Remote Control Laboratory Via Internet Using Matlab and Simulink

R. PUERTO, L.M. JIMÉNEZ, O. REINOSO

Department of Industrial Systems Engineering, University Miguel Hernández, Elche, Alicante, Spain

Received 26 February 2008; accepted 21 July 2008

ABSTRACT: This article describes the general architecture and application of a remote laboratory for teaching control theory based in Matlab/Simulink. The proposed system allows solving the time and spatial limitations of laboratories that rely on real physical systems used in control courses. In this way, control lab assignments with various physical processes present in the remote laboratories can be performed. Also, some examples that show the validity and applicability of the presented architecture are introduced. © 2009 Wiley Periodicals, Inc. Comput Appl Eng Educ 18: 694–702, 2010; View this article online at wileyonlinelibrary.com; DOI 10.1002/cae.20274

Keywords: control systems; real-time; Internet remote control; Matlab/Simulink

INTRODUCTION

The work presented in this article provides a general architecture for remote execution in real-time of physical processes using the Matlab platform. The main motivation is, on one hand, the lack of lab scale models or real physical systems, and on other, the fact that students of Control Engineering courses have to adapt to the laboratory schedules available in their schools, which are usually too restrictive. This application will allow the students, using the Internet, to simulate the operation of a controller designed for a physical process and then to test this same controller in the real physical process available in the remote laboratory.

Distance learning through the use of remote laboratories via Internet is a highly topical issue as a consequence of the large potential to increase the quality in the process of teaching and learning [1]. A distance laboratory allows users to perform experiments from a remote location, and can be divided into two classes: virtual labs and remote labs [2]. In the virtual labs the students can run simulations remotely with possible animations. Some laboratory skills, such as the statistical analysis of data, can be learned outside the laboratory [3]. However, experimental design can only be learned from using real equipment in real experiments, often through a certain amount of trial and error. On the other hand, remote labs are laboratories where students can interact with actual experiments via the Internet. A more detailed study between remote and virtual labs is presented in Ref. [4]. Without a doubt, the use of real physical systems allows the students to acquire knowledge in a more efficient way than using only simulated exercises. Through remote laboratories, users can plan and manage experiments, analyzing the experimental data as if they were physically present in the laboratory at any time.

Different real experiments through remote laboratories have been proposed in a large number of disciplines [5,6]. In the automatic control discipline, practical experiments with physical systems are significantly important [7–9]. Most of the experiences presented combine different developments that allow make real experiments on control systems through the Internet [10,11].

One of the main features of the system proposed in this article consists of using a set of tools extensively known by the students of automatic control such as Matlab and Simulink [12,13]. These tools enable educators and students to focus on control systems design, implementation, and evaluation rather than on time-consuming, low-level programming. In this sense, Schmid [10] presents a virtual laboratory, which uses Matlab/Simulink for simulations using virtual reality. Bonivento et al. [14] proposed a remote control laboratory based on Matlab/Simulink but with a specific software application developed in Visual C++ to interact with the physical models. In Ref. [15] the authors use the Matlab Web Server tool to allow the remote execution of Matlab from any computer having a web browser.

So, we have chosen the Matlab/Simulink platform (with additional toolboxes) as the development tool of the application proposed in this article for the following reasons: first of all, Matlab, Simulink, and additional toolboxes constitute a reliable platform widely used, with an adequate technical support and an extensively use in control applications. Second, it is possible to develop an application in the laboratory in a lesser time than with other tools or platforms. Also, Matlab provides several

[1]Matlab and Simulink are trademarks of Mathworks Company.

Correspondence to R. Puerto (r.puerto@umh.es).

tools for the remote execution of programs and a real-time control toolbox to manage a physical system through a data acquisition system. Finally, a lot of researches use this platform as a development tool for both, simulation and real-time control of physical systems.

The remainder of the article is organized as follows. Second Section shows a general description of the architecture is introduced. Third Section presents two physical systems connected to RECOLAB with some examples of control design. Fourth Section shows the experience of student usage. Finally fifth Section presents the conclusions.

GENERAL DESCRIPTION OF THE SYSTEM

The main goal on developing the system was to achieve accurate real-time executions over physical systems through Internet, transparently connecting parameterized Simulink schemes with user's control design. In order to do this, it has been necessary to integrate a specific hardware and software architecture.

The general scheme of the application architecture is shown in Figure 1. In this diagram, the hardware and software elements are split into two main blocks: local area where the user works, and remote area where the whole physical system and control elements are located. The detailed elements of local and remote areas are the following:

1. Local area:
   - Computer with Internet connection and an HTTP 4.0 client application. The application is optimized for Internet Explorer 6 and Firefox 1. $\times$ with a minimum resolution of 800 $\times$ 600.
2. Remote area:
   - High speed Internet connection.
   - Computer server: the current implemented system consists of a PC Pentium IV running Microsoft Windows XP operating system.
   - Data acquisition system: NI 6024E acquisition board with analogue and digital I/O.
   - Physical system to control: Two models are currently implemented, a DC motor model 33-002 from Feedback and a ''Airflow Slider Cylinder.''
   - Images capture system and Web video server: an Axis network camera with MPEG-4 video compression streams video and static images to the user.
   - HTTP Server Apache v.2.0.54 with PHP 5.0 module. This server allows the communication of the computers using the http protocol.
   - MATLAB R12 with SIMULINK V. 4.1: executes the program that makes possible the real-time control of the system and the generation of the results in a file.
   - Real-time Windows Target Toolbox V.2.1: this toolbox allows executing Simulink schemes in real-time. For this purpose, it provides the necessary blocks for the interaction with the data acquisition system.
   - Control System Toolbox 5.1.

Functionality of Software Application

This application has two aspects clearly differentiated:

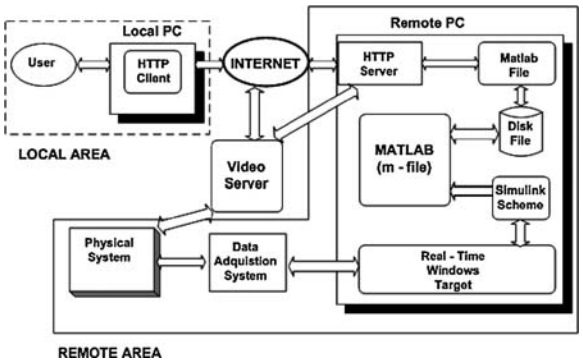

Figure 1 General architecture diagram.

1. Web application: this includes client-server communication using HTTP/HTML protocol, the user interface, user's access control, and the main Common Gateway Interface (CGI) application. CGI is a standard for interfacing external applications with information servers, such as HTTP or Web servers. This application is in charge of resource access and communication between Apache and Matlab.
2. Real-time application: this is a set of predefined Simulink control schemes and Matlab code, based on Real-time Windows Target toolbox, which implements the real-time execution of Simulink schemes over a specific physical system.

Different programming languages and development tools has been used for each part. The non-critical task such as user interface, security access and resource sharing, have been coded in PHP (v.5.0) [16] running over an Apache HTTP server [17]. PHP is a popular script language that has been chosen as far as it is an open language widely supported by most Web servers and O.S. platforms, and with an extensive library that supports every network protocol and data base access. PHP code runs on the web sever so it shows a controlled environment for the programmer and can communicate with any other process running in the server (Matlab application in our case).

Of course, PHP as a scripting language is not suitable for real-time applications. The hard real-time core of RECOLAB (feedback control of physical systems) is developed in Simulink and compiled with the Real-Time Windows Target tool [18]. Figure 2 shows the functional flow graph of main system application. Following is a description of the main components of the software application.

User Interface of RECOLAB. The user interface of RECOLAB is based on standard HTML 4.0 mark-up language. The HTML code is generated dynamically through PHP scripts. The HTML layout is separated from the contents in order to make of RECOLAB a flexible platform. All formatting of data is based in CSS-styles and PHP layout predefined functions for forms. The data content is stored in configuration text files making quite easy and flexible to add new physical systems, and control schemes without any modification in the PHP code.

The user interface shown in Figure 5 is based in HTML forms dynamically generated in PHP from the configuration

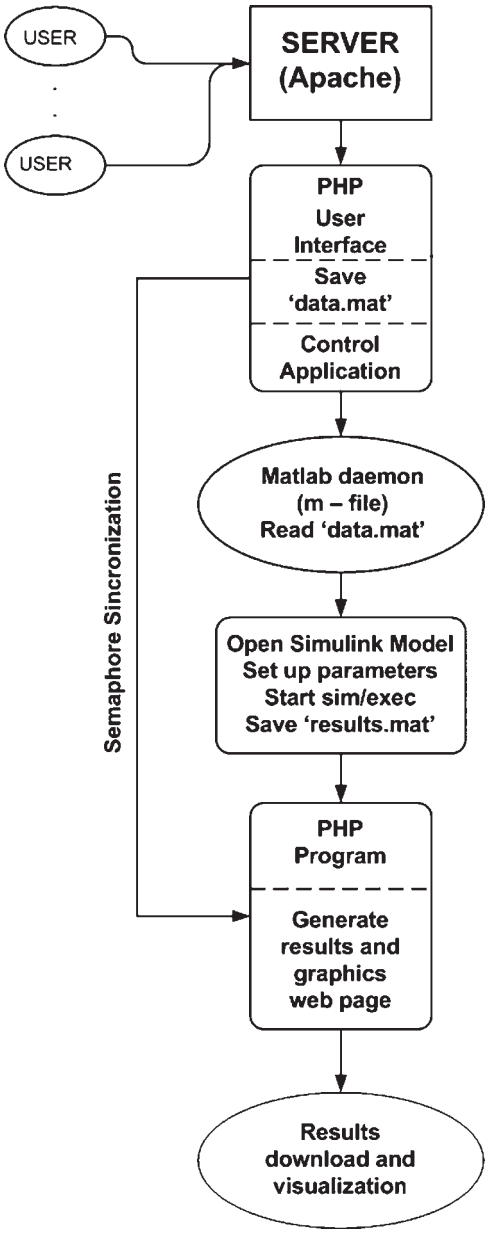

Figure 2 Functional description of RECOLAB application.

files. These pages also act as CGI using GET/POST parameters for passing state information. Interactive actualization of forms under user selection is done through CGI calls to the server and not through client-side scripting. In this way security access can be managed more effectively.

Every control Simulink scheme can be executed in simulation or in real-time over the physical systems using an acquisition board. The user can choose different control strategies and regulators, and can select its parameters according to the simulation experiments that are being carried out.

After an execution, the whole sampled or simulated variables of the process can be downloaded. The results page shows part of this information as a graph (the output of the system) (see Fig. 6).

CGI Application. This application links the input data from the user with the RT control Matlab program, synchronizes the different modules, and records back the output data to the user. The non-RT management tasks have been developed in PHP, using files as basic mechanism to communicate with real-time modules. In the next section we comment how this communication is performed.

An additional task performed by this module is Access Control. Most of Recolab site can be accessed from Internet without any restriction, as far it is an open teaching tool. The most critical point in Recolab is the real-time execution of the Simulink schemes over physical systems. In order to have a more efficient use of the laboratory resources, this processing needs to be done under a validated access.

The validation system used for Recolab is integrated with the CGI application and it is focused in access control of application resources. The validation system is coded in PHP so it permits to be integrated with the main application. Rendered options by the PHP code are based on the validation done by user. In this way there is no need to maintain different PHP code for validated or not validated users. If an user request an option that needs validation, the PHP code itself request for user validation.

The validation process is based in a user name and a password. The control access to the resources is based on a user name and a priority level. Most of the resources in Recolab use an access control by level.

Real-Time Control Application. This application runs the real-time feedback control using the specified control scheme over the physical system. This application has been developed using the following tools: Matlab, Simulink, and Real-Time Windows Target. This last tool allows generating the real-time code to execute the Simulink schemes using an acquisition board (and therefore, over the physical system, connected to it). The fact of executing a Simulink scheme directly is an added advantage, as far as the complexity and spent time working with Simulink are drastically decreased, allowing an easy and fast design, and modification of control schemes.

Real-time control using Simulink is performed through Real-Windows Target. This toolbox generates source code that translates feedback control of Simulink scheme. This source code is compiled with Watcom C compiler in order to generate the real-time execution code. Each Simulink scheme depends on the type of regulator, control method and the physical system to be controlled, but there are some common elements:

- A Regulator (specific in each case, P, Pi, PD, PID, state feedback, etc.).
- Acquisition board interaction block that provides Analog and digital input/output interface with the physical system. In our case this hardware is a Nation Instruments 6024E board that is managed through an specific library block.
- Auxiliary blocks for setting an initialization of actuators, and sensors.

Once designed and compiled the Simulink scheme, this code can be executed from command line using *set_param* Matlab instruction [18]. This is the mechanism used in RECOLAB to connect the real-time execution with the user interface. A Matlab code based in this instruction implements

the selection of control model, type of regulator and more important, updates dynamically the parameters of regulator without any additional recompilation, as far the structure of regulator or control scheme does not change. For example, a PID regulator has several adjustable parameters: gain, poles, and zeros (vectors of rank 2). These five variables can be accessed from command line using *set_param* instruction allowing been modified in each real-time execution without recompiling the Simulink scheme. Finally, this Matlab instruction can be used to start and stop the real-time execution. Table 1 shows an example of real-time execution of a precompiled Simulink scheme.

Real-Time Task and CGI Interprocess Communication. The interface to real-time control execution is handled by a CGI application developed in PHP. This application starts the Matlab real-time control task with the parameters requested from the user and synchronizes its execution using a file-based semaphore mechanism. The whole detailed procedure implemented to run a Simulink simulation or a real-time execution is as follows:

(1) The user (client) connects to the RECOLAB server and chooses through the user interface the simulation or real-time execution of a specified Simulink scheme. For example, the user can select the type and the parameters of the controller, giving the values of the coefficients of the polynomials that conforms the numerator and denominator of its transfer function or the state feedback gain matrix.
(2) PHP module executes the CGI associated to the user interface webpage, stores the user data in a file. At this point launches the Matlab control session in background, dealing with access privileges and resource sharing between the concurrent connections. It also tracks for any problem in the real-time code execution. Together with this action the specific M-file is executed under Matlab. This script code reads the configuration file written by the CGI and loads the adequate Simulink file, updating its parameters (selected by the user), and launching the real-time execution or simulation.
(3) Once finished the simulation or the real-time execution, all the result data is stored (included the figures with time response) and the PHP task is signaled, by means of semaphore mechanism, showing that the execution is ended. Main control CGI (PHP) reads the experiment and generates the HTML page with the graphs of the results. In this same page, the user can download the experiment data file (Matlab .mat file type) for later analysis in local computer.

This procedure is shown schematically in Figure 2. There must be remarked that only one execution can be done at the same time, as far as it must use the physical system and real-time resources. This is why a semaphore control mechanism based in files has been implemented. When more than a request is accepted by the Apache Web server they are queued waiting until previous execution ends or a timeout expires. Obviously, only one user can execute a real-time experiment with the servomotor at the same time.

The system permits also simple simulations of Simulink schemes, indeed it is a recommended step before a real-time execution. In this case, and when only simulations are requested, they can be done concurrently.

EXPERIMENTAL SYSTEMS AND EXAMPLES

This section describes the physical systems connected to RECOLAB [19] and some control examples. We only describe real-time examples (not simulation) since this is the most interesting feature of RECOLAB. Currently, the laboratory has two physical systems connected to RECOLAB, which allows performing a wide-range of control experiments. These systems are a DC motor and a sliding cylinder.

The DC motor consists of a Feedback 33-100 mechanical unit [20]. The mechanical components of this unit are: the DC motor itself, an analogical tachometer, an analogical potentiometer with a position signal, absolute incremental digital encoders and a magnetic brake. Figure 3 shows this system with a camera, which allows sending the acquired images to the video server.

The sliding cylinder consists of a tube made of methacrylate where an object slides propelled by an air flow obtained from a DC fan. A photoelectric sensor placed in the top of the tube obtains the position of the object. Figure 4 shows an image of the sliding cylinder.

When a user enters to the system, he is allowed to perform a simulation or a real-time execution for each one of the systems mentioned above. The different types of experiments and control schemes applicable to each system are the following:

Table 1 Matlab Example Code for Real-time Execution of Simulink Scheme

```
% Matlab code for Simulink file emmotorvelpid.mdl
open_system('emmotorvelpid');
set_param('emmotorvelpid','stopfcn','generares');
set_param('emmotorvelpid/Ref','after',instruct.Ref);
set_param('emmotorvelpid/Kp', 'Gain', instruct.Kp);
set_param('emmotorvelpid/PID', 'denominator', instruct.den);
set_param('emmotorvelpid/PID', 'numerator', instruct.num);
save_system('emmotorvelpid');
set_param('emmotorvelpid','simulationcommand','connect');  %  Connect
Real - Time Kernel System
set_param('emmotorvelpid','simulationcommand','start');
```

Figure 3 Feedback 33−100 mechanical unit. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

- DC engine
  - System identification.
  - PID controllers for velocity and position control.
  - Algebraic controllers based on poles placement technique.
  - State feedback control with and without reference following.
- Sliding cylinder
  - PID controllers.
  - Algebraic controllers based on poles placement technique.
  - Algebraic controllers based on minimum- and finite-time.
  - Minimum variance controller with and without integral action.
  - State feedback control.
  - Predictive control.
  - Optimal control.

As can be seen, the identification experiment is only allowed for the DC engine, whereas this is not the case for the sliding cylinder because of its complexity. However, this complexity allows on the other hand performing more complex controllers, which have no sense in the DC engine case.

Once the user has accessed to the system, a page appears in which all needed data to perform the real-time execution are requested. Then, the process to follow in order to specify the type of experiment to perform is the following:

1. Select the physical system in which the experiment is going to be performed (DC engine or sliding cylinder).
2. Select the control model to apply. Depending on the type of the chosen physical system, these models will allow different options. For example, for the DC engine control we can use position and velocity feedback, state feedback, etc.
3. Select the type of controller. For example, PID controllers, state feedback, predictive control, etc. As mentioned before, the types of controllers applicable in each case will depend on the system and the chosen control model.

Figure 4 Sliding cylinder.

4. Simultaneously to the selection of the previous options, the page is dynamically modified to show the chosen system (the Simulink diagram which will be executed in real-time) as well as the parameters associated to the chosen controller, allowing to input these parameters.
5. Once the user has introduced all data, the experiment can be performed, and when it is finished, the output signal is shown in the screen. Moreover, the application allows to download a ''.mat'' file with the values of the most significant signals (output, control action, etc.) in order to be analyzed by the user.

To illustrate the described process, an example of velocity DC control using a PID controller is presented next. To do this, the system, the execution model and the controller type are chosen. Then, the webpage shows the Simulink diagram to be executed as well as the PID controller transfer function, allowing the user to input data. In this case, the controller transfer function has the following form:

$$R(z) = \frac{K_p(z-a)(z-b)}{z(z-1)}$$

The parameters of the controller transfer function are computed from the specifications described in the laboratory assignments for the students, achieving these specifications using the root locus design method. Figure 5 shows the content of the webpage before and after the execution, in which all the described items can be seen.

After the execution, a webpage with the graph of the output signal is presented to the student (in this case the engine velocity). This page is shown in Figure 6, which also shows that

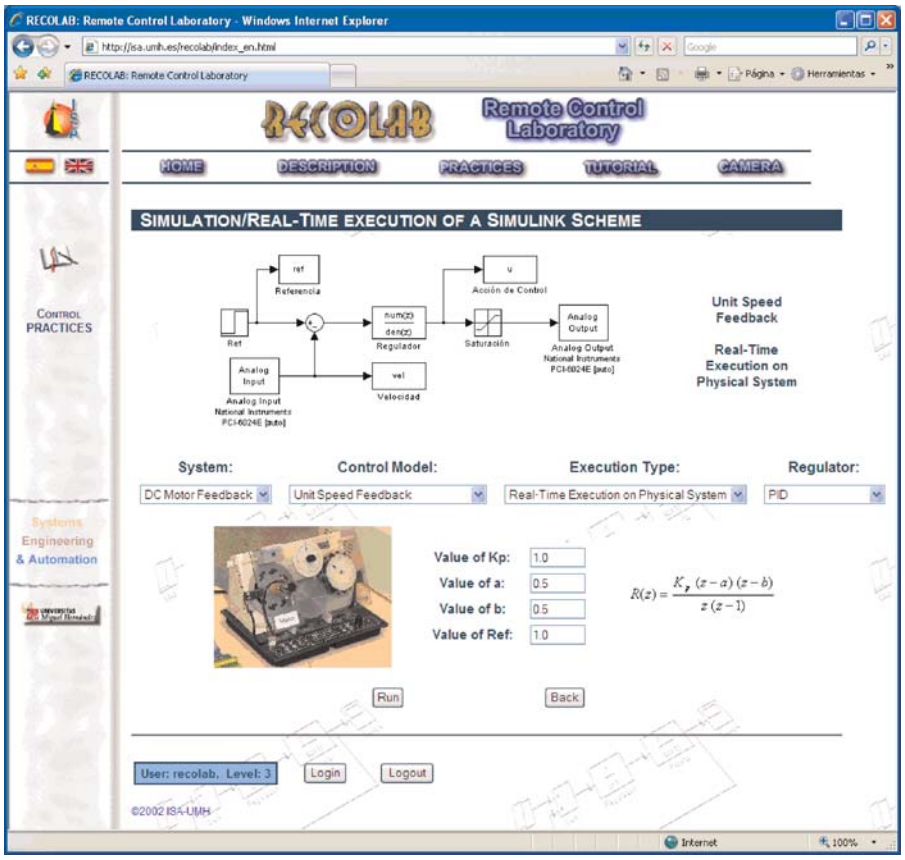

Figure 5 User interface of RECOLAB. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

the system allows the download of all the signals involved in the execution in order to be analyzed by the students.

SYSTEM USAGE

RECOLAB system started being used in 2003. So far, the response of the students has been satisfactory. First, there have been a high number of accesses to the simulation system; and more importantly, there have also been a high number of accesses to the real-time control of the physical equipment.

Considering the four subjects where this physical equipment was used during practice sessions from the 2003 to 2004 academic year, there were 172 students which could be potential users of the RECOLAB system. Even though these students had attended the practice sessions (it was compulsory, as the system was not fully tested), 59% of them accessed and used the system to repeat remotely some of the practice sessions they have already attended. During the academic year 2004/2005, there were 183 students in the four subjects where RECOLAB system was used. In this academic year 74% of the students accessed to the system and used it to repeat some practice sessions. During the academic year (2005/2006) there were 195 students enrolled in such subjects. In this case, most of the students (91%) connected to the system. Finally, during the last academic year (2006/2007), from a 173 students enrolled, 170 used the system to make remote practice through Internet. Figure 7 shows the percentage of students using this tool to practice on control systems through Internet. As a conclusion, the students were able to continue working after the laboratory sessions: some of them performed additional experiments with different parameters or different control schemes; while others simply finished the work they have not been able to finish in the laboratory due to time limitations.

The average number of accesses per student was 5.3; meaning that students preferred to split the work in different sessions. The average connection time per session was 35 min. This behavior is practically similar to the previous academic years. Figure 8 shows the number of accesses per student and the connection time per session made by the students during 2006/2007 academic year. These are other advantages of the RECOLAB system when compared to a traditional laboratory exercise: it is possible to choose the number and duration of the sessions used to perform a certain experiment with the physical system.

All these data show that the tool developed helps the students, giving them a high degree of flexibility to perform the practice work whenever it suits them.

The effectiveness of the tool was evaluated by the participants. Table 2 summarizes the survey results from 694 undergraduate students at the Department of Industrial Systems Engineering at Miguel Hernández University, Spain, during the last years. Overall, the survey clearly revealed that the tool received a very positive feedback. However the tool can be

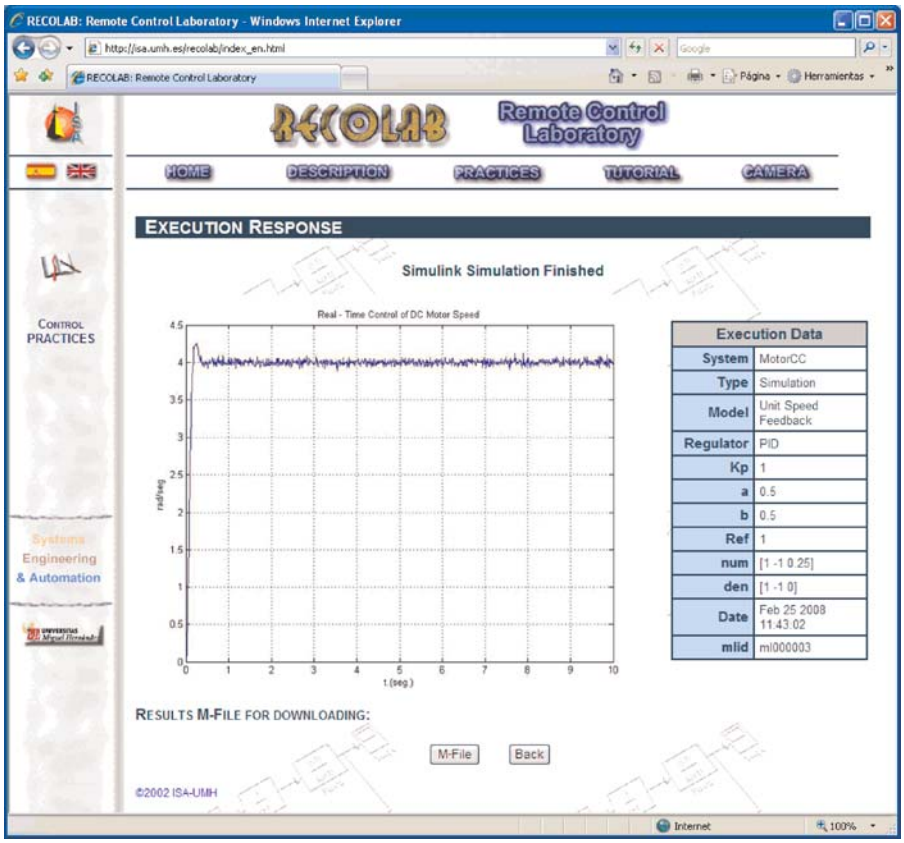

Figure 6 Results webpage. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

improved in partial aspects. Students requested more physical systems were included in the Internet laboratory so that they could manage and prove the control systems over more physical systems. Also, in some specific occasions some students had connection problems as a consequence of other students were using the remote physical system to prove the designed control scheme and the access time was higher than it was to be expected.

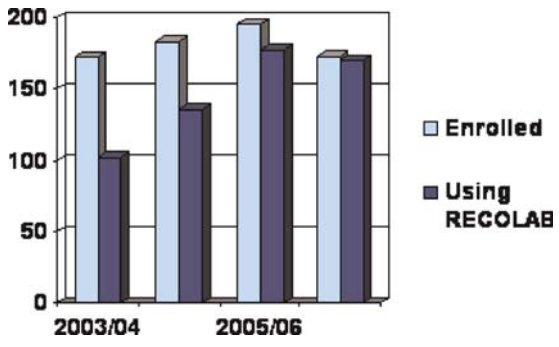

Figure 7 Percentage of students using RECOLAB as a no-compulsory system to make practice in control engineering. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

CONCLUSIONS

In this article, an architecture that allows real-time control through the Internet has been presented. Besides, the proposed system allows data interchange (results, data plots, video, etc.) between the user and the remote laboratory. This architecture based on Matlab and Simulink has been applied on a DC motor remotely controlled. However, even though the application example presented in the article is focused on the control of a DC motor, the same Internet-based real-time control scheme can be applied to many other laboratory equipments.

The main advantage of the proposed system is that it helps the student to perform practice experiments remotely. Apart from this teaching application, the tool developed and the ideas presented in this article can also be used to test new control schemes over different physical equipments. As a consequence, and due to the excellent results achieved in the use of the remote control laboratory, during this academic year (2006/2007) professors and lecturers of the involved courses decided to

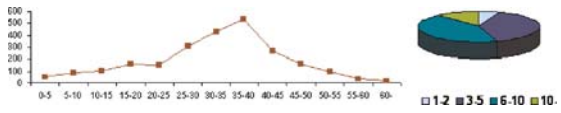

Figure 8 Number of accesses per student and connection time per session during 2006/2007 academic year. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Table 2 Students Survey Results

Questions	Fully disagree (%)	Partial disagree (%)	Partial agree (%)	Agree (%)	Fully agree (%)
a. Were the learning procedures clearly defined?	0	0	10	23	67
b. Was clear the use of the system?	0	0	5	18	77
c. What about the help system?	0	0	0	12	88
d. Do allow the system help to understand the importance of practical experiments in the subjects enrolled?	0	0	0	10	90
e. What is your opinion about the use of this tool to improve your knowledge in control engineering?	0	0	2	15	83
f. Overall, how would you rate this tool?	0	0	1	14	85

make compulsory for the students the use of this remote laboratory.

Even though the current version of RECOLAB is operative and working, the system is continuously under development. Some improvements for future versions include: supervision of variables using a new Java client; increase in the number of practice sessions and control schemes available; transmission of higher compressed video in order to deal with limited bandwidth networks; and addition of control theory tutorials.

REFERENCES

[1] A. Ausserhofer, Web-based teaching and learning: A panacea? IEEE Commun Mag 37 (1999), 92–96.

[2] M. Casini, D. Prattichizzo, and A. Vicino, The automatic control telelab: A user-friendly interface for distance learning, IEEE Trans Educ 46 (2003), 252–257.

[3] S. Ayasun and G. Karbeyaz, DC motor speed control methods using MATLAB/Simulink and their integration into undergraduate electric machinery courses, Computer Appl. Eng. Educ 15 (2007), 347–354.

[4] M. Exel, S. Gentil, F. Michau, and D. Rey, Simulation workshop and remote laboratory: Two web-based training approaches for control, Proc Am Control Conf 5 (2000), 3468–3472.

[5] M. Shor, Remote-access engineering educational laboratories: Who, what, when, why, and how? Proc Am Control Conf 4 (2000), 2949–2950.

[6] R. Robson and M. Shor, A student-centered feedback control model of the educational process, Front Educ Conf 2 (2000), 14–19.

[7] N. A. Kleir, K. J. Astrom, D. Auslander, K. Cheok, G. F. Franklin, N. Masten, and M. Rabins, Control systems engineering education, Automatica 32 (1996), 147–166.

[8] R. Pastor, J. Sanchez, and S. Dormido, Related: A framework for publish web laboratory control system, In IFAC Internet Based Control Educ 1 (2001), 207–212.

[9] R. Puerto, O. Reinoso, R. Neco, N. Garcia, and L. M. Jimenez, Remote lab for control applications using matlab, In IFAC Internet Based Control Educ 1 (2001), 121–127.

[10] C. Schmid, Virtual control laboratories and remote experimentation in control engineering, Proc 11th Annual Conference on Innovations in Education for Electrical and Information Engineering, 213–218 (2001).

[11] R. Puerto, L. M. Jimenez, O. Reinoso, C. Fernandez, and R. Neco, Remote control laboratory using Matlab and Simulink: Application to a dc control model, In IFAC Internet Based Control Educ 1 (2004)), 53–59.

[12] W. Dixon, D. Dawson, B. Costic, and M. de Queiroz, Towards the standardization of a matlab-based control systems laboratory experience for undergraduate students, Proc Am Control Conf (2001), 1161–1166.

[13] W. Dixon, D. Dawson, B. Costic, and M. de Queiroz, A matlab-based control systems laboratory experience for undergraduate students: Toward standardization and shared resources, IEEE Trans Educ 45 (2002), 218–226.

[14] C. Bonivento, L. Gentili, L. Marconi, and L. Rappini, A web based laboratory for control engineering education, Second International Workshop on Tele-Education in Engineering Using Virtual Laboratories, 2002.

[15] J. L. Diez, M. Valles, A. Valera, and J. L. Navarro, Remote industrial process control with Marlab web Server, Internet Based Control Educ 1 (2002), 139–143.

[16] Online PHP documentation http://www.php.net.

[17] Online Apache documentation http://www.apache.org.

[18] Online Matlab documentation http://www.mathworks. com.

[19] RECOLAB web portal http://isa.umh.es/recolab/.

[20] Online Feedback educational models documentation http://www.fbk.com.

BIOGRAPHIES

Rafael Puerto was born in Alicante (Spain) in 1969. He received his MSc degree in Computer Science in 1994 and his is working in his PhD in Control Engineering at the Miguel Hernández University of Elche (Spain). From 1996 to 1999 he worked in the Polytechnic University of Valencia. His current position is as assistant professor of Automatic Control (1998–), Miguel Hernández University of Elche, teaching subjects in the area of control engineering. His research interests include Real-Time Systems, Multirate Control, Advanced Control Systems, and Control Engineering Education. He has taken part in several national and European research projects.

Luis M. Jimenez was born in Avila (Spain) in 1967. He received his MSc degree in industrial engineering in 1992 from the Polytechnical University of Madrid (Spain). His current position is as assistant professor of Automatic Control (1998−), Miguel Hernández University of Elche, teaching subjects in the area of control engineering. From 1995 to 1998 he worked as assistant professor at University of Alicante. His research interests include Robotics control, Real-Time Systems, Advanced Control Systems, and Engineering Education.

Oscar Reinoso received the industrial engineer and PhD degrees from Polytechnic University of Madrid (UPM) in 1991 and 1996, respectively. From 1994 to 1997 he worked in the Research & Development Department of Protos Desarrollo in a visual inspection system. Since 1997, he has been at the Miguel Hernández University, as professor in control, robotics, and computer vision. His research interests include Robotics, Teleoperated Robots, Climbing Robots, Visual Servoing, Visual Inspection Systems. He is author of several books, articles, and communications in the cited topics. Prof. Reinoso is a member of the CEA-IFAC and IEEE.