A Web Based Remote Control Laboratory

Claudiu Chiculita, Laurentiu Frangu
Department of Automatic Control, University of Galati, ROMANIA
Claudiu.Chiculita@ugal.ro, Laurentiu.Frangu@ugal.ro

ABSTRACT

The paper deals with the structure and implementation of a remote laboratory, in the technical area, for university purposes. The requirements of such an application are presented, considering the technical, didactical, security and financial issues. The limitations of the available commercial software products are examined. The main part describes an original laboratory system (it doesn't use Matlab, Labview or other similar products), meeting most of the above mentioned requirements. It was created by the authors and tested in the university, but it isn't a commercial product. Its main features are: the execution of the code on the server (laboratory's computer) only, the didactical issues (such as private directory for every student and access of the teacher to students' solutions), low-cost solution for the student and experimentation on physical equipment. The last feature is specific for the technical education and allows the student to write its own control algorithm, in order to find solutions to the laboratory's subject, but without being limited to predefined algorithms. However, the system limits the access of the student to the allocated resources only (this is one of the security feature). Even if the described remote laboratory was thought for automatic control subjects, it can be easily adapted for other technical areas.

I. INTRODUCTION

The usefulness of a remote laboratory in teaching environments is already known: it saves equipment, personnel for the institution and it saves time and money for the remote students. It also allows the same equipment to be used in research purposes by more teams, through Internet, and is a good case study for the informatics and control aspects of the telepresence and teleoperation applications ([Gillet, 1997]). This laboratory can be used for remote learning, equally in academic or industrial context. There are three different aspects of the remote laboratory implementation: the technical equipment used in the experiment (specific to the taught subject), the communication software and the teaching software. This paper deals with the possibility of using the equipment in the laboratory by the remote students, mainly from the point of view of the informatic system. It concerns the communication software and the solutions to some important didactical requirements.

The final purpose of the remote laboratory is to allow the student to take full control of the equipment, in order to fulfill the task required by the teacher. Simultaneously, the laboratory must be protected against malicious software and incorrect operation. To prepare remote laboratories, two steps have to be completed: to formulate the requirements of such activity and to make a survey about the available software. This paper performs both steps mentioned above and proposes an original software, meeting the most part of the requirements. The presented software was validated through an application in the field of automatic control education, but it is also applicable and easy to set up in different other fields.

The paper continues, in chapter II, with the requirements formulated for the remote laboratory, as they resulted from the experience of our laboratory of Automatic Control and Electronics and from the study of the software products. These requirements include the didactical aspects of the laboratory activity, equally important for our university and for virtual universities. Chapter III contains the survey of the existing commercial software products and their advantages and drawbacks. Chapter IV contains the description of our experimental implementation, including the hardware, software and communication solutions. Finally, the conclusions are presented in chapter V.

II. REQUIREMENTS FOR A REMOTE LABORATORY

There are today many universities that organize remote education, including for the technical field. The courses and the simulation laboratories are distributed on electronic supports or by e-mail. However, the laboratory experiments that use physical equipment are not replaceable, because the student must have a direct contact with the real technical environment. Part of the work has to be organized in the institution, while the other part may be performed from the remote site. The usefulness of the remote laboratory proposed in this paper is justified by the opportunity of organizing the remote experiments, saving time and money for the students.

In the sequel, the structure of such a laboratory will be described. The following terms are used throughout the paper: the server is the computer situated in the laboratory, interfaced with the physical equipment; the client is the student's computer, remotely situated. A very general laboratory structure must include:

- the equipment (the regulated object, in the case of automatic control laboratory) associated with a server computer and a data acquisition card;
- an interface program, running on the server computer, that drives the signals to and from the data acquisition

card and manages the information about the students' work;
- a server software that deals with the communication between the laboratory and the remote students;
- the informatic environment (usually Intranet or Internet);
- a client software running on the remote (student's) computer, that provides the Graphic User Interface and communicates with the server.

In order to find the best solution for a laboratory, the first step is to formulate the requirements. Part of them resulted from our experience, including the didactical aspects of the activity, while the other part resulted during the study of the available commercial products.
The following criteria were considered:

- technical possibility for the student to remote control the equipment;
- security of the computers;
- didactical criteria;
- cost of the solution, mainly the cost for the student.

Technical requirements:
- the local physical equipment should not need operation nor maintenance, during the laboratory experiment;
- the equipment has to be connected to a data acquisition card, that allows both receiving feedback signals from the equipment and sending to it the commands issued by the computer. The equipment and the card are supposed to be compulsory parts of the laboratory, because the simulation activity does not constitute the object of our survey;
- the server software has to provide the interface to the acquisition card and the real-time capability. It also has to be versatile, with respect to the controlled equipment and to allow the student's algorithm to have full control over the equipment (for purposes of identification, control, etc.);
- the time required for the communication has to be maintained low, to avoid overloading the network.

Security requirements:
- the server software has to protect the equipment against wrong commands, exceeded limits or accidentally multiple users;
- the server also has to be safe about network attacks (should allow work behind a firewall);
- the remote student should not send to the web server compiled files, because they may contain or be infected with malicious hidden software.

Didactical requirements:
- every student must be provided with a separate directory, containing data files and his own solutions of the problems. The directories must be private, to avoid interference;
- the results of the solutions provided by the student have to be recorded, both for the teacher and for own use;
- the teacher must have access to the solutions, in order to mark students' activity;
- the access of the students must rely on ID and password basis.

Financial requirements:
- the client software should not require additional license. A common browser could be used by the student, as a good solution from the cost point of view;
- the system must use common acquisition cards and as much common software as possible.

Obviously, some of the above requirements may not be satisfied simultaneously. In general, some compromises have to be accepted, excepting the security. For example, using an own TCP/IP based protocol gives a faster communication than HTTP, but this is not usually allowed by the security software (the firewall). That is why the Internet solution could slightly differ from the Intranet solution.

III. SOFTWARE FOR REMOTE LABORATORIES: STATE OF THE ART

There are not many commercial software products, allowing remote operation of the equipment, for teaching purposes. All of them are derived from the simulation software. This is explainable, because the first step when teaching in the technical area is to model and to simulate. In the domain of automatic control, the option to link the remote operation software to the simulation software is favorable, because of the current use of simulation tools in the automatic control laboratories. The most spread simulation software products are Matlab + Simulink, from MathWorks, and LabView, from National Instruments.

Matlab is an environment that allows simulation of systems, on the basis of a powerful collection of mathematical and plotting routines. Additionally, Simulink provides a graphic interface that simplifies the building of a simulation scheme. Although this is the most popular simulation software for teaching automatic control and for research, it is not suitable for the purpose of our work. The recent variant that allows operation through network (Matlab Web Server) does not run the remote written software, in order to control some local equipment. When simulating a control loop, the controller has to be installed on the server and it accepts only the parameters sent from the client. Regarding the license, only the laboratory has to possess one.

LabView is an environment that allows the simulation on the basis of a powerful graphic interface. It also allows the access to local equipment, through data acquisition cards (National Instruments). Although the configuration from the remote site is possible, the control actions are limited to sending the set of parameters. Similar to Matlab, LabView does not allow the student to write his own controller and to run it on the server. The license is necessary for the laboratory only.

A more powerful solution is provided by the products of Quanser. First of all, it uses the popular environment Matlab (with Simulink), for which the universities already

have licenses. There are some restrictions with respect to the general performances of Simulink, but they are not important. The server software includes the real-time capabilities and the drivers for the data acquisition card. The client compiles the Simulink simulation scheme and sends it to the server, where it will be run. The client receives, in return, the signals from the controlled equipment (e.g. a control loop on the local process). During the run, the only action allowed to the client is to stop the experiment. The controller is limited to the structures that may be simulated by Simulink, but these are usually enough for reasonable controllers. On the contrary, the solution of sending an executable file to the server is a serious drawback, because of the security problems. Additionally, this solution requires the Quanser acquisition card, the license for Matlab and the licenses for the Quanser server and client (at the laboratory and at the remote site). This is an expensive solution, indeed.

Studying the previous solutions, some conclusions may be drawn. The available commercial products are not intended to solve the problems of the remote technical education, so they do not present facilities for the didactical aspects of the remote laboratory experiments (such as private student directory, access of the teacher, etc.). Regarding the technical aspects, one can notice that, in the field of automatic control, where the student must implement his own control algorithm, only the Quanser product is suitable. This statement is justified for all technical teaching fields. However, the Quanser product uses compiled files that could complicate the security problems. This solution does not allow the change of parameters, during the experiment. Finally, about the financial aspect, the conclusion is that the most suitable product (Quanser) is also the most expensive.

IV. THE PROPOSED REMOTE LABORATORY SYSTEM

IV.1. The structure of the system

Our goal is to offer to the student the possibility to carry on laboratory experiments on the existing physical equipment. In this purpose we propose a system structured as remote laboratory that meets the most part of the above mentioned requirements. The application is in the field of automatic control (identification, control, optimization). The hardware structure is presented in 'Figure 1' and it contains: the regulated object, the data acquisition card, the server and the client. The chosen regulated object is a level control system, single input - single output, presenting nonlinear behavior. The data acquisition card is simple (PCL711, Advantech), because it needs only an analog input and an analog output. A single computer runs the interface to the acquisition card and the web server, under Windows environment. Because the process is slow, the real-time capability is provided by the usual functions, avoiding the necessity of sophisticated real-time kernels.

As an important advantage of this solution, the student isn't confined to use predefined controllers. He is allowed to write his own full-range controller (fuzzy, neural, etc.), testing any algorithm he can imagine. Having this goal in mind, we chose as support for writing the algorithm a powerful and easy-to-use programming language that will be introduced in the next paragraph.

Even if the mentioned application field is the automatic control, other technical areas can also benefit from the same structure. For instance, robotics, mechanics, ship model testing, data acquisition and management laboratories can use it.

IV.2. Implementation of the software

The software structure is the result of a series of experiments. It consists of two different parts. The client computer runs a web browser only, which opens a web page loaded from the server. The server computer runs the software written by the authors, which cumulates the functions:

- interface to the physical equipment (through the data acquisition card);
- interpreter for the student-written program;
- web server, dealing with all communications between server and client;
- manager of didactical information.

Figure 2 illustrates the parts of the software running on the server. The main features of the software and the reasons for choosing this structure are described in the sequel.

Server-side executed code.
There are two aspects concerning the execution: where the controller runs (or other software part that drives the physical equipment) and where the source code is compiled (or interpreted). Running the software on the client implies that the control loop includes the server, the client and the network. This approach introduces large and unpredictable dead-time in the loop, making it suitable for slow processes only [Buss 1999]. In order to obtain a convenient solution also for fast processes, the presented laboratory system executes the code on the server. The second issue concerns

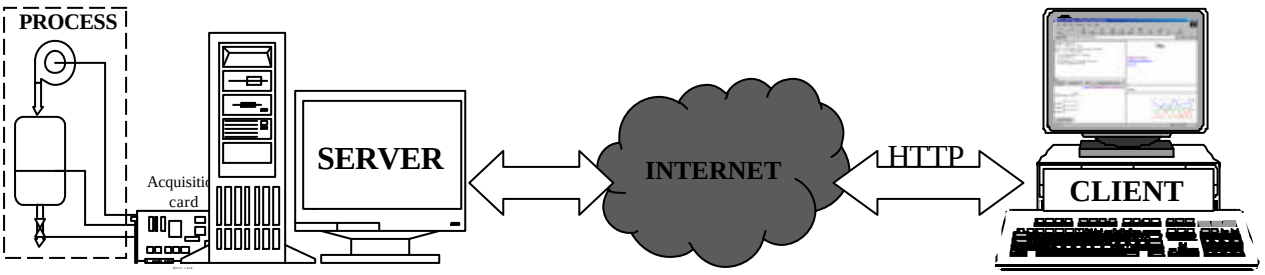

Figure 1: Structure of the remote laboratory

the type of data sent from the client to the server. As previously mentioned, sending binary code from the client to be executed on the server (this is the solution adopted by Quanser's product) creates security problems and requires a compiler to be present on the client. The adopted solution is to edit the source code of the controller at the client and to send it to be interpreted and executed at the server. To avoid security problems caused by the source code, this one is filtered by a restrictive environment prior to reach the interpreter.

The communication solution.
The communication between server and client may be done in several ways. The direct TCP connection requires dedicated programs, running on the two computers. The client is not portable and creates security problems. A better solution, the connection by Java applet, requires all software written in Java and a browser supporting this language. Finally, a dedicated web server, communicating with the client general web browser, is more flexible. Even if it requires more effort to write it, we chose this last solution, because of the interface with the process that had to be integrated in the server. Additionally, leaving only a web browser to run on the client provides a portable and low cost solution for the student.

Web browser client interface.
The student's graphic interface is presented in Figure 3. It is loaded from the server, when starting the editing session, and it runs inside the browser. The top left frame contains the edited program, which will be sent to and interpreted by the server. The bottom left frame contains the values of the process parameters which can be changed during the experiment. The right side frames contain data received from the process and will be detailed in IV.3.

Powerful language: Python.
We chose to use Python as the student programming language. Python is an interpreted, high level, object oriented language. The main reason for using it is its feature to allow building a restrictive environment that filters the interpretation of the Python client source code. This feature solves some of the security problems: the client has restricted access to the internal resources of the server computer. In the present the restrictions introduced by the authors (using the above mentioned feature) concern the access to the controlled object and to the file system.

Python is also very simple to learn; this feature has an obvious didactical motivation, because it allows shortening the student's requested time for accommodation to the language.
Other advantage of Python is that it produces short source code (up to three times shorter than other languages). So the code written by the student is more visible in the limited size window, provided by the web browser. A last reason for choosing this language is the fact that it is free.

Solutions for the didactical aspects.
The presented software allows the student to have access to a private directory on the server, which contains own programs and data. Only the professor has access to all directories in order to evaluate and to mark student's activity. Both accesses are on a password basis.

IV.3. Description of the experimental session

For illustrating the use of the proposed remote laboratory system, a short description of the experimental session is presented in the sequel.
At the beginning of the session, the student logs in if there is no other session in progress. For this purpose he launches a web browser and loads the web page of the laboratory. The server asks him the name and the password. Once logged in he can read in the web page corresponding to his subject, the tasks required by the teacher and some examples of programs written in Python. To start a solution, the student must edit the source of his program and to send it to the server. During the execution, he can read the data from the process and change the parameters. As presented in Figure 3, the right bottom frame contains the diagrams of such data, but it can also contain image from a video camera, showing the process. In the same figure, the upper right frame displays the output data of the program. At the end all data can be saved, both on server or client. Furthermore, he can study how appropriate is his controller, that drives the physical equipment, and can improve his solution of the problem. Finally the teacher reads the results and marks the activity of the students. Because the access of the remote students is not synchronous, this solution saves equipment and space in the laboratory. Usually the equipment does not need current operation and maintenance, so there is no need for supplementary personnel (the laboratory may be accessed even during the night).

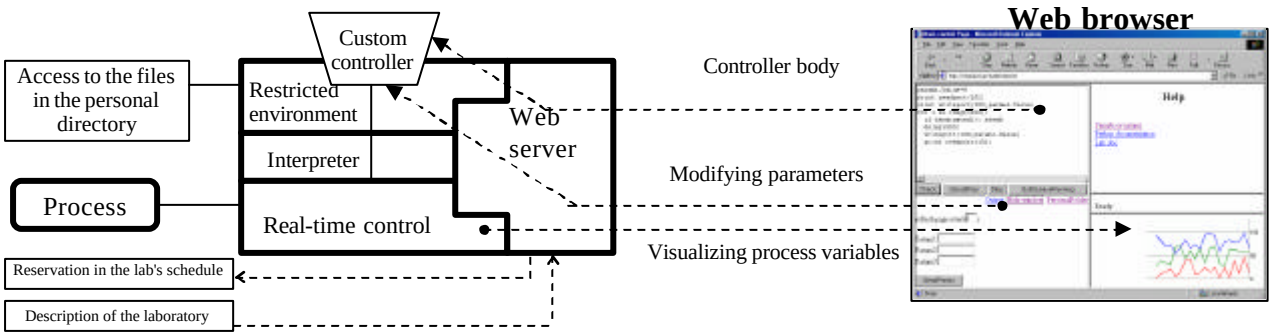

Figure 2: Structure of the server software, including the communication with the client

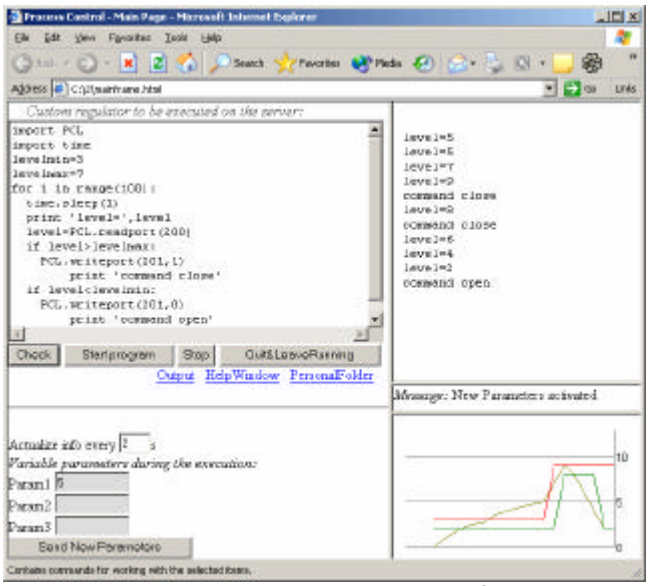

Figure 3: The client graphic interface

V. EXPERIMENTAL RESULTS AND CONCLUSIONS

The solution for the described system was adopted after testing and comparing more possible variants (such as running the code at the client). Then it was tested inside the university (Intranet), in order to verify the didactical, technical and security functionality. During the tests the students learned quickly the language and performed the tasks asked by the teacher. So it is confirmed that the structure is functional and the goal of the work was reached. The main features of the proposed system, which is not a commercial product, are:

- it satisfies some didactical requirements, such as easy to learn language and private directory (as far as we know, it is an original feature);
- it is a low-cost solution for the student (this is an important issue for the success of the remote learning, in virtual universities);
- it introduces laboratory experiments on physical equipments (this is not a common feature and it is important for technical education);
- it simplifies the programming and security problems, because the client runs a web browser only;
- it is a safe solution because the student has access to the allocated resources only.

We consider that the didactical and security aspects of the application may be improved, but the general require ments, formulated in chapter II proved to be appropriate. Further work may concern the graphic interface, the didactical aspects and the application to other equipments.

REFERENCES

D. Gillet, Ch. Salzmann, R. Longchamp, D. Bonvin, “Telepresence: An Opportunity to Develop Practical Experimentation in Automatic Control Education”, *European Control Conference*, Brussels, Belgium, 1997

D. Gillet, Ch. Salzmann, and E. Gorrochategui, “Remote Manipulation with LabVIEW for Educational Purposes”, Computer Based Measurement and Automation in Education, 1998 Conference Proceedings, National Instruments, Austin, TX.

M. Buss, G. Schmidth, “Control problems in multi-modal telepresence systems” in Advances in control, Highlights of ECC’99 (Paul M.Frank, Ed.), Springer 1999.

Guido van Rossum, Python Documentation, www.python.org