Proceedings of the 5th IFAC International Symposium on Intelligent Components and Instruments for Control Applications (SICICA 2003), Aveiro, Portugal, July 9-11, pp. 39-44, 2003.

A VIRTUAL LABORATORY ENVIRONMENT FOR REAL-TIME EXPERIMENTS

Giuseppe Carnevali and Giorgio Buttazzo

University of Pavia, Italy
Email: carnev75@libero.it, buttazzo@unipv.it

Abstract: A virtual laboratory is a computing system that allows to share the physical resources available in a laboratory with remote users connected on the internet. This paper describes a particular virtual laboratory environment for running real-time robot control experiments over the internet through a standard web browser. Using the proposed approach, a remote user can connect to the virtual lab, run a control experiment on a physical robot device, change the control parameters, and interact with the real-time system to receive the desired state information. The robotic application considered in this paper is a ball balancer, a two-degrees-of-freedom rotating plate that has to be controlled to keep a ball in a desired location. Ball position and speed are detected by vision, through a fixed CCD camera.

Keywords: Real-time, Virtual Laboratory, E-learning, Ball Balancing.

1. INTRODUCTION

Recent advances of internet technology greatly extended the ways in which people can access the web. Since a few years ago, the internet could only be used to search for information, or make it available to other users. Today, the internet can also be exploited to conduct actual experiments on physical devices available in remote locations. Such a technology opens new interesting possibilities for education, training, and collaborative investigations. Students are enabled to access resources they do not have, and can run experiments much longer than typically allowed in university labs. They can visit several virtual laboratories available in the world, thus comparing the different approach and solutions for the same problem. Scientists and technicians can cooperate by running joint experiments from their own office on large and expensive equipment, reducing travel costs and allowing observation of results to other people in real-time.

Distance learning has also several economical benefits. In fact, the classical educational model requires the usage of a large amount of resources, in terms of people, space, travel, and time for teaching. In certain disciplines, especially those related to engineering, physics and chemistry, a practical laboratory experience is required to complete the course of study and corroborate the theoretical parts with direct experience.

In the absence of real laboratories, or when the available resources are not sufficient for satisfying the actual demand from the students, experiments are performed via simulation, through specific software tools (Bassem, 2000; Ph, 1998; Alhalabi and Marcovitz, 2000; Hamza *et al.*, 2000). Using such simulation environments, the knowledge acquired by the students depends on several factors, such as the model used in the simulation, the

[1] This work has been partially supported by grants from the European Union, under contract IST-2001-34820, and from the Italian National Research Council (CNR), under contract CNRC002FE3.

types of imposed constraints, and the functionality of the tool. On the contrary, a real laboratory experience offers the student the possibility to know the actual behaviour of a system, including non linearity and noisy data. A virtual laboratory environment does not preclude a student to face with these problems. Although there is not direct interaction with the remote system, the student can send input data to the real system and can receive back all output data produced during the experiment.
Finally, using this approach, shared devices are more protected against inappropriate usage. In fact, a suitable user interface can disable dangerous operations, or bring the system in a safe state when it detects critical conditions that might cause any damage.
Different kind of experiments can be implemented using a virtual laboratory approach. For example, a user could acquire data from expensive devices, such as infrared cameras, stereo vision systems, 3D laser scanners, or mobile cameras, in order to test his algorithms on real data. Control experiments can also be carried out on robotic devices, by changing input set points and control parameters.
There are already several virtual laboratories on the web, mainly developed in university research labs, that propose specific control experiments on robot devices. For example, the control group at the University of Siena developed a virtual environment, The Automatic Control Telelab, with a set of three experiments, that incudes a position controller for a dc motor, a water level controller, and a levitation control system (Casini *et al.*, 2001).
The Automatic Control Telelab allows a user to execute remote control experiments using three modes: Predefined Controller, User-defined Controller, and Student Competition Experiment. In the first mode, the student can choose one among different predefined controllers, like a simple PID controller, and change its parameters. In the second mode, the student can submit a simulink personal controller, whereas in the third mode a user-defined controller is submitted and evaluated against certain performance specifications. Controllers are then compared and ranked.
At the University of Illinois at Urbana Champagne, an inverted pendulum can be remotely controlled by sending its own control algorithm, and its performance can be evaluated in terms of angular errors on the balancing pole (Schwarz *et al.*, 2000). The environment allows the user to send its own controller and a fault tolerant technique protects the system from errors in experimentation and malicious attacks, by switching to a safe backup controller when critical conditions are detected.
Another example of virtual laboratory has been developed at the Carnegie Mellon University Computers, where the user is connected to engineering test equipment, like oscilloscopes and function generators (Stancil, 2000). When students enter the system, they are able to control both the computer and the equipment to test different electronic circuits.
At the Hagen University, a robot vehicle can be remotely accessed to perform different tasks, ranging from kinematics and dynamics analysis to controller design and system identification (**?**).
In this paper, we present a virtual laboratory environment that allows a remote user to interact with a real-time operating system to perform a control experiment with a ball balancing device. The peculiarity of the proposed lab consists in the use of a hard real-time kernel, called Shark (P. Gai, 2001), that can be tested on a real application to verify the sensitivity of timing constraints (such periods and deadlines) on the control performance. The system has been developed as a support for the real-time Systems course at the University of Pavia.
The rest of the paper is organised as follows. Section 2 provides a general description of the user interface. Section 3 illustrates the software architecture and the communication with the remote user. Section 4 is dedicated to the hardware description. Section 5 and 6 presents the detail of the specific robotic experiment. Section 7 states our conclusions and future work.

2. GRAPHICS USER INTERFACE

Before describing the details of the system architecture, we will briefly present the main operations that can be performed on the environment. When accessing the URL of the virtual lab, the user has to select the experiment to be performed. Then, a proper interface allows the user to set a number of configuration parameters before and during the execution of the experiment. In particular, there is the possibility of selecting the particular controller to be used and setting the control parameters.
The user can receive two types of information from the remote device. The variables describing the state of the device are sent to the user and displayed in suitable graphical form. In addition, the user has the possibility to visualize the image of the real laboratory, coming from a web-cam located near the physical device. This feature allows the user to see what is happening in the lab, thus catching all the aspects of the experiments that cannot be understood from the analytical representation, but can only be provided by a direct observation. The user interface displayed on the client browser is illustrated in Figure 1.

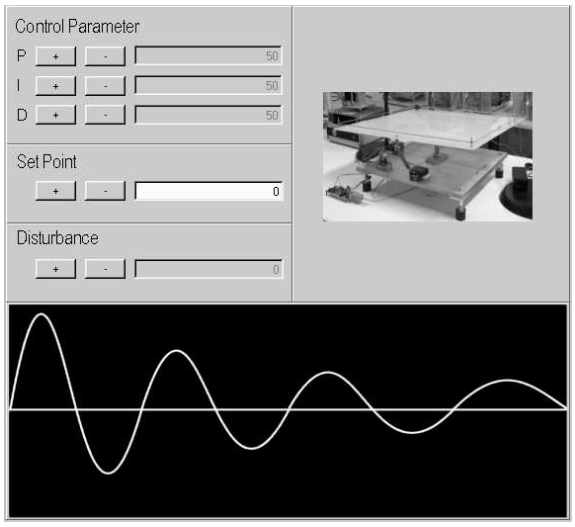

Fig. 1. User interface showing a sample image produced by the remote webcam and the graphical output of the error.

The interface also handles the case in which the system is busy, because is being used by another user, and the case in which the experiment exceeds a maximum duration. This is done for allowing several students to run their own experiments to avoid occupying the lab for a long time. Before the timeout, the user receives an informative message from the system. Then, at the occurrence of the timeout, the experiment is terminated and the user is disconnected to make the lab available to another user.

3. SOFTWARE ARCHITECTURE

The system is implemented using a three-layer architecture, as shown in Figure 2. The reason

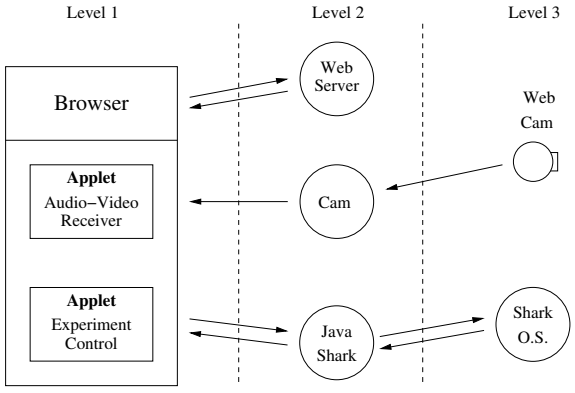

Fig. 2. Software architecture of the virtual laboratory environment.

for having three layers is manly due to the use of the real-time kernel, which cannot be executed on the same machine that acts as a server and handles the internet connection. A physical separation between the network server and the real-time system also allows to achieve a highly predictable

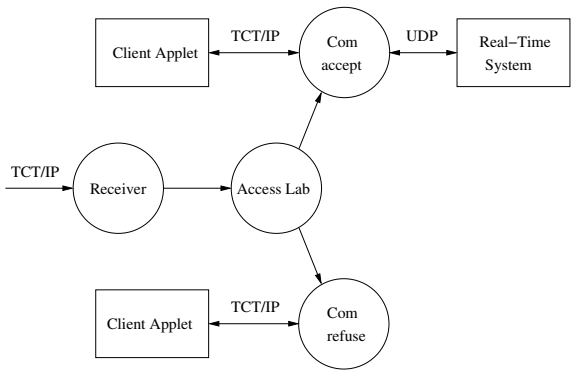

Fig. 3. Main components of the Java/Shark server.

behaviour on the tasks dedicated to the control applications visualize .

The first layer consist of an applet. As also done in (Sadoski and Commella-Dorda, 2000), its purpose is to handle the user graphical interface.

The applet is responsible for changing the parameters of the experiment, visualizing feedback data from the server and displaying the images coming from the web cam.

The main goal of the applet is to achieve high usability, meaning that users should be able to utilize it with any browsers, without downloading any additional software. This goal has been reached by using the Java 1.1 release. In fact, although this release does not have the latest Java features, it has the highest browser compatibility. Moreover, a Java 1.1 compatible virtual machine is embedded within the most common browsers, therefore users do not need to download any plug-in to run the applet. The second layer consists of three different servers: a web server, that answers the requests coming from the client; a servlet (based on the module for handling servlets in the web server), that sends the images from the webcam to the client; and a Java server, called "Java/Shark", that manages the communication between the client and the real-time kernel. This is the most important part of the proposed architecture.

The main components of the Java/Shark server are illustrated in Figure 3.

The *Receiver* is a daemon and its purpose is to wait for a connection from a client. When this event occurs, it passes the connection's reference to *Access Lab* and returns to waiting for a new connection request. *Access Lab* is the laboratory's handler that checks whether the Laboratory is available for an experiment. If the laboratory is busy, the client is assigned a unique identifier and it is enqueued in a waiting list, otherwise it is connected to the server. Any experiment has a maximum running time, after which the client is disconnected for timeout. After a timeout, the connection is stopped, a message is sent to the client, and a new connection is established

with the first client in the waiting queue, if any. When the lab is free, the handler activates the *Com accept* object, which starts the experiment's countdown and activates two threads. The first thread receives data from the real-time subsystem and sends them to the applet, whereas the second thread does the opposite. The *Com refuse* object is called when a connection occurs and there are other references in the queue (busy laboratory). It runs a thread that sends a message to the applet and a *busy laboratory* warning message is displayed on the user browser when the message is received. If the user leaves the virtual laboratory before starting his experiment, his reference is properly removed from the waiting queue. The third layer consists of the real-time controller, whose purpose is to run the actual experiment on the physical device. Its main function is to set the initialization parameters sent by the user, run the control application, and send periodically to the user the output data produced by the system.

4. HARDWARE ARCHITECTURE

In this section we illustrate the main characteristics of the hardware architecture, focusing on the structure of the connections and on the relations that occur among its physical components. The hardware architecture follows the same scheme presented in the previous section and consists of three layers. The first layer consists of the computers used by the clients, connected to the internet. They can be of any type, can have any operating system and can use any kind of browser.

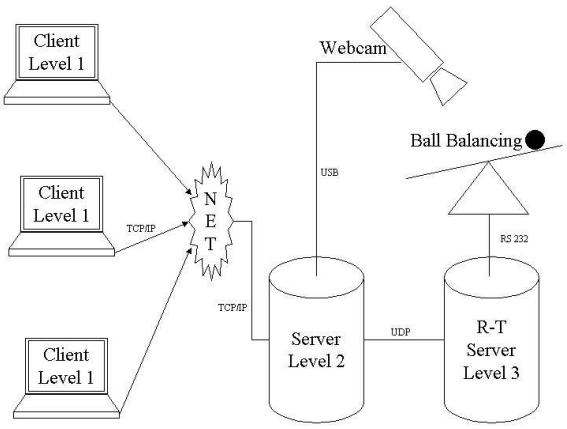

Fig. 4. Hardware architecture.

The system server includes two computers that constitute the second and third layer. The second layer manages the communication with the client, hiding the third layer (dedicated to the real-time application) from the user. The machine at the second layer is the only one visible on the internet, and this allows to isolate all security issues on this layer. This computer also handles the images produced by the web cam.

The computer at the third layer is not visible to the external world. It communicates with the server using an UDP connection on a dedicated local network. This computer runs the Shark real-time kernel (P. Gai, 2001) for executing the control application. A number of peripheral devices are available for connecting this computer to the controlled system, such as serial ports, parallel ports, data acquisition boards, and frame grabbers.

5. EXPERIMENT DESCRIPTION

The virtual laboratory presented in this paper allows the implementation of two basic components: the *laboratories* and the *experiments*. A laboratory is a container for experiments, while an experiment is a specific procedure for using a physical device available in the laboratory.
In order to test the system, we implemented a laboratory that contains an experiment. The specific experiment we decided to develop is the control of a ball balancing device. This system consists of a two-degrees-of freedom rotating plate that has to be controlled to keep a ball in a desired position. A picture of the system is illustrated in Figure 5. The graphical interface on the client allows the user to modify the parameters of a PID controller, and the periods of the real-time tasks. In addition,

Fig. 5. The ball balancing device.

it is possible to visualize the system though a web cam and follow the temporal evolution of the error in a graphic diagram (see Figure 1). Finally, the user can also introduce a disturbance on the plate to verify the stability of the system and the measure the response times required to bring the error below a certain desired value.

6. IMPLEMENTATION DETAILS

The scheme used to develop the ball balancing application is illustrated in Figure 6. It consists of

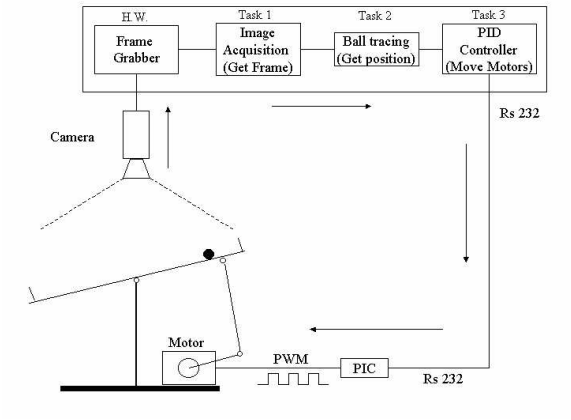

Fig. 6. Block diagram of the application.

a squared plate with raised edges that prevent the ball to fall out of its surface. The plate is free to rotate around two degrees of freedom by means of a spherical joint attached at its centre (from below). The joint is then fixed on a rigid shaft attached on an aluminium base that supports the whole structure.
The motion of the plate is transmitted by two servomotors connected by two staffs to its external border. The servomotors are driven by PWM signals generated by a Programmable Interrupt Controller (PIC 16F876), which receives position commands from a PC through an RS232 serial port. The position of the ball is detected by a CCD camera, whose images are acquired by the PC using a frame grabber.
To achieve a correct behaviour of the system, all sensing, control and actuation activities need to be executed within precise timing constrains. The control application is developed in C under the Shark real-time operating system, which is able to handle periodic and aperiodic activities with explicit timing constraints, with a highly predictable behaviour. The most important feature of Shark is that it can be easily configured to run different scheduling algorithms and resource management protocols, that can be selected among several existing modules available in the kernel. Its modular design also allows replacing certain classes of algorithms without modifying the application.
The application consists of five main periodic tasks that are scheduled with the Earliest Deadline First (EDF) scheduling algorithm (Liu *et al.*, 1973). To avoid priority inversion phenomena, Shared resources are accessed through the Stack Resource Policy (SRP) (Baker, 1991).

The tasks are described below.

- A periodic task (`get_frame`) is dedicated to the image acquisition from the camera. It runs with a period of 40 ms and puts the image in a shared memory buffer.
- A periodic task (`get_position`) analyses the image produced by the camera and computes the position of the ball on the plate. To speed up the computation, the ball is identified by a simple thresholding operation and its position is derived by computing the first order momentum. Velocity is also estimated from the difference with the previous values.
- A periodic task (`move_motors`) implements a PID controller and generates the reference positions for the two motors. It is a hard periodic task running with a period of 40 ms.
- A periodic task (`draw_frame`) displays the image produced by the camera on the screen. It is a soft periodic task running with a period of 100 ms.
- A periodic task (`draw_data`) visualizes some additional information on the screen, such as the state variables and other system parameters. It is a soft periodic task running with a period of 100 ms.

7. CONCLUSIONS

In this paper we described our experience in the development of a virtual laboratory environment aimed at the interaction with a real-time system for running control experiments. The real-time experiment we have implemented consists in controlling a two-degrees-of-freedom plate balancing device for keeping a ball in a desired position.
The results achieved in this work allow remote users to connect through the internet to run real-time experiments available in the virtual laboratory and interact with the system using a simple and intuitive graphical interface.
The peculiarity of the proposed environment is to remotely interact with a real-time kernel to verify how the timing constraints defined on the application task affect the performance of the control system.
As a future work, we plan to include the possibility for the client to send not only the parameters of the controller, but the entire control algorithm. In this case, the system must be designed to be tolerant to malicious attacks or software errors

in the controller, by switching to a safe backup algorithm when critical conditions are detected.

REFERENCES

Alhalabi, Bassem and David Marcovitz (2000). Remote labs: An innovative leap in the world of distance education.

A. Jochheim, C. Rhrig (1999). The virtual lab for controlling real experiments via internet. *IEEE International Symposium on Computer-Aided Control System Design,CACSD'99, Hawaii.*

Baker, T.P. (1991). Stack-based scheduling of real-time processes. *The Journal of Real-Time Systems* pp. 67–100.

Bassem, Sam Hsu (2000). A java-based remote laboratory for distance learning.

Casini, Marco, Domenico Prattichizzo and Antonio Vicino (2001). Automatic control telelab: un laboratorio remoto per l'ingegneria dei controlli.

Hamza, Khalid, Bassem Alhalabi and et al. (2000). Remote labs!

Liu, C.L. and J.W. Layland (1973). Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM* pp. 40–61.

P. Gai, L. Abeni, M. Giorgi G. Buttazzo (2001). A new kernel approach for modular real-time system development. *Proc. 13th IEEE Euromicro Conf. on Real-Time System, Delft, Netherland.*

Ph, Bassem Alhalabi (1998). Virtual labs vs remote labs: Between myth reality.

Sadoski, Darleen and Santiago Commella-Dorda (2000). three tier software architettures.

Schwarz, Janek, Andreas Polze, Kristopher Wehner and Lui Sha (2000). Remote lab: A reliable tele-laboratory environment. *International Conference on Internet Computing.*

Stancil, Daniel D. (2000). The virtual lab: Engineering the future.

E. Bini, G. C. Buttazzo and G. M. Buttazzo, "A Hyperbolic Bound for the Rate Monotonic Algorithm", *Proceedings of the* 13^{th} *Euromicro Conference on Real-Time Systems*, pp. 59-66, June 2001.

C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard real-Time Environment", *Journal of the ACM* 20(1), 1973, pp. 40–61.

M. Spuri and G.C. Buttazzo, "Scheduling Aperiodic Tasks in Dynamic Priority Systems," *Real-Time Systems*, 10(2), 1996

C.D. Knight and S.P. DeWeerth, "World wide web-based automatic testing of alnalog circuits",Midwest symposium Circuits and System,1996

M.Shaheen and k. Loparo and M. Buchner, "Remote laboratory experimentation", American Control Conference, Philadelphia,1998

B. Aktan and C.A. Bohus and A. Crowl and M.H. Shor, "Distance learning applied to control engineering loboratories",IEEE Transactions on education,1996

A. Bhadari and M.Shor,Access to an instructional control laboratory experiment trough the world wide web, Proc. of American Control Conference,1998,

J.W. Overstreet and A. Tzes, "An Internet-based real-time control engineer laboratory", IEEE Control System Magazine,1999

J. Zhang and J. Chen and C.C. Ko and S.S. Ge, "A web-based laboratory on control of a two-degree-of-freedom helicopter",Proc. of Conference on Decision and Control, 2001

T.f. Junge and C. Schmid, "Web-based remote experimentation using a laboratory-scale optical tracker",Proc. of American Control Conference,2000

J. Apkarian and A. Daves , Interactive control education with virtual presence on the web, Proc. of American Control Conference,2000