Preprints of the 16th IFAC World Congress, Prague, Czech Republic, 4-8July 2005, paper 02242, 6p

AN INTERNET-BASED LABORATORY FOR DISTANCE LEARNING IN CONTROL ENGINEERING

Tudor Buzdugan * **Ioan Naşcu** * **Robin De Keyser** **

* *Faculty of Automation and Computer Science, Technical University of Cluj-Napoca, Romania*
** *Electrical energy, Systems & Automation, Ghent University, Belgium*

tudor.buzdugan@control-lab.info ioan.nascu@aut.utcluj.ro
rdk@autoctrl.UGent.be

Abstract: The motivation, structure and performances of an educational Internet based control engineering laboratory are presented. It is intended to be a general tool, able to work with a large category of real processes and to provide support for the most common and specific laboratory experiments in the field of control engineering. It is a ready to use solution and a user friendly system from both points of view, student and professor.

Keywords: distance learning, laboratory experiments, real-time, control engineering, internet

1. INTRODUCTION

Nowadays, in control engineering schools, the real life experiments are often replaced by computer simulations, this being a cheap alternative to providing laboratories with support for a large number of students. Also, distance learning in engineering fields, especially in control engineering is difficult, and the hands-on experience of the student is obtained from simulations only.

Unfortunately, simulations cannot provide an accurate description of the real process and therefore nor the required experience for an engineer. Simulations represent a good development tool, but the obtained results have to be tested on the real process in order to verify their accuracy. In fact, it is highly probable that in a real experiment, the results will differ from the ones obtained via simulation.

The purpose of this tool is to help teachers and students rediscover the laboratory experiments, with less resources and lower costs. Therefore, the main objective of this project is the design and implementation of a control engineering laboratory able to support large numbers of students, distance learning and to provide real-plant experiments. As nowadays Internet connections are widely available, the straightforward option is an Internet based laboratory.

Some other attempts in the field of online laboratories have been analyzed. The general conclusion was that they are either simulations, either designed for a set of plants and therefore not a general solution, either rely on expensive software such as LabVIEW or Matlab/Simulink (B.C.Seet and K.V.Ling, n.d.).

A complete system, able to connect the student's workstation with the real plant has been developed. The student can now perform laboratory

experiments in the laboratory or at home, via internet, without too many constraints. The proposed system is applicable to a large category of real processes and provides support for the most common and specific laboratory experiments in the field of control engineering.

2. PERFORMANCE SPECIFICATIONS

Such an internet based laboratory should provide:

- *generality* - the system should be able to work with a large category of plants, provide the most general experiments that a teacher would like to present to his students and have a user interface that can run on any workstation.
- *transparency* - from user's work station to the real plant the system should be transparent, have no interference, such that the user doesn't need to bother about implementation related issues and focus on the experiment only.
- *feedback* - it is essential that the system transmits to the student as much feedback as possible, especially in multimedia form, in order to make the student feel closer to the real experiment
- *ease of use* - the system has to be user-friendly from both points of view, teacher and student.

The *target plant* to be connected to the system presents the following characteristics:

- *fast* - in order to minimize the time needed to run an experiment and to maximize the number of experiments, fast processes are recommended. The smallest sampling time the system allows is of one millisecond.
- *safe* - as the experiments will be executed remotely, it is very important that the plant presents self-diagnosis mechanisms able to detect dangerous situations before any damage occurs. In these cases (which are very likely to appear in such an educational system) the plant has to be able to reset to a safe-state and report the situation.
- *an example* - one of the plants used during system development was a DC motor with variable load. It is a fast process, with a time constant of about 0.2 seconds, allowing experiments to take less than a minute. A temperature sensor has been attached to the motor and if its temperature exceeds a threshold, power to the motor will be turned off and the system will be signalized that the plant is momentarily unavailable. After the motor cools down, experiments can continue.

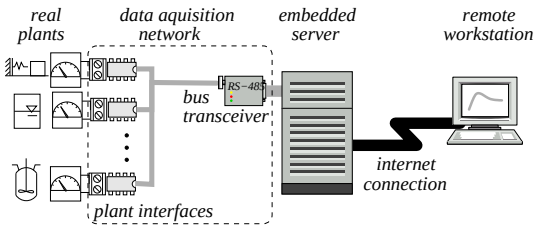

Fig. 1. Structure of the System - General View

3. THE STRUCTURE OF THE SYSTEM

The chosen structure for this system is presented in Figure 1. This architecture gives the possibility to have several real plants simultaneously connected to a single server, that controls and supervises them in parallel.

At the student side, a simple PC connected to internet and a web browser is needed. All experiment data and relevant plots are provided by the system.

At the laboratory side, the system is composed of the following parts:

- *the server* - the part of the system responsible with running the experiments, based on parameters provided remotely by the user. The control algorithms run on the server within a real-time environment;
- *the data acquisition network* - the link between the experiments running on the server and the real processes.

The most attractive approach would be an internet connected plant, able to receive inputs and send outputs over internet in real time. Unfortunately such a connection is never deterministic, i.e. the time needed to receive or send data over interned cannot be specified as it can change at any time (Burns and Wellings, 2001). Therefore such a connection cannot assure constant sampling time and it is not suitable for control engineering experiments.

Instead of directly controlling a plant, a more realistic solution is to control an experiment. Controlling an experiment means setting it's parameters, starting and stopping it, operations that do not require any special timing constraints. This is also a better approach, the student having directly available an already set-up platform so he can focus on the experiment itself and not on implementation details.

4. THE SERVER

The server is the central part of the application and the most complex one, integrating a wide range of technologies. One of the most important attributes of this part is a high transparency

between the student and the experiment. The user shall not be concerned about implementation details. The system as a whole must function in such a way that it does respond exactly to the user's requirements without any interference except for reasonable safety constraints.

From control engineering point of view, the most critical real time activity is sampling signals at a constant rate (Dutton, 1997). Therefore a real-time environment is needed to run data acquisition and control algorithms routines in order to satisfy these constraints.

On the other hand the internet connection and user interface shall not be placed in a real-time environment as these parts do not satisfy real-time requirements as described above.

Therefore the chosen solution for the embedded server has two functionalities:

- to provide the *real-time environment* for running the experiments.
- to provide an *internet user interface* for controlling these experiments.

From the *hardware* point of view there are no special requirements, any common PC can be used. Standard *serial port* and a *network card* are sufficient.

On the other hand the *operating system* has to be very flexible and to allow *access at any level* for the developer in order to be able to integrate all the functionalities in a transparent manner. Therefore an *Open-Source operating system* (e.g. Linux) is needed for this application.

In order to be easy to use from professor point of view, the system runs *"live on CD"*, meaning that the system runs directly from the CD, without the need of installing anything on the hard-disk.

4.1 The Real-Time Environment

The proposed solution is to use real-time techniques for running the experiments and leave everything else out of the real-time constraints, as a background job, that gets executed whenever there is time left after the critical sections did their tasks. Such environments, able to run both real-time constrained tasks and not constrained tasks are offered by operating systems extensions such as RTAI(Real Time Application Interface) (Bruyninckx, 2002).

Modern processor architecture offers the possibility of running programs in two hardware protection levels called *kernel space* and *user space*. The first level allows direct access to all the resources, while the latter has more protection against erroneous accesses at the cost of larger latencies(Bruyninckx, 2002). The real-time extensions add a *third layer*, the *real-time space*. This is in fact nothing else but a part of kernel space, used in a particular way (Proctor, n.d.). The main idea is to provide a virtual interrupt emulation to standard Linux, and to offer a kernel space microkernel with real-time scheduled threads. RTAI intercepts all hardware interrupts, checks whether an interrupt is destined for a real-time service routine (and launches the corresponding ISR if it is), or forwards them to Linux in the form of a virtual interrupt, which is held until no real-time activity must run. In this scheme, Linux is never able to disable hardware interrupts. This kernel takes over the real hardware from Linux, and replaces it with a software simulation (Mantegazza, n.d.).

For our system, real time functionality is needed for controlling the real plant. This means the Real Time Environment is responsible for running control experiments while the HTTP service is responsible for configuring the experiments and return relevant results to the user.

Data Flow through the Server is presented in Figure 2.

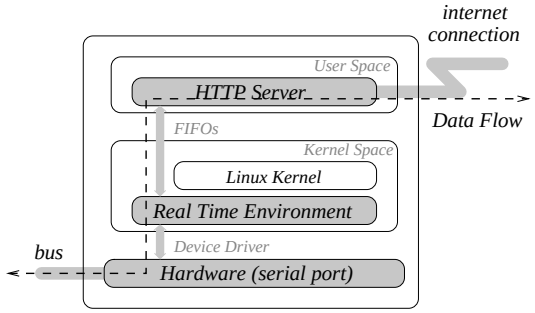

Fig. 2. The Data Flow through the Server

The HTTP service receives experiment parameters and start command from the user. It will pass these parameters through RTAI FIFOs to the Kernel, after checking their validity. Passing parameters through FIFOs means writing these values in a special *device file "/dev/rtf0"*. These values are not written on the hard-disk but stored in *kernel memory* which is *shared with the real time environment*. After the parameters have been uploaded in kernel space, the control routine is signalized to start the experiment.

At this point, the real time environment has all the data it needs and is now ready to start the requested experiment. First thing is to select the required control algorithm and to set the sampling time.

At each sample time control signals are sent to the plant and measured outputs are received from it through the bus. The communication between the real time environment and the data acquisition

network is done by a specialized *device driver* that controls all activity on the bus.

All experiment data obtained at each sample in the real time environment (such as plant inputs and outputs, time, reference etc.) are passed to the user space through another FIFO (*"/dev/rtf2"*). All this data is stored in a file and a relevant plot is generated. At the end of the experiment, the plot along with the file containing all the relevant sampled signals measured or computed during the experiment are made available to the user by the HTTP service.

4.2 User Interface

The experiment's user interface shall be very flexible and general. The most used internet protocol for user interface is nowadays HTTP. This protocol is based on a server that delivers the information in the form of a page, providing also functionalities for user feedback.

On the other hand, the servers can provide dynamic content, as result of user's feedback or certain events. As well, HTTP servers can interact with other applications. For this project, the interactions with other applications (the real-time environment) consists on parameter update, start and stop commands.

As all the needed functionality for the user interface can be successfully accomplished by a HTTP server, also resulting in being the most general solution, this is the one chosen for this application.

An accounting system has been implemented in order to allow the system generate personalized user interface. This means that the system remembers parameters and results of previous experiments for the same user, in order to make student's work easier. Another positive aspect of the accounting system is that the professor can directly see and analyze results of each student and be able to give specific advices to each individual according to the obtained results. One special user is the administrator or the professor responsible for the experiments. He has a special interface from where he can change and apply various settings of the system.

The administrator/professor is able to:

- add/remove users.
- add/remove new acquisition interfaces - each real-plant is connected to the server through a data acquisition interface; these interfaces can be automatically detected by the system but only upon administrator request; after an interface has been successfully detected, the other users can perform experiments with the new added plant.
- suggest experiment parameters - when the student will login for the first time and will choose one experiment, the parameters suggested by the professor will be automatically filled in.
- apply restrictions on the parameters the system will accept - each experiment parameter can be restricted to take values within a certain range imposed by the professor; this represents a minimal safe-operation constraint.

All these operations can be executed from a web-browser, so the professor does not need to be present near the server. The only time when human physical presence is required is when starting the system and when connecting new real processes to the data acquisition network.

The user interface can be seen as an *Expert System* assisting the student in performing the experiments. It gains initial knowledge (suggested parameters and allowed ranges) from the human expert (the professor) and then it adapts, remembering last values of the parameters for each experiment, for each student. It will prevent starting an experiment with unsafe parameters.

5. THE DATA ACQUISITION NETWORK

This part of the system is used to connect several plants to the server. The main functionalities provided are:

- *multi-point communication* - in order to be able to connect several real plants to the server and be able to perform experiments on several, different systems without requesting local intervention.
- *high speed* - to be suitable for real plants with small time constants.
- *medium distance* - in order to connect experiments located in different places, not necessarily in the same room with the server.

5.1 The bus

This data acquisition network is based on a RS-485 bus to which up to 120 interfaces can be connected. The interfaces are powered through the bus, on separate power lines, making the system simpler and more flexible.

The data control and transmission protocol is application dedicated, adapted to the need of high speed. Serial asynchronous protocol has been chosen, as dedicated hardware for this protocol is already available in most computers. A transceiver is needed to convert voltage levels of PC serial port and RS-485 bus. The transmission speed is 115 kbps, allowing bus lengths of 200 meters.

Matlab and LabVIEW drivers have been developed in order to allow the use of this network in separate, independent applications, very useful for network control systems research.

5.2 The plant interface

The plant interface is responsible with analog-numeric conversions and with data transmission. A low power micro-controller is used to implement these functionalities.

A number of interfaces have been developed, allowing 2, 3 and 4 inputs/outputs, with 8 and 10 bits accuracy. The inputs (with respect to the interface) are analog signals in the range of [0..5]V, while the outputs are PWM signals.

A digital input signal is used to sense if the plant is ready to be used and a digital output signal is used to inform the plant to turn off when not used, between experiments.

For some real experiments numerical filtering is also needed and therefore the interface is able to perform this operation as well, on demand.

6. EXPERIMENTS AND RESULTS

The project intends to provide a general set of built-in experiments, able to cover most of the needs of control engineering disciplines. The experiments have been grouped into the following categories:

- *Systems Identification* - providing *Step Response*, *Stair-Case Response*, *Custom Signal Response* experiments;
- *Classic Controllers* - *PID Controller*, *PID Relay Auto-tuning*, *Poles-Zeros Allocation*;
- *Fuzzy Controller* - a Mamdani based *Quasi-PID Fuzzy Controller*.

An artificial, additional dead-time can be added to the real process and it is possible to control it for all the experiments presented above. Noise/load control for the SISO experiments is also possible (if the plant allows this).

For *Systems Identification*, the most general experiment is the *Custom Signal Response*, that allows the user to feed into the real plant any test signal. The student will provide this custom signal as a file containing the values at each sample time. This experiment can also be used for open-loop control experiments.

In Figure 3 the results of a *Custom Signal Response Experiment* are presented. The signals were sampled with eight bit accuracy, so the best and most general way to represent all these signals on the same plot is to represent them as eight bit integers, that means, all the values lie in the range [0..255].

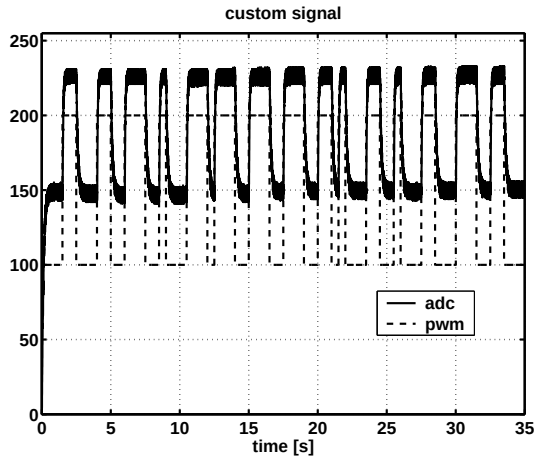

Fig. 3. Custom Signal Response Experiment Results

For *Classic Controllers*, the most important parameters are the sampling time, the setpoint, the load and the parameters of the controller. The reference and load values and dead-time can change once per experiment from their initial values at given times, provided by the user.

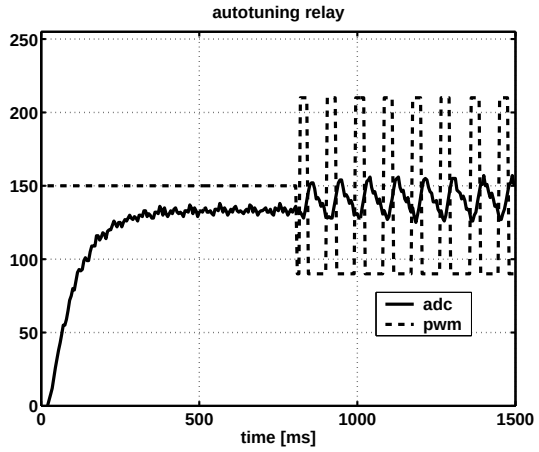

Fig. 4. Auto-tuning Relay Experiment Results

In Figure 4 results of an *Auto-tuning Relay Experiment* are presented.

while the way a *PID Controller* experiment interface looks in the user's browser can be seen in Figure 5.

In order to make the student "feel" and understand how plant signals are transformed into control signals by the controller, a *fuzzy controller* experiment has been added. The fuzzy control makes use of linguistic terms for describing signal values and semantic rules to generate the output, being therefore closer to human language than mathematical formulas.

In this case the user interface is more complicated, the student having to specify the set of rules that

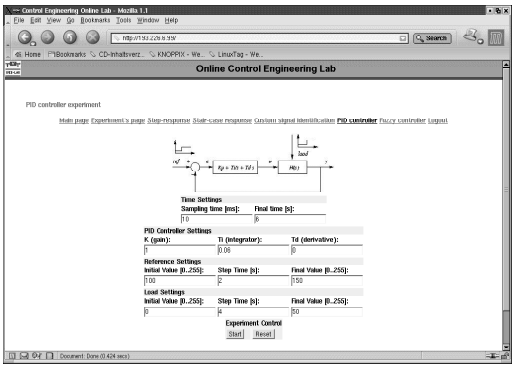

Fig. 5. PID Controller Interface

becomes very big with the increase of the number of linguistic terms. This problem is solved by the expert system behind the user interface, based on the fact that for a quasi-PID fuzzy controller, the set of rules respects a typical pattern and only few of the rules are needed in order to build the whole set.

Normalization, fuzzification, defuzzification and inference are performed by the system, the student providing adequate parameters for each routine. Results of such an experiment are presented in Figure 6.

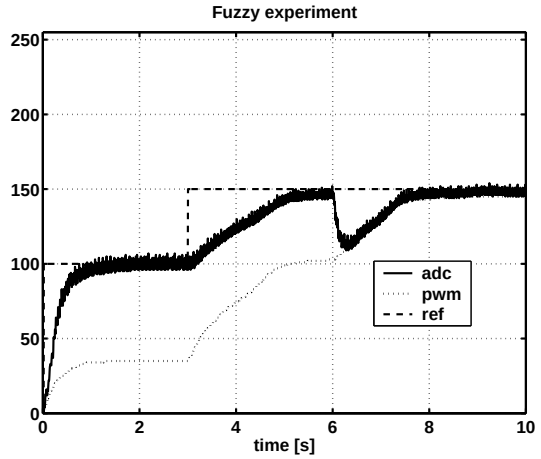

Fig. 6. Fuzzy Experiment Results

7. CONCLUSIONS

An internet based control engineering laboratory has been developed. The implemented system is generally applicable to a large category of real processes. Any plant that accepts PWM control signals as input and has analog output within the range of 0-5V can be used. Also noise/load control is possible, if accepted by the plant.

The plant control is very reliable, providing safety mechanisms (turning off the plant between experiments) and restricting experiment parameters as indicated by the experiment's administrator. The system is password protected, so that only authorized users can perform online experiments.

Experiment results are made available for the user both as database and as a plot of relevant signals. Experiment parameters provided by the user are remembered by the system such that performing experiments becomes easier.

The user interface works as an expert system assisting the student while performing experiments. Based on knowledge provided by the teacher it suggests experiment parameters and applies restrictions on dangerous experiment parameters. The system also helps the student in providing experiment parameters when their number becomes very big, such as for the *Fuzzy Controller*. The user interface is user friendly and self explaining.

The system works *"live on CD"* simplifying the work of the professor, as there is no need to install anything on the server's hard-disk. At the student's side only a standard browser is required.

REFERENCES

B.C.Seet and K.V.Ling (n.d.). An internet-based laboratory for control engineering education.

Bruyninckx, Herman (2002). *Real-Time and Embedded Guide*. K.U.Leuven Mechanical Engineering.

Burns, Alan and Andy Wellings (2001). *Real-time Systems and Programming Languages*. Addison-Wesley.

Dutton, Ken (1997). *The Art of Control Engineering*. Addison-Wesley.

Goldin, J. (1996). Ten ways to bulletproof rs-485 interfaces. *National Semiconductor, Application Note AN-1057*.

Mantegazza, Paolo (n.d.). A hard real time support for linux. http://www.rtai.org/.

Proctor, Frederick (n.d.). Linux and real-time linux. http://www.ddj.com/documents/.