A Java/Matlab-Based Environment for Remote Control System Laboratories: Illustrated With an Inverted Pendulum

José Sánchez, Sebastián Dormido, Rafael Pastor, and Fernando Morilla

***Abstract*—In this paper, a novel environment is described that provides 24-h-a-day access to a Web-based lab for the remote control of different didactic setups. The control of an inverted pendulum is used to demonstrate the use of such an environment. The main attributes of this Web-based lab are: 1) the on-line interactivity with the didactic setup, 2) the possibility of defining different experiments by using parameter files, and 3) the open architecture of the environment which allows easy development of new experiments with other didactic setups. The structure of this Web-based lab not only provides students with quantitative information feedback but also allows visual supervision. Now, a remote-controlled camera plays an important role within a remote experimentation environment with mobile parts. Students can handle the camera on-line, just as they can control the didactic setup over the Internet.**

***Index Terms*—Distance education, laboratory education, remote control, telelab, teleoperation, Web-based instruction, Web-based lab.**

I. Introduction

MODELING and dynamic simulation are now considered to be the basic tools for verifying theoretical subjects [1]. Experimentation *in situ* with a plant cannot be replaced by simulations [2], and practical education needs to be based on real aspects that occur in mechanical, electrical, or chemical systems [3], [4]. However, the face-to-face laboratories are subject to a series of factors restricting their use.

— *The high number of students enrolled in the subject*. Students do not have enough time to complete the tasks nor the opportunity to practice with other didactic setups. Teachers must reduce their research time or work overtime to prepare the labs.

— *The distance-learning-based educational model*. Distance learning offers flexibility for those who are subject to limited time, distance, or physical disability.

— *Insufficient economic resources*. There are few laboratory experiments because they are very expensive.

Teleoperation is a possible solution to these problems. This concept has been studied for some years [5]–[8], but it is currently accessible to students thanks to the diffusion of the Internet network. If the term teleoperation is applied to gain access to the face-to-face laboratories, then one has the concept of the *WWW-based laboratory*.

This paper focuses on *the description of a Java-based environment that adapts to the necessities of experimentation across the Internet with different physical systems*. First, the main characteristics of the environment are enumerated in order to highlight the differences and analogies with respect to other similar works. Next, a case study for the control of an inverted pendulum is presented in more detail. Finally, the system evaluation and the conclusions are presented.

Manuscript received March 6, 2002; revised June 4, 2003. This workwas supported by the Spanish CICYT under Grant DPI2001-1012.

The authors are with Universidad Nacional de Educación a Distancia (UNED), Escuela Tecnica Superior de Ingenieria en Informatica, Departamento de Informatica y Automatica, 28040 Madrid, Spain (e-mail: jsanchez@dia.uned.es).

Digital Object Identifier 10.1109/TE.2004.825525

II. Main Characteristics of the Environment

In order to differentiate this contribution from other works related to the development of Internet labs [9]–[16] and to highlight the primary differences and analogies, the main characteristics of this methodology are mentioned as follows.

- *Easy to use and understand*. The experimentation interface is friendly and has been tested successfully in other simulation environments developed in the department [17].
- *Multiplatform client software*. The experimentation graphical interface is a 100% pure Java applet. Students just need a Java-enabled browser to carry out the experiments. No plug-ins, nor specific navigators, nor *ad hoc* software are needed as in some previous works [9], [10], [15], [16].
- *Global access to the remote laboratory using a data-sharing approach*. A small application protocol has been developed to exchange information between the server and the clients. When using the protocol it is possible
  - — to know the user's profile: student or teacher,
  - — to transmit the parameter and data streams by closing the tuning loop across the Internet, and
  - — to manage the control loop (stop, start, new sampling time, etc.) from the client interface.

  This approach is not new [12], [13], but in this case the developed protocol is more simple and light. It has been reduced to a set of basic commands regardless of the didactic setup. This approach is accomplished with the use of Matlab/Simulink as a front-end with the physical system.
- *On-line access to the physical resource*. Interaction with the system is dynamic and in real time. During the experimentation stage, every change in the input variables is immediately shown in the experimentation graphical user interface (GUI). Users can thus visualize on-the-fly how the

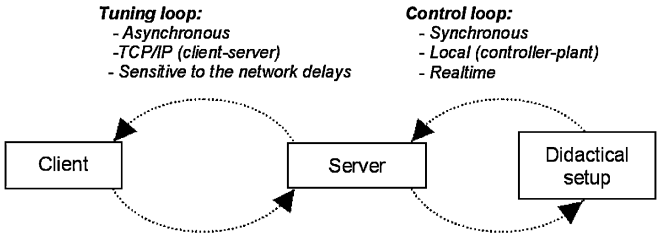

Fig. 1. Teleoperation diagram with two information loops: tuning and control.

system behavior evolves according to the values of the interactive variables. Yet when the experimentation stage is over, one can download and collect the results in a MAT file. Later on, students can carry out a quantitative analysis of their experiment, since the file has the samples of all the system signals. At the same time, the environment design allows students to work in batch mode: to tune the parameters, to launch the job, and finally to recover the data in a MAT file once the time is over. This design makes it possible to work with the didactic setup regardless of the network delays and the dynamic characteristic of the physical system.

- *Closed control loop running at the server side*. This well-known design approach [13] involves two information loops: the *synchronous control local loop*, and the *asynchronous tuning loop* (Fig. 1). The real-time control loop runs on the computer connected to the didactic setups. The control loop is always closed at the server side and never across the Internet because closure happens with the tuning loop. This last loop reads the state of the plant and sends this information to the remote client. The information circulating via the tuning loop contains small data arrays that are produced in response to the protocol commands that clients have sent to the server. The video stream is completely independent since it is not considered to be part of the tuning loop, as is in [13].
- *The separation of the control and tuning loop so that the current physical system is replaced by another system* almost transparently, since it would suffice to adapt the composition of the data exchanged between client and server to the new characteristics of the plant and the experimentation interface.
- *Definition of different experiments*. The teacher can include new experiments in the telelab by defining the corresponding *experiment definition files.* Thus, different experiments can be defined for one physical system: the customizing of control possibilities to be used in the user interface, the modifying of experimentation time, the changing of sampling time, the substituting of control strategy, etc. This feature is not present in any previous work, and it allows the contents of the telelab to be designed to meet the necessities of a specific control course.
- *Generation of disturbances in the process variables*. Most of the Web-based experimentation environments allow the introduction of disturbances in the system signals in a manual way, but not by software programming. In this framework, the teacher can preprogram a different set of disturbances for each experiment. Therefore, the system can be used as a flexible and didactic tool to test how students tune control parameters for reducing the disturbances.
- *Replacement of the controller*. In this environment, the software packages used to design the control structure are Matlab/Simulink and the Quanser WinCon environment. Thus, many control strategies with the same didactic setup can be tested by different Simulink block diagrams. Teachers do not need to learn a new language to define new controllers [9], [11] since Matlab/Simulink is considered *de facto* a standard language in the field of automatic control. At the same time, students can design new controllers using Simulink templates and, thus, introduce new experiments in the environment.

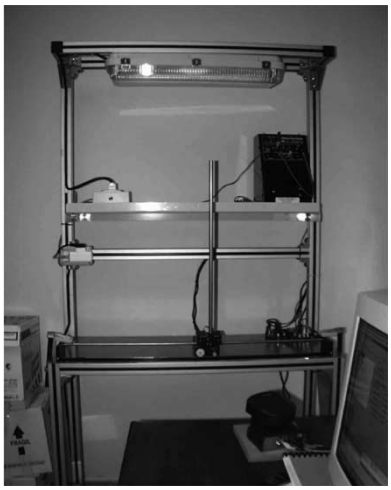

Fig. 2. Inverted pendulum adapted to the remote experimentation.

III. The Pendulum Experiment: A Case Study

In order to describe the features of the environment in more detail, a case study is presented in which the steps to create and carry out the experiment are described. The physical system used is an inverted pendulum with a linear motion cart, manufactured by Quanser [18], installed on an aluminum chassis (Fig. 2). This chassis, as well as the pendulum and its associated electronic equipment (power stage, data acquisition board, server computer), supports the visualization and lighting system (control box, lamps, motorized camera, and video hardware server). A system with these characteristics (mobile parts, fast dynamics) is very appropriate for use on the Internet since students perceive the results of their control actions with both quantitative and visual information. Another reason for choosing an inverted pendulum is that it is a classical plant designed for teaching advanced control systems theory. This didactic setup is useful to illustrate how to design and tune control laws for balancing a pendulum by means of different strategies.

The aim of this experiment is to obtain the parameters of a controller, using an LQR design that stabilizes the rod and keeps the cart in the desired position. As in previous steps, students must know control theory and the characteristics of the physical

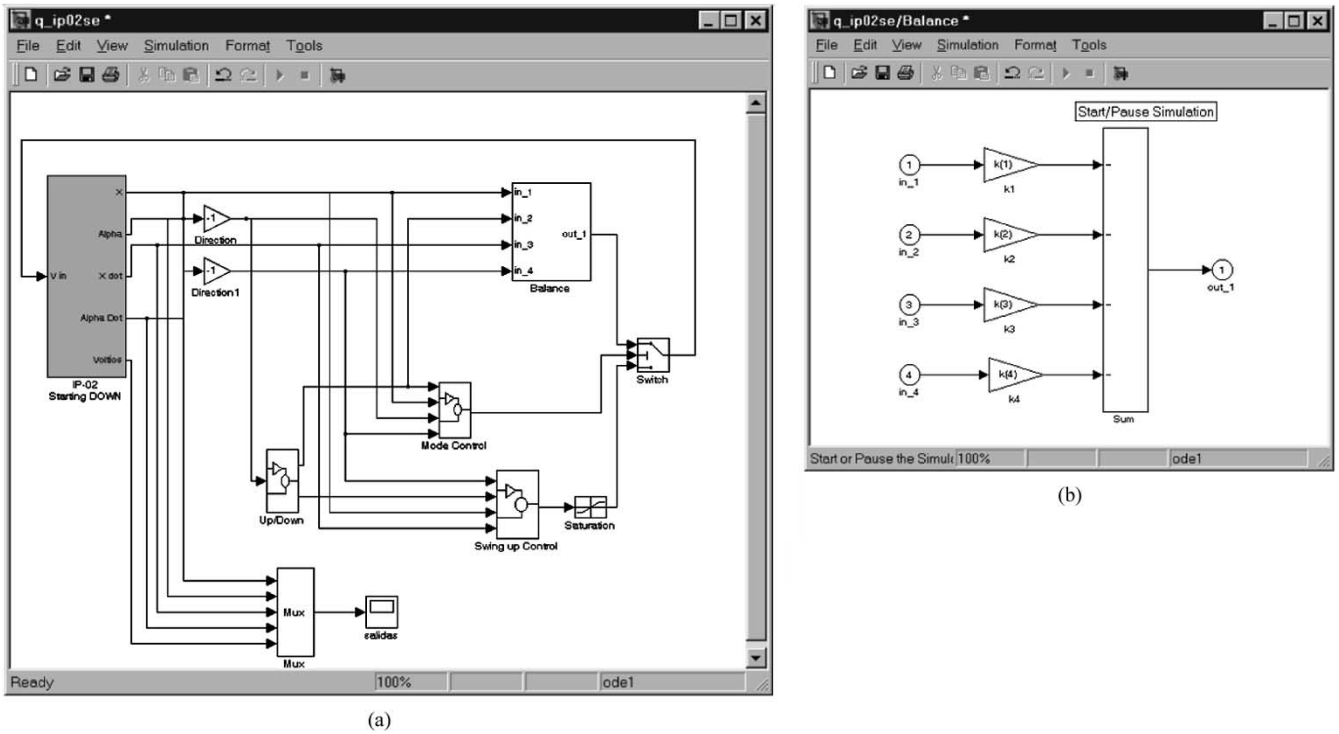

Fig. 3. (a) Simulink diagram with the control strategy. (b) Detail of the balance block with the gains.

system. The goal is to obtain the optimal feedback gain $K = [K_1, K_2, K_3, K_4]$ and, afterwards, to test to see if the solutions can stabilize the pendulum and keep the cart in the same position in which the experiment was started.

The control strategy to raise an inverted pendulum consists of two steps: swinging up and balancing. In this simple case study, balancing is the control law that students must tune by calculating previously the values of matrix K.

All the control strategy is programmed in a Simulink file by the block library of the WinCom software. Using the GUI of the environment and the documentation (PDF files), students can modify the values of the Simulink blocks in a remote way and, in this example, introduce the feedback gain K (Fig. 3). A complete description of the mathematical model can be found in [18].

IV. Environment Elements

The telelaboratory software architecture is based on the client–server paradigm. For this reason, one can distinguish two different parts: the client side or experimentation applet, and the server side, which is made up of laboratory equipment adapted for access and manipulation via the Internet (i.e., the inverted pendulum).

A. The Client Side

On the client side, the interface of the experimentation applet consists of two parts: the GUI, and the visualization system. The GUI is a Java applet with a browsing window and an experimentation window.

1) The Browsing Window: This window (Fig. 4) allows the experiments to be classified according to the structure of a textbook. Thus, each experiment is associated with a lesson, and each lesson with a chapter. Hence, the window has three browsing levels: chapters, lessons, and experiments.

The environment could have i chapters, each chapter j lessons, and each lesson k experiments with different control strategies or plants. In this simple way, tutors/instructors can tailor the environment for teaching a course with a specific profile, and students/operators can select an experiment from all the existing ones according to the concepts or situations they want to study or observe. A plain text file, the *browsing file*, stored at the HTTP server, sets up this window. An example of a browsing file is shown in Fig. 5. A chapter, a lesson to test the environment, and two experiments associated with the lesson are defined in this example. As can be observed, the inclusion of an experiment within a lesson implies writing the following lines: the name of the experiment that appears in the browsing window, the description file, the parameterization file, and the resource used for the experiment.

2) The Experimentation GUI: The experimentation GUI (EGUI) designed for the remote control of the inverted pendulum is shown in Fig. 6.

A generic EGUI consists of the following parts: the process diagram, the control panel, univariate scopes, a multisignal scope, and the historical log (as a MAT file). The process diagram is made up of a graphical diagram of the plant with alphanumeric visualization of the most important signals and units. A typical control panel consists of three types of elements: buttons, sliders, and fields; that allow users to control the experiment (start, stop, change sampling time, etc.), adjust the value of input variables, and set points, or modify the control parameters (for instance, K_p, T_i, and T_d of a PID controller) and the control mode (manual, automatic, cascade, etc.).

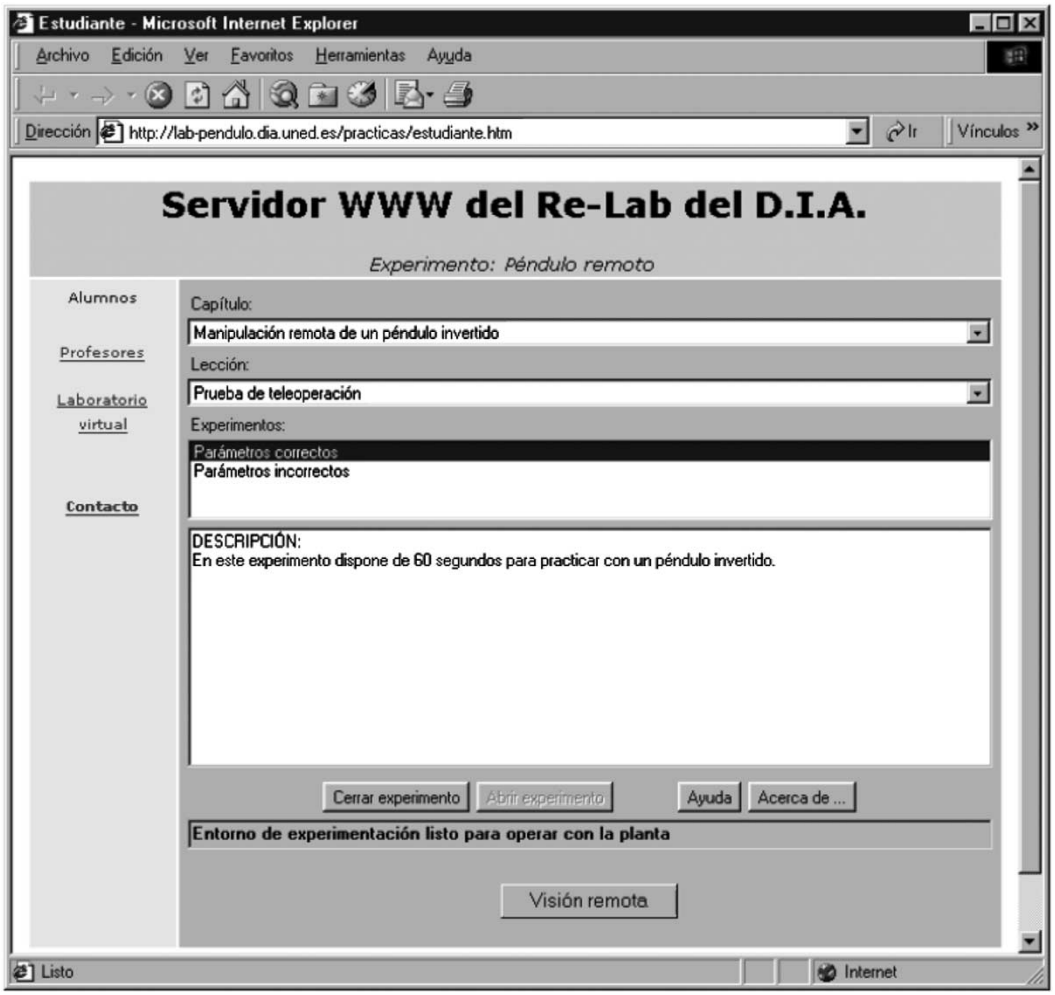

Fig. 4. Browsing window.

```
[groups]
total= 1
group1= “Teleoperation of an inverted pendulum"  // Name of the chapter 1
[group1]
total= 1
subgroup11= “Testing the environment"          // Name of the lesson 1
[subgroup11] total= 2
name1= “Correct parameters"                    // Name of the experiment 1
description1= experiment_1_1_1.des             // Name of the description file 1
parameters1= experiment_1_1_1.exp              // Name of the parameterization file 1
plant1= pendulum                               // Kind of the plant for the experiment 1
name2= “Wrong parameters"                      // Name of the experiment 2
description1= experiment_1_1_2.des             // Name of the description file 2
parameters1= experiment_1_1_2.exp              // Name of the parameterization file 2
plant1= pendulum                               // Kind of the plant for the experiment 2
```

Fig. 5. Example of a browsing file syntax.

In the case study presented here, the control strategy to stabilize the pendulum just needs the tuning of the gain K. Thus, four sliders, are required to modify the gain of the LQR controller implemented by Simulink blocks at the server side [Fig. 3(b)].

Besides scopes displaying the pendulum state with all the relevant signals, the EGUI provides animation of the mobile parts of the pendulum to enhance user perception. The user thus has a quantitative and qualitative view of the physical system.

The EGUI is an applet designed as an object-oriented application. Thus, each of the EGUI parts becomes a Java class allowing adaptation of the applet according to the didactic setup that is remotely controlled. Thus, if the environment is tailored to conduct another physical system (for example, a servo motor), just a few lines of Java code must be changed. For example, the information of the input data vector about the state of the plant must be connected to the corresponding

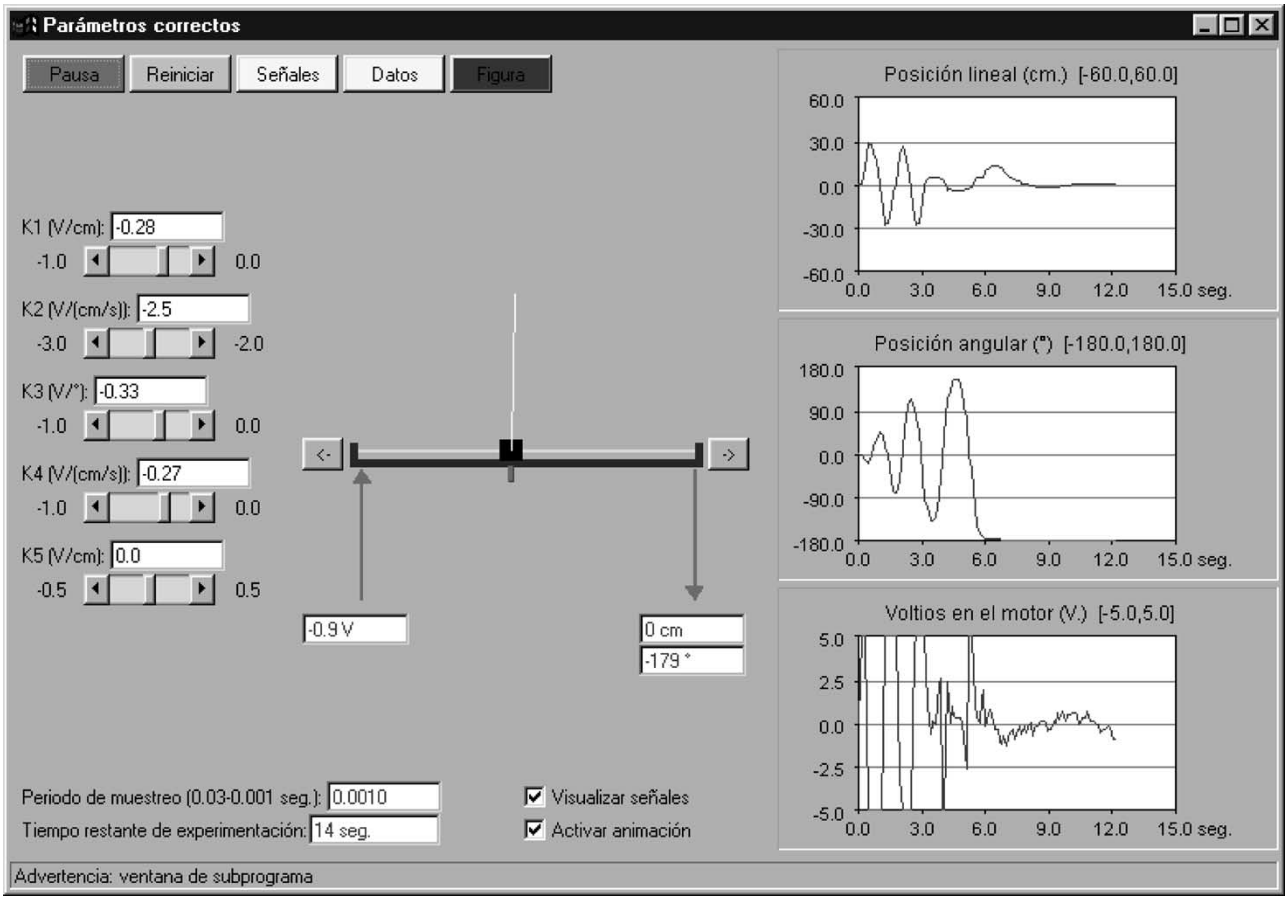

Fig. 6. Experimentation window.

GOAL: Student must establish knowledge of the LQR design in a control system.
DESCRIPTION: In this exercise, a LQR controller must be designed to stabilize the rod and keep the cart in the initial position. Using the information available in the pendulum documentation, the gain K must be calculated and tested using the graphical interface. Before starting the experiment, use the button to move the cart to the center of the track to avoid collisions.

- Process variables: Angle of the rod, position of the cart.
- Control signal: Motor voltage input.
- Disturbances: None.

Fig. 7. Example of description of an experiment.

scopes, fields, and graphical elements, or the composition of the output data sent to the server side must be changed in order to modify the blocks of the Simulink file that implements the control strategy.

3) Defining a New Experiment: One of the main characteristics of the environment is the definition and addition of new experiments. The creation of a new experiment implies two steps: description and parameterization. For both the creation of two files is necessary: a description file (extension.*des*) and a parameter file (extension.*exp*). The description file contains the textual description of the experiment's objectives and is viewed in the browsing window when one experiment is chosen from the list of experiments available for the lesson. Fig. 7 shows an example of a description file for an activity in which the pendulum is used.

Once the description file is created, a parameter file has to be generated in which the parameters define the plant behavior and the EGUI. As in the browsing file, the parameters are classified into groups or categories, and the number of groups varies according to the system. The creation of a parameter file requires filling a template with other values. If another didactic setup were used, a new template would have to be defined. This template would include all the relevant parameters of the new system (plant variables, controller parameter, disturbances, sampling time, etc.). An example file is shown in Fig. 8.

The *interactive_variables* category contains information on the process input variables and allows one to set the degree of interactivity and appearance of the experimentation interface. Each interactive variable has its textual label, initial value, maximum and minimum value, the possibility of interaction, and a list of disturbances. Disturbances are interpreted as changes in the value of a variable at specific time instants. Thus, when disturbances are defined for a variable, the total number of programmed variations have to be pointed out, followed by a list

```
[interactive_variables]
Label_Sampled_Time= "Sampling time (0.03-0.001 sec.):"
Minimum_Sampled_Time= 0.001
Maximum_Sampled_Time= 0.03
Initial_Sampled_Time= 0.001
Variable_Sampled_Time = panelable
Disturbances_Sampled_Time= 0
Label_K1= "K1 (V/cm.):"
Minimum_K1= -1.0
Maximum_K1= 0.0
Initial_K1= -0.28
Variable_K1 = sliderable
Disturbances_K1= 2 5 -0.5 40 -0.8
.....
[pendulum_parameters]
.....
[info_variables]
Variable_Cart_Position= yes
Variable_Angular_Position= yes
Variable_Signal_Motor= yes
[experiment_parameters]
Matlab_file= d_ip02se                  // Initial parameters
Simulink_file= q_ip02se                // Control strategy
Experimentation_time= 60               // Seconds
```

Fig. 8. Example of a parameter file.

of pairs instant/signal value, i.e., $ni_1v_1i_2v_2 \ldots i_nv_n$. Another important category is *experiment_parameters*. In this category, the total experimentation time and the Matlab/Simulink files used to generate the control loop are specified. The category *info_variables* indicates the visualization of output variables, and *pendulum_parameters* contains specific information on the plant (pendulum length, cart dimension, rail length, etc.). When the definition of the new experiment is over, the browsing file has to be modified to include the new experiment in the browsing window.

Depending on the kind of plants to be used, new categories will have to be introduced to define the behavior of other elements, such as PID temperature controllers (TC_*parameters*) or PID flow controllers (FC_*parameters*) when experimenting with a heat exchanger. The introduction of new categories involves the modification of the EGUI Java code that analyzes the parameter files and sets the values (label, maximum, minimum, disturbances) for the elements of the EGUI. Examples of other EGUIs can be found in [17].

B. The Server Side

In order to exchange information with the client applet, several applications operate on the server side (Fig. 9).

- *An HTTP server*. It provides the HTML pages, experimentation applets, browsing file, and definition files for each experiment (description file and parameter file). It also controls access to the pages according to user's profile and provides access to the remote visualization environment.
- A *concurrent server, called plant server*, developed in Java, with a twofold purpose. First, it interacts with the data acquisition board (DAB) and supplies a stream with the state of the plant to the tuning loop in order to update the EGUI. Second, it manages the Simulink+Wincon environment through the Matlab workspace. The control loop is thus managed (starting, stopping), and control parameters can be changed (controller parameters, sampling period). The dialog between this server and the experimentation applet is based in the application protocol.
- *WinCon.* It is a real-time Windows application that runs Simulink-generated code, using the Real-time Workshop to achieve digital real time on a PC equipped with a DAB. The DAB, the inverted pendulum, and the WinCom software are made by Quanser Consulting.
- *The Matlab/Simulink environment*. The design and construction of the control loop is carried out with the Simulink blocks and the WinCom Simulink block library. The Simulink diagram is managed from the Matlab workspace with different Matlab commands sent across the tuning loop from the EGUI. According to the experiment selected in the browsing window, different Simulink files will be opened to generate the real-time control loop, changing the control strategy, and allowing design of new experiments without changing the physical system.

C. Element Communication

The local communication on Windows between the Java language and Matlab is carried out using the ActiveX Automation protocol. Using the Matlab-type definition file called mlapp.tlb, a group of Java classes for wrapping the ActiveX interface with Matlab can be generated simply and quickly from the Microsoft Visual Java ++ environment (MVJ ++). Thus, using the methods provided by these classes, the Java code can communicate with the Matlab workspace and can, therefore, manage the control loop generated with Simulink/WinCom.

Accordingly, the commands that are sent from the EGUI to the plant server via the tuning loop are classified.

- *Inquiry commands*: The answer to these commands is a vector with the state of the system. This vector is built by reading the DAB encoders using low-level library routines. This response vector is used to update the EGUI. These commands also allow the retrieval of a file MAT with all the data generated during the experimentation stage.
- *Control commands*: They are sent at the request of the user to manage the control loop. These commands are stopping, starting, modifying the sampling time, changing the parameters of the Simulink blocks. These kinds of commands connect with the control loop via the Matlab workspace.

V. Remote Visual Supervision

In order to provide 24-h-a-day access to the lab described above, the hardware, the Java applets, and a server-side Java

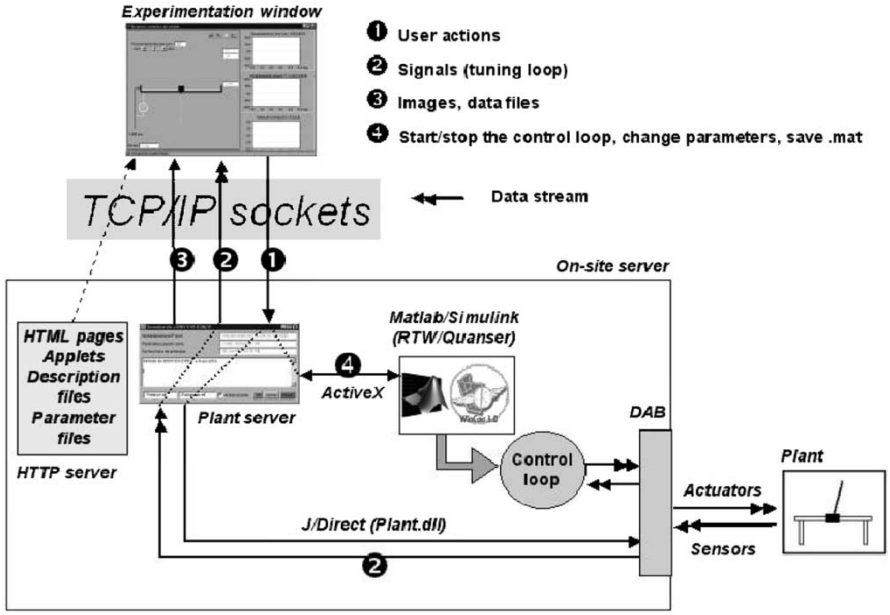

Fig. 9. Interaction between the client applet and server-side elements.

application have been developed for the remote control of a system consisting of several lights and a computer-controlled video camera. The system design allows teachers to reconfigure several camera setups. Students, therefore, concentrate their attention on specific points of the physical system that operate across the Internet.

A. Structure and Operation

Fig. 10 illustrates schematically the software architecture of the teleoperation environment. In this approach, the HTTP server supplies an HTML page consisting of three applets: one for the remote control of the plant (Section IV), another for the control of a SONY EVI-D30/D31 video camera, and a third one to download the video images. Evidently, the type of applet downloaded for the control of the camera depends on the user: the teacher applet has more functions than the student applet.

Once the WWW browser has loaded the applet for controlling the camera, it establishes a communication channel with the command server making use of TCP sockets. Owing to its software architecture, the server forks a new child process for attending every control applet. Thus, the server can go on listening to any new student or teacher connection request.

After that, the server accepts the connection, and the camera control applet issues commands and inquiries to the SONY camera. A simple protocol, named *Command and Inquiry Protocol* (CIP), has been developed to exchange camera control information between applets and the server. Therefore, the server receives the CIP packets, translates them into low-level packets, and sends this information to the camera through the on-site server serial port. This low-level protocol is VISCA [19].

Furthermore, once the video applet is loaded in the WWW browser, it establishes communication with the AXIS hardware video server [20]. This element recovers the images with the same quality and size that have been programmed. The choice of a video server is a smart solution for this kind of environment, for it releases the on-site server from the video grabbing tasks, and it avoids disturbances being introduced in the control local loop.

B. Camera Control Interfaces

There are two applets for managing the camera: teacher and student. The teacher applet [Fig. 11(a)] allows the configuration of the camera interactively according to experiment requirements: plant location, lighting, zoom for emphasizing some details, etc. The student applet [Fig. 11(b)] has fewer functions than the teacher applet.

Evidently, the construction of a remote experimentation environment involves not only physical independence on the part of the student, but also temporal independence. For this reason, a prototype for the remote control of lighting has been designed so that the view of the physical system is guaranteed at the very instant that a connection is established to conduct an experiment. The hardware prototype is based on electromechanical relays.

VI. Evaluation

Currently, the environment is used by a group of automatic control students at the university as a complementary activity to their on-site laboratory work. After working at home with the system, students must attend the face-to-face lab, run the same control experiments, and check the previous results. Compared to students who do not use the environment, the analytical skills and performance of those who do are superior during the design and tuning phase with the real systems. Students working at home perform better at exams and compose well-reasoned commentaries, improving their laboratory evaluation scores.

The results are very satisfactory because students acquire a new qualitative and practical view of their theoretical knowledge. In addition, by working with the experimentation environment, they know in advance the behavior and reactions of the plant, and thus shorten their preparation time at the lab.

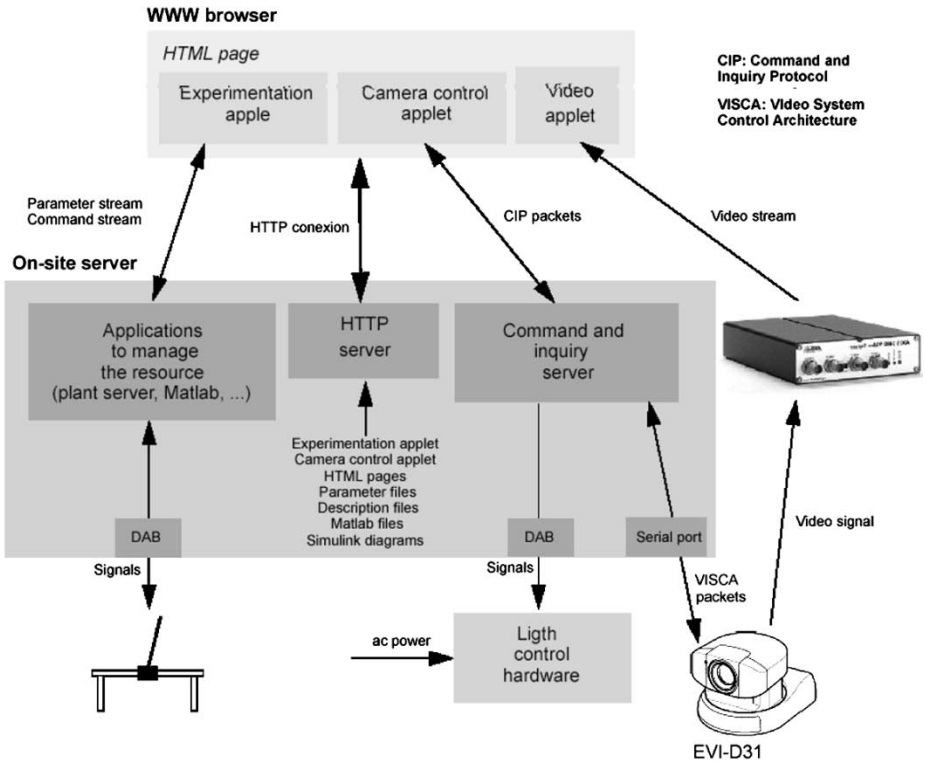

Fig. 10. Complete outline of the remote control environment.

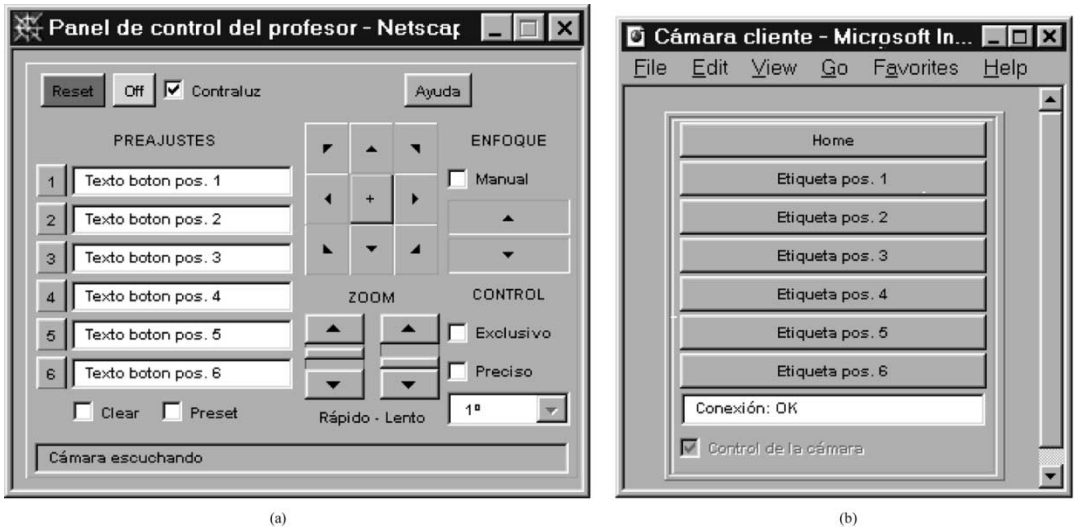

Fig. 11. Camera control interfaces. (a) Teacher. (b) Student.

VII. Summary

This paper provides a detailed description of an environment for the teleoperation of real plants via Internet, locating the control loop in the server and establishing a supervision loop between clients and plant server. The physical system that has been used consists of an inverted pendulum with a linear motion cart. This choice, in spite of its apparent complexity, has allowed the demonstration of the feasibility of environments created for the remote experimentation of real plants irrespective of their apparent difficulty.

The system that has been built is a development of other environments presented in earlier works. As regards the experimentation interface, it has all the characteristics of earlier environments: interactivity, extensibility, and adaptability. Furthermore, the server-side software has been designed and programmed so that the physical system characteristics are hidden. Thus, the server-side software can easily adapt to the remote control of other plants available in the laboratory. Matlab/Simulink/Quanser have shown to be the right choices, since this software provides the teacher with a highly flexible instrument for creating new practical experiments and also allows the user to have proper control of the control loop.

The visual supervision environment has also been presented. This system is characterized by presenting panels for controlling the camera according to the user's profile. Moreover, the vi-

sual supervision system has lighting control mechanisms which means that the laboratory can be used 24 hours a day.

ACKNOWLEDGMENT

The authors wish to thank the referees of this paper for their constructive comments.

REFERENCES

[1] S. Dormido, "Control learning: Present and future," in *Proc. 15th IFAC World Congr. Automatic Control*, Barcelona, Spain, July 2002, pp. 81–103.
[2] D. Cooper and D. Fina, "Training simulators enhance process control education," in *Proc. American Control Conf.*, San Diego, CA, June 1999, pp. 997–1001.
[3] N. A. Kheir, K. J. Åmstrom, D. Auslander, K. C. Cheok, G. F. Franklin, M. Masten, and M. Rabins, "Control system engineering education," *Automatica*, vol. 32, no. 2, pp. 147–166, Feb. 1996.
[4] P. Antsaklis, T. Basar, R. DeCarlo, N. Harris, M. Spong, and S. Yurkovich, "Report on the NSF/CSS workshop on new directions in control engineering education," *IEEE Control Systems Mag.*, vol. 19, pp. 53–58, Oct. 1999.
[5] T. B. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Trans. Robotics Automat.*, vol. 9, pp. 592–606, Oct. 1993.
[6] B. Aktan, C. A. Bohus, L. A. Crowl, and M. H. Shor, "Distance learning applied to control engineering laboratories," *IEEE Trans. Educ.*, vol. 39, pp. 320–326, Aug. 1996.
[7] D. Gillet, C. Salzmann, R. Longchamp, and D. Bonvin, "Telepresence: An opportunity to develop real-world experimentation in education," in *Proc. Eur. Control Conf.*, Brussels, Belgium, 1997.
[8] S. E. Poindexter and B. S. Heck, "Using the Web in your courses: What can you do? What should you do?," *IEEE Control Syst. Mag.*, vol. 19, no. 1, pp. 83–92, Feb. 1999.
[9] J. W. Overstreet and A. Tzes, "An internet based real-time control engineering laboratory," *IEEE Control Syst. Mag.*, vol. 19, pp. 19–34, Oct. 1999.
[10] C. Schmid, "A remote laboratory using virtual reality on the web," *Simulation*, vol. 73, no. 1, pp. 13–21, July 1999.
[11] A. Zolnay, A. Lassó, H. Charaf, and I. Vajk, "Configurable remote, platform independent control system," in *Proc. IFAC Symp. Advances in Control Education (ACE'2000)*, Gold Coast, Australia, 2000.
[12] D. Gillet, H. A. Latchman, C. Salzmann, and O. D. Crisalle, "Hands-on laboratory experiments in flexible and distance learning," *J. Eng. Education*, vol. 90, no. 2, pp. 187–191, Apr. 2001.
[13] D. Gillet, C. Salzmann, and P. Huguenin, "A distributed architecture for teleoperation over the internet with application to the remote control of an inverted pendulum," in *Lecture Notes in Control and Information Sciences 258: Nonlinear Control in the Year 2000*. London, U.K.: Springer-Verlag, 2001, vol. 1, pp. 399–407.
[14] C. C. Ko, B. M. Chen, J. Chen, Y. Zhuang, and K. C. Tan, "Development of a web-based laboratory for control experiments on a coupled tank apparatus," *IEEE Trans. Educ.*, vol. 44, pp. 76–86, Feb. 2001.
[15] K. Ling, Y. Lai, and K. Chew, "An online internet laboratory for control experiments," in *Proc. IFAC/IEEE Symp. Advances in Control Education*, Gold Coast, Australia, Dec. 2000.
[16] W. Sheng, L. Choo-Min, and L. Khiang-Wee, "An integrated internet based control laboratory," in *Proc. IFAC/IEEE Symp. Advances in Control Education*, Gold Coast, Australia, Dec. 2000.
[17] J. Sánchez, F. Morilla, S. Dormido, J. Aranda, and P. Ruipérez, "Virtual control lab using Java and Matlab: A qualitative approach," *IEEE Control Syst. Mag.*, vol. 22, pp. 8–20, Apr. 2002.
[18] Homepage, Quanser Inc. [Online]. Available: http://www.quanser.com
[19] [Online]. Available: http://www.j3soft.com/~cables/
[20] Homepage, Axis Inc. [Online]. Available: http://www.axis.com

José Sánchez received the computer sciences degree from Madrid Polytechnic University in 1994 and the Ph.D. degree in science from Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain, in 2001.

Since 1993, he has been working at the UNED Department of Computer Sciences and Automatic Control as an Assistant Professor. His current research interests are the design of new systems for control education, virtual laboratories, telepresence, multimedia, and the use of the Internet in education.

Sebastián Dormido received the physics degree from Madrid Complutense University, Madrid, Spain, in 1968.

In 1981, he was appointed Full Professor of Control Engineering at the Faculty of Sciences, Universidad Nacional de Educación a Distancia (UNED), Madrid. His scientific activity includes various topics from the control engineering field: computer control of industrial processes, adaptive systems, model-based predictive control, robust control, and modeling and simulation of continuous processes.

Rafael Pastor received the physics degree from Madrid Complutense University, Madrid, Spain, in 1994.

He is currently working on his thesis on virtual and remote laboratories. Since 2000, he has been working in Department of Computer Sciences and Automatic Control at Universidad Nacional de Educación a Distancia (UNED), Madrid, as an Assistant Professor. His current research interests are real-time control systems, distributed systems, virtual and remote labs and their applications in higher degree courses.

Fernando Morilla received the degree in physics from Seville University, Seville, Spain, in 1979 and the Ph.D. degree from Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain, in 1987.

In 1983, he joined the Department of Computer Sciences and Automatic Control at the UNED. His current research interests include process modeling, simulation and control, tuning and autotuning of PID controllers, predictive control, and control education.