SecondLab: A Remote Laboratory under Second Life

J. García-Zubia, J. Irurzun, I. Angulo, U. Hernández
Faculty of Engineering
University of Deusto
Bilbao, Spain
zubia@eside.deusto.es

M. Castro, E. Sancristobal
IEECD – Spanish University for Distance Education (UNED)
Spain

P. Orduña, J. Ruiz-de-Garibay
DeustoTech – Tecnológico
Fundación Deusto
Bilbao, Spain

Abstract—**The present work describes the implementation of a new remote lab, SecondLab, that allows students to control a microbot from Second Life. SecondLab works over WebLab-Deusto, the remote lab of the University of Deusto, giving the students the chance to work with real experiments from a social 3D-based immersive environment. This approach places the remote lab closer to the students, trying this way to increase their motivation to study science and engineering.**

Remote experimentation; Second Life; e-Learning

I. INTRODUCTION

The new learning trends suggest both entertainment and highly immersive environments as a way to achieve an effective learning process. In particular, socialware and game-like implementations are presented as attention-captive solutions applied to virtual education [1]. The recently published research from the Massachusetts Institute of Technology focused on Physics experiments is a commendable example on this direction [2].

The present work follows this line proposing a tool that can contribute to solve the current decrease of enrollments in scientific and engineering-related degrees. To achieve this purpose, SecondLab takes advantage of the technical possibilities of remote labs and the increase of popularity of social immersive environments among teenagers.

The present paper is structured in two main sections. Section II first shows an overview of the projects involved in the design, while Section III describes the implemented solution from a technical point of view. The research finishes in Section IV with the conclusions and future plans.

II. BACKGROUND COMPONENTS

The aim of this project is to offer an attractive electronic experiment to be used from a virtual 3D environment, taking advantage of other projects with different well-defined goals. Pursuing this aim, the research has basically involved an integration effort. The three components which SecondLab is based on are WebLab-Deusto, Second Life and the SecBot microbot (see Fig. 1).

The authors would like to acknowledge to the Spanish Science and Innovation Ministry for the support in the project TIN2008-06083-C03/TSI "s-Labs – Integración de Servicios Abiertos para Laboratorios Remotos y Virtuales Distribuidos" and to the Regional Government of the Basque Country for the support in the project SAIOTEK S-PE08FD03.

Figure 1: The three background components

A. WebLab-Deusto

WebLab-Deusto [3] is the remote lab of the University of Deusto, which makes possible to offer real experiments to a certain group of users through any computer network, such as Internet.

This remote lab has been used since February 2005 by the students of the University of Deusto as an essential tool for their practice works in four different engineering-related subjects.

WebLab-Deusto provides a distributed software architecture (see Fig. 2) that makes easy to integrate new experiments inside, so every experiment served this way automatically takes advantage of all the common features implemented by WebLab-Deusto:

- *Authentication*. WebLab-Deusto offers an extensible authentication system that currently supports three ways to verify the user credentials: username and password stored in a MySQL database, authentication through a remote LDAP server and trusted authentication based on the IP address of the client that is trying to log in.

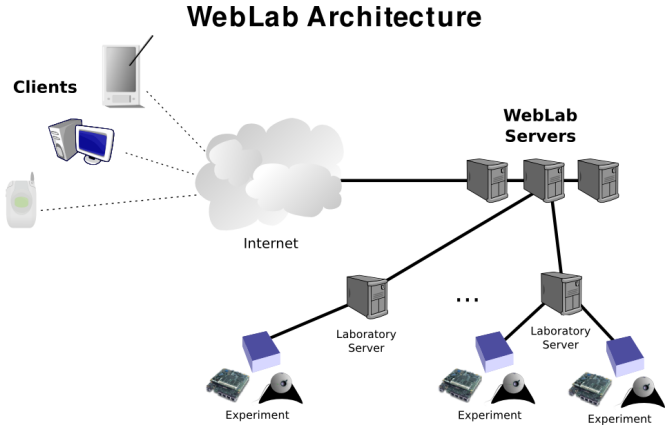

Figure 2: WebLab-Deusto architecture

- *Reservations*. WebLab-Deusto offers a generic reservation system that makes possible to reserve any available experiment for a non-scheduled period of time (i.e: 200 seconds), which is defined as a parametrizable experiment-dependant property.
- *Queue management*. WebLab-Deusto manages different reservation queues for the different available experiments, making the experiments themselves completely unaware of the users that are trying to use them.
- *Scalability*. The WebLab-Deusto architecture is horizontally scalable since more hardware can be added to improve not just the performance but also the availability of the remotely served experiments. The more experiment hardware devices of the same type are added, the more users will be able to use the same experiment simultaneously.
- *Security*. The WebLab-Deusto distributed architecture keeps in isolation the experiment-dependant hardware and software, so any problem related to a wrong use of the experiment will never put the whole remote lab at risk. Besides that, this same architecture considers the Login server separate from the others, so the most sensitive information (usernames and passwords) could not be accessed in case of eventual security attacks (see Fig. 3).
- *Deployment*. The WebLab-Deusto deployment system makes easy and flexible to configure the distributed network map in which all the servers and experiments are involved. Every WebLab-Deusto server can be configured with an easy-readable .xml file to declare the protocols which it will be communicated through, being five different ones available up to now: XML-RPC, Python-dependant SOAP messages, TCP sockets, UNIX sockets and "Direct" (direct method calls inside the same program instance).
- *Logging*. Everything that may happen during the use of the remote lab can be logged at different risk levels. The amount of events to log is up to the WebLab-Deusto administrator.

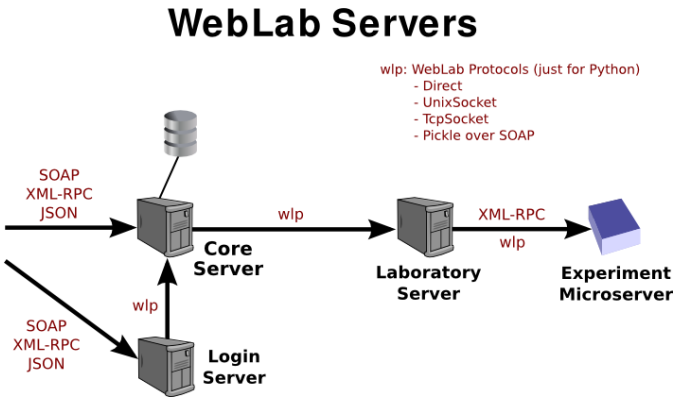

Figure 3: WebLab-Deusto servers

- *Administration*. As a common feature that every remote lab requires, WebLab-Deusto offers administration tools that make easy both to monitor who is currently using the remote lab as well as to add, edit or remove users, permissions and experiments.

B. *Second Life*

Second Life [4] is an end-user MMOW (Massive Multiplayer Online Worlds) software developed by Linden Lab since 1999. It provides a 3D environment where people can create their own customized avatar (a virtual representation of a person) and interact with other people. Its goal is to offer a real-world-like platform with the same physic laws, rules of conduct and opportunities (to get a job, to buy things…) that everybody has in the real life.

The most remarkable feature of Second Life is that there are no established goals for its users. Due to this, the platform has been mainly used in an entertainment way, but during the last few years it has also been used as an educational tool [5].

Second Life consists on two main software components: the private server farms at Linden Lab and the free-software client. The servers manage the business logic and the persistence tier, so the client is just responsible for rendering the 3D world and notifying the servers all the events carried out by the user. Therefore, developing software in Second Life entails both designing 3D graphics as well as programming server-side software.

C. *SecBot*

SecBot is an auto-programable microbot specifically designed for this project. Despite that, SecBot can be considered as a background component, since any other experiment could have been offered in SecondLab instead (from a technical point of view) and the microbot itself could also be used in a different context.

The reasons for choosing a microbot as the experiment offered in SecondLab are both educational and technical. The educational reason points to the attractiveness of the experiment for young students, which works in the direction of promoting entertainment while learning. The technical reason is related to the state-of-the-art in auto-programmable microcontroller memories. Now it's possible to create an autonomous microbot with the capability of programming its own instructions memory, so it can completely change its behavior in run-time. This is exactly the feature required to design the autonomous remotely-programmable microbot desired for the educational aims of the project.

This customized microbot, called SecBot, has been built over the *Robot Sumo Terminator S300230* from SuperRobótica [6], a commercial mechanical structure commonly used in the fighting modality of robotics competitions. As a first prototype version, SecBot offers the student the following components:

- 1x PIC18F4520 microcontroller from Microchip [7]
- 1x MD22 engine controller circuit from Devantech [8]
- 1x GP2D12 infrared sensor from Sharp [9]

III. SECONDLAB

SecondLab merges all the previous components in order to create a remote lab that allows students to program a microbot from Second Life. The best way to understand how SecondLab works from the student point of view is looking at the "happy path" scenario, described by the following steps:

1. Write a C program with the instructions that the microbot shall follow.
2. Compile the .c file to obtain the assembler .hex version of the program.
3. Open Second Life and log in with the avatar's name, surname and password.
4. Travel to SecondLab by typing "SecondLab" in the search tool.
5. Enter the laboratory with the avatar (see Fig. 4) and press the big green START button.
6. Upload the .hex program (see Fig. 5).
7. Wait until the screen in front shows the microbot working under the programmed orders (see Fig. 6).
8. Press the big red STOP button and leave the laboratory.

A. Architecture

Since SecondLab takes advantage of the previously described background components, its architecture is completely determined by them. In particular, WebLab-Deusto determines the global architecture, as it is the real remote lab underneath.

The WebLab-Deusto architecture follows an experiment-agnostic approach [10]; that is, it makes all its features completely independent of the specific requirements of the remotely-served experiments. Thus, integrating a new experiment in WebLab-Deusto becomes as simple as developing two components: a microserver at the very end of the server side (capable of interacting with the microbot), and a web-client module at the highest layer of the provided web-client application (presentation layer). Since the client platform had to be Second Life in this project, the most remarkable challenge became making Second Life work as a WebLab-Deusto client. Figure 7 shows the interaction among all these components.

Figure 4: The SecondLab entrance

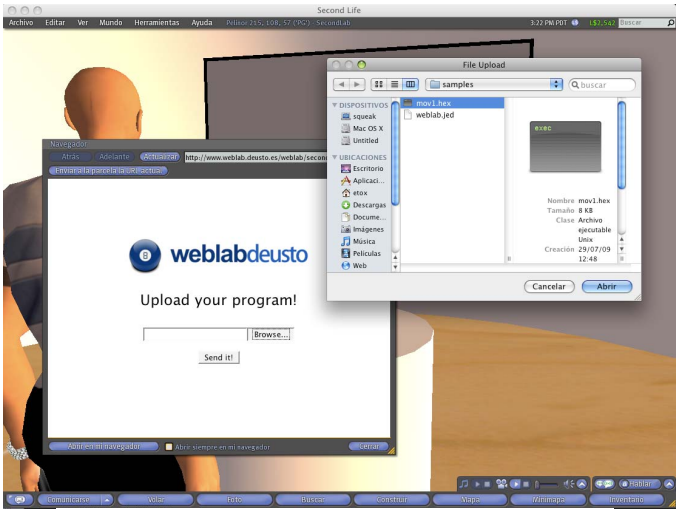

Figure 5: Uploading the program

A third smaller component, Web-uploader, had to be developed too in order to solve a certain Second Life technical limitation that will be explained later.

B. Microserver

The function of a WebLab-Deusto microserver is to communicate the experiment-agnostic set of WebLab-Deusto servers with the final hardware of the experiment (the SecBot microbot in this project). The only requirement established by WebLab-Deusto to develop a microserver is to implement the following interface (neutral language):

- test_me(value)
- start_experiment()
- send_file_to_device(file, file_info)
- send_command_to_device(command)
- dispose()

This interface can be implemented in any of the following supported protocols:

- Direct (function calls from/to the same process)
- Pickle over UNIX Sockets (Python dependent)
- Pickle over Internet Sockets (Python dependent)
- Pickle over SOAP (Python dependent)
- XML-RPC (language independent)

As figure 7 shows, the microserver in this project had not just to translate the messages received from WebLab-Deusto to the microbot, but also to manage with two different communication protocols: Bluetooth and any other from the

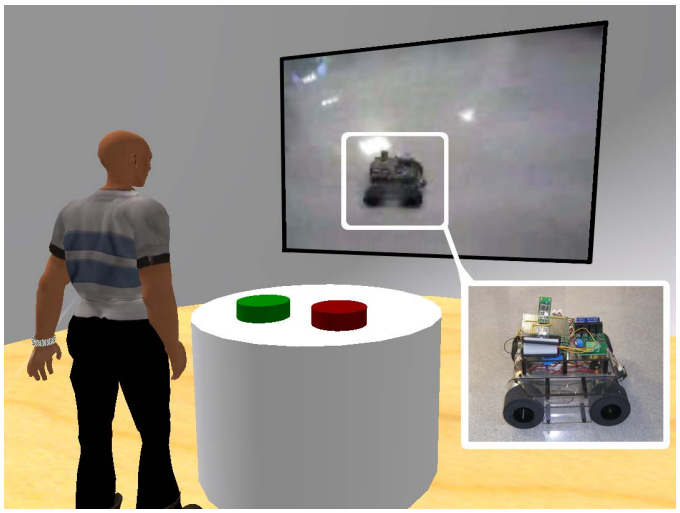

Figure 6: Watching the microbot working

five above. For this reason, a high-level and well-communicated language as Visual Basic .NET was chosen to develop this component. Therefore, XML-RPC was the only possibility to implement the listening interface.

C. Client

As previously explained, the most significant challenge in this project was to make Second Life work as a WebLab-Deusto client. Since Second Life provides a generic world where people can live and build their lifes, it also provides technical features so developers can program the created objects and buildings with certain behaviors.

Several steps must be followed to develop a custom piece of virtual world in Second Life. Building is not allowed for free, so buying a plot of land is the first step in order not to lose the work; secondly, every desired object must be graphically designed (buildings, items, textures…) using the 3D model editor provided by Second Life; and lastly, as many programmed behavior as wanted can be added to the world thanks to the scripting language created by Linden Lab (*LSL, Linden Scripting Language*). Every object in the world can be linked with many LSL scripts, which will be executed when the different programmed events happen in the virtual world (e.g.: an avatar touches a certain object).

From this scenario, developing a WebLab-Deusto client in Second Life entailed two big steps: designing a graphical 3D laboratory where the students could go with their avatars and communicating this laboratory with the remote lab so it could work exactly as a valid WebLab-Deusto client. Since the graphical task involves a creative work, the technical challenge was programming a communication layer under the conditions of a limited language like LSL.

The WebLab-Deusto architecture presents the server side as a webservice with the following public interface facade (neutral language):

- login(username, password)
- login_based_on_client_address(username, address)
- get_user_information(session)
- list_experiments(session)
- reserve_experiment(session, experiment)
- get_reservation_status(session)
- send_file(session, file, file_info)
- send_command(session, command)
- poll(session)
- finished_experiment(session)
- logout(session)

This facade can be called using any of the three following format protocols:

- SOAP
- XML-RPC
- JSON

Second Life offers both XML-RPC and the underlying HTTP protocols; however, the provided XML-RPC support is too limited to be used with a minimally complex service (only one integer and one string values can be retrieved at a time). Therefore, an XML-RPC layer has been manually implemented in LSL so the provided HTTP service is definitely used to communicate Second Life with WebLab-Deusto. This developed XML-RPC layer is limited to the specific use done in this project, so it cannot be considered a generic XML-RPC library for LSL. Although the original design considered it, the technical memory restrictions found in Second Life (64 KB/script including bytecode, stack and heap) made impossible that desirable approach.

Finally, as figure 7 shows, the communication between Second Life and the webcam that points to the microbot is done through the RTSP streaming protocol, thanks to the easy-to-use RTSP client support offered by Second Life, which only requires to specify the URL where the streaming is received from. The network camera used in SecondLab is the D-link DCS-5220, although any IP-based webcam with RTSP support could be used.

D. Web-uploader

The web-uploader server module covers a Second Life limitation. The LSL programming API does not offer any user interface component to allow the user to select a file from its computer. This shortage becomes a critical issue in a project

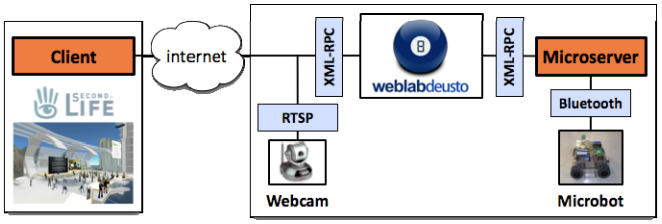

Figure 7: Architecture of SecondLab

like SecondLab, where sending a file from the student's computer is the main action in the interaction scenario.

This module takes advantage of the most flexible utility offered by the LSL programming API, which is a function that shows an embedded web browser with the specified URL loaded. The solution has been developed like a set of two server-side Python scripts. The first one offers a traditional HTML form with a *file* field that allows the user to choose a file from its hard drive (see Fig. 5). As figure 8 shows, when the submit button is pressed, this script sends a HTTP POST request to the second one, which really sends the selected file to WebLab-Deusto and notifies Second Life that the file has been sent, so the web browser can be closed and the user can be told to look at the virtual screen in front.

IV. Conclusions and Future Work

SecondLab proves that it is possible to mix virtual environments with real experimentation in a first approach (prototype version). However, it also proves that developing a remote lab for end-users under Second Life is not recommended due to its serious technical limitations:

1. *Limited scripting language*. The Linden Scripting Language is not comparable with any modern programming language. Its most remarkable handicap is the limited support offered for data types, which makes impossible to manage minimally complex data structures in memory. Suffice it to say that LSL does not offer basic structures (struct in C) or dictionaries, being a very limited (non-nestable) list the most complex structure supported.
2. *Few User Interface components*. Second Life was not designed as a real development environment but as a platform where average-users could create buildings and interact with other avatars through simple question-answer dialogs. Therefore, the LSL programmer can only get input data from the user by two simple UI components: a dialog window with customizable answer-buttons (llDialog) and a text box where the user can write characters (llTextBox). As already explained, sending the user's .hex file in SecondLab had to be implemented using the embedded browser provided by Second Life in a quite unorthodox way (see Fig. 8).
3. *Memory restrictions*. Every LSL script is only provided with 64 KB of total memory (bytecode, stack and heap included) at the Linden Lab servers. Having tested during the development of this project that 1000 lines of LSL source code consume approximately 55-60 KB of memory, this restriction becomes a significant problem that makes Second Life a really limited environment where almost none ambitious project can be developed at. However, Linden Lab also enables to set up a personal server and connect it to their grid, so the environment restrictions imposed by the Linden Lab service might be avoided. Although SecondLab has not been deployed this way (since renting a land was enough for a prototype), this other approach may solve the memory limitations.
4. *Lack of high-level communication protocols*. As already said, although according to the documentation LSL offers both XML-RPC and HTTP as communication protocols, the provided XML-RPC support is too limited to be used with a minimally complex service. Therefore, HTTP becomes the only well-supported communication protocol, so any higher-level one must be manually implemented (as done with XML-RPC in this project).

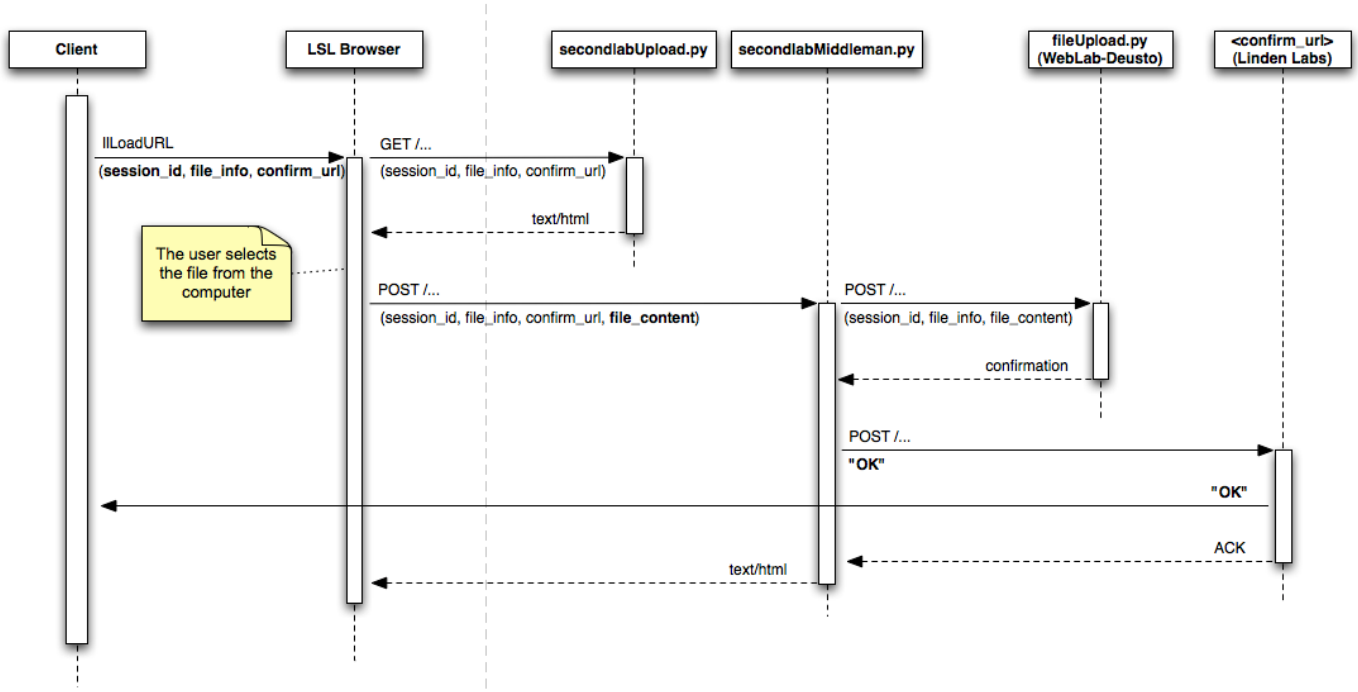

Figure 8: The web-uploader sequence

According to these limitations, a production-ready 3D-based remote lab for end-users should consider the idea of choosing a different 3D environment, such as Wonderland [11] or OpenSimulator [12].

After getting around of the current Second Life issues, future plans for this project consider technical improvements both in the software and hardware sides.

The main software improvement will extend the functionality of the remote lab (based on Second Life or any other 3D environment) to offer two real microbots for two students at the same time, so different kinds of competitions may take part remotely in order to increase the students motivation.

In addition, the next SecBot version will rely on an auto-rechargeable battery system, so the microbot will be completely autonomous. This mechanism will be based on a circular customized ring. The walls of this ring will be filled with two conductor strips connected to the power supply, so as the microbot stops being used it can move forward and stay attached to any wall charging its battery.

REFERENCES

[1] F. M. Schaf, D. Müller, C. E. Pereira, F. W. Bruns. "Computer Supported Collaborative Social Environment for Education, Training and Work". Remote Engineering and Virtual Instrumentation 2008. June 23rd–25th Dusseldorf, Germany.

[2] T. Scheucher, P. H. Bailey, C. Gütl, V. J. Harward. "Collaborative Virtual 3D Environment for Internet-accessible Physics Experiments". Remote Engineering and Virtual Instrumentation 2009. June 22rd–25th Bridgeport, CT, USA.

[3] WebLab-Deusto, http://www.weblab.deusto.es

[4] Second Life, http://www.secondlife.com

[5] T. Ritzema, B. Harris. "The use of Second Life for distance education". Journal of Computing Sciences in Colleges. Volume 23 , Issue 6, Pages 110-116. June 2008.

[6] SuperRobótica, http://www.superrobotica.com

[7] Microchip, http://www.microchip.com

[8] Devantech, http://www.devantech.co.uk

[9] Sharp, http://www.sharp-world.com

[10] P. Orduña, J. García-Zubia, J. Irurzun, E. Sancristobal, S. Martín, M. Castro, D. López-de-Ipiña, U. Hernández, I. Angulo, J. M. González. "Designing Experiment Agnostic Remote Laboratories". Remote Engineering and Virtual Instrumentation 2009. June 22rd–25th Bridgeport, CT, USA.

[11] Wonderland, https://lg3d-wonderland.dev.java.net

[12] OpenSimulator, http://opensimulator.org

AUTHORS

J. García-Zubia is with the University of Deusto, Electronics and Automation Department, Avenida de las Universidades 24, 48007 Bilbao (Spain, he is Head of Dpt. Of Industrial Electronics, Control Engineering, and Computers Architecture of the Faculty of Engineering. He is the responsible of the Remote Lab at the University of Deusto (WebLab-DEUSTO: http://www.weblab.deusto.es). WebLab-Deusto has been implemented using web 2.0 techniques (AJAX, SOAP, etc), a novelty approach in Europe. Different works have been published explaining the results and the technology of this weblab and the evolution of WebLab-DEUSTO has been supported by different projects. (e-mail: zubia@eside.deusto.es).

J. Irurzun is a Computer Engineer by the University of Deusto, Bilbao 2009. He is Research Intern at the WebLab-Deusto Research Group and MSc student in Development and Integration of Software Solutions at the University of Deusto. (e-mail: irurzun@deusto.es).

I. Angulo is with the University of Deusto in the Dpt. of Industrial Electronics, Control Engineering, and Computers Architecture of the Faculty of Engineering. He is the lead hardware designer and developer of WebLab-Deusto. (e-mail: ignacio.angulo@deusto.es).

P. Orduña, is a Computer Engineer by the University of Deusto, Bilbao 2007. Nowadays he is Research Assistant at the Ambient Intelligence department of DeustoTech - Tecnológico Fundación Deusto, and PhD student at the University of Deusto; his research is focused on Remote Laboratories. He is the lead software designer and developer of WebLab-Deusto. (e-mail: pablo.orduna@deusto.es).

J. Ruiz-de-Garibay, is a Computer Engineer by the University of Deusto, Bilbao 2003. Nowadays he is Research Associate at the Ambient Intelligence department of DeustoTech - Tecnológico Fundación Deusto, and PhD student at the University of Deusto; his research is focused on Remote Laboratories based on Robotics. (e-mail: jonathan.garibay@deusto.es).

U. Hernández is with the University of Deusto in the Telecommunications Department at the Faculty of Engineering. He is developer of the research group on web-based laboratories and he is in charge of the Remote Labs based on instruments control. He is involved too in the deployment in University of Deusto of the VISIR project leaded by the Blekinge Institute of Technology (Ronneby, Sweeden). (e-mail: unai.hernandez@deusto.es).

M. Castro, Electrical and Computer Engineering educator in the Spanish University for Distance Education (UNED), has an industrial engineering degree from the ETSII (Industrial Engineering School) of the Madrid Polytechnic University (UPM) and a doctoral engineering degree from the same University. He received the Extraordinary Doctoral Award in the UPM and the Viesgo 1988 Award to the Doctoral Thesis improving the Scientific Research about the Industrial Process Electricity Application. He works as researcher, coordinator and director in different projects, ranging from solar system and advanced microprocessor system simulation to telematics and distance learning systems, acting now as and senior technical director. He is now with the UNED (Spanish University for Distance Education) as Professor in the Electronics Technology subject inside the Electrical and Computer Engineering Department as well as he is Director of the Department. (e-mail: mcastro@ieec.uned.es).

E. Sancristobal is a Computer Science Engineer by the Salamanca Pontifical University (UPS), Madrid 2002, has a Technical Engineering degree in computer networks (UPS), Madrid 1998. He finished his Doctoral Studies in Electronics Technology by the Electrical and Computer Engineering Department from the Spanish University for Distance Education (UNED), Madrid 2005. He worked for the University Distance Education Institute (IUED). Nowadays is working for the Computer Science Service Centre of the Spanish University for Distance Education (UNED) and is an associate professor Electrical and Computer Engineering Department from UNED. (e-mail: elio@ieec.uned.es).