poor usage of the remotely available programmable hardware devices is achieved. We believe that better software architectures for WebLabs should lead us to more user-friendly, cost-efficient, reliable and scalable WebLabs.

In this paper we illustrate the successive improvements applied to the software architecture of our WebLab (see Fig. 1) to progress from a 1-1-1 (1 user, 1 server, 1 programmable device) to an N-1-N WebLab. Our final goal is to achieve a reliable, cost-efficient, user-friendly, collaborative and scalable WebLab. Progressively we examine the advantages and disadvantages of the different iterations of our software architecture to conclude with a definition of what we consider may be the canonical architecture for WebLabs.

The structure of the paper is as follows. In section 2 we review the WebLab concept. Section 3 shows the different evolutions of the software architecture of our WebLab and proposes a new architectural model for WebLabs which will allow among other things collaborative work. Finally, section 4 draws some conclusions.

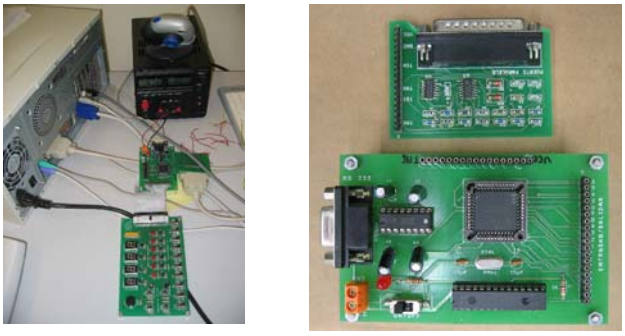

Fig. 1. University of Deusto's WebLab-PLD

II. WEBLAB OVERVIEW

A remote WebLab, from now on WebLab, is a hardware/software system which allows remote control and monitoring of an electronic programmable device via web. WebLabs are usually employed in universities to enable students to access the lab resources from anywhere with an Internet connection. In a nutshell, a WebLab is a ubiquitous lab. A WebLab allows for:

- the adjustment of electronic devices on which a task is deployed
- the programming of the control device if there is one,
- the induction of inputs to the programmable device,

- the personalisation and configuration of the programmable device,
- the data acquisition by means of a DAQ card or virtual equipment,
- monitoring the evolution of an experiment by means of a WebCam or data capturing program,
- the WebLab administration: login, password, security, etc.
- learning with material complementary to the assignment.

Thus, an analog electronics WebLab [1] allows a student to configure using hardware or software the electronic circuit to analyse, adjust the function generator and oscilloscope or observe the results in the PC's terminal. On the other hand, a Programmable Logic Device (PLD)-based [12] WebLab can be used to remotely program an integrated circuit (or any other programmable device CPLD or FPGA device), induce inputs into it, or monitor the device outputs by means of WebCams or data captures exported to files.

In the first example, the WebLab configures the electronic circuits with some limitations whereas in the second, the inputs and outputs are always the same. In the latter example, the WebLab allows the programming of a device, whereas in the former there is no device to program. Moreover, in the analog electronics example there is a need for an oscilloscope or a data acquisition card, whereas in the PLD case the outputs may be LEDs or moving engines viewed from a WebCam.

The usual components of a WebLab are:

- A server machine (WebLab server) together with the software that exports the service to students. The best solution is to offer a web front-end to control the remote device. However, we often encounter a simple socket server which accepts commands and inputs and sends outputs through a TCP/IP channel.
- A PC where the client software is run, which allows the student to connect to the WebLab server and complete an assignment remotely. The device from which the user interacts could potentially also take the form of a PDA or mobile phone capable of communicating through HTTP.
- The piece of hardware over which the assignment is defined: an analog filter, a PLD with switches and LEDs, a robot, and so forth.
- Traditional electronic instrumentation to excite inputs and analyze the outputs: function generator, oscilloscope, data acquisition cards and so on. The instrumentation hardware requires of a communication means and the associated software for remote configuration, such as GPIB, LabView, RS-232, etc.
- A hardware/software equipment to excite logic inputs like switches: PIC, RS-232, etc.
- A WebCam and perhaps a microphone to receive live feedback of the experiment evolution.

From the above we can observe that PCs, software applications, hardware cards, networks and communication protocols are needed in the construction of a WebLab. An unfortunate trend observed is that each available product seems to suggest its own communication protocol. Therefore, there is an opportunity to standardise the interface to WebLabs by process interaction standards such as Web Services.

III. SOFTWARE ARCHITECTURE EVOLUTION

The software architecture of our WebLab has gone through the following four iterations:

1. Socket and Applet-based Proprietary solution [3].
2. Web-based solution [12].
3. AJAX-based Web solution [12].
4. MicroServer-based AJAX-based Web solution.

As a result of this iterative process we have envisioned the architecture of a next-generation WebLab which will allow mainstream access to WebLabs worldwide, we have called this architectural concept "Universal WebLabs".

A. Socket and Applet-based Proprietary Solution

Fig. 2. shows the first iteration of the software architecture we devised for our WebLab. A proprietary standalone client implemented in C communicated using the SDLnet library with the WebLab server. This server was in charge of communication through RS-232 with a PIC acting as bridge of a programmable PLD. In parallel to the command-line application remotely controlling the programmable device, an ActiveWebCam applet by PySoft was used to observe in real-time the status of the hardware being programmed. The WebLab server kept user-access and usage control. Each time only one user could be accessing the remote device. This was a prototype only used by lecturers and guests.

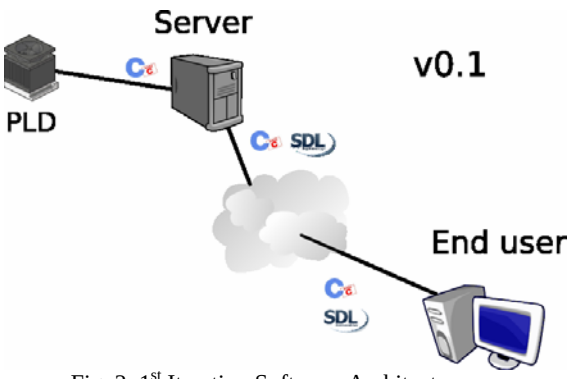

Fig. 2. 1st Iteration Software Architecture

The main drawbacks of this solution were:

- *Interoperability issues*. Both the client and server solutions could only be run on the MicroSoft Windows platform.
- *User-friendliness issues*. The users needed to start two independent applications, the controlling standalone C-based application and the Java viewing applet. Moreover, the controlling client offered a primitive command-interface through which FTP-like commands could be used to upload new logic to the programmable device, induce inputs and read outputs.

- *Security issues*. On the server side, the firewall has to be configured to enable traffic offer two non well-known ports rather than using already opened ports such as 80 for HTTP. In addition, there was not built-in user access control. Consequently, there was fear to open the WebLab to the public, and it was only used for demonstration purposes within the University's LAN.

B. Web-based Solution

Fig. 3 shows the second iteration of our software architecture. Here, the server-side was composed of three elements: a) an Apache web server hosting a webpage with the controlling and viewing applets, b) a Python server which communicates though the serial port with the PIC that controls a PLD and c) a webcam server broadcasting the images captured. In this iteration, the client application was totally based in Java, accessible through a web browser with a pre-installed Java plug-in. The controlling applet communicated with the controlling server, whereas the viewing applet connected with the webcam server.

The WebLab server's logic was updated to keep user-access and usage control. Each time only one user could be accessing the remote device for a maximum period of time (120 secs). The only requirement imposed to students was to use a browser with a pre-installed Java plug-in.

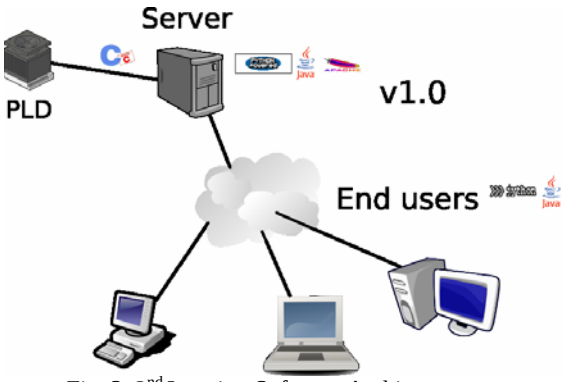

Fig. 3. 2nd Iteration Software Architecture

This solution still presented some issues regarding user-friendliness and security:

- *User-friendliness issues*. We had two independent applets executing on the same webpage. The download of the applets took some time and required the user browsers to have installed the Java plug-in.
- *Security issues*. A security alert was raised every time the user downloaded the controlling applet since this required access to the file system of the user in order to upload a file with the new programming logic. Moreover, we still had to keep opened two ports in the firewall: one for the webcam server and another for the controlling server. This supposed a hassle for the firewall maintenance.

With this iteration, we finally gave access to students of the "Programmable Logic" module to access the system from an Internet browser outside the University.

C. AJAX-based Web Solution

The third iteration of our WebLab, currently in use, is shown in Fig. 4. A single client application shown in the user's browser communicates with the server through HTTP. We now have a web-based firewall-safe system programmed with AJAX (Asynchronous JavaScript and XML). The main benefit of this web development approach is that it only uses tools readily available on any web browser, i.e. XHTML, DOM and JavaScript. Therefore, no plug-in installations are required on the users' browsers.

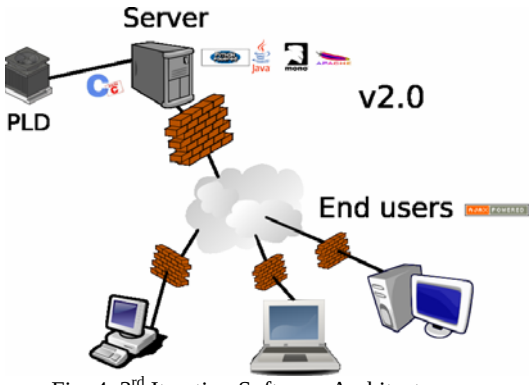

Fig. 4. 3rd Iteration Software Architecture

The server side is composed of the following elements: a) a Java server continuously capturing images from a WebCam and saving them into a directory exported by an Apache web server, b) a Python server controlling the communication with the programmable device and c) an ASP.NET application based on Mono and running on the Apache Web Server offering a web-service interface to client applications.

The client application is now a pure HTML/JavaScript solution which follows the AJAX web interaction model, i.e. rather than changing the full content of a page every time there is an interaction between the client and server, only the portion of the page affected by the interaction is modified. This technology is being applied successfully to sophisticated web applications such a Gmail, Google Maps or Flicr. The key of this technology is that the control commands, responses and images are transmitted asynchronously, without interrupting the user interaction with the system, by means of the JavaScript's XmlHttpRequest object [18].

The data exchanged between the AJAX client and the Mono-based server is through the standard Web Services transport protocol, namely SOAP. The Mono-based server delegates the arriving web-service method invocations to the Python server controlling the programmable device. The latest captured image is continuously being retrieved through HTTP by the AJAX-based client by accessing to a well-known URL.

The main drawbacks of this solution are:

- *Interoperability issues*. Although the client-side is multi-platform, the server software still relies on the Windows platform. Both the serial communication and storing software programs only run on Windows.
- *Server Software Maintenance issues*. Far too many technologies are used on the server side: Java, Python and ASP.NET. For maintenance purposes it would be interesting to concentrate all the functionality in a single component developed with only one programming technology.
- *Scalability issues*. The server provides service to only one user accessing the remotely programmable device each time. Ideally we would like to network N devices controllable by the same server instance, and accessible simultaneously by N users.
- *Image Streaming issues*. The reception of the remotely programmable device images is still far from optimum. Each image is transmitted as a JPEG file instead of a streaming solution which would allow for a more up to date and reliable tracking of the remote device's activities.
- *Security issues.* This iteration still lacks a semantic verification of the programs uploaded to the programmable device which would prevent the upload of hazardous software. However, now only port 80 is used in the communication between the client and server side of the system. Therefore, this solution is firewall-safe.

D. MicroServer AJAX Web-based Solution

We are currently progressing to the WebLab architecture shown in Fig. 5. This solution will be web-based, firewall-safe, more scalable (will provide several programmable devices) and support cooperative work among group members. N groups of users from any client platform will be able to access simultaneously to any of the N networked programmable devices.

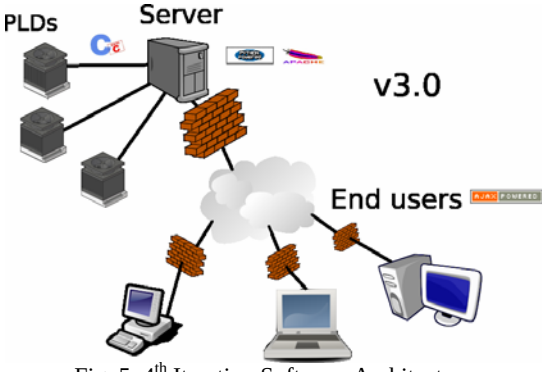

Fig. 5. 4th Iteration Software Architecture

In our third WebLab iteration, the communication and control of I/O was performed through RS-232 (see Fig. 6) by means of a PIC microcontroller acting as a bridge between the server and the electronic prototype. Moreover, the WebCam was connected to the server by means of an USB port. Therefore, if we wanted a single server to control several prototypes and WebCams we would need several serial and USB ports together with the corresponding coordination protocol for all those devices.

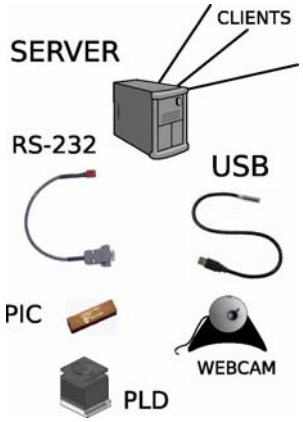

Fig. 6. 3rd Iteration Electronic Prototype Connectors.

In the currently ongoing development of the fourth iteration of our WebLab we will replace the PIC microcontroller by an assortment of IP-accessible MicroServers, as shown in Fig. 7. The adoption of MicroServers will turn our WebLab into a much more flexible and scalable distributed system:

- The WebLab server will no longer have to deal with the low-level RS-232 communication details. It will instead communicate through HTTP by means of data encoding standards as XML.
- The MicroServers will allow the set of programmable devices within a WebLab to be connected in a LAN. The MicroServers will connect either through an Ethernet port or will host an IEEE 802.11 chip to allow them to be wirelessly connected among themselves and the controlling WebLab server.
- The electronic prototypes attached to the MicroServers will also be capable of exchanging information among themselves. The information does not only flow between the electronic prototype and the server, but it also can flow among prototypes with the help of the MicroServers.

With the incorporation of MicroServers, each programmable device in a WebLab will be transformed into a networked node. Therefore, network administrators will now have to deal with a new type of device and ensure it is operational on a 24x7 basis.

An interesting application of this more scalable WebLab, now we can have N students simultaneously accessing to the N available programmable devices, is that its use could be shared with organisations external to our University. For instance, taking into consideration the hour zone differences between Spain and South America, our WebLab could be accessible to South American Universities during Spanish night hours. That activity would not suppose a big disadvantage for our students, since their use of the WebLab is very marginal at night.

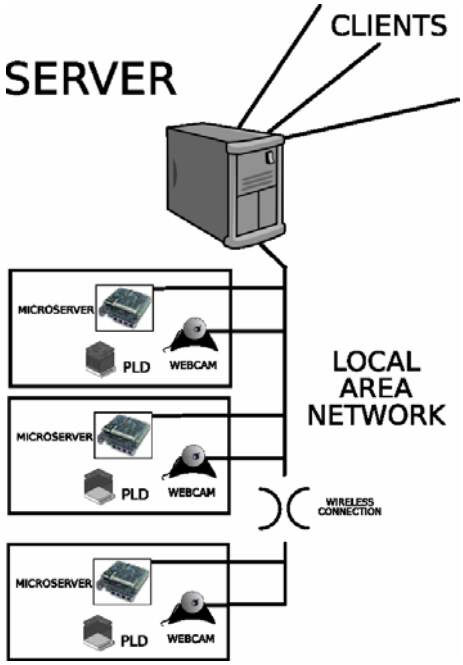

Fig. 7. 4th Iteration Electronic Prototype Connectors.

On the client-side, the current AJAX-based solution will be improved to add groupware features, i.e. the capability of working in group, and better image retrieval features. On the other hand, the WebLab server will concentrate all the functionality currently dispersed in three components: the webcam server, the controlling Python server and the Web Services hosted in Apache. It will be reimplemented in a single programming language (probably Python). This server will now communicate with N autonomous MicroServers, small hardware devices attached to each programmable device, providing two main function: (1) writing programs and configurations into the remote devices, induce inputs and capture outputs, and (2) be accessible through TCP/IP. Each MicroServer will implement a cut-down web server. Access control to each of these MicroServers will be regulated by the WebLab server. In addition, we will incorporate IP cameras also accessible through TCP/IP, without having to attach them to a PC.

In conclusion, this fourth iteration will provide us with a cross-platform, secure, collaborative multi-user multi-device solution, which maximises the use of the hardware resources allocated. The core idea behind our fourth iteration architecture will be to "push away" from the WebLab server all the functionality specific to a given programmable device.

E. Towards Universal WebLabs

In this section we explain our vision on the future of WebLab architectures.

The adoption of IP-accesible MicroServers giving access to attached programmable devices removes the previous location dependency between the server and its associated programmable devices. Before, the server and the electronic prototypes have to be co-located in the same lab premises. However, now they can be placed in any network accessible location. For instance, university A may implement programmable logic prototypes and host the WebLab server whilst university B may provide an assortment of process control prototypes, still controlled by university A's server.

There is a clear analogy between the World Wide Web and MicroServer-based Multi-Device WebLabs. Everybody can create a new web page, store it in a web server and automatically make it accessible on the web. Likewise, any organisation could create a set of network accessible prototypes and register them with a controlling WebLab server. In essence, we could be talking about "WWW hardware".

However, before this vision can become reality is necessary to standardize the controlling interfaces offered by a WebLab MicroServer. Every compliant WebLab MicroServer should implement the same Web Service interface, so that a given WebLab server can act as proxy between the users' browser and the MicroServers controlling any kind of programmable device. More ambitiously, we could even consider the concept of "WebLab MicroServer Plug&Play". The software stored in a MicroServer could implement automatic discovery, interaction and registration mechanisms similar to the ones provided by UPnP [19]. Thus, a manual registration of each MicroServer added with the WebLab server would not be required any longer. In essence, we would be moving from a centralised (all programmable devices in one physical location) to a distributed cross-organitational WebLab [20].

Finally, from a didactical point of view collaborative work is paramount. It is usually applied to sharing documents. Applied to WebLabs, collaborative work will enable to do the same with hardware devices. WebLabs clearly improve the possibilities to share devices in a remote way.

We have not found any references in the literature mentioning the possibilities for cooperative work opened by WebLabs. We believe that the software architecture proposed by our fourth generation WebLab presents very promising collaborative features and will truly approach to the final goal of a WebLab, i.e. to allow almost the same kind of interaction as the one achieved by a group of people working in the same physical lab.

IV. CONCLUSIONS

Traditionally little attention has been payed to the software part of WebLabs. This paper has shown the benefits of aiming better software solutions. A good software solution should lead to a more efficient use of hardware resources. Consequently, we have applied an iterative process to the software architecture of our own WebLab in order to progress from a 1/1/1 (user/server/programmable device) to an N/1/N WebLab.

As a result of our iterative study we suggest a new canonical software architecture based on the concepts of Web Services and MicroServers which presents the following features: cross-platform, secure and firewall-safe and scalable (multi-user and multi-device).

V. ACKNOWLEDGMENTS

WebLab-PLD has been part of the project WebLab-PLD supported by the Regional Government of the Basque Country (Spain): Department of Industry, Commerce and Tourism, SAIOTEK 2004, S-OD04UD18.

VI. REFERENCES

[1] Gustavsson, I. et al. “A flexible remote electronics laboratory". *2nd International Symposium in Remote Engineering and Virtual Instrumentaion, REV 2005*, Brasov (Romania), July 2005

[2] Barrón, M. “Laboratorios virtuales para enseñanza por internet" en *I Jornadas de Tendencias sobre eLearning, TEL 2005*, Madrid (Spain), febrero de 2005.

[3] García Zubía, J. “Programmable Logic and WebLab” V European Workshop on Microelectronics Education, *Proceedings of the 5th European Workshop on Microelectronics Education* ISBN: 1-4020-2072-4, pp: 277-282, 2004.

[4] Ko, C.C.; Chen, Ben. M.; Chen, Jianping. *Creating Web-Based Laboratories*, Springer-Verlag 2004, London, ISBN: 1-85233-837-7

[5] Alamo, J.A., MIT Microelectronics Weblab, Marzo, 27, 2001. http://web.mit.edu

[6] Pérez M. et al. “Laboratorios de acceso remoto. Un nuevo concepto en los procesos de Enseñanza-Aprendizaje”. http://digital.ni.com/worldwide/latam.nsf/web/all/F54369A0EC8C0B4486256B5F006565A9

[7] Kahoraho Bukubiye, E., Larrauri Villamor, J.I. "A WebLab System for the Study of the Control and Protection of Electric Motors", *Proceedings of Telecommunication, Electronics and Control*, pp. 7. Cuba 2002. ISBN: 84-8138-506-2, 2002

[8] Rodrigo, V.M; Bataller, F.M.; Baquero, M. and Valero, A. “Virtual Laboratories in Electronic Engineering Education”. Proceedings ICEE International Conference on Engineering Education. ISBN: 84-600-9918-0, 5 pp in CD, Valencia, España, 2000.

[9] Almeida, P., Viera Coito, F., Brito Palma, L. “An Environment for Remote Control”. *1st International Workshop on e-learning and Virtual and Remote Laboratories*, VIRTUAL-LAB’2004, Setubal, August 2004.

[9] Casini, M.; Prattichizzo, D. y Vicino, A. “e-Learning by Remote Laboratories: a new tool for control education” . *The 6th IFAC Conference on Advances in Control Education*, Finland, 2003.

[10] Cabello, r. et al. "EMERGE: A European Educational Network for Dissemination of OnLine Laboratory Experiments". *Innovations 2004*, Ed. iNNER, 2004.

[11] Soysal, O. "Computer Integrated Experimentation in Electrical Engineering Education over Distance" *Proceedings of ASEE 2000 Annual Conference*, Saint Louis, MO, June 2000.

[12] García-Zubia, J et al. “A new approach for implementing remote laboratories: a practical case”. *2nd International Symposium in Remote Engineering and Virtual Instrumentaion, REV 2005*, Brasov (Romania), July 2005

[12] Bagnasco, A.; Ponta, D.; Scapolla, A.M. “Remote esperiments in electronics: pedagogical issues". *2nd International Symposium in Remote Engineering and Virtual Instrumentaion, REV 2005*, Brasov (Romania), July 2005

[13] Gomes, C. “Distance Learning Remote Laboratories using LabVIEW”. *1st International Workshop on e-learning and Virtual and Remote Laboratories*, VIRTUAL-LAB’2004, Setubal, August 2004

[14] Aliane, N.; Martínez, A.; Fraile, A.; Ortiz, J. "LABNET: laboratorio remoto para control de procesos". *Actas de las XI Jornadas de Enseñanza Universitaria de la Informática, JENUI 2005*, pp: 515-522, ISBN: 84-9732-421-8, Julio 2005, Madrid (Spain)

[15] Pelcz, A. et al. “Remote experiments using JAVA: Inplementations in the Virtual Electro Lab project". *1st International Workshop on e-learning and Virtual and Remote Laboratories*, VIRTUAL-LAB’2004, Setubal, August 2004

[16] Gomes, L.; Costa, A. “Embedded sytems introductory course supported by remote experiments ". *1st International Workshop on e-learning and Virtual and Remote Laboratories*, VIRTUAL-LAB’2004, Setubal, August 2004.

[17] Bagnasco, A.; Chirico, M.; Scapolla, A.M. "XML Technology to Design Didactical Distributed Measurement Laboratory (RmwLAB) Instrument", *IEEE Transactions on Instrumentation and Measurement*, VOL. 54, Nº 1, february 2005

[18] McLellan, D. “Very Dynamic Web Interfaces”, O'Reilly Xml.com, http://www.xml.com/pub/a/2005/02/09/xml-http-request.html, February 2005.

[19] The Universal Plug and Play Forum, 2005, http://www.upnp.org/.

[20] Rasche, A. et al. “Distributed Control Lab". *1st International Workshop on e-learning and Virtual and Remote Laboratories*, VIRTUAL-LAB’2004, Setubal, August 2004.