Education Research in Engineering Studies: Interactivity, Virtual and Remote Labs

J.L. Guzmán, H. Vargas, J. Sánchez,
M. Berenguel, S. Dormido, F. Rodríguez

Abstract

In few years, new teaching methods in the education field have appeared, taking advantages from the *New Information and Communication Technologies* (NICT). They allow teachers to find innovative techniques in order to enhance student's motivation and improve their education: multimedia tools, hypertext systems, interactive tools, information exchange between teachers and students through Internet, information access from any part of the world without temporal constraints, etc. The rapid evolution of computers, Internet, and the distributed computing facilitates the development of feasible and cost-effective solutions. However, sometimes it is necessary to incorporate contents with a strong experimental component and let the students apply the acquired knowledge on real systems. Traditionally, local simulation tools or local laboratories have been used for this purpose, but with spacial and time constraints. Nowadays, and thanks to advances in the NICT, especially in Internet technologies, the laboratory environment can be transferred to distance education. Based on these premises, two new concepts have appeared within the distance education framework: virtual laboratories, which are a new kind of simulation tools allowing the simultaneous use of remote simulation modules by students; and remote laboratories, which permit students to remotely put into practice concepts without requiring their presence at the place where the hardware is placed in such a way that they can control and monitor physical devices 24 hours a day at any time and anywhere. This work shows the development of a virtual and remote lab using as case-study the control level of a tank, where *Easy Java Simulations* (EJS) has been used as programming environment. Firstly, the real process is described, as a typical case study in an introductory course on Automatic Control. The process model, lab sessions, and the student work are included. Later, the complete development of this virtual lab is described, showing its most relevant steps: the programming of the model, the lab window configuration and, the operating controls to facilitate the student's understanding of the process to be controlled. Furthermore, based on the paradigm Model-View-Controller, the transformation of the virtual lab into a remote one is presented. Finally, it is described the integration of these labs in a high level environment that includes the other actors involved in any traditional engineering laboratories, for instance, teachers, user's manuals, assignments,, exercises, etc. For this task, *eMersion*, a tool to develop

flexible learning resources using WWW services, has been used. The lecturer must transmit his/her experiences and ideas to the students but not all, some of these ideas have to be found by themselves. These tools are very powerful elements helping the students to enhance their skill, motivation, and ability to understand and solve engineering problems. The experiences with the kind of tools used in this work are considered a great stimulus for developing the student's engineering intuition and allow students of any part of the world (including those of developing and transition countries) to access to (sometimes) scarce lab resources.

Nomenclature

Acronym	
CAD	Computer Aided Design
DOF	Degrees of Freedom
EJS	Easy Java Simulations
EPFL	Ecole Polytechnique Fédérale de Lausanne
HTML	HyperText Markup Language
IEEECSS	IEEE Control System Society
IFAC	International Federation on Automatic Control
IP	Internet Protocol
IT	Interactive Tool/Tools
LAN	Local Area Network
MIMO	Multiple Inputs Multiple Outputs
MVC	Model-View-Control
NICT	New Information and Communication Technologies
ODE	Ordinary Differential Equations
PID	Proportional Integral Derivative
PPP	Point-to-Point Protocol
SCADA	Supervisory Control and Data Adquisition
SISO	Single Input Single Output
TCP	Transmission Control Protocol
WWW	World Wide Web

1 Introduction

New teaching methods and innovative techniques have lately appeared showing the influences, advantages, and drawbacks that NICT have provided to the teaching field. As soon as the technology progressed, new teaching tools began to appear. These advances have produced a creation of new techniques and methods opening a wide range of possibilities not only in the educational field (distance education, interactive information, remote, and virtual laboratories,...), but also in the industrial area (remote control, remote management, flexible timetable,...). The rapid development of Internet and distributed computing facilitates the development of feasible and cost-effective solutions [17], [32].

The main advantages of NICT are related to teleaccess, teleoperation, and telecontrol. Until recently, the exchange of information was done using local networks for several reasons: information safety, narrow bandwidths, limited tools for the exchange of information, etc. Thanks to the advances in Internet technologies, a new method for accessing the information has appeared, the *teleaccess*, providing a way to access the information safely from any part of the world without temporal constraints. Both in the industrial and research/educational fields, high costs are often related to the restricted use of (usually expensive) systems with usage-time constraints, or to the displacements required to control them. With *teleoperation* (extension of sensorial capabilities and human skills to a remote place) and *telecontrol* (the specific part of teleoperation whose goal is to send commands to the actuators) technologies, it is possible to control systems remotely through the Internet, thus helping to reduce displacement costs and allowing one to extend the use of time-limited resources or equipment [8], [17].

In the same way, the presentation of information by the teachers has been ameliorated along the years, from the classical lectures based on blackboards and sliders to the use of digital sliders and the providing of information through Internet. However, this fact has considerably been improved with the appearance of *Interactive Tools* (IT). Although Interactivity can be defined in several ways, from a teaching point of view an IT can be defined as a collection of graphical windows whose components are active, dynamic, and/or clickable; and that is intended to explain just a few concepts. The use of interactive and instructional graphic tools would reinforce active participation of students. For educators, this kind of tools can provide a very useful way to test main ideas and to realize how difficult explaining a particular concept to students is [19], [15], [16], [34].

Many advances on education are supported by the fact that the current society is involved in this technological revolution. Currently, the students begin to work with a computer at high school, being able to perform digital information treatment, Internet navigation, and communicate by email. So, students have usually a great knowledge about the use of computers before starting university studies. Therefore, these skills allow the university teachers to advance and create new teaching methods based on the new technologies. Nowadays, it is very common to find theoretical lessons given by means of digital sliders or computer-based simulations as complements to the traditional lectures, web sites for the subjects where students can find digital information, virtual courses where students learn at

home through Internet, interactive information to motivate students, and so on. These previous advances aim at enhancing students' motivation and improving their education though multimedia tools, hypertext systems, interactive systems, information exchange between teacher and student through the Internet, access to information from any part of the world without temporal constraints, etc. All these methods are involved and/or described by new terms such as: online learning, e-learning and distance learning [35]. Online learning constitutes just one part of technology-based learning and describes learning via the Internet, intranet and extranet. The levels of sophistication of online learning vary. A basic online learning program includes the text and graphics of the course, exercises, testing, and record keeping, such as test scores and bookmarks.

A sophisticated online learning program also includes animations, simulations, audio and video sequences, expert discussion groups, online mentoring, links to material on a corporate intranet or the Web, and communication with corporate education records. The term *online learning* can be used synonymously with *Web-based learning* or *Internet-based learning*. The term *e-learning* covers a wide set of applications and processes, including computer-based learning, Web-based learning, virtual classrooms, and digital collaboration. E-learning can be defined as the delivery of content via all electronic media, including the Internet, intranets, extranets, satellite broadcast, audio/video tape, interactive TV, and CD-ROM. For this purpose the term e-learning is used synonymously with technology-based learning. Terms like e-learning, technology-based learning and Web-based learning are defined and used differently by different organizations and user groups. Distance learning is formed by all characteristics of e-learning plus text-based learning and courses conducted via written correspondence [35]. *Web-based experimentation* differs from more traditional e-Learning solutions in the sense that it is more service-oriented than content-oriented. What the students do with the resources is more important than the resources themselves. The key services sustaining Web-based experimentation include remote accesses to real laboratory resources, their simulations, analysis tools and share spaces. These services have to be fully integrated and able to exchange data in real-time to effectively support typical collaborative hands-on learning activities. The ideas presented in this chapter are focused on Web-based experimentation [39].

All these advantages are useful for subjects without a strong practical component, but there exist other subjects with strong experimental contents that require a new element allowing the students to apply the acquired knowledge. Traditionally this element has been a local simulation tool or a local laboratory, this being used by the students to perform several practical exercises using real systems but with space and time constraints. Nowadays, thanks to advances in the NICT, especially in Internet technologies, the laboratory environment can be transformed for its use into distance education. By this, two new concepts have appeared within the distance education framework, *virtual laboratories* and *remote laboratories*. The first of them is a new kind of simulation tool much more powerful than the traditional ones, allowing the simultaneous use of remote simulation modules by the students. On the other hand, remote laboratories allow the students to perform the main laboratory activities remotely, without requiring the presence of the students at the place

where the hardware is placed, in such a way that the students can control and monitor physical devices 24 hours a day at any time and anywhere, interacting with the teacher without having to move to the university. Remote labs help the students to put into practice what they have learned by remotely accessing real systems [8].

Another important impact of the NICT on education has been the *Interactivity*. Traditionally, all the information was presented as static text and formulas. In this way, the students' motivation was difficult to enhance, being necessary turning to exercises or general questions in order to keep the students' attention. Based on a traditional saying which expresses *"an image means more than one thousand words"*, the information presentation has been improved along the years, first with the use of slides, and after that using digital slides. These new elements are a great support to traditional education, being possible to improve by means of *visualization*; that is, the way of acting with explicit attention to potential specific representations in order to explain abstract concepts. This capability is currently well developed in the Computer Science field, where high quality 3D objects can be represented in a computer reaching a very good reality approximation. However, when one looks at some elements or graphics, it is usual to think: what can i do with it?, or how could this element be changed?. That is, people look for some interaction with the graphic elements. A cause-effect relation is required where visual information is not enough, but also the available actions to be done are required. This feature of providing interaction to the elements is known as *Interactivity*. So, the use of Interactivity on education provides a wide range of possibilities to both teachers and students. Teachers can use interactive presentations where not only the meaning of the concepts is provided, but also how these concepts are related with others, or how such concepts are affected by some input modifications. On the other hand, students can use interactive web sites or interactive computer-based tools, as a way to study theoretical concepts abstracted by means of interactive elements [19].

These advances have had, and currently have, a great impact in engineering education. A good example is the Automatic Control field. The professional success of an engineer strongly depends on the education received during the academic years. For this reason the search of new teaching methodologies and techniques has required a continuous attention by teachers and researchers at the university. The advances and incentives for teaching improvements are promoted by special committees, existing even exclusive research lines on this topic. Examples are the IFAC [23] and the IEEECSS [21], supporting the *IFAC Technical Committee on Control Education* [24] and the *IEEE Technical Committee on Control Education* [22] respectively, which evolve:

- University education and continuing education issues in control engineering.
- Methodology for improving theory, practice, and accessibility of control systems education.
- Control engineering laboratories, experiments, computer aided design, distance and virtual education technologies.
- E-learning and Internet-based teaching technologies.

- Cooperation and technology transfer between academia and industry.

So, the presence of these committees in the most important Associations on Automatic Control indicates the great interest placed on the education area, requiring new advances and research in the same way that in other technical areas. In [30] the authors recommend to the control community to *invest in new approaches to education and outreach for the dissemination of control concepts and tools to nontraditional audiences.* An important feature of Automatic Control education is the continuous use of experiments and the development of new laboratories and software tools. These are much easier to do than ever before and also more important. Laboratories and software tools should be integrated into the curriculum, including moving beyond their current use in introductory control courses to increased use in advanced (graduate) course work.

The development of such complex interactive resources is time consuming. Then, their reusability among the academic community is limited by the development overhead necessary to adapt them to different educational contexts and needs. Finally, interoperability with other resources integrated in learning environments cannot easily be implemented and ensured. This chapter describes an easy and simple approach to integrate both simulation and teleoperation Java applets within a Web-based environment that is dedicated to collaborative Web-based experimentation. The simulation Java applets are developed using EJS authoring tool [10], open-source solution designed to help educators and scientists creating discrete computer simulations in Java enabling virtual experimentation. The teleoperation applets connected to real distributed laboratory resources enable remote experimentation. It is also shown how the combination of these virtual and real learning resources complement each other from a learning point of view providing they can easily exchange data. This combination is performed using *eMersion* [39]. The eMersion environment is a learning environment dedicated to Web-based experimentation. It is developed and deployed at the EPFL in Switzerland since 2000 to support interactive simulation and hands-on laboratory activities in engineering education. The proposed approach should help educators facing the challenge of combining high-level Web services or components provided by various academic institutions to integrate them into comprehensive learning frameworks.

Therefore, this chapter describes how a specific lab session can be represented in a Java Applet providing interactive virtual simulation capabilities and also including remote experimentation. After that, the developed tools will be included in a collaborative Web-based experimentation web site. Hence, the first section is devoted to describe the case study used as example along the chapter. EJS is presented in section 3. The development of the virtual and remote laboratory is detailed in section 4. Section 5 is devoted to show how the eMersion environment is used to include EJS applications in a Web-based experimentation web site. Finally, some conclusions are outlined.

2 Case study: tank level control

This section describes the real process that will be used through the chapter in order to develop a virtual and a remote laboratories, both being included in a Web-based collaborative environment. The selected example is focused on the Automatic Control field and it is represented by a two-coupled tank system. The description of the process will be presented as a case study for teaching the main concepts of an introductory course on Automatic Control.

2.1 Introduction & setup of the lab session

The lab session is oriented towards the review of basic continuous time modelling, simulation and control design concepts, using as a test bed the tank level control problem. As is common in control engineering practice, the students have to follow the main steps in the design:

1. To identify the process variables to be controlled and the required precision.
2. To identify the process variables to be governed by the control actuators to maintain the controlled variables within prescribed limits.
3. To select/analyze the sensors.
4. To select/analyze the plant actuators required to implement the control actions.
5. To select/analyze the other elements of the control system: computer, supervisory control and data acquisition system (SCADA), communications, etc.
6. To develop a mathematical model describing the main system dynamics, including the identification of unknown parameters, validation with new data and comparison with real results.
7. To design controllers able to fulfil the required specifications and to analyze the performance of the closed loop control system when applied to the mathematical model developed in step 6, considering different setpoint profiles, disturbances and analyzing the effect of plant-model mismatch, thus modifying the control system to achieve the desired performance.
8. To implement the control system.
9. To test the controller in the real plant to assess the fulfilment of the control specifications.

The installation over which the students perform the lab sessions is a part of a quadruple tank process [25], that permits the implementation of both SISO and MIMO control schemes (see Figures 1(a)-(b)). The main objective of the lab session is to control the level

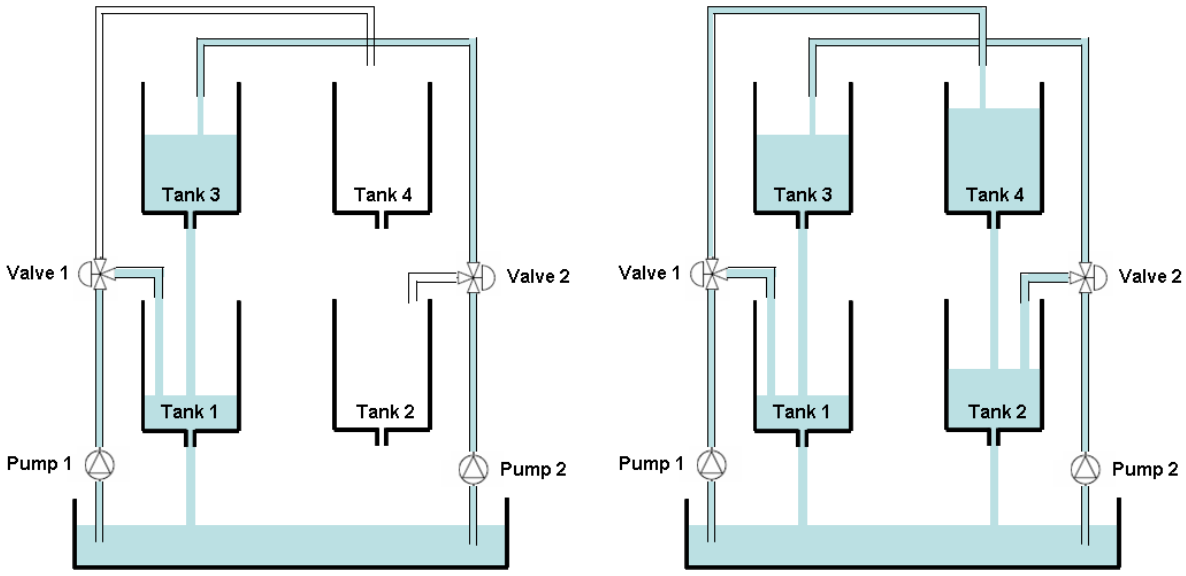


Figure 1: Quadruple tank process for SISO (left) and MIMO (right) control schemes.

of one of the lower tanks in the face of nonlinearities, unmodelled dynamics and disturbances coming from the upper tank (an scheme is shown in Figure 2). Moreover, the installation is composed of industrial instrumentation, what is also useful for students training purposes. For the first five steps of the nine mentioned above, the students have to analyze the complete setup of the installation, composed by:

- Flow pumps *Grundfos* UPE 25-40, which are equipped with expansion modules *Grundfos* MC 40/60. These pumps can be regulated by an analogical signal (0-10 V) and have characteristic nonlinearities like hysteresis and dead zone, that the students have to analyze.
- Three-way pneumatic valves *Samson* 3225 with regulator 3760.
- Level sensors constituted by pressure transmitters *Endress & Hauser PCM 731*.
- SMC on/off electrovalves used to simulate disturbances on the tanks' levels.
- *Amidata* flow sensors 257-025. The control signals used are the voltages (0-10 V) that are sent from the computer to the regulator of the pumps. As the pumps have a strong nonlinear behaviour, a cascade controller has been implemented in such a way that the students have to design control algorithms providing the desired flow to the pumps to achieve the setpoint levels in the tanks, while an internal loop provides the required voltage to achieve such flow, using the measurement from the flowmeters (see Figure 2).

- Data acquisition system *Opto 22 Snap Ultimate I/O* used as interface between the applications and the scale model. This unit offers a powerful input/output system for local and remote monitoring, industrial control and data acquisition. Based on the IP, the Snap Ultimate I/O offers a great flexibility in the physical network (wired or wireless LAN Ethernet, PPP) and flexibility in the software application.
- A SCADA tool has been developed to monitor and supervise the plant. It has been programmed with the graphical development environment LabVIEW 7.1 from National Instruments [29].

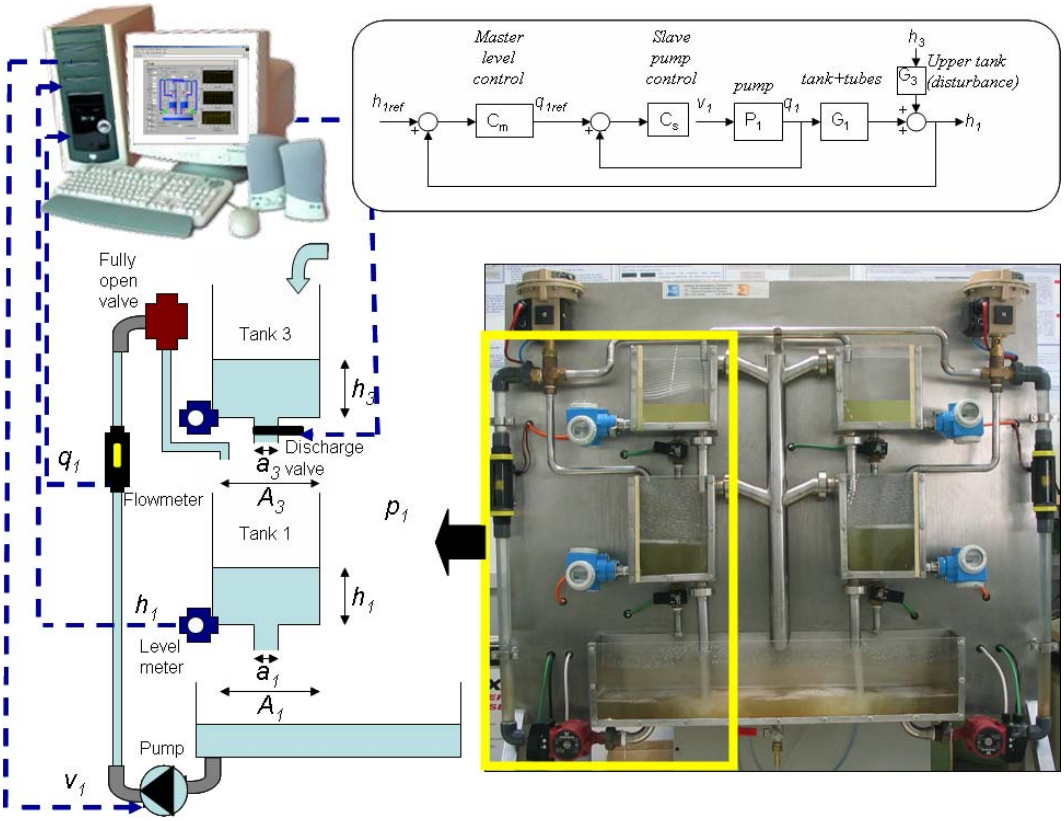


Figure 2: Setup used in the lab session

2.2 Modelling issues

Once the students have understood how the installation works, they are involved in step 6, that is, modelling the system. The simplified dynamical description of the complete quadruple-tank system is given in [25]. The students have to perform the modelling based on physical principles of the double-tank subsystem represented in Figure 2.

For the complete quadruple-tank process, a simple model can be obtained using the Bernouilli and mass balance equations. In this way, a model relating the input flow to each tank ($q_j; j = 1..2$) with the level of the four tanks ($h_i; i = 1..4$) can be obtained.

$$\begin{aligned}
\frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1}{A_1}q_1 \\
\frac{dh_2}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2}{A_2}q_2 \\
\frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)}{A_3}q_2 \\
\frac{dh_4}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)}{A_4}q_1
\end{aligned} \tag{1}$$

where A_i is the cross-section of the tank i, a_i represents the cross-section of the outlet hole and h_i the water level of tank i, for $i = 1..4$, $\gamma_j \in [0, 1]; j = 1..2$ are the positions of the three-way valves ($\gamma_j = 1$ means that all the flow goes to the lower tanks). Finally, g denotes the acceleration of gravity. Differential equations (1) constitute a very simple, intuitive and understandable model of this process. However, this model is based on several hypotheses that could be wrong for a physical implementation of the process. For example, linear behaviour is supposed for three-way valves and pumps, but this condition may not be kept in some situations. The main parameters describing the system are included in table 1.

Table 1: Physical parameters of the quadruple-tank process

Parameter	A_i	g	$a_1; a_2$	$a_3; a_4$
Value	389.16	981	2.1382	0.9503
Units	cm^3	cm/s^2	cm^2	cm^2

If differential equations (1) are linearized around an operating point, the corresponding transfer matrix is given by:

$$\begin{bmatrix} \frac{\gamma_1 c_1}{T_1 s+1} & \frac{(1-\gamma_2)c_1}{(T_3 s+1)(T_1 s+1)} \\ \frac{(1-\gamma_1)c_2}{(T_4 s+1)(T_2 s+1)} & \frac{\gamma_2 c_2}{T_2 s+1} \end{bmatrix} \tag{2}$$

Where:

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}} \quad \text{and} \quad c_i = \frac{T_i}{A_i} \text{ for } i = 1..4 \tag{3}$$

For a defined steady-state operating point ($dh_i/dt = 0$) characterized by h_i^0 and q_i^0, from the linear model it can be deduced that:

$$\begin{aligned}
\frac{a_1}{A_1}\sqrt{2gh_1^0} &= \frac{\gamma_1 k_1}{A_1}v_1^0 + \frac{(1-\gamma_2)k_2}{A_1}v_2^0 \\
\frac{a_2}{A_2}\sqrt{2gh_2^0} &= \frac{(1-\gamma_1)k_1}{A_2}v_1^0 + \frac{\gamma_2 k_2}{A_2}v_2^0 \\
a_3\sqrt{2gh_3^0} &= (1-\gamma_2)k_2 v_2^0 \\
a_4\sqrt{2gh_4^0} &= (1-\gamma_1)k_1 v_1^0
\end{aligned} \tag{4}$$

Where it is supposed that the relationship between the flow given by each pump (q_j) and the voltage supplied to the regulator of the pump (v_j) is linear. As has been mentioned, in the laboratory installation this is not true, but the inclusion of a feedback loop with a PI controller adequately tuned linearizes the system, and thus, as the dynamics of the flow are quite fast when compared to that of the tanks, the relationship between voltage and flow can be approximated by a gain, or what is equivalent, the output of the level controller is the desired flow to track the level reference in such a way that the dynamics of the internal loop governing the pump through the voltage sent to its regulator can be neglected and considered as a block with unitary gain. Then, following the steady-state analysis, in order to find an univocal relationship between the steady state inputs ($q_1^0 = k_1 v_1^0, q_2^0 = k_2 v_2^0$) and the corresponding outputs (h_1^0, h_2^0), the following condition has to be fulfilled $\gamma_1 + \gamma_2 \neq 1$.

To work only with tanks number 1 and 3, the three-way valves have to be positioned such that $\gamma_1 = 1; \gamma_2 = 0$, in such a way that by means of flow q_1, the level of tank 1 can be controlled, while water from tank 3 acts as a disturbance. The model for this system can be easily obtained from that of (1) as follows:

$$\begin{aligned}
\frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{1}{A_1}q_1 \\
\frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{1}{A_3}q_2
\end{aligned} \tag{5}$$

The steady-state operating points are then given by

$$\begin{aligned}
\frac{a_1}{A_1}\sqrt{2gh_1^0} &= \frac{1}{A_1}q_1^0 + \frac{1}{A_1}q_2^0 \\
a_3\sqrt{2gh_3^0} &= q_2^0
\end{aligned}$$

and the linearized model by

$$H_1(s) = \underbrace{\frac{c_1}{T_1 s + 1}}_{G_1(s)} Q_1(s) + \underbrace{\frac{c_1}{(T_3 s + 1)(T_1 s + 1)}}_{G_3(s)} H_3(s) \tag{6}$$

Equation (6) provides a great flexibility in the design. If disturbances are not considered coming from the upper tank (tank 3), the transfer function relating the flow given by pump 1 with the level of tank 1 is $G_1(s)$, that is a first order transfer function to which it is relatively easy to design a PI controller. If tank 3 is used as a source of disturbances, its influence is given by $G_3(s)$, as can be seen in Figure 2, where the complete cascade control loop is represented. Notice that these linear models are only valid when working around a particular setpoint given by (q_1^0, h_1^0). This aspect is also very important for the students, that have to understand the concepts involved in the linearization of nonlinear systems.

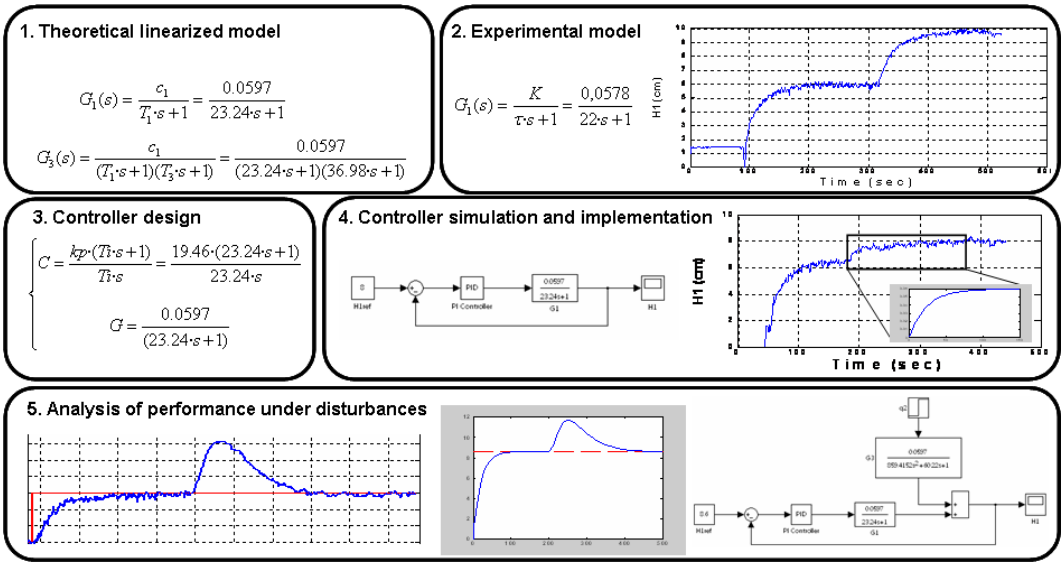


Figure 3: Summary of the results obtained in a typical lab session

2.3 Students' work

The main activities that the students have to perform are related with steps 6-9 indicated in the introduction of this section, mainly concerning model validation and controller design, simulation and implementation. The sequence of activities proposed to the students are (see Figure 3):

1. To study the system, to understand the differential equations describing the main dynamics and the linearization process.

2. Working only with pump number 1 and with the three way valve fully opened ($\gamma_1 = 1$), take the system to a defined operating point described by $h_1^0 = 6$ cm, $q_1^0 = 232$ cm^3/s, manually. Then, introduce an open loop step in the pump flow up to $q_1 = 299.5$ cm^3/s and after reaching steady-state, check that the tank level is around the theoretical new steady-state ($h_1^0 = 10$ cm). Obtain a linear model of the system using

the reaction-curve method and compare the results with the theoretical linearized model. Comment the results.

3. Design a PI controller to control the level of tank 1 using as setpoint $h_{1ref} = h_1^0 = 8$ cm. The design specifications are that the system must have null steady-state error and the closed loop system dynamics those of a first order system with a dominant time constant of 20 seconds. Implement the control loop in Simulink [27] and simulate the system.

4. By manually approaching the system to an operating point close to the setpoint, test the performance of the developed controller and compare with simulation results.

5. Using pump number 2 with a flow of 250 cm^3/s and with the second three-way valve fully closed ($\gamma_2 = 0$), analyze how the resulting control system behaves when subjected to disturbances coming from tank number 3. Design a feedforward controller to attenuate for such disturbances and test it under simulation (real tests are left to another session as feedforward control is compared to decoupling multivariable control).

Figure 3 summarizes the results obtained in a typical lab session by one of the students.

3 An open source tool for developing virtual and remote laboratories: Easy Java Simulations

Easy Java Simulations, EJS for short, is an open source software tool that helps creating dynamic interactive virtual and remote laboratories in Java language [11, 34]. The tool is targeted for both science/engineering students and teachers with low profile in programming but deep knowledge on certain technical and scientific fields. Within EJS, virtual laboratories are created by specifying a model for the simulated process and by building a view that continuously visualizes the state of this model and that readily responds to user interactions. Models of processes can be described in EJS applications either using its self-contained facilities (ODE solvers and Java code editors) or Simulink block diagrams. When using the first possibility, EJS produces as final product a Java application or applet that can be run in any computer or browser. However, when the second possibility is used, EJS can be considered a tool that wraps the Simulink model with a Java-made layer of interactivity. In this case, the final result is a Java application or applet that interchanges data (input parameters and states variables) with the Simulink model through the Matlab's [28] workspace (Figure 4) [7].

In spite of the fact that EJS was originated for the teaching of Physics, the tool has been being adapted and improved to serve the necessities of control engineering education [2, 9, 18, 20, 26, 36, 37]. This is the main reason to enable EJS to exchange information with Simulink since the later tool is considered as a de facto standard in the control engineering community. But trying to broaden the use of EJS in the control community, new options for exchanging data with other tools widely used have been included in last EJS releases. Currently, it is already possible to exchange data with Scilab [41], Scicos [40], and SysQuake [31].

When Easy Java Simulation is used to develop remote laboratories, the state vector is recovered from a remote computer connected to the real plant, and not from the simulated model. In this case, the developer has to write its own Java code to make the Internet connections since EJS provides just some helps, as for example, to get the video stream from IP cameras. In this way, a virtual and remote laboratory can coexist inside of the same application keeping exactly the same graphical interface and the level of interactivity.

For all these reasons, Easy Java Simulations can be considered as a very convenient framework for the control education community in the way to adopt an interactive-based approach to be on the brink of new pedagogical tendencies, such as those Physics instructors are boosting with different world-wide open-source initiatives [4, 5].

Because of its interest, the next section makes a description of EJS in more detail and presents a complete case-study of development of a virtual and remote laboratory for control engineering education. In this case study, we will use the model of two non-interactive tanks described in the previous section to present how we can turn this model into a real interactive, dynamic simulation by using EJS, and after that, in a remote laboratory. Although we do not assume the reader familiar with EJS, we will not describe all its features, but concentrate only in what is needed for the goals of this chapter. Readers interested in

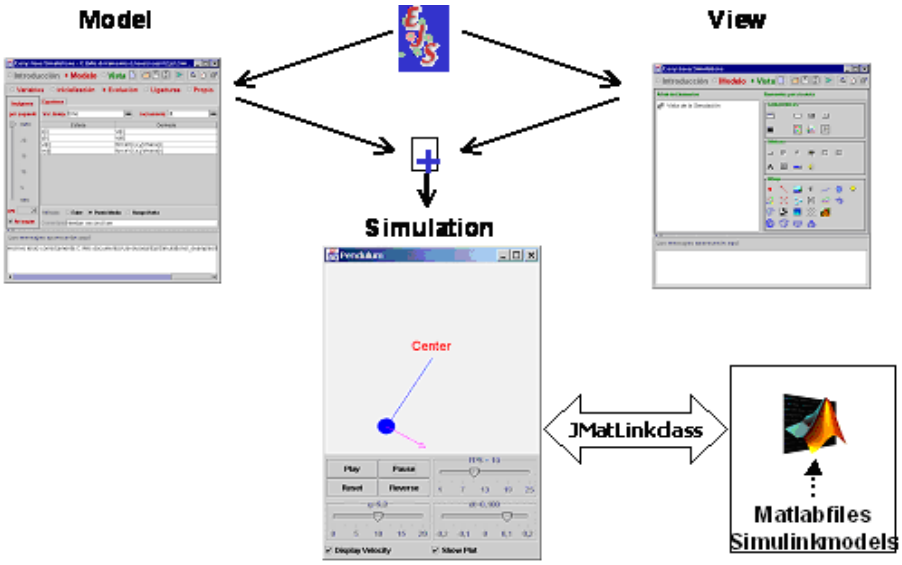


Figure 4: Building a simulation using EJS and MATLAB/Simulink.

EJS itself and its last features can download the software, a great number of examples and a detailed manual from [10].

3.1 Brief survey of Easy Java Simulations

A quick description of Easy Java Simulations places it in the category of *code generators*. This means that users need to provide only the most relevant core of the simulation's algorithm and the tool automatically generates all the Java code needed to create a complete interactive simulation, including a wide range of sophisticated software techniques (such as handling computer graphic routines, communication protocols, multithreading and others). What makes EJS very special inside this category of software is that it has been conceived by teachers and for teachers and students who are more interested in the understanding of the simulated phenomena that in the underlying computer specific aspects.

The tool uses an original simplification of the successful Model-View-Control software paradigm (MVC), and structures simulations in two main parts: model and view. The MVC paradigm states that a simulation is composed of three parts:

1. The model, which describes the process under study in terms of
   - Variables, that hold the different possible states of the process,
   - Relationships among these variables (corresponding to the laws that govern the process), expressed by computer algorithms.
2. The control, which defines certain actions that a user can perform on the simulation.

3. The view, which shows a graphical representation (either realistic or schematic) of the different states that the process can have.

These three parts are deeply interconnected. The model obviously affects the view, since a change in the state of the model must be made graphically evident to the user. The control affects the model because control actions can (and usually do) modify the value of variables of the model. Finally, the view affects the model and the control, because the graphical interface can contain components that allow the user to modify variables or perform the predefined actions. As mentioned above, EJS further simplifies the construction of a simulation by merging the control half into the view, half into the model.

To assist developers to follow the MVC paradigm, the EJS interface is divided into three main sections: *Introduction*, *Model* and *View*. The *Introduction* is where the developer can write any didactical or pedagogical information related to the application (for example, instructions about the simulation, explanations of the process, exercises to carry out, questionnaires to fill out, etc.) by using a built-in editor (Figure 5). Once the developer finishes the writing of the simulation and compiles it, all this information is transformed into HTML pages. The *Model* section is where the different parts of the model are specified. Due to its special importance, it will be described in more detail along the following paragraphs. The *View* section is where the graphical representation of the virtual laboratory is built and also where the level of interactivity is specified. As well as the Model, this section will get our attention later on.

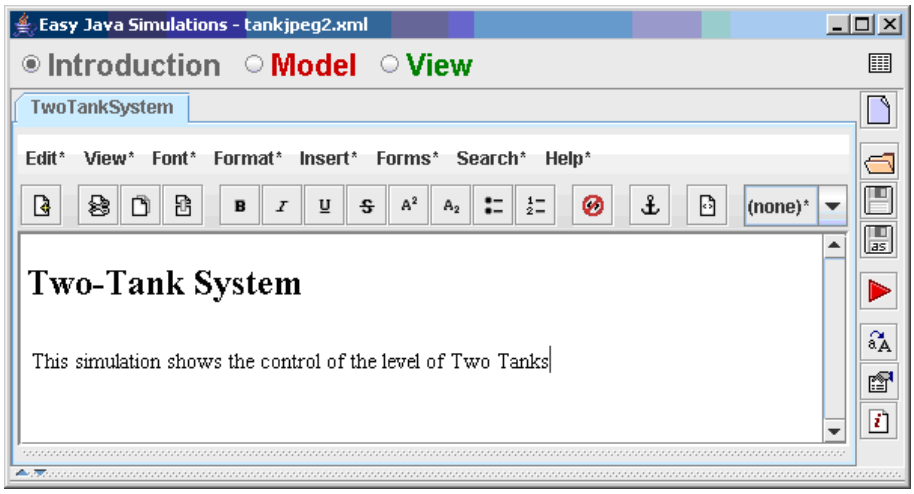


Figure 5: Built-in editor of the *Introduction* section.

The model is what supports the scientific part of the simulation and is thus the responsibility and the main interest of the user. Control teachers use it to describe their algorithms in terms of mathematical equations and to express them using a given computer language, or by means of high-level tools such as MATLAB/Simulink or Scilab/Scicos. The main target when creating the model of a simulation is to concentrate in the analytical description of

the process, the content, and the accuracy of the simulation. However, the creation of the necessary graphical user interface (that is, the view) is the part of the simulation that demands more knowledge of advanced programming techniques. And, to make the situation even worse, the addition of interactivity to this interface involves mastering sophisticated software techniques which demand a big investment of time and effort.

EJS helps its users through both tasks. The tool provides extensive scaffolding to define the model, while still retaining the flexibility of a general programming language so that it is possible to specify almost any type of algorithms. This is pedagogically important, since the process of learning good control fundamentals consists, to a great extent, in understanding the basic principles required to build models.

In order to define the model in EJS, the user follows a prescribed sequence of actions:

- Action 1: To declare the variables that describe the system,
- Action 2: To initialize these variables to a correct initial state,
- Action 3: To describe how the value of these variables change in time, and
- Action 4: To establish how they affect each other when the user interacts with the system and modifies one or more of their values.

The last two steps consist in translating the mathematical equations or physical laws that rule the process under study into computer algorithms. Sometimes, though, the actual implementation of these algorithms can be a difficult task. For this reason, and as it was said before, EJS provides different extra facilities. The first one is a built-in editor and solver for systems of ordinary differential equations (ODE). The user writes the equations in a way much similar to what he/she would write on a blackboard, and the system automatically generates the code that numerically solves the equations using one of several of the provided standard algorithms (Figure 6). The editor also includes support to handle simple state-events of the ODE. The other facilities are connections with MATLAB/Simulink, Scilab/Scicos and SysQuake that lets users design and solve their models with the help of these tools, but the description of these features is out of the scope of this text.

Regarding the prescribed sequence of actions to define the model, the *Model* section of the EJS interface is divided into five subsections: *Variables*, *Initialization*, *Evolution*, *Constraints* and *Custom*. All these subsections will be comment when the construction of the virtual laboratory is described in the next section.

In order to help create the view, EJS provides a set of advanced graphical elements (Figure 7) which build on top of both standard Java *Swing* components (containers, buttons, text fields, sliders, combo boxes,...) and on specific scientific two- and three-dimensional visualization classes from the Open Source Physics project [3,6] (particles, vectors, images, vector and scalar fields, etc.) All these graphical elements can be used in a simple drag-and-drop way to build the interface. The user just needs to design the view so that it will offer a visualization of the process appropriate to the desired pedagogical goals. In particular, the view should encourage students to explore the process under different engineering perspectives in order to gain a better insight of the system.

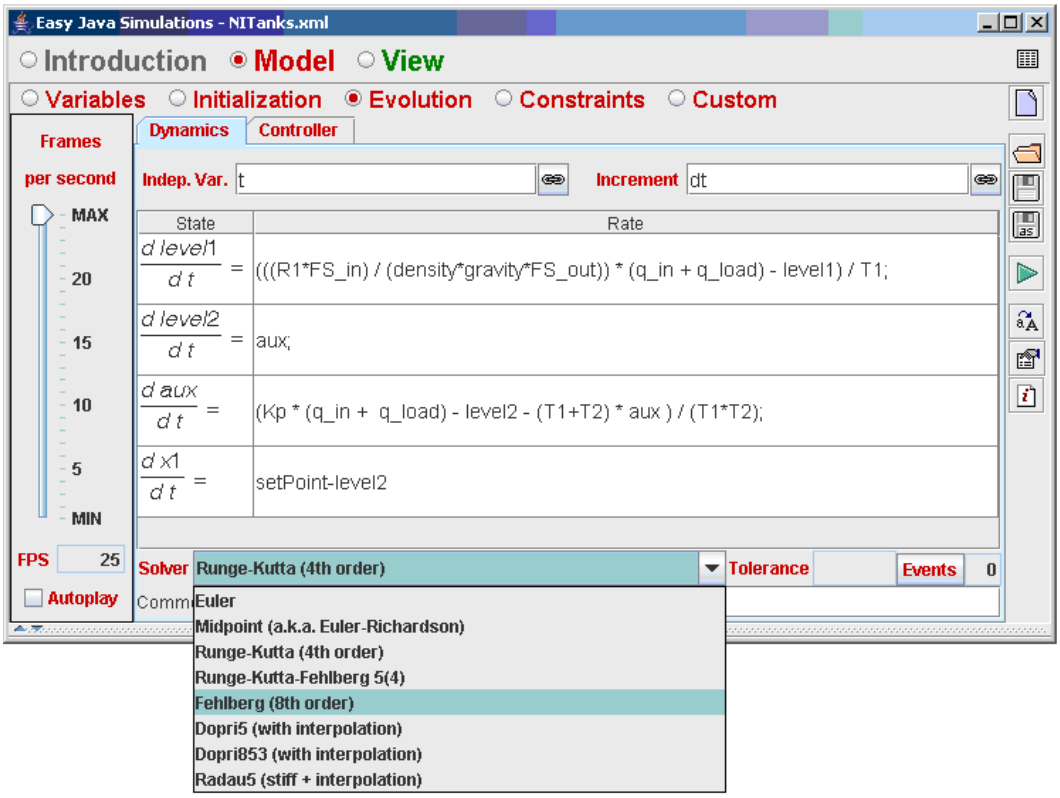


Figure 6: Built-in editor to write a ODEs and select the solver.

Also it is worth to notice in Figure 7 the existence of a specific category of graphical elements to let users the design of control schemes in a very simple way. So, the construction of control diagrams can be done directly by using these elements and it is not required to make up the diagram with basic graphical elements (for example, circles, squares, labels, segments, etc.) An example of it is shown on the left part of Figure 7, where it can be appreciated the control elements of a view developed to represent two non interactive tanks.

To complete the view, its different elements have to be instructed to display on the screen according to the values of the variables of the model. Every graphic element of EJS has certain internal values, called properties, that can be customized to make the element look and behave in a particular way (change its displayed value, position, size, colour, etc.) The user can very easily connect the properties of the graphical elements of the view to the value of the different variables of the model. Some of these properties can also be instructed to trigger user-defined actions (typically Java routines defined in the model) when the user interactively changes them. This procedure, which we call *linking* the model and the view, is what turns the simulation into a real dynamic, interactive application. This mechanism configures a simple, though very effective, way of designing and building advanced interactive user interfaces. A couple of examples can be observed in Figure 8 for some interactive

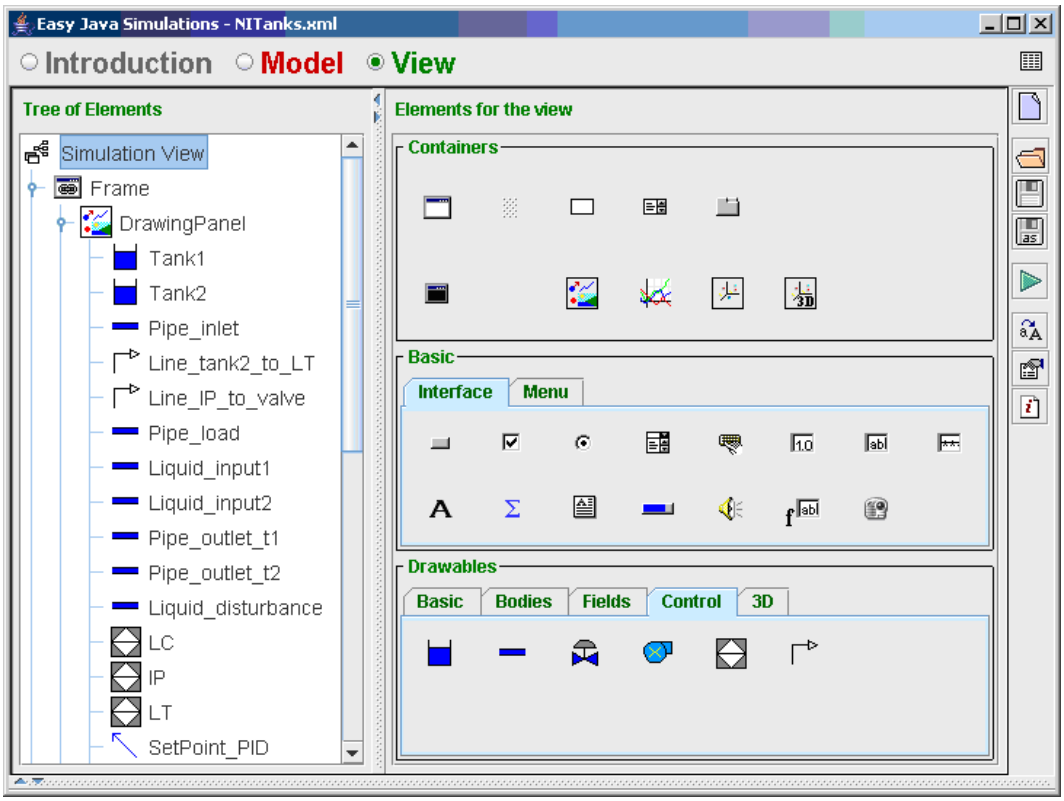


Figure 7: Library of graphical elements offered by EJS.

virtual laboratories.

The reason for this is that linking is a two-way connection. When the variables of the model change, the view is informed so that it immediately displays the new state of the model. In return, since the view elements have built-in interactive capabilities, any interaction of the student with the interface immediately affects the model variables that have a connection to it. For example, let us imagine an EJS simulation of the control level of a tank. The tank and the water inside it are represented in the view by means of two coloured rectangles. If the dimensions of the tank are modified by dragging with the mouse the corners of the outer rectangle, the variables representing the tank width and height in the analytical model will reflect this change (thus affecting the dynamic of the system). Similarly, if the water level changes as a consequence of the model's evolution, the visible height of the rectangle will change to display the current water level.

With all the high-level information, which only the user can provide, EJS takes care of all the low-level procedures needed to create the final simulation. It will generate the Java code that handles all the internal tasks, compile it into Java classes, pack the classes in a compressed file, and end up with a ready-to-use simulation. Simulations created using EJS can either be used as stand-alone applications under different operating systems (for

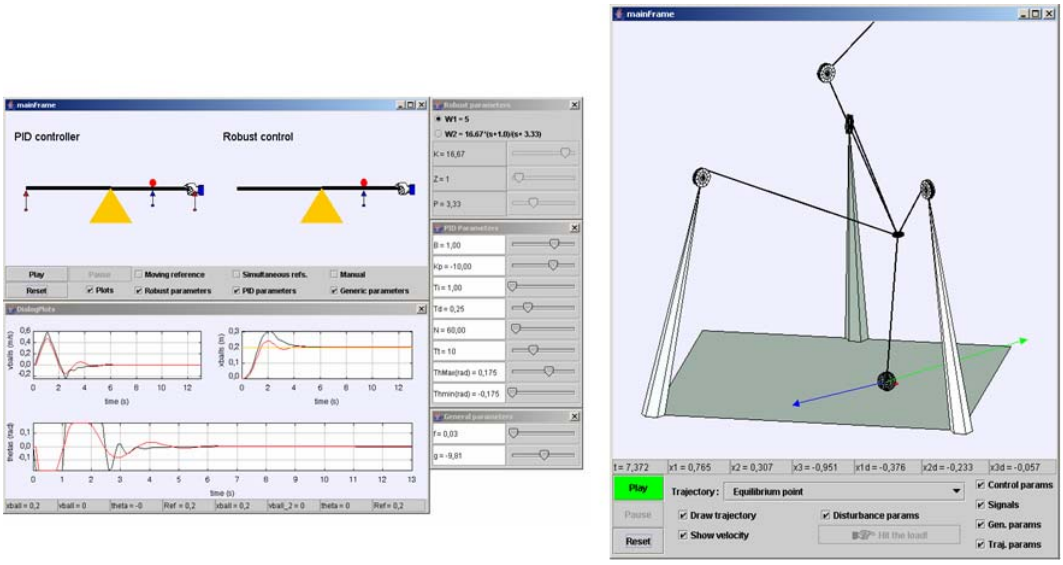


Figure 8: Two examples of interactive application developed using EJS. This example corresponds to a prototype of virtual laboratory designed for research on advanced control strategies.

instance, a .BAT file is provided when running under Windows), or be distributed via Internet and run as applets within HTML pages (those pages created with the built-in editor located in the *Introduction* section) by using any Java-enabled web browser. At present, it is already possible to use MATLAB/Simulink when running simulations as applets and not only just as applications. Last versions of EJS generate distributed applications with Java interfaces and Simulink models running in different computers using the Internet to exchange information.

3.2 Development of an interactive virtual and remote laboratory using EJS

The virtual laboratory developed with EJS is based on the two-tank model described in previous sections. Along next paragraphs it will be shown the most relevant steps that have to be carried out in the development of the virtual lab, but details about the construction of some specific and tricky parts will not be included. Readers interested in a more in-depth knowledge of the EJS file can download the source file from [38].

One of the most important things to bear in mind when developing a virtual laboratory is the correct configuration of windows and operating controls to facilitate the student's understanding of the process to be controlled. In this case study, the view (using EJS terminology) is divided into several elements as displayed in Figure 9. Because the simulation is to be used for didactical purposes, it is important to provide a view that displays the physical system as realistically as possible, but that also provides support for elements commonly used in control engineering (scopes, diagrams, input devices, controllers, etc.) Another important requirement to fulfill when interactivity is included in the virtual laboratory is to provide a set of interactive elements that can be manipulated by students to make changes in a dynamic way during an experiment.

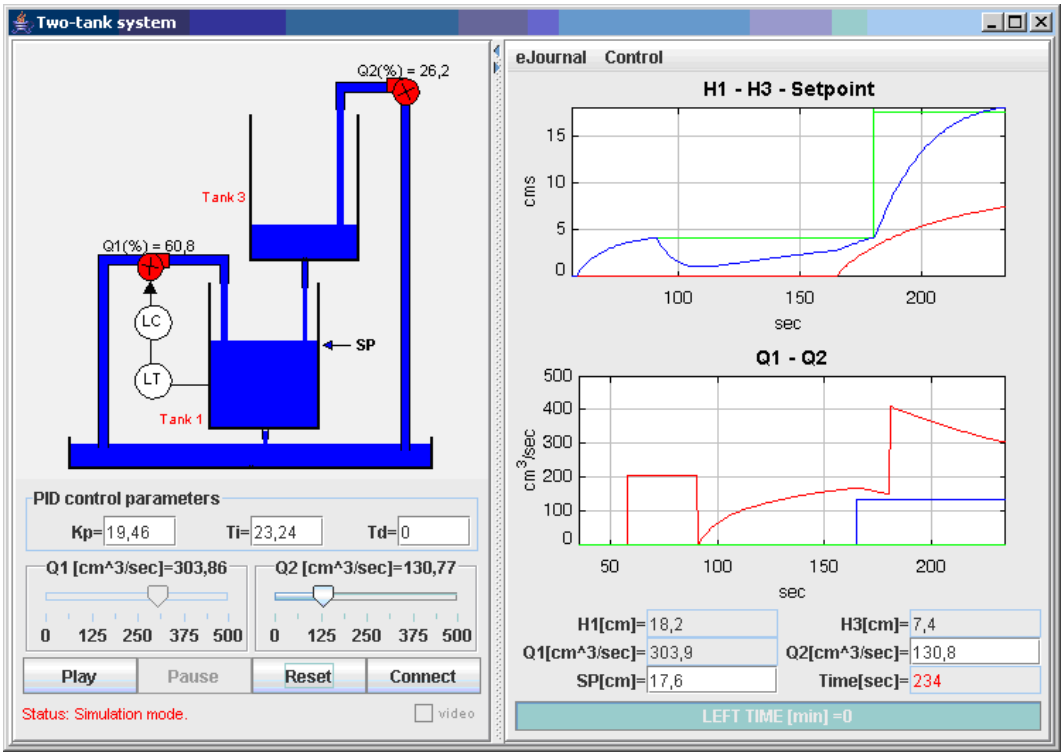


Figure 9: Graphical interface of the virtual laboratory to experiment with a two-tank system.

According to that, the view has been structured in one main window divided into two parts as Figure 9 shows. The part on the left is composed of two sections: the control dia-

gram of the two-tank system (top) and the control panel (bottom). The diagram schematically displays the process to control and allows the user to change interactively the set point by dragging the arrow up and down. The control panel is made up of buttons, sliders and text fields to let users choose the operating conditions of the system. With these elements the user can select:

- To play, pause and reset the simulation (*Play*, *Pause* and *Reset* buttons).
- To change the input flows Q1 and Q3 with the sliders. It must be noticed that when the user is controlling the liquid level in manual mode both sliders are enabled, but when working in automatic mode, just slider Q3 is on. It is due to Q1 corresponds to the control variable and Q3 is a second input flow designed to introduce disturbances in the plant.
- To tune the parameters of the PID controller by the text fields.

At the bottom of the left part of the main window there is a status bar to display information about the connection when the remote mode is enabled, that is, when the model of the two-tank system is replaced by a TCP connection to the real plant. To the left of the status bar, there is a check button, to begin the transmission of video in real time of the two-tank system.

Continuing with the description of the virtual laboratory, the right part of the main window is essentially for information purposes. This part is composed of two mixed-signal scopes and six text fields to present the value of most important state variables of the system: time, input flows, tank levels, reference, and control actions. Above the scopes, there is a bar with two different menus: *Control* and *eJournal*. The first one lets users change the control mode, that is, manual or automatic using the PID controller. The *eJournal* menu is used to save state information in the collaborative environment (this feature will be described further).

3.3 Building the model of the process

Once the description of the virtual laboratory has been given, let us describe the most relevant steps to build this application with EJS. First of all, it is necessary to declare and initialize the variables to be used in the programming of the model, in the building of the graphical view, and in the performing of the user actions. The declaration of variables in EJS is done in the *Variables* section by using pages, as it can be appreciated in Figure 10. In this example, there are nine variable pages and each one has a different label to identify the purpose of its variables. The variable page, labelled *Tank1*, contents the variables to define the physical characteristics and elements involved in the model of the tank 1 and its graphical representation.

After the declaration of the variables, the next step is the writing of Java code to initialize different parts of the laboratory, for example, some elements of the view, the controller parameters, the initial state of the system, etc. As well as happens with the variables, there

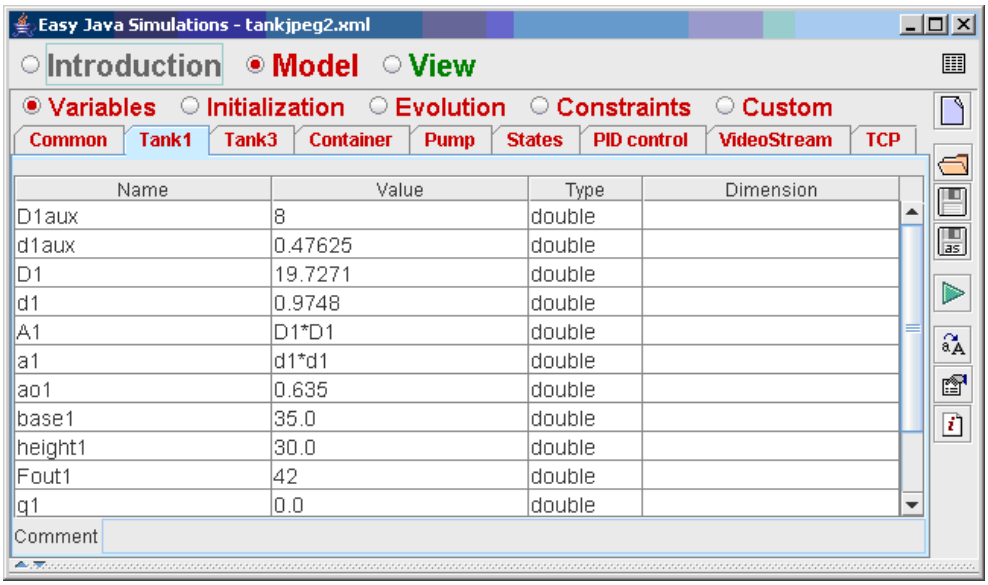


Figure 10: Pages to declare variables in the *Variables* section.

is an *Initialization* section where it is possible to create pages to enter the Java code. In Figure 11 it can be seen that there are three initialization pages in the virtual laboratory and that the *Init controller* page contents the Java methods used to set the PID controller parameters. It is important to say that the code written in these pages is just executed in any first-time play or reset of the virtual laboratory.

Following with the construction of the virtual laboratory, the next step consists in the writing of the elements presented in the control loop to be simulated, that is, the process and the controller. In this example, the process is a two non-interactive tank system and a PID controller. The construction of these two elements is done in the *Evolution* section and, as well as in the *Variables* and *Initialization* sections, it is possible to create different pages to define each element. In this example, the *Evolution* section is composed of two pages: one to define the differential equations corresponding to the two-tank system and a second one to write the code of the discretized PID controller.

The *Evolution* page created to declare the ODE system corresponding to the two-tank system is shown in Figure 12. This page presents many elements that deserve to be commented. First, it must be noticed that this page is an ODE editor, so it is necessary to indicate who is the independent variable and the increment that this variable experiments in every integration step. In this example, this variable is the time and it is represented by the variable t and its increment by dt (both ones have been previously declared in the variable page named *Common*). Second, in the central part of the page it is found the editor to write the differential equations and, below it, there is a menu to choose the integration methods to solve the ODE system (in this example, the method is a Runge-Kutta of fourth order).

Continuing with the writing of the differential equations in the EJS built-in editor, there

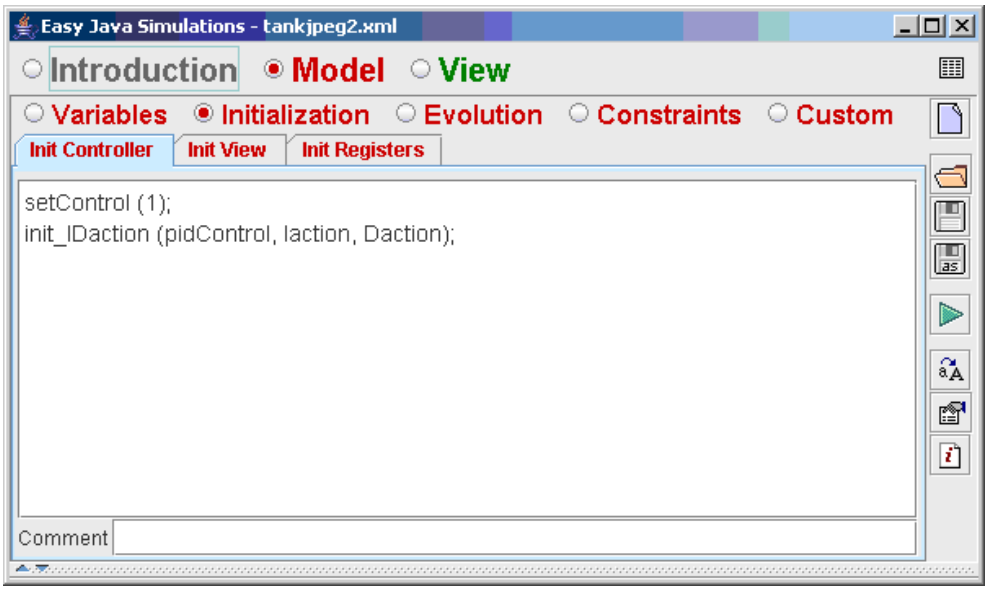


Figure 11: Pages to write initialization code in the *Initialization* section.

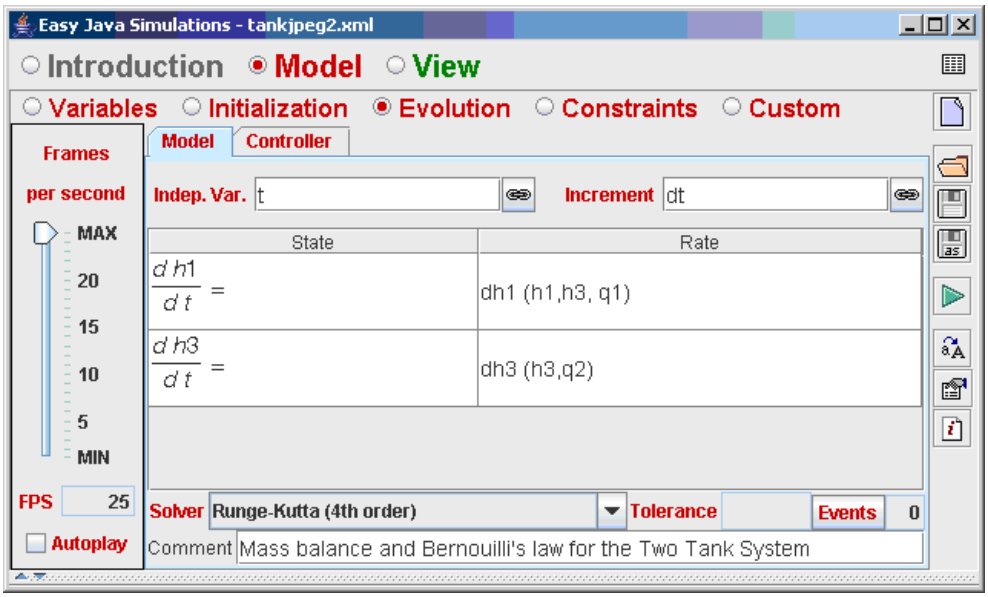


Figure 12: Page of the *Evolution* section to define the two-tank model by two ODEs.

are two options: to write directly the equations, as for example happens in Figure 6, or to write the signature of a Java method that contents the equations plus some additional code, as happens in the example of Figure 12. In this example, the differential equations have been replaced by calls to two Java methods defined by the developer: $dh1$ and $dh3$. There

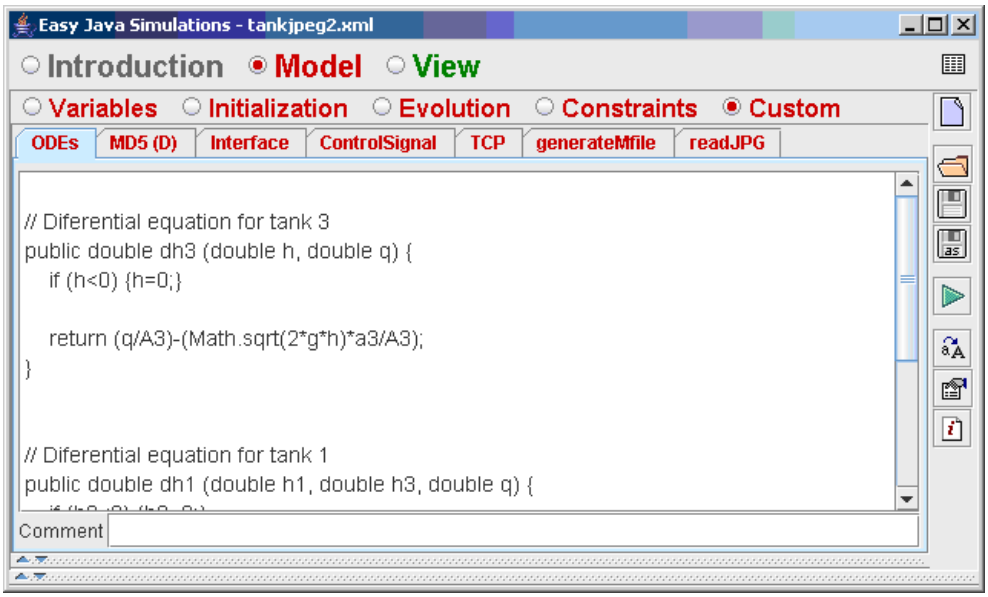


Figure 13: Page of the *Custom* section where the Java methods to specify the differential equations have been written.

is a section in EJS, named *Custom*, where developers can create pages and write its own Java code. As we will see further, this section will have a key role in the adaptation of the virtual laboratory to become a remote one.

Figure 13 presents the page of the *Custom* section where the code of the methods $dh1$ and $dh3$ has been written. In this example, this approach for writing the ODEs has been used to detect the anomalous situation that happens when the level of the tank becomes negative (see the if instruction in the Java code of Figure 13). The rest of the Java code of the method corresponds to the differential equation already described in equation (5).

Once the model of the two tanks has been introduced, the next step to complete the model of the full control system is the writing of the control code in the second *Evolution* page, named *Controller*. Figure 14 presents this page and, at a first glance, there is a big difference regarding the previous *Evolution* page where the two ODEs were defined. In this case, the page permits to specify a model by writing pure Java code. So, it is possible to use control structures or call Java methods previously written in a page of the *Custom* section. It can be appreciated in Figure 14 the structure of the control code and how an if-control structure and a boolean variable, named *connected*, are used to switch to virtual (using the model of the process, that is, the ODEs) or to remote mode (using the real process).

Let us dedicate a few lines to comment the code of Figure 14. When the user connects to the remote plant (variable *connected* = true) and the control mode is manual (*manualMode* = true), the value of the control action is stored in the variable $q1$ (by moving the slider Q1) and sent to the remote computer by the method *sendData*. When the automatic mode is on (*manualMode* = false) then the interface sends to the server-side the set of PID parameters

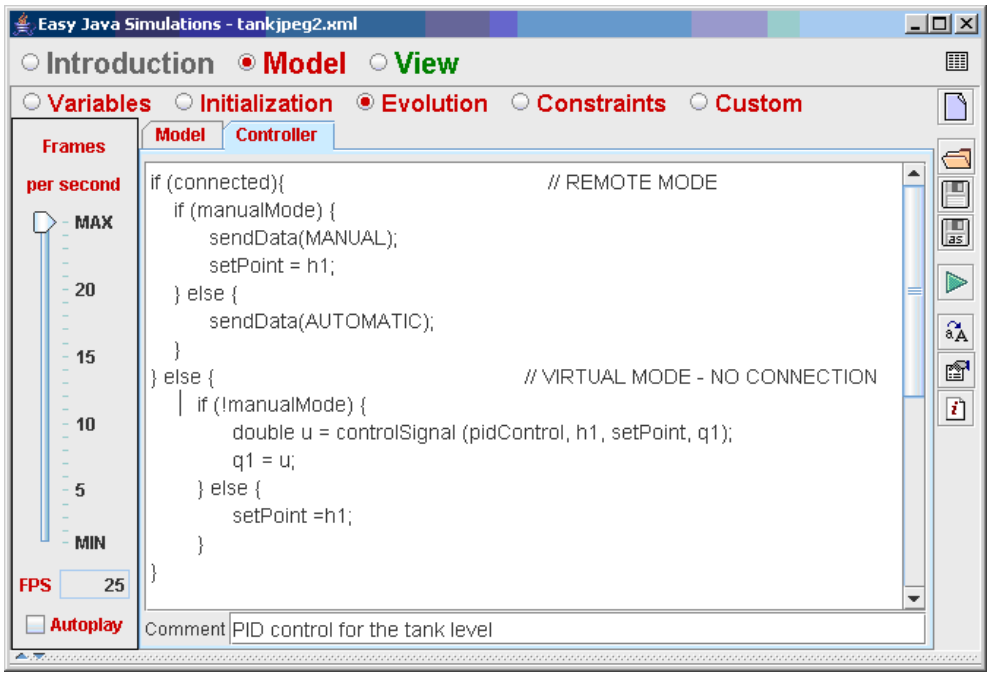


Figure 14: Page of the *Evolution* section where the value of the input flow to the tank 1 is obtained.

tuned by the user (proportional gain K, integral time Ti, and derivative time Td). In both cases, the method *sendData* is used to send the information to the remote side but using a different parameter depending on the control mode. Moreover, when the user works in virtual mode, the value of the input flow to the tank 1 is obtained too by manipulating the slider Q1 and stored in $q1$ but, this time, the value is used by the model of the process. In case that the user works in automatic mode, the value of the control action is calculated by the PID controller (method *controlSignal*) and the result is stored in the variable $q1$, that is, the input flow to the tank 1. It is important to bear in mind that the Java code corresponding to the discretized PID controller, that is the *controlSignal* method, is located in a page of the *Custom* section. Readers interested in the Java implementation of the PID controller can consult [1].

To finish with the development of the model, it is necessary to describe the *Constraints* section. The main goal of the code written in this section is to specify restrictions that the simulation or the remote system must fulfill. As example, Figure 15 shows the content of the page created in the *Constraints* sections in this case study. In this case, the Java code lets avoid the overflow of both tanks by setting the input flows to $0\ cm^3/s$ when the level of the tanks is over a certain limit. As well as in the Initialization and Evolution sections, the developer can call to Java methods written in the *Custom* section.

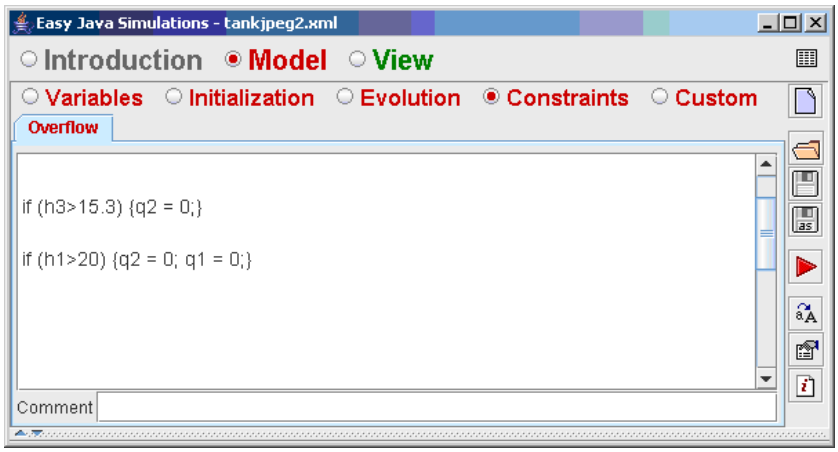


Figure 15: Page of the *Constraints* section where some safety actions are introduced to avoid overflows.

3.4 Building the interactive view

Once the model of the control system has been described to some extend, it is time to know how the view of the virtual laboratory has been developed with the EJS graphical library and how the interactivity is added to the view. To begin with, let us have a look to Figure 16 that corresponds to the *View* section of this case study. The left part shows the tree-representation of the graphical elements chosen for building the view (see also Figure 9). In Figure 16, just the elements corresponding to the control scheme can be appreciated (tanks, pumps, pipes, etc.), but below the rest of elements (buttons, slider, text fields, menus, etc) are located to make up the full view of the virtual laboratory.

As a first example about how to add interactivity to the view elements, Figure 17 displays the table of properties of the graphical element *Tank3* that allows users to change the level of liquid with the mouse since the property *Enabled* is set to *true*. This way, when the user drags up and down the liquid, the variable $h3$ that represents the level will be increased/decreased simulating a disturbance. Another example of interactivity is presented in Figure 18 by means of the property panel of the pump $Q1$. In this case, since the property *Enabled* of this element is *true*, the user can place the mouse over the pump symbol, click it and dragging up and down the value of the variable $q1$ linked to this symbol will be modified, simulating a disturbance in the input flow.

3.5 The remote laboratory for the two-tank system

Up to now, the most of the existent approaches to design remote laboratories involve two information loops: the synchronous control local loop, and the asynchronous tuning loop (Figure 19). The real-time control loop runs on the computer connected to the didactical setups. The control loop is always closed at the server-side and never across the Internet

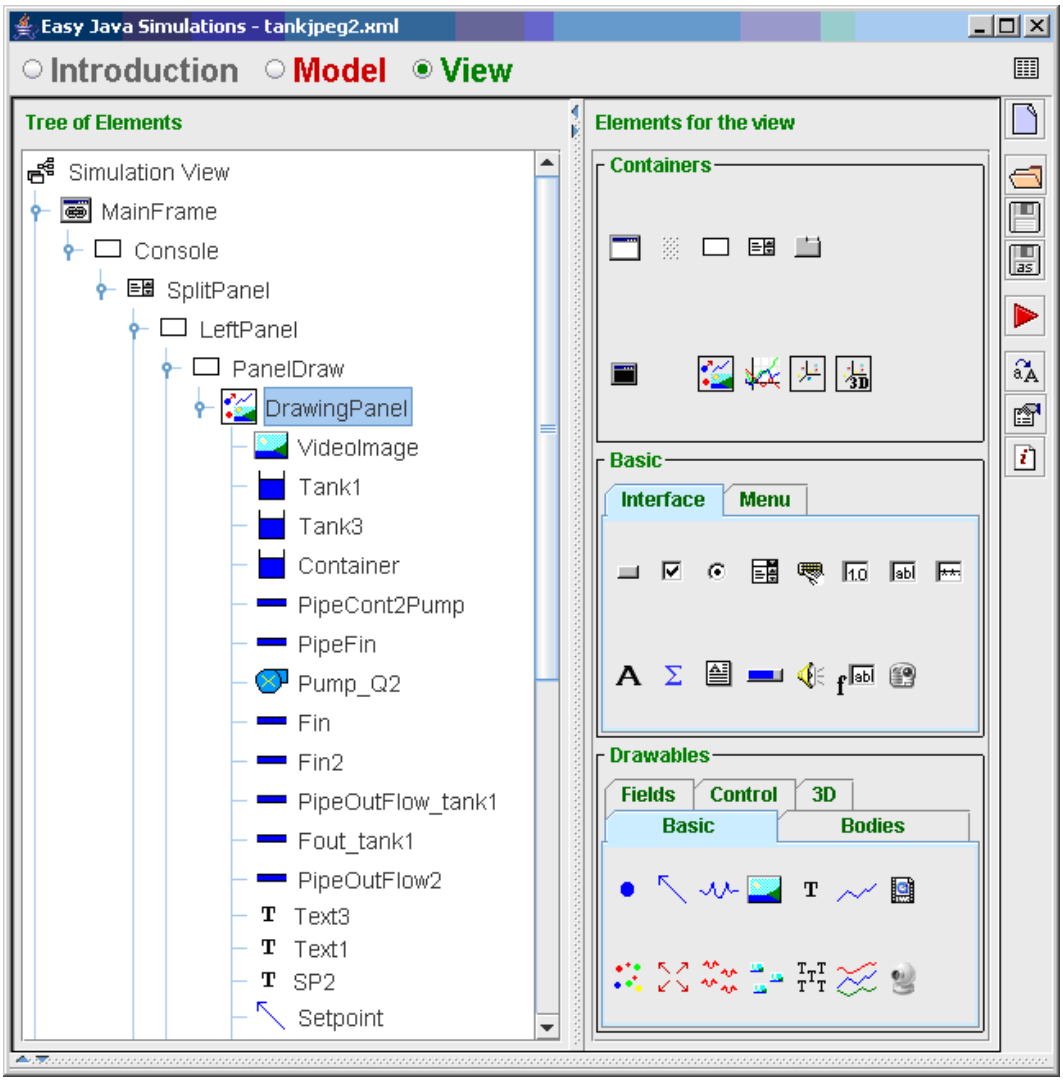


Figure 16: Elements for the view of the two-tank system.

because closure happens with the tuning or information loop. The information circulating via the tuning loop belongs to two classes: commands that clients sent to the server (for example, controller parameters) and data packets with the state of the process in response to the commands. This separation of the control and tuning loop is very convenient since it allows easily the replacement of the current physical system by another one 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.

For the case study described in this chapter, the approach presented in Figure 19 has been implemented using a single client/server structure, that is, the same computer performs

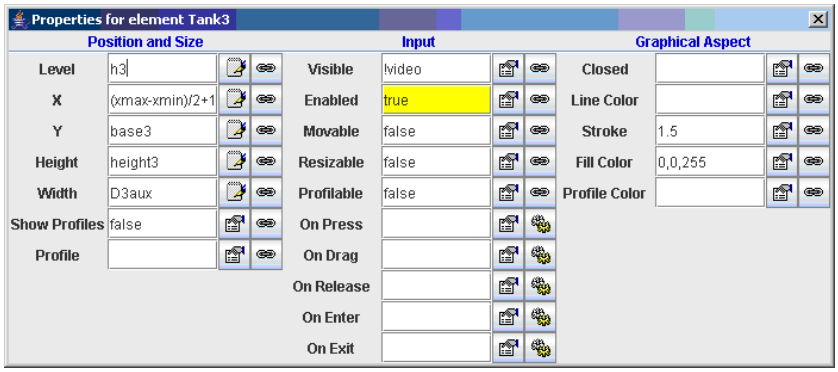


Figure 17: Properties of the *Tank3* element.

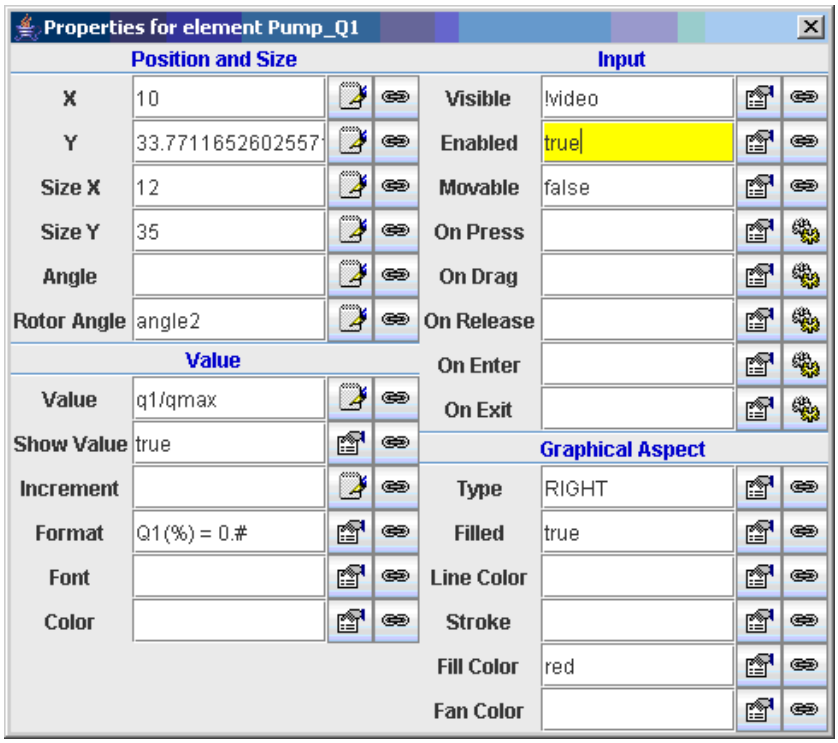


Figure 18: Properties of the *Pump_Q1* element.

as both the web server and the controller (Figure 20). The server is a computer running the control-server application and a web server to let students download the EJS applet, that is, the view described in the previous section. This interactive view is able to play either as virtual lab by using the mathematical model or as remote lab making a TCP connection with the controller running at the server-side. That is, a TCP channel among the EJS view and the server replaces the local communication channel between the view and the model

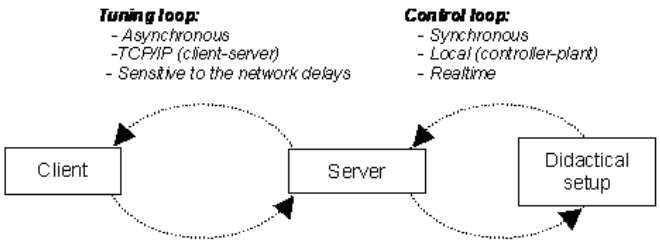


Figure 19: Teleoperation diagram with two information loops: tuning and control.

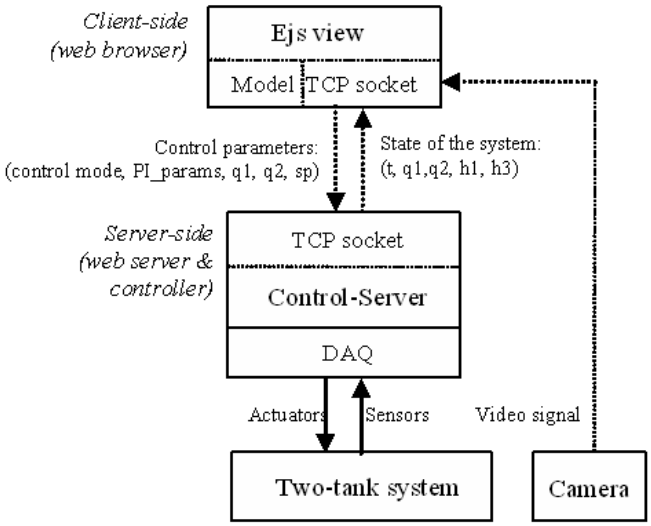


Figure 20: Single client/server structure implementing the tuning and control loop.

when the interface is used to access to the remote didactical setup.

The control-server application has been developed with the graphical programming language of LabVIEW [14, 33]. As in Figure 18, this application is composed of two different information loops: the asynchronous tuning loop, and the synchronous control loop. Regarding the transmission direction, the information circulating via the tuning loop can contain the user's actions or the system's state. The control loop is running in real-time at the frequency specified by the instructor. At the server side, both loops exchange information by common global variables. These variables are the PID controller parameters, the setpoint of tank 1, the pumps settings, and the two levels. In this way, every time a user changes any value in the EJS view, the new value is sent via the information loop to the server-side, and it is written in its corresponding global variable; every sampling time the control loop reads this variable and applies the new value to the system.

3.6 Transforming the virtual laboratory into a remote one

Thanks to the Model-View-Controller paradigm followed by EJS, and therefore in the construction of the virtual laboratory, the programming effort to transform it into a remote laboratory is not too high. The key modification is the replacement of the source where the vector state is generated, that is, the model when virtual mode, the didactical setup when remote. According to that, the only noticeable modifications are the writing of the Java code to open the TCP connection and to exchange information in both directions. The question of grabbing real time video from the server-side does not constitute a problem since in last EJS releases there is a graphical element to connect IP cameras by just dragging this element to the tree-like structure and indicating the IP number of the video camera in its properties table.

To let users switch between the virtual and the remote labs, there is a *Connect* button in the view located to the right of the *Reset* button. To commute to remote, users just need to click on this button and the information begins to be exchanged with the server-side and the two-tank diagram is replaced in the view by real-time video coming from the real system (Figure 21). From this moment, the control loop located in the server-side starts to control the two-tank system according to the data packets sent from the EJS view. So every user action sends a data array to the server, obtaining as response a vector with the current state of the plant. In this case-study, the length of the data packets is 28 bytes for the user data and 20 bytes for the system state. The control parameters sent from the view to the server are: the control mode (manual/automatic), the PID parameters (Kp, Ti, Td), the manual pumps settings (q1, q2) and the set-point of the tank 1. The information returned to the view is composed of five values: the sampling time, the input flows of both pumps (q1, q2), and the liquid levels in tanks (h1, h3).

More details about the Java code to communicate the view with the remote server are not going to be given, and readers interested in this question are encouraged to download the EJS file and examine it. However, it is interesting to have a look at Figure 22 that corresponds to the properties panel of the *Connect* button. It is worth to observe that the interactivity of this button is added by the *Action* text field, so when the user clicks on the button, the *openTCP* method will be invoked. This Java method and the others for exchanging information with the server-side are located in the Custom section.

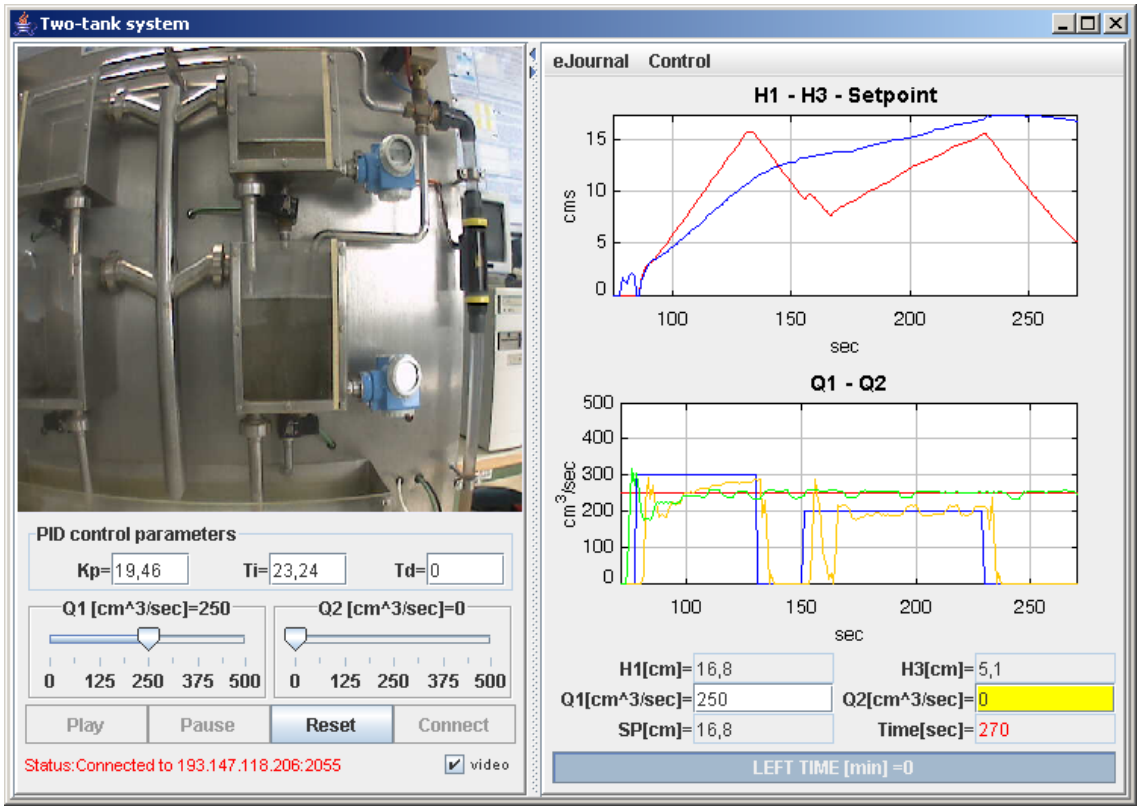


Figure 21: Interface of the web-based laboratory when working in remote mode.

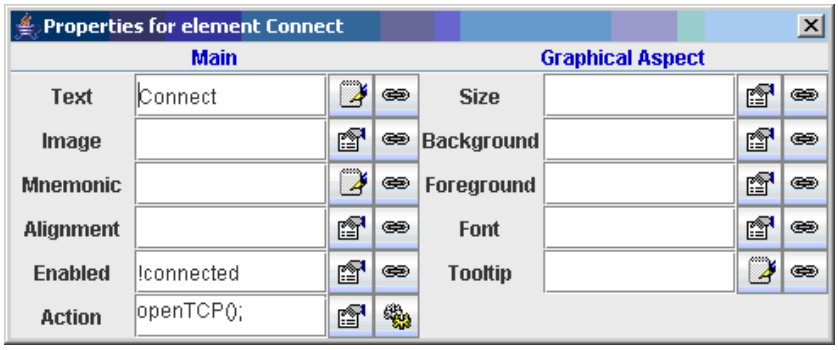


Figure 22: Properties panel of the *Connect* button.

4 Integration of EJS applications in the eMersion environment

The construction of a virtual and remote laboratory is not the last step in the development of an integral web-based experimentation environment for teaching/learning purposes. To obtain that, it is necessary the integration of the virtual and remote laboratories in an environment that provides to students, instructors and teachers the different services that they need, demand, and use when they attend traditional engineering laboratories. Elements to be available in these environments should be: math/engineering utilities, user's manuals, lab assignments, hands-on exercises, shared workspaces with the other actors in the stage, etc. and, obviously, the graphical interfaces to perform the experiments with the processes either in remote or virtual mode. The name given in the literature to such a kind of applications is *collaborative environments*. In a broad sense, a collaborative environment is a web-based application that put together and interconnects under the same framework, commonly a web browser, the different tools that the actors involved in the teaching and learning processes need to reach their goals, that is, hands-on learning for students and teaching for instructors.

However, according to the subject under study, the requirements and tools for an experimentation environment are not always exactly the same. It is obvious to observe that an experimentation environment focused to biology, English literature or control engineering should provide different facilities whereas the learning/teaching processes were always carried out across Internet. Indeed, it could happen that if the three environments were correctly designed the unique element in common would be the web browser and each environment would offer different utilities. So, it is important to bear in mind that the requirements that an experimentation environment must fulfil are of extreme importance and they must be in concordance with the kind of subject to study or practice. And this a consequence that the words ”experimentation” and ”laboratory” present different meanings according to the context where they are used.

Focusing us in the control engineering studies, to describe the architecture of a web-based experimentation environment is common to use a two-layer approach consisting in an experimentation layer plus an e-learning layer. The experimentation layer should include the creation of the interactive client-side interfaces (the EJS applets) and at the server-side the applications for running the real-time server and the transmission components for exchanging information with the remote user interfaces (in this case, the LabView server). The e-learning layer must gather all the fundamental aspects related to the web-based collaborative relationships existing among students and instructors. Nowadays, it is possible to browse Internet and find many commercial and open-source tools to create web-based collaborative environments just by a few clicks of the mouse. However, all these tools have been designed to be as generic as possible, and they are not oriented to a specific subject of the sciences and humanities studies. It is true that many of these tools are articulated in programmable modules but, too it is true that they would need a considerable programming effort if they were adapted to an engineering framework oriented to remote experimentation.

4.1 What is eMersion?

Fortunately, this problem was detected some years ago by a research group at the EPFL, and after analyzing the specifications that a collaborative environment must satisfy, they produced eMersion [12, 13]. eMersion came to life during 2000 and currently the environment is being improved and spread thanks to the experiences and suggestions of many users. Using the own authors' words obtained from their web server [39], eMersion is defined as "*a multidisciplinary initiative to deploy innovative pedagogical scenarios and flexible learning resources for supporting web-based experimentation in engineering education*". Some of the services integrated within eMersion include remote experimentation consoles, a shared laboratory journal (*eJournal*), a contextual protocol, an analysis toolkit (*SysQuake Remote*) and awareness support. One of the last refinements in EJS has been the inclusion of a set of built-in methods to integrate EJS applications in eMersion as a new service, that is, as experimentation console [12, 13].

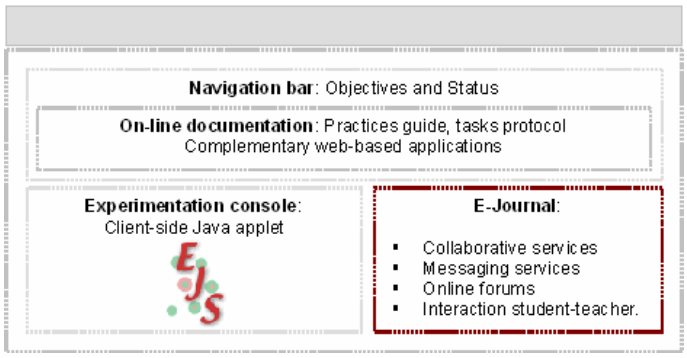


Figure 23: Architecture of the web-based experimentation environment based on EJS and eMersion.

Figure 23 shows the functional structure of the collaborative environment using EJS and eMersion. The parts that compose the system are four: the *eJournal*, the experimentation console, the on-line documentation and the navigation bar. The *eJournal* module provides a shared space to facilitate the communication and collaboration among students with instructors during all the learning process. In this workspace students can store, retrieve, share and exchange their experimental results and documents generated while they perform their experiences. The experimentation console module contains the applets developed with EJS. Using it, students can perform their laboratory assignments in a similar way to a traditional laboratory, thereby, simulating the physical phenomenon by a mathematical model (the virtual laboratory) or accessing to the real laboratory resources through a local network or Internet (the remote laboratory). The on-line documentation module provides a portfolio with theory, assignments, task protocols and user's manuals of the environment plus any additional information to complete successfully an experimental session. The protocol describes the tasks to be performed or the sequence of activities that students must perform during an experimental session. These tasks can belong to two groups: prelabs

and labs. The first ones are assignments that students must carry out in simulation mode to gain previous knowledge about the use of the application in remote mode, that is, about the real process. So once the prelabs are mastered, students obtain remote access to the plant located in the laboratory. The navigation bar module contains a set of icons to allow users the browsing of the environment for selecting the different modules aforementioned. Additionally, eMersion provides tools to simplify the organization of work-groups, offers instant messaging, news and e-mail notification services.

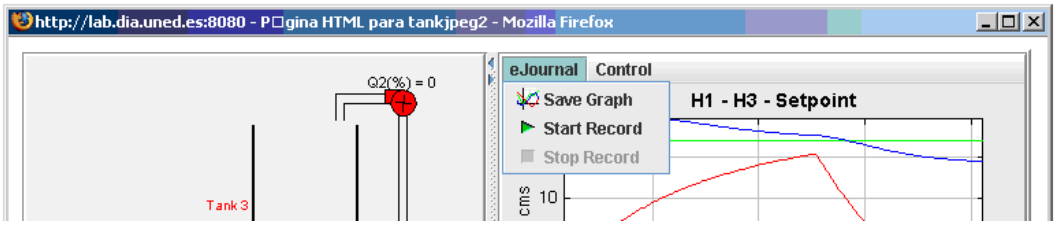
4.2 How to integrate EJS in eMersion

To describe the integration of an EJS applet in eMersion, the case study corresponding to the two-tank laboratory will be used. Since this experimentation console developed with EJS is a pure Java applet, the integration in eMersion will just consist in permitting that the applet exchanges data and snapshots of the mixed-signal scopes with the *eJournal*.

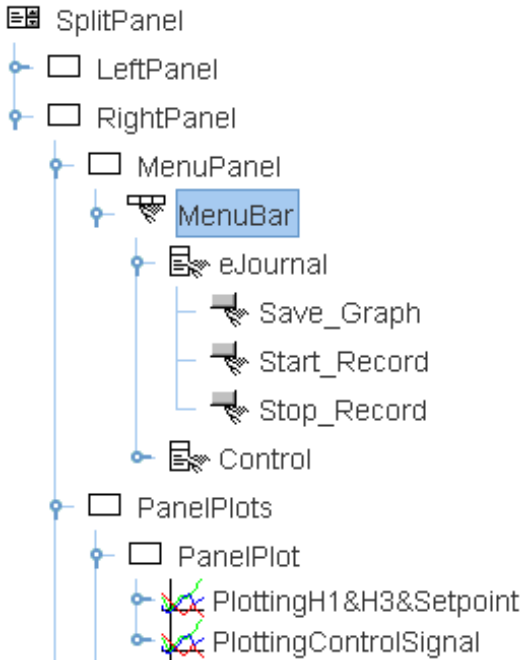
To begin with, readers must observe in Figure 24(a) the three commands located in the *eJournal* menu of the web-based laboratory. The first one, *Save Graph* lets users to save snapshots of the signal scopes, *Start Record* begins the recording of the system variables in a plain text file, and *Stop Record* stops the recording. All the information obtained by using these commands is available in the *eJournal* workspace and it can be used further for different purposes, as for instance, to analyze and design the control system the control system using tools as SysQuake, Matlab or SciLab, to complete reports, to share information with other workmates about the experiences, etc.

In normal conditions, the recording of all this information in eMersion would demand a lot of programming effort but to avoid that a set of built-in methods have been incorporated to EJS. Figure 24(b) presents the three-like structure of the three graphical elements that compound the *eJournal* menu. As well as any other EJS graphical element, each one of these menu elements has its own properties panel where the built-in methods to exchange data with the *eJournal* are invoked according to the user's actions. For instance, Figure 24(c) corresponds to the properties panel of the *Save_Graph* element and it can appreciate that in the Action text field is called the *_saveImage* method with the argument *PanelPlot*. Since this name corresponds to the EJS panel that encompasses the mixed-signal scopes (see bottom part of Figure 24(b)), the result of this invocation will be the storage in the *eJournal* of an image file named *OneTankGraph* (the first argument of the *_saveImage* method) with a snapshot of the two scopes. Other built-in methods to integrate EJS in eMersion are *_saveText* (records the temporal evolution of the system state along an experiment), *_saveState* (stores the system state in a Matlab file) and *_readState* (reads a file previously recorded with *_saveState*).

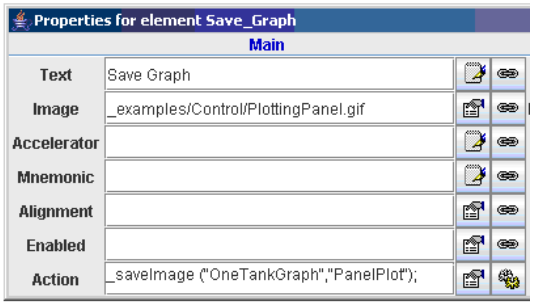
To finish up this section, Figure 25 presents three of the web pages that constitute the web-based experimentation environment based on EJS and eMersion. Figure 25(a) shows the window with the objective, the navigation and the current task panel. Figure 25(b) is the EJS experimentation console and Figure 25(c) is the *eJournal*. In this last image it can be clearly appreciated some of the files and images obtained during different experiments by the *eJournal* menu located in the EJS console.



(a)

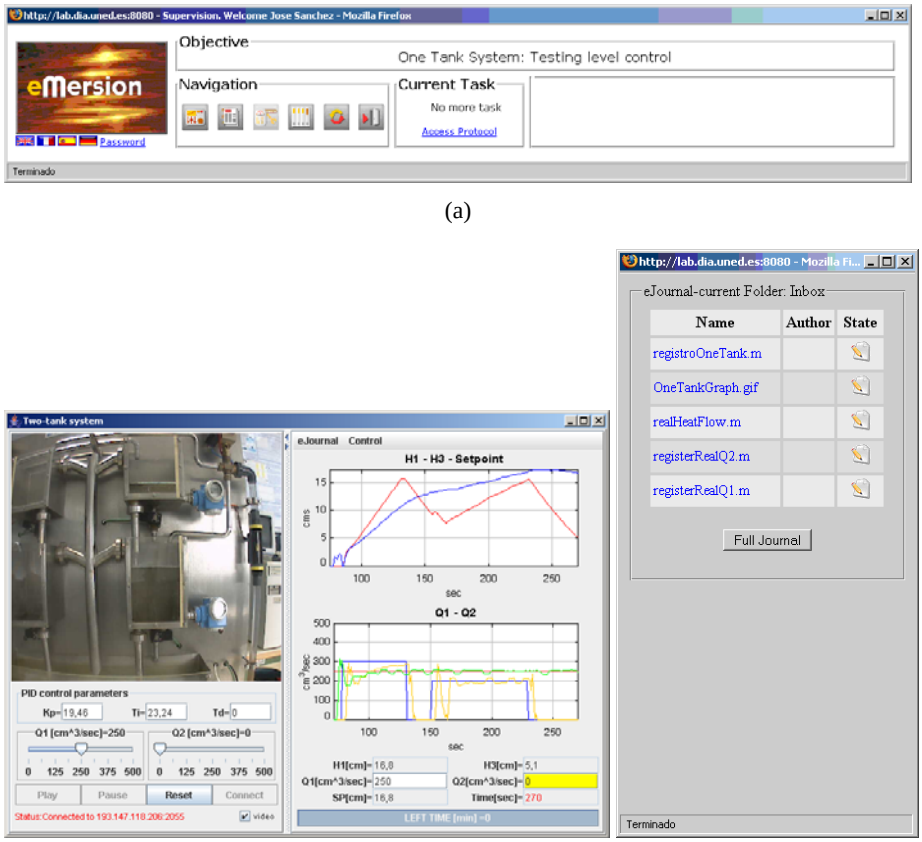


(b)



(c)

Figure 24: Integration of the EJS experimentation console in eMersion.



(a)

(b) (c)

Figure 25: Experimentation console integrated in eMersion.

5 Conclusion

This chapter has presented an easy and effective approach to develop interactive virtual and remote laboratories for education purposes. Easy Java Simulation has been used as programming environment providing a powerful and abstract way to implement this kind of laboratories. The advantages of this approach have been applied to the engineering field being focused on the Automatic Control area. In this way, a case study based on a two-coupled tank system has been presented and used along the chapter showing how virtual and remote laboratories can easily be developed. Furthermore, these laboratories have been included in a Web-based collaboration environment, eMersion, allowing students and lecturers to exchange information, services, and exercises in an organized and systematic way. The eMersion environment and its eJournal can be seen as a container and a mechanism for educators to combine and to enable communication between heterogeneous and distributed laboratory resources, Web services or Web components.

Therefore, this chapter has shown how the advances on the NICT have provided great advantages in the education field. These advantages are even more important in specific education areas with a strongly practical component, as for instance, the Automatic Control field. Our recommendation for control-engineering education is that students be made aware of problems and potential solutions arising from the increasing complexity of our technological systems. Laboratory experience is extremely important as part of control learning. Good education in automatic control recognizes the importance of laboratory experiments and unequivocally balances the theoretical and practical contents [8].

Hence, from the tools described in the chapter engineering students can learn theoretical concepts through eMersion, to study this theory by simulation using interactive virtual laboratories, and finally testing their knowledge on the real plant by a remote laboratory. In our personal experience, these tools are an excellent elements as support to teaching and learning which allow to enhance the motivation and participation of the future engineers. Authors animate to the readers to test these new ideas and features in education.

6 Acknowledgements

The authors would like to thank CICYT-FEDER for partially funding this work under grants DPI2004-01804 and DPI2004-07444-C04-04.

References

[1] K.J. Åström and B. Wittenmark. *Computer Controlled Systems - Theory and Design*. Prentice Hall, Upper Saddle River, USA, 1997.

[2] D. Bucceri, J. Sánchez, S. Dormido, P. Mullhaupt, and D. Bonvin. Interactive 3D Simulation of Flat Systems: The SpiderCrane as a Case Study. *44th IEEE Conference on Decision and Control and European Control Conference*, 2005. Seville (Spain).

[3] W. Christian, M. Belloni, and D. Brown. An Open Source Framework for Authoring Curricular Material. *Computing in Science in Engineering*, 8(5):51–58, 2006.

[4] W. Christian and M. Belloni. *Physlets: Teaching Physics with Interactive Curricular Material*. Prentice Hall, Upper Saddle River, New Jersey, 2001.

[5] W. Christian and M. Belloni. *Physlet Physics: Interactive Illustrations, Explorations, and Problems for Introductory Physics*. Prentice Hall, Upper Saddle River, New Jersey, 2004.

[6] W. Christian. *Open Source Physics: A User's Guide with Examples*. Addison-Wesley, 2004. http://www.opensourcephysics.org.

[7] S. Dormido, F. Esquembre, G. Farias, and J. Sánchez. Easy Java Simulations: A software tool to create scientific simulations in Java. *44th IEEE Conference on Decision and Control and European Control Conference*, 2005. Seville (Spain).

[8] S. Dormido. Control learning: present and future. *Annual Reviews in Control*, 28(1):115–136, 2004.

[9] N. Duro, R. Dormido, H. Vargas, S. Dormido, J. Sánchez, and R. Pastor. The Three-Tank System: A Remote and Virtual Control Laboratory using Easy Java Simulations. *44th IEEE Conference on Decision and Control and European Control Conference*, 2005. Seville (Spain).

[10] F. Esquembre and J. Sánchez. *Easy Java Simulations 3.3. How to use Ejs with Matlab and Simulink*. http://fem.um.es/Ejs/, 2004.

[11] F. Esquembre. Adding interactivity to existing Simulink models using Easy Java Simulations. *Computer Physics Communication*, 156:199–204, 2004.

[12] G. Fakas, A. Nguyen, and D. Gillet. The Electronic Laboratory Journal: A Collaborative and Cooperative Learning Environment for Web-Based Experimentation. *Computer Supported Cooperative Work*, 14:189–216, 2005.

[13] D. Gillet, A.V. Nguyen, and Y. Rekik. Collaborative web-based experimentation in flexible engineering education. *IEEE Transactions on Education*, 48(4):696–704, 2005.

[14] D. Gillet, C. Salzmann, and E. Gorrochategui. *Remote Manipulation with LabVIEW for Educational Purposes*. Internet Applications in LabVIEW, Prentice Hall PTR. National Instruments - Virtual Instrumentation Series, 2000.

[15] J.L. Guzmán, K.J. Åstrom̈, S. Dormido, T. Hägglund, and Y. Piguet. Interactive learning modules for PID control. *Proceedings of the 7th IFAC Symposium on Advances in Control Education ACE06*, 2006. Madrid (Spain).

[16] J.L. Guzmán, M. Berenguel, and S. Dormido. Interactive teaching of constrained generalized predictive control. *IEEE Control Systems Magazine*, 25(2):79–85, 2005. Available: http://aer.ual.es/siso-gpcit/.

[17] J.L. Guzmán, M. Berenguel, F. Rodríguez, and S. Dormido. Web-based remote control laboratory using a greenhouse scale model. *Computer Applications in Engineering Education*, 13(2):111–124, 2005.

[18] J.L. Guzmán, F. Rodríguez, M. Berenguel, and S. Dormido. Virtual lab for teaching greenhouse climatic control. *Proceedings of the 16th IFAC World Congress*, 2005. Prague (Czech Republic).

[19] J.L. Guzmán. *Interactive Control System Design*. Phd thesis, University of Almería, Spain, 2006.

[20] A. Hernández, M. Mañanas, and R. Costa-Castelló. RespiLab: A virtual laboratory for the analysis of human respiratory control system. *7th IFAC Symposium on Advances in Control Education*, 2006. Madrid (Spain).

[21] IEEECSS. IEEE Control System Society. http://www.ieeecss.org/.

[22] IEEECSS. Technical Committee on Control Education. http://www.math.ku.edu/ksacg/ CSS/tcce.html.

[23] IFAC. International Federation of Automatic Control. http://www.ifac-control.org/.

[24] IFAC. Technical Committee on Control Education. http://www.gu.edu.au/centre/icsl/edcom/.

[25] K.H. Johansson. The quadruple-tank process - a multivariable laboratory process with an adjustable zero. *IEEE Transcations on Control Systems Technology*, 8(3):456–465, 2000.

[26] A. Mago, W. Medina, L. Fermín, and J. Grieco. Control systems simulator for wheeled robots using Easy Java Simulations. *7th IFAC Symposium on Advances in Control Education*, 2006. Madrid (Spain).

[27] The MathWorks Inc. *Using Simulink. Dynamic System Simulation for Matlab.*, 1999.

[28] The MathWorks Inc. *Using Matlab. The Language of Technical Computing*, 2002.

[29] R. Medina. *LabVIEW User Manual*. National Instruments Corporation, 2002.

[30] R.M. Murray, K.J. Åström, S.P. Boyd, R.W. Brockett, and G. Stein. Future directions in control in a information-rich world. *IEEE Control System Magazine*, 9(2):20–33, 2003.

[31] Y. Piguet. *SysQuake 3 User Manual*. Calerga S'arl, Lausanne (Switzerland), 2004.

[32] F. Rodríguez, M. Berenguel, J.L. Guzmán, and S. Dormido. A virtual course on automation of agricultural systems. *International Journal of Engineering Education*, 22(6):1197–1209, 2006.

[33] C. Salzmann, D. Gillet, and P. Huguenin. Introduction to Real-time Control using LabVIEW^{TM} with an Application to Distance Learning. *International Journal of Engineering Education. Special Issue: LabVIEW Applications in Engineering Education*, 16(3):255–272, 2000.

[34] J. Sánchez, S. Dormido, and F. Esquembre. The learning of control concepts using interactive tools. *Computer Applications in Engineering Education*, 13(1):84–98, 2004.

[35] T.D. Urdan and C.C. Weggen. Corporate e-learning: exploring a new frontier. http://www.elearning.nl/publicaties/marktonderzoek/New_Frontier.pdf, 2000.

[36] H. Vargas, J. Sánchez, S. Dormido, G. Farias, N. Duro, R. Dormido-Canto, S. Dormido-Canto, and F. Esquembre. Web-based Learning Resources for Vocational Training & Automation Technicians. *7th IFAC Symposium on Advances in Control Education*, 2006. Madrid (Spain).

[37] A. Visioli and F. Pasini. A virtual laboratory for the learning of process controllers design. *7th IFAC Symposium on Advances in Control Education*, 2006. Madrid (Spain).

[38] Virtual and remote labs. *UNED/University of Almería (Spain)*, 2007. http://lab.dia.uned.es:8080/emersion/.

[39] eMersion. http://emersion.epfl.ch.

[40] Scicos: Scilab's block diagram modeler/simulator. http://www.scicos.org/.

[41] Scilab. http://www.scilab.org/.

Contact information

José Luis Guzmán[1⋆]
Héctor Vargas[2]
José Sánchez[2]
Manuel Berenguel[1]
Sebastián Dormido[2]
Francisco Rodríguez[1]

1. Dep. Lenguajes y Computación, University of Almería. Address: Ctra. Sacramento s/n, 04120, Almería, Spain. Tel: +34 950015849, Fax: +34 950015129. Email:{joguzman,beren,frrodrig}@ual.es.

2. Dep. Informática y Automática, National University of Distance Education. Address: C/. Juan del Rosal 16, 28040, Madrid, Spain. Tel: +34 913987151, Fax: +34 913986697. Email:hvargas@bec.uned.es,{jsanchez,sdormido}@dia.uned.es.

⋆ Corresponding author.