Remote Laboratory Architecture for the Validation of Industrial Control Applications

Houcine Hassan, Carlos Domínguez, Juan-Miguel Martínez, Angel Perles, and José Albaladejo

***Abstract*—The seven engineering degrees of the Higher Technical School of Design Engineering (ETSID), Technical University of Valencia, include in their formation programs, subjects, and laboratory projects to instruct students in the aspects of the design, development, and validation of applications for process control, automation, industrial informatics, and embedded systems. Moreover, the authors participate in European projects of education such as the International Network of Embedded System (INES) and the European Project Semester (EPS), where exchange students remotely perform the first phase of their projects at ETSID from their home universities. To significantly reduce the cost of installing a huge number of real prototypes in labs and to fulfill the distance requirements of the exchange students, a remote laboratory architecture, i.e., simPROCes, has been designed. simPROCes not only permits the teleoperation of simulators/real prototypes but also allows that complete control applications be remotely tested and validated. simPROCes is precisely specified to be independent of the model of computer, data acquisition card, programming language, and operating system, and is transparent to the programmer and easy to use. This system is useful both for the education and development of control application purposes. A water tank process shows the installation procedure of simPROCes to test and validate control applications. The experience of applying simPROCes in the electronic engineering degree and within the framework of INES and EPS has been successfully rated by student surveys.**

***Index Terms*—Embedded system, input/output (I/O) system, international team, process control, project, remote laboratory, simulation.**

I. Introduction

THE SEVEN engineering degrees of the Higher Technical School of Design Engineering (ETSID), Technical University of Valencia (UPV), include in their formation programs, subjects, and laboratory projects to instruct students in the aspects of the design, development, and validation of applications for process control and embedded systems. These subjects require a process prototype (i.e., plane or tank), and the aim is to design a computer-based system for controlling the process. The computer system can be either an embedded computer or a personal computer (PC).

Each subject uses various prototypes (i.e., the industrial informatics subject uses a tank, a robot arm, and a greenhouse) [1]. The mean of the number of students per subject is 70, and the team project per prototype is composed of two students; therefore, for each subject, 35 prototypes are required. If this number is multiplied by the number of subjects with the aforementioned characteristics, the cost (laboratory maintenance, technicians, and materials) of having completely equipped labs is very high and consequently unaffordable.

Moreover, the authors participate in two European projects of education, i.e., the International Network of Embedded System (INES) [2] and the European Project Semester (EPS) [3]. Within these projects, international and interdisciplinary teams of students perform a project at ETSID consisting of the development of an embedded controller for industrial application. Due to the short time duration of in-placement at ETSID (two months), students have to develop the first phase of their project remotely from their home university to successfully complete the work.

To significantly reduce the cost of maintaining a huge number of real prototypes in laboratories, to have more flexibility in the lab timetables, and to accomplish the distance requirements of exchange students, a laboratory architecture, i.e., simPROCes, that permits remote interaction with process simulators and/or real prototypes has been designed. Simulators of the real prototypes are developed and run by simPROCes in a server host. Consequently, huge student groups interact with the simulators to progress their projects without being limited either by time, space, and distance constraints or by the unavailability of real prototypes. Only two real processes of each type are bought and connected to the input/output (I/O) data acquisition (DAQ) cards in the server. Real prototypes are used in the validation phase of the control application after being tested with simulators what leads to the reduction of malpractice costs.

simPROCes is carefully designed to be independent of any particular model of computer (i.e., ARM9 and PC), DAQ cards (i.e., PCI-9112 and Nudam-RS485), programming language (i.e., Borland C++ and LabView), and operating system (i.e., Windows and Linux). Other advantageous characteristics that have been considered in the design are that simPROCes is transparent to the programmer/student and easy to use, i.e., without the need for special training.

This paper presents the experience of applying simPROCes for the validation of industrial control applications in the degree of electronic engineering (EE) at ETSID and within the framework of two European projects of education where ETSID is participating, i.e., the Intensive Program INES and EPS.

After the introduction, Section II presents the most representative work performed in remote and virtual labs.

Manuscript received April 12, 2007; revised August 14, 2007.

The authors are with the Departamento de Informárica de Sistemas y Computadores, Universidad Politécnica de Valencia, 46022 Valencia, Spain (e-mail: husein@disca.upv.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2007.907015

The ETSID, INES, and EPS learning methodology based on projects using simPROCes is described in Section III. The components of the remote lab architecture simPROCes are detailed in Section IV. The case study of the water tank, i.e., to show how to configure simPROCes and how to validate control applications, is analyzed in Section V. Section VI presents the evaluation results of simPROCes by ETSID and foreign students. The conclusions are sum up in Section VII.

II. Related Work

Ferrolho and Crisóstomo [4] designed a collection of software tools that allow the development of industrial applications of monitoring and controlling robotic networks and computer numerical control (CNC) machines inserted in flexible manufacturing cells. In [5], the trajectory tracking of a car-like mobile robot is performed wirelessly and through the Internet. The time delay transmission is tackled by a quality-of-service module. An Internet-based teleoperation system for the control of an autonomous mobile robot with a two-layer fuzzy controller is implemented in [6]. To robustly transmit the surroundings and control information of the robot, packet-type data are employed. A system architecture of the Internet-based telelaboratory allows the remote control and programming of two educational online robots [7]. The delay associated with Internet communications is solved by using predictive-display techniques. Chang *et al.* [8] propose a real-time control of a swinging load through the Internet. Command shaping is applied to move a cable-suspended load at the end point of a robot to minimize load swing. The effects of transmission time delay are assessed. The design of an agent-based control system for industrial manufacturing is presented in [9]. The system was integrated with the Distributed Component Object Model and Transmission Control Protocol (TCP)/IP communication. A distributed network control system that provides a flexible way to control an industrial plant is proposed in [10]. TCP/IP connectivity is provided. This system is applied in the control of production plants. A framework for the Internet-based interaction of humans, robots, and environments by using agent technology is proposed in [11]. The system decreases the time for Internet connection. A control network for industrial battery management is discussed in [12]. The battery network is locally managed by control nodes that communicate with the remote user. Kofman *et al.* [13] present the teleoperation of a robot manipulator by a human operator for dynamic environments. The teleoperation is carried out by using a vision-based human–robot interface. A model for sharing real laboratories on the Internet has been developed in the field of electronic measurements [14]. The online laboratory allows the execution via Web of real experiments.

The aforementioned papers present useful systems that are focused on teleoperation in specific application fields like manufacturing, robotics, etc. The key issue of the work presented in this paper is to design an architecture that allows remote development and validation of embedded control applications. simPROCes is not a simulator of a specific process, it is a framework to integrate process and I/O simulators with control

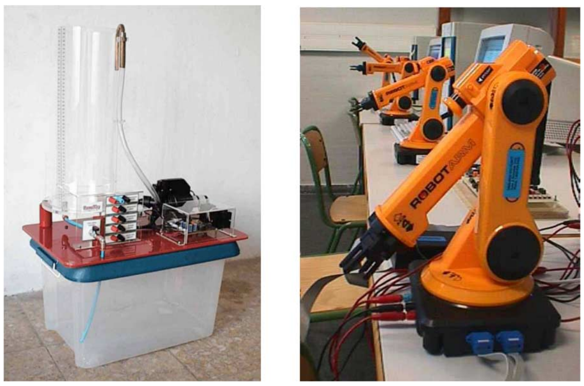


Fig. 1. Real prototypes: water tank and robot arms.

applications. These simulators should be developed for each application accordingly to the interfaces of simPROCes. The controller that is tested on simulators can be connected to real processes without source code changes what will minimize errors in the integration phase. The switching between process simulators and the real prototypes will permit configuring labs according to budget projects. This model is useful both for the education and development of control application purposes.

III. ETSID-INES-EPS Projects-Based Learning

In the EE degree at ETSID and within the framework of INES and EPS projects, students require the use of prototypes of real processes to complete their engineer formation in the aspects of the design of industrial control applications.

A. Project Organization in EE

An innovative educational methodology [1] based on cooperative and active learning theories [15] that foster the work in teams to perform a project [16] is applied in the EE degree. A multidisciplinary project composed of five compulsory subjects (i.e., industrial informatics, process control, analog electronics, digital electronics, and instrumentation) is proposed. Each of the subjects needs interaction with different industrial prototypes running on a server (i.e., green house and robot), as well as with I/O interfaces (i.e., PCI-9112 lab card [17] and simseny [18]).

Three control system projects are offered in the current academic year, i.e., water tank, greenhouse, and robot arm.

Due to the huge number of students and the high cost of maintaining each of the prototypes, ETSID cannot provide each of the 35 teams/class with an exclusive prototype. The solution is to have two real prototypes of each type (six in total) connected to a server via the DAQ PCI-9112 card.

The real processes (see Fig. 1) are used only in the validation phase of the project and to run process simulators on the server (see Fig. 2), so as all teams interact with them during the project sessions to test their application control. simPROCes allows both the real prototypes and the simulators to be accessed by clients through TCP/IP.

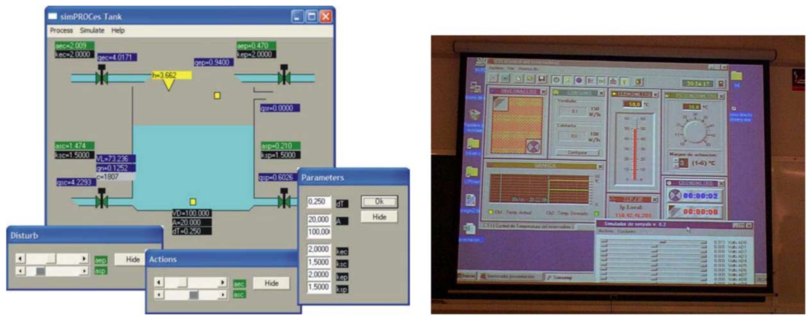


Fig. 2. Process simulators: water tank and green house.

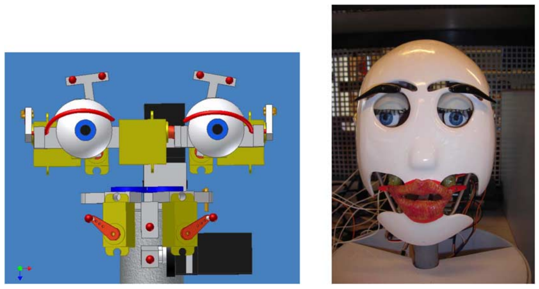


Fig. 3. Robot assistant and simulator.

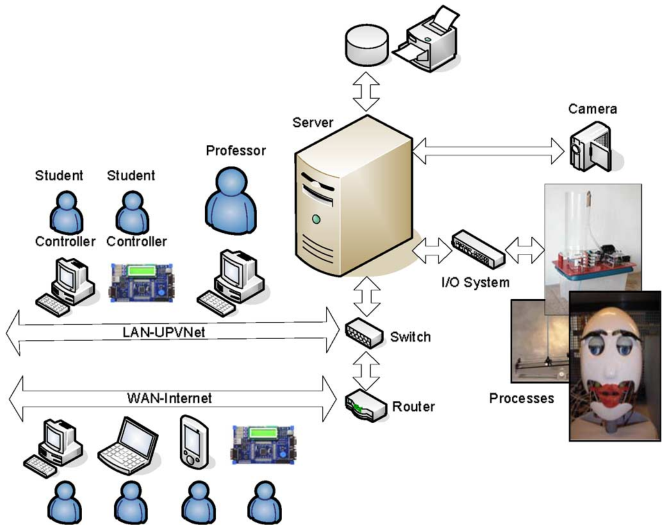


Fig. 4. Hardware architecture.

B. European Educational Projects

The authors participate in European projects of education such as INES and EPS. Within these projects, students from different countries perform their project at ETSID by using the facilities of the remote lab architecture. In INES, students should develop an embedded controller of a water tank (see Figs. 1 and 2) based on the ARM9 processor [19]. The control application consists of the control of the water tank level.

Regarding EPS, international and interdisciplinary teams perform a project consisting of the development of an assistant robot controller [20] (see Fig. 3) and the programming of different artificial emotions [21] that the robot presents when interacting with people [22].

The Keil MCBSTR9 Evaluation Board [23] is used to implement the tank regulator and the robot controller.

The main constraint of the INES and EPS projects is the short time available for completing the project, i.e., two months. If teams would not have the possibility of accessing the process before their in-placement for progressive development and testing, they could not perform their project in such a short period of time. To overcome this situation, simPROCes is provided to allow teams to learn about the specifications and functionalities of the water tank and the robot assistant from their home university. Likewise, they can start the design parts, the implementation of the controlling programs, and testing their solution. In the final month, at ETSID, students access the real prototype connected to the server through simPROCes to validate the control application programmed in the embedded controller.

IV. Remote Laboratory Architecture

simPROCes is the remote laboratory that permits interaction with the following: 1) process simulators to test the control applications developed by programmers; and 2) real prototypes for the validation of the developed controllers. simPROCes has been developed taking into account the following considerations:

- Useful to develop controllers
  - — simulate processes and I/O DAQ cards;
  - — several simultaneous simulations;
  - — switch real prototypes and process models;
  - — independent of programming languages and I/O DAQ.
- Flexible
  - — installable in any node of the network;
  - — transparent to the user, no special training required, focus on design.
- Cost effective
  - — reduce the number of necessary real processes;
  - — reduce maintenance costs regarding personnel and materials.

A. Hardware Architecture

The laboratory is organized as can be seen in Fig. 4.

The process prototypes are connected to the server host through four PCI-9112 DAQ I/O cards. The process simulators are running in the server over the simPROCes software.

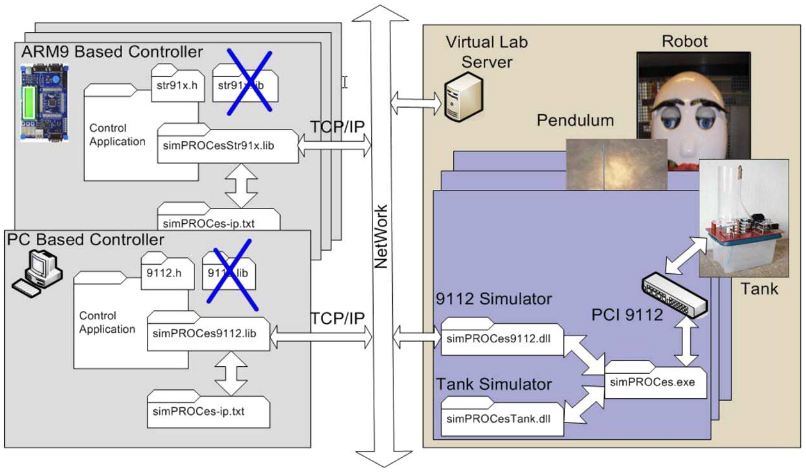


Fig. 5. simPROCes software architecture.

Both the processes and the simulators can be accessed from clients connected at the UPV local area network and from the EPS and INES clients of the partner universities. TCP/IP has been employed to allow connectivity.

Currently, the clients can develop control applications based on two platforms, i.e., embedded computer and PC. The embedded computer is the STR91X based on the ARM9 processor. The Keil MCBSTR9 Evaluation Board incorporates this processor and is available for use in the labs with simPROCes. For the MCBSTR9 board, the applications are developed and cross compiled with the Keil ULINK USB-JTAG Adapter and the μVision IDE and Debugger. For the PC, they are developed with visual programming environments such as Borland C++ builder [24]. To allow students to watch the process in progress, an IP camera (see Fig. 4) has been installed in the lab. A video server accepts calls from any remote location, establishing a direct video link that is based on Rapid Transport Protocol for real-time video streaming.

B. *Software Architecture*

The simPROCes software is mainly composed of a server of real processes and process models (right of Fig. 5), and a set of clients that develop control applications (left of Fig. 5).

The basic elements of simPROCes are the following:

- server of real processes/simulators;
- connector of control applications and remote processes/ simulators via TCP/IP;
- emulator of the DAQ card calls by capturing DAQ calls and redirecting to the server;
- a set of physical process models;
- an electrical signal viewer (see Fig. 8).

These components are described in the next subsections.

1) simPROCes Components: The main component *simPROCes.exe* of the architecture permits the programmer to select and load in run time the library of the process prototype/simulator (i.e., simPROCtank.dll of the water tank simulator) and the library of the I/O DAQ card/simulator (i.e., simPROCes9112.dll for the PCI-9112 card). simPROCes.exe is the link between the process and the I/O system. It defines the connections of the ports and the physical signals.

The I/O emulation is performed with two modules linked over a TCP/IP connection, what will permit the remote control of the processes. One of the modules is the library loaded by simPROCes in the server (i.e., simPROCes9112.dll), and the other is the library that should be linked with the control application in the client (i.e., simPROCes9112.lib). The main idea is to capture all the calls to the functions of the DAQ card library and redirect them via TCP/IP to the process simulator or to the real process connected to the server DAQ cards.

The use of a dynamic link library (dll) makes this tool applicable in programming environments such as Microsoft Visual C++, Borland C++, Visual Basic, Borland Delphi, National Instruments LabView/LabWindows, HP VEE, etc.

From the viewpoint of the control application programmer, the only change to perform is to replace the library of the I/O DAQ/controller manufacturer (i.e., 9112.lib) by the library of the simPROCes (i.e., simPROCes9112.lib) in the linking process. The code of the application and the compilation process are not affected by this change because the same header file is used in both the cases. That is, the I/O emulator presents the same interface as the I/O DAQ (i.e., 9112.h).

During the testing phase, the programmer of the control application checks the developed controller with the process simulator running in the server. After, in the validation phase, without changing the code of the application, the programmer validates the application with the real process and the I/O DAQ of the manufacturer by selecting in the mounting process the library that corresponds to the real process.

The remote access to the processes and simulators is allowed by configuring in the local host the simPROCes-ip.txt file. The IP address of the server host where simPROCes is running should be included in this file. This file should be available in a directory listed in the O.S. *path* environment variable. Otherwise, if the programmer has obtained from the authors the simPROCes software and libraries [25], he could execute the whole simulation in the local host. In this case, before starting the simulation, the autoloop address 127.0.0.1 should be written in the simPROCes-ip.txt file.

```
int initialise_p(char *util);
int parametrize(double *pars);
int start(double *state);
int stop(void);
int continue(void);
int action(double *accs);
int disturb(double *dist);
int observe(double *obs);
int inform_p(int *options);
```

Fig. 6. Process interface.

```
int initialise_io(char *util);
int connect(int nIN, *IN, int nOUT, *OUT);
int IN(double *param);
int OUT(double *state);
int inform_io(int *options);
```

Fig. 7. I/O interface.

2) simPROCes Interfaces: The simPROCes architecture also defines the interfaces (see Fig. 6) that the processes and I/O dlls should respect to connect for the interaction with the control application.

initialise() permits simPROCes to determine the basic characteristics of the simulator/prototype related to the sensory and action signals, as well as to a set of state variables. parametrize_p() configures the modeled system such as the surface of the base or the constants of the valves. start(), stop(), and continue() control the state of the process/simulator, and start() configures an initial state of the process. action() and disturb() produce the excitations of the process. The actions come from the control application, and the disturbances are generated by simPROCes. observe() gets the information from the sensors in the server and sends it to the control application in the client. inform_p() shows a graphical representation of the process state (see Fig. 3). Although the students develop their Graphical User Interface for their control applications, this view helps them in the specification phase.

Fig. 7 shows the interface that the I/O modules should use to interact with simPROCes.

initialise_io() permits simPROCes to determine the basic characteristics of the I/O simulator/prototype related to the type of signals it can manage. connect() describes the I/O ports of the sensors and the actuators. IN() gets information from the sensors. OUT() sends the signals to the actuators. inform_io() shows a graphical representation of the I/O system to see and/or change the state of the signals, as can be seen in Fig. 8.

As well as in the interaction with real prototypes as in the simulation, the DAQ is performed using the same header file of the manufacturer. For the example of the PCI-9112 DAQ card, the functions that are used to initialize the DAQ card and to establish operation ranges and modes are $W\_9112\_Initial$, $W\_9112\_AD\_Set\_Mode$, and $W\_9112\_AD\_Set\_Range$. To perform the digital inputs of the sensors, we have W_9112_DI. For the analog outputs, we have W_9112_DA. Moreover, to analyze the analog sensors, we have $W\_9112\_AD\_Set\_Channel$, $W\_9112\_AD\_Soft\_Trig$, and $W\_9112\_AD\_Aquire$.

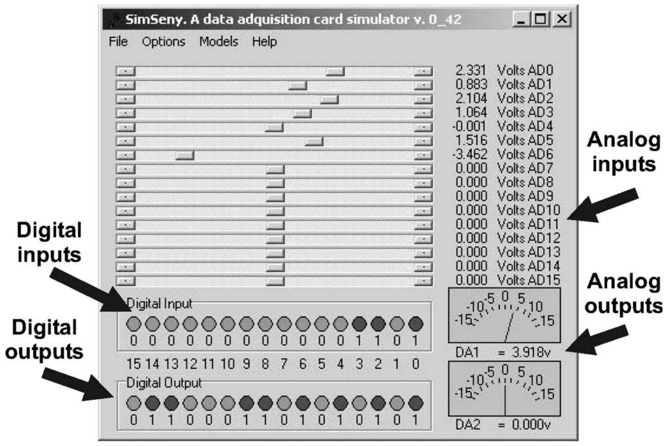


Fig. 8. PCI-9112 simulator.

A set of utility tables of the process and the I/O system have been developed and stored in a database for simPROCes testing and configuration.

C. Security Aspects

The scope of simPROCes does not require costly encryption, but access to the system must be restricted. Only students of the projects can access the remote server, so a validation mechanism has been implemented. When starting the application, a windows pop-up requires the validation with a username and password. Students at ETSID have a unique "username/password" to access electronic services, so it has been decided to maintain the same approach in the application. The validation is done using the validation servers of UPV. To maintain the same user/password tuple, first the existence of the user login through the lightweight directory access protocol (LDAP) service is checked, and then a data encryption standard (DES) key of the username/password of the student at UPV services is required. When the user logs in the remote simulation system, the name and password are coded to DES in the local side and transferred to the server.

V. Case Study: Water Tank Control

This section details the steps to configure the components (processes, simulators, I/O, and network) of simPROCes to validate control applications. The configuration is illustrated using the water tank case study and the PC as the platform for development of control applications. The steps to validate control applications are as follows:

1) Build the control application to be validated (client).
2) Start an instance of simPROCes architecture (server).
3) Configure the IP address of the server.
4) Test the control application with the tank simulator.
5) Validate the control application with the real tank.

A. Control Application

The programmer should develop the control application that it intends to validate. Fig. 9 shows an example of a water

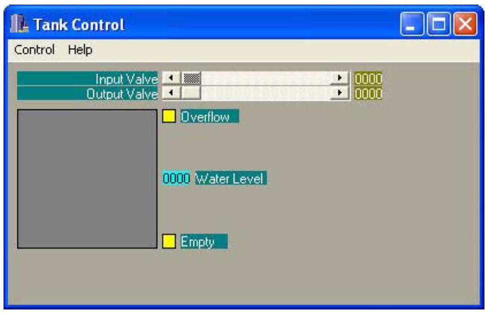
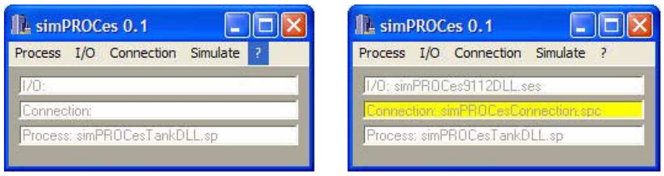


Fig. 9. Tank control application example.



I/O:
Connection:
Process: simPROCesTankDLL.sp

I/O: simPROCes9112DLL.ses
Connection: simPROCesConnection.spc
Process: simPROCesTankDLL.sp

Fig. 10. simPROCes and water tank setup.

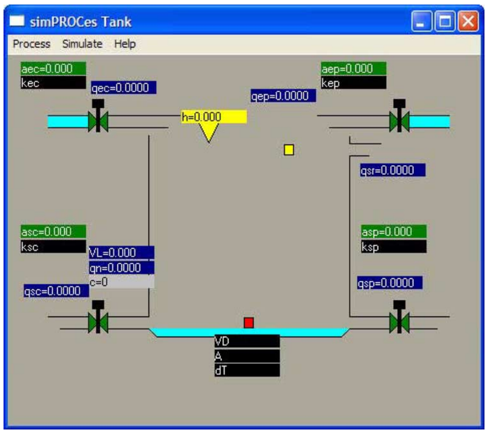


Fig. 11. Water tank instance.

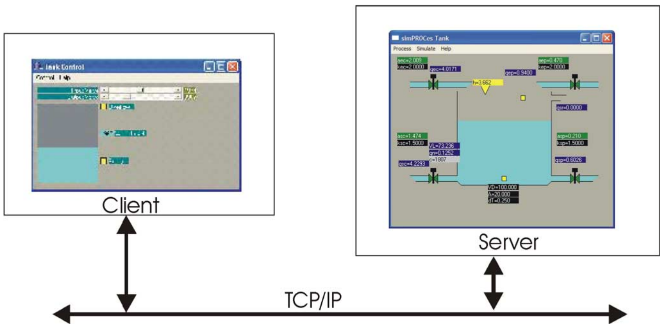


Fig. 12. Testing the control application with the water tank simulator.

tank control. Other examples are provided with the simPROCes toolkit [25].

The application in Fig. 9 performs the control and monitoring of the liquid level of the tank. The programmer creates the application in the local host where it implements the controller (i.e., proportional-integral-differential regulator) and the graphical interface that he wants to test. The application program interacts by TCP/IP with the water tank prototype/simulator running in the server host (i.e., open/close input and output valves). This application has been developed with Borland C++ Builder 5 [22]. The code of the application remains the same either when using the real tank or the simulator. The use of simPROCes does not affect the compilation of the program. Recall that the "9112.h" header file of the manufacturer is used for both cases. However, to use the remote server, in the application mounting process, the library of the manufacturer "9112.lib" is changed by the "simPROCes9112.lib" library.

B. simPROCes Configuration

To test the control application, simPROCes should previously be initialized. The first step of the initialization is to run simPROCes and to load the process prototype/simulator (dynamic link with the water tank prototype/simulator "simPROCesTankDLL.sp"), as can be seen in Fig. 10.

Second, the I/O should be loaded to link the simPROCes modules with the DAQ PCI-9112 "simPROCes9112DLL.ses." After that, the schemes of connections of the process variables and I/O simulator channels are established. These connections are linked to the tank simulator by loading the "simPROCesConnection.spc" file. Finally, the water tank instance of the simPROCes simulator is launched (see Fig. 11).

C. Network Configuration

To have remote access to the tank, the "simPROCes-ip.txt" should be edited in the local host. This file has to include the IP address of the server host where simPROCes is running. This information is provided each course by the lab technician.

D. Control Application Testing Based on Tank Simulator

After having build and executed the control application in the local machine, and initialised and run the simulator of the water tank in the remote host, the programmer proceeds to the testing of the developed controller (see Fig. 12).

The client program sends control actions to the server simulator and visualizes the levels of the tank that receives from the server when it applies open and close actions of the input and output valves.

E. Control Application Validation Based on Tank Prototype

The application is validated with the real water tank, as can be seen in Fig. 13. To validate the control application, the programmer should select, when starting simPROCes, the library that manages the real water tank "simPROCesRealTank.dll" instead of the library of the simulator of the tank "simPROCes-Tank.dll." The client application sends control actions to the real tank and visualizes its state that receives from the server.

The same experiments of varying input and output valves have been carried out with the real tank and the simulator (Figs. 12 and 13). The results of the regulation of the level with both configurations can be seen in Fig. 14.

The *Reference* label is the liquid level value that a controller should fulfil either with the simulator (*Sim Level*) or with the

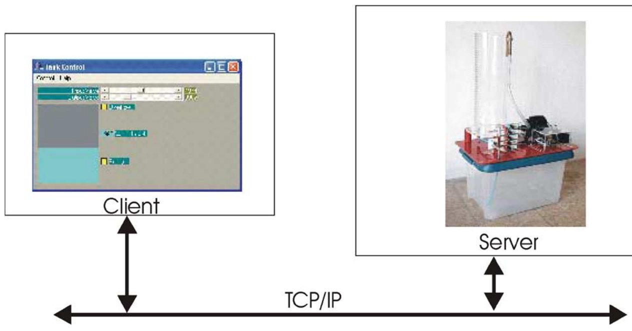


Fig. 13. Validation of the control application with the real tank.

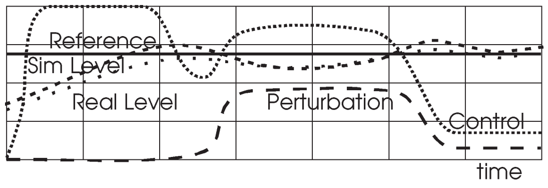


Fig. 14. Tank liquid level regulation.

1. Has been easy to install simPROCes?
2. Has been easy to use simPROCes?
3. Is it better to perform the project first with the simulator and then to validate it with the real process?
4. Has simPROCes simulator permitted you to progress your application more than if you had worked only with the real process?
5. Has simPROCes permitted you to replace easily the simulator by the real process?
6. Would you recommend the use of simPROCes to students for the next course?

Fig. 15. Questions of simPROCes survey.

real tank (*Real Level*). The same disturbance has been applied for both cases to analyze the behavior of the controller. It can be appreciated that, as well as with the simulator as with the real tank, the controller fulfils the reference. Moreover, both systems reach the steady level approximately at the same time. The control of the simulator is faster because the model of the process does not consider all the inertial laws of a real water tank.

VI. Remote Laboratory Evaluation

simPROCes has been evaluated by ETSID students and by the exchange students of the European projects. For three years, four groups per year of 70 students of EE at ETSID have been using the platform during the development of their project. At the end of the project, they answered the questions contained in the query (Fig. 15).

The answers are rated from A to E marks. Table I shows the meaning of each mark.

The results of this assessment can be seen in Fig. 16.

simPROCes has been rated very well by the ETSID students. The big part of them does not have difficulties to install and use simPROCes (82%). The results of question 2 are no better than question 1 because students need a period of adaptation to the system. Related to questions 3, 4, and 5, students prefer to

TABLE I
Significance of Marks

Mark	Meaning
A	Strongly disagree
B	Disagree
C	Unsure
D	Agree
E	Strongly agree

The results of this assessment can be seen in Figure 16.

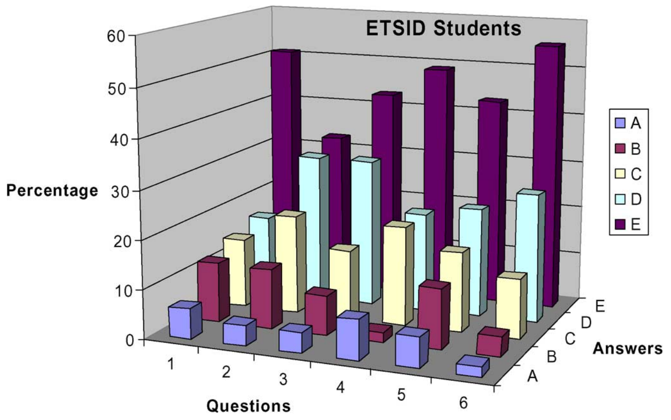


Fig. 16. ETSID student opinion.

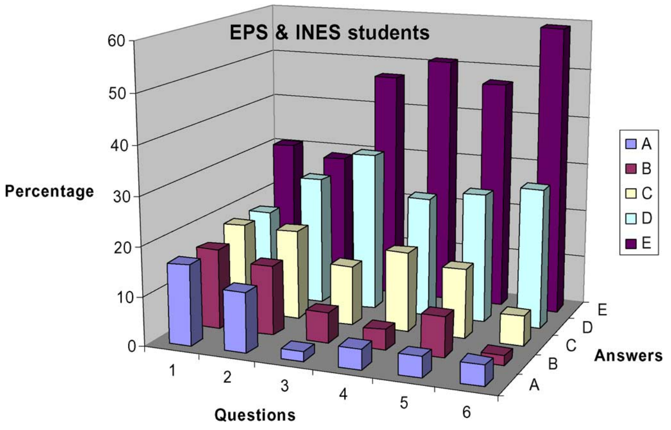


Fig. 17. EPS and INES student opinion.

use the simulator and after validate the application with the real process because they progress more in the development of their work. Likewise, they do not find it difficult to switch from the simulator to the real process. Only 6% of the surveyed students would not recommend simPROCes.

Concerning EPS and INES students, for two years, two groups per year of ten students of the EPS project and two groups per year of 12 students of the INES project have used simPROCes to perform their project. They have assessed simPROCes by answering the questions of the same survey, and the result of their opinion is plotted in Fig. 17.

Although simPROCes is overall very well rated by foreign students, they find little more difficulties in the installation and initial usage of the system than ETSID students. The reasons are mainly the reduced communication between the students and the professor/technicians, and the partial perception of the lab and the network requirements. However, for questions 3, 4, and 5, foreign students have better rated simPROCes than ETSID students because the former are students of last years; hence, they have more skills in the design of control applications and strongly appreciate the availability of a simulator to progress the project and to switch to real process with such a facility for its validation. Furthermore, only 4% of the exchange students would not recommend simPROCes.

VII. Conclusion

To reduce the costs (laboratory maintenance, technicians, and materials) of having completely equipped labs with huge number of real prototypes and to allow exchange students to remotely perform their project at ETSID from their home universities, a remote laboratory architecture, i.e., simPROCes, that permits the testing and validation of control application programs of real/simulated processes has been designed.

simPROCes has been precisely specified to be independent of any particular model of computer, DAQ cards, programming languages, and operating systems.

The successful experience of applying simPROCes in the EE degree at ETSID and in two European projects, i.e., INES and EPS, has been presented.

The components of the hardware and software architecture of simPROCes and their configuration have been detailed.

A case study of a water tank process has been used to show how to configure the components of simPROCes to test and validate control applications.

The evaluation of simPROCes both by ETSID students and by the exchange students has been performed. The surveys have shown that students have a very good opinion of simPROCes, which is encouraging the authors to continue improving this platform for future courses.

In the future work, the development of more process models and real prototypes both for the EE degree and for the rest of the ETSID degrees is intended. The improvement of the communication protocol to cope with control applications that have critical real-time constraints will be undertaken.

Acknowledgment

The authors would like to thank the Dean of the school for the continuous support of our ideas in innovation in higher education and to the rest of their team, the Coordinator of EPS for his comments that have improved our experiences, the Coordinator of INES for all his ideas that helped us design our specific project, the Lecturers of the EE degree who participated in this project for their effort and willingness to make this experience a success, and the anonymous reviewers for their helpful and valuable suggestions.

References

[1] H. Hassan, J. M. Martínez, C. Domínguez, A. Perles, and J. Albaladejo, “Innovative methodology to improve the quality of electronic engineering,” *IEEE Trans. Educ.*, vol. 47, no. 4, pp. 446–452, Nov. 2004.

[2] R. Peters, and H. Hassan, “International network of embedded systems,” *Advanced Logic in Control Programming Intensive Programme.* Alice Project Internal Report 2007. [Online]. Available: http://relint.etsid.upv.es/->alumnado->Intensive_Programme

[3] L. Gustafsson, P. Fuentes, and H. Hassan, *European Project Semester,* Mar. 2007. http://relint.etsid.upv.es/-> Exchange student-> European Project Semester. [Online]. Available: http://www.ihk.dk/english/eps/index.asp

[4] A. Ferrolho and M. Crisóstomo, “Intelligent control and integration software for flexible manufacturing cells,” *IEEE Trans. Ind. Informat.*, vol. 3, no. 1, pp. 3–11, Feb. 2007.

[5] H. Chih-Lyang, C. Li-Jui, and Y. Yuan-Sheng, “Network-based fuzzy decentralized sliding-mode control for car-like mobile robots,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 574–585, Feb. 2007.

[6] S. Kwee-Bo, B. Kwang-Sub, and H. Fumio, “Internet-based teleoperation of an intelligent robot with optimal two-layer fuzzy controller,” *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1362–1372, Aug. 2006.

[7] R. Marín, P. J. Sanz, P. Nebot, and R. Wirz, “A multimodal interface to control a robot arm via the web: A case study on remote programming,” *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1506–1520, Dec. 2005.

[8] T. Chang, P. Jaroonsiriphan, M. Bernhardt, and P. Ludden, “Web-based command shaping of Cobra 600 robot with a swinging load,” *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 59–69, Feb. 2006.

[9] A. Walter-Colombo, R. Schoop, and R. Neubert, “An agent-based intelligent control platform for industrial holonic manufacturing systems,” *IEEE Trans. Ind. Electron.*, vol. 53, no. 1, pp. 322–337, Feb. 2006.

[10] J. García, F. Rogelio, A. Luque, C. Aracil, J. M. Quero, D. Carrión, F. Gámiz, P. Revilla, J. Pérez-Tinao, M. Moreno, P. Robles, and L. G. Franquelo, “Reconfigurable distributed network control system for industrial plant automation,” *IEEE Trans. Ind. Electron.*, vol. 51, no. 6, pp. 1168–1180, Dec. 2004.

[11] N. Dong To, O. Sang-Rok, and Y. Bum-Jae, “A framework for Internet-based interaction of humans, robots, and responsive environments using agent technology,” *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1521–1529, Dec. 2005.

[12] D. Lim and A. Anbuky, “A distributed industrial battery management network,” *IEEE Trans. Ind. Electron.*, vol. 51, no. 6, pp. 1181–1193, Dec. 2004.

[13] J. Kofman, X. Wu, T. J. Luu, and S. Verma, “Teleoperation of a robot manipulator using a vision-based human-robot interface,” *IEEE Trans. Ind. Electron.*, vol. 52, no. 5, pp. 1206–1219, Oct. 2005.

[14] M. Chirico, A. M. Scapolla, and A. Bagnasco, “A new and open model to share laboratories on the Internet,” *IEEE Trans. Instrum. Meas.*, vol. 54, no. 3, pp. 1111–1117, Jun. 2005.

[15] J. R. Buck and K. E. Wage, “Active and cooperative learning in signal processing courses,” *IEEE Signal Process. Mag.*, vol. 22, no. 2, pp. 76–81, Mar. 2005.

[16] H. Hassan, A. Giner, and R. Lomas, “Robot arm control. A multidisciplinary project in electronic engineering,” in *Proc. IEEE Conf. Mechatronics Robot.*, Aachen, Germany, Sep. 2004, pp. 1703–1708.

[17] AdLink Technology Inc., *PCI-9112, 16 ch.12-bit Advanced Multi-function DAS Card.* Datasheet. [Online]. Available: http://www.adlink.com.tw

[18] A. Perles, J. Albaladejo, J. V. Capella, J. M. Martínez, H. Hassan, and C. Domínguez, “SimSeny: A data acquisition card simulator applied to electronic engineering studies,” *J. Comput. Appl. Eng. Edu.,* to be published.

[19] A. Sloss, D. Symes, and C. Wright, *ARM System Developer’s Guide*. San Mateo, CA: Morgan Kaufmann, 2004. 704 pages.

[20] G. Kim and W. Chung, “Tripodal schematic control architecture for integration of multi-functional indoor service robots,” *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1723–1736, Oct. 2006.

[21] H. Prendinger and M. Ishizuka, “Symmetric multimodality revisited, unveiling users physiological activity,” *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 692–698, Apr. 2007.

[22] B. Jensen, N. Tomatis, L. Mayor, A. Drygajlo, and R. Siegwart, “Robots meet humans—Interaction in public spaces,” *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1530–1546, Dec. 2005.

[23] Keil Arm Product News, *Keil MCBSTR9 Board for Embedded Networking Applications*, Jun. 2006, Plano, TX. [Online]. Available: http://www.keil.com/pr/article/1095.htm

[24] Borland Inc., *Borland C++ Builder 6.0 Users Guide*. [Online]. Available: http://www.borland.com

[25] C. Domínguez, H. Hassan, A. Perles, J. Albaladejo, J. V. Capella, and J. M Martínez, *simPROCes Remote Lab Development Tool.* Valencia, Spain: Higher Tech. School Design Eng. (ETSID), Tech. Univ. Valencia (UPV). Internal License 03-2006.

Houcine Hassan received the M.S. and Ph.D. degrees in computer engineering from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1993 and 2001, respectively.

Since 1993, he has been with the Industrial Informatics Group, UPV, where he is currently participating in several research and educational projects. Since 1994, he has been an Associate Professor with the Departamento de Informárica de Sistemas y Computadores, UPV. His research interests focus on a number of aspects of the development of hardware and software architectures for embedded systems, real-time systems, and robotic systems.

Carlos Domínguez received the M.S. degree in electrical engineering from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1991.

Since 1989, he has been involved in different projects in the Industrial Informatics Group, UPV. Since 1992, he has been an Assistant Professor with the Departamento de Informárica de Sistemas y Computadores, UPV. His research interests include the development of intelligent agents for mobile robots, real-time systems, and planning and scheduling integration.

Juan-Miguel Martínez received the B.Eng. degree in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1986, 1993, and 1999, respectively.

Since 1986, he has been with the Departamento de Informárica de Sistemas y Computadores, UPV, where he is currently an Associate Professor. His research interests include power saving, deadlock handling, and congestion control mechanisms for computer interconnection networks. He has been involved in different projects in innovation in higher education since 1990.

Angel Perles received the M.S. and Ph.D. degrees in computer engineering from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1994 and 2003, respectively.

Since 1997, he has been an Assistant Professor with the Departamento de Informárica de Sistemas y Computadores (DISCA), UPV. He is also a member of the Fault Tolerant Systems Research Group, UPV. His research interests include discrete-event-based modeling and simulation, real-time systems, and industry-applied computers.

José Albaladejo received the M.S. degree in fundamental physics from the University of Valencia, Valencia, Spain, in 1984 and the Ph.D. degree in computer engineering from the Universidad Politécnica de Valencia (UPV), Valencia, in 2003.

Since 1989, he has been an Assistant Professor with the Departamento de Informárica de Sistemas y Computadores (DISCA), UPV. Since 2000, he has been a member of the Fault Tolerant Systems Group. His research interests are fault injection, hardware and software codesign, and reconfigurability.