In Web-Based Control and Robotics Education. Series: Intelligent Systems, Control and Automation: Science and Engineering, Vol. 38, Tzafestas, Spyros G. (Ed.), Springer, 2009.

Chapter 11

IMPLEMENTATION OF A REMOTE LABORATORY ACCESSIBLE THROUGH THE WEB

Giuseppe Carnevali, University of Pavia, Italy
Giorgio Buttazzo, Scuola Superiore Sant'Anna, Italy

E-mail: carnev75@libero.it, giorgio@sssup.it

1 Introduction:

In spite of the big crisis of the years 2000, Internet has continued to grow. The broadband has reached more and more people all around the world and the used technology are always more dynamic and user friendly. The worldwide distribution of the Net joined with the simplicity of information sharing has created new economical and social models in several fields. The methodology of learning, which is a primary form of information and knowledge sharing, was also significantly involved in such a process, evolving in what the E-learning paradigm. At the beginning, E-learning enabled users to share E-books, web pages with exercises and have simple chats with the teachers. Then, videoconferences and virtual desktops were introduced to improve the learning experience of the students. Today, the Internet can also be exploited to conduct actual experiments on physical devices available in remote locations. This approach is twofold. From one hand, students can access remote physical devices from anywhere and at any time, conduct longer experiments and compare different approaches among various laboratories. On the other side, scientists and technicians can collaborate from different locations 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 also introduces other advantages with respect to traditional models. In fact, it has fewer requirements in terms of space, time, money and travels. These aspects are more important in scientific and technical fields where it is necessary to perform practical laboratories activities to complete the course of study and improve the theoretical knowledge with direct experience.

Giving students a complete and instructive laboratory experience is a hard challenge for the E-learning field. Two solutions are typically adopted. The more straightforward one consists of using a simulated environment accessed through specific software tools [1][2][3][4][5]. This means that all the experiments are performed in a synthetic environment, where physical processes are defined and

simulated using mathematical models. The main advantage of this solution is that it is simple to implement, but the knowledge acquired by the students depend on several factors, such as the model used in the simulation, the types of imposed constraints, and the functionality of the tool. The main disadvantage of this approach is that several details may be neglected, taking the students away from practical problems.

The other solution, which is more sophisticated, consists of implementing a remote control and monitoring system that interacts with physical devices. This solution is more difficult and more expensive to implement, but provides a better laboratory experience, because it allows the students to interact with real devices and face with the actual behaviour of the system, including non-linear phenomena and noisy data.

Another advantage of this solution is that students are protected against unexpected and dangerous behaviours of the devices and, at the same time, sophisticated devices are more protected against inappropriate usages. In fact, a suitable user interface can disable dangerous operations, or bring the system in a safe state under 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 visual systems, 3D laser scanners, or mobile cameras, in order to test specific algorithms on real data. Control experiments can also be carried out on robotic devices, by changing input set points and control parameters.

2 Examples of existing virtual labs:

There are already several virtual laboratories on the web, mainly developed in university research labs, which 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, including a position controller for a dc motor, a water level controller, and a levitation control system [6]. Figure 1 illustrates a screenshot of the initial window related to the magnetic levitation system. The Automatic Control Telelab allows a user to execute remote control experiments using three operational modes: Predefined Controller, User-defined Controller, and Student Competition Experiment. In the first mode, the student can select 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.

Figure 1: The virtual laboratory developed at the University of Siena

The real-time group at the University of Illinois at Urbana Champagne, develzzoped an inverted pendulum that 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 [7]. The environment, illustrated in Figure 2, allows the user to send its own controller and a fault tolerant technique protects the system from possible mistakes in the control algorithm and malicious attacks, by switching the user controller to a safe backup controller when critical conditions are detected.

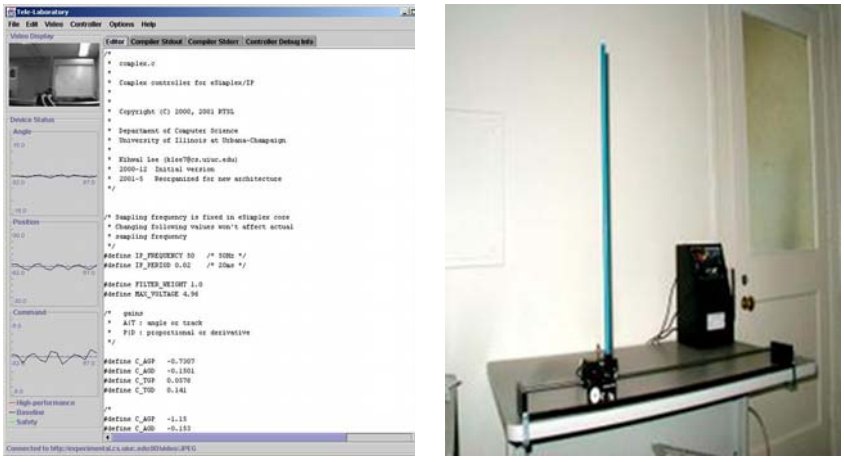

Figure 2: The virtual laboratory developed at the University of Illinois

At the University of Pisa, the control group of the Centro “E. Piaggio” developed a remote laboratory named “Breaking the Lab's Walls: The Tele-Lab” [8]. The environment provides a number of devices on which the user can conduct control experiments: an anthropomorphic robot, a simple DC motors, a magnetic levitator and an interactive environment intended to allow fair and thorough com-

parison of different techniques to solve a basic problem in nonholomic motion planning.

Figure 3 illustrates the interface for the DC motor experiment. The environment allows the user to modify parameters like set points, controller type and sampling time. By clicking on another button the user can start a teleconference application that visualizes the scene under control. At the end of a control session, it is possible to get numerical and graphical data of the experiment.

Figure 3: The virtual laboratory developed at University of Pisa

Another example of virtual laboratory has been implemented at the Polytechnique Federale de Lausanne. Here the students can carry on experiments on an inverted pendulum through an innovative graphical interface (see Figure 4).

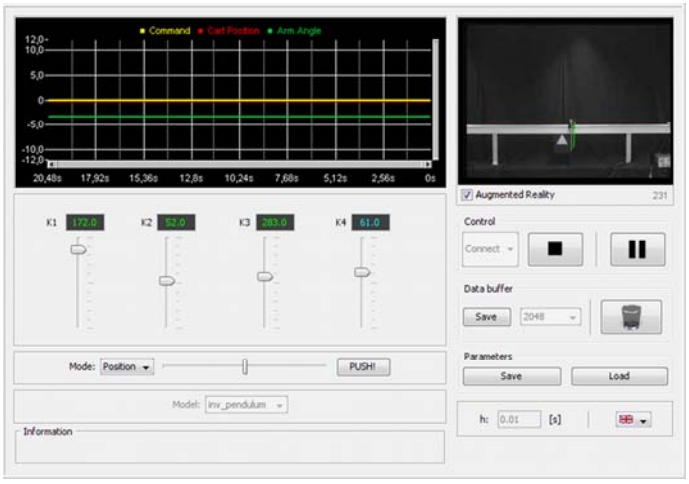

Figure 4: The inverted pendulum at the Polytechnique Federale de Lausanne.

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 [10].

In this chapter, we present a virtual laboratory environment that allows a remote user to interact with a real-time operating system to perform a control experiments. The peculiarity of the proposed lab consists in the use of a hard real-time kernel, called Shark [11], that can be tested on a real application to verify the sensitivity of task 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.

3 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 graphical interface is built using the Java applet technology. This technology is based on a virtual machine so that different platforms can run the same application without need of extra work. Figure 5 illustrates a generic interface for a PID controller working on a feedback system.

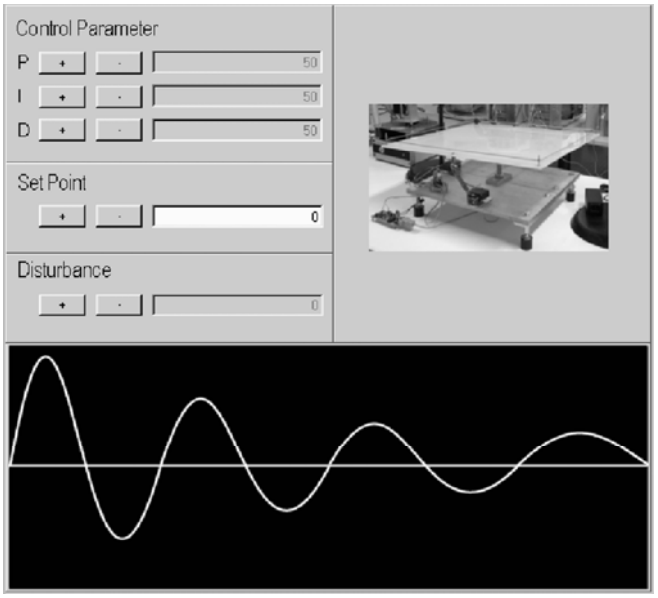

Figure 5: Interface for the PID controller

The graphic interface consists of three different areas. The first one is used to change the experiment parameters, the second one shows the experiment data and the third one shows the laboratory scene.

The first area is divided in different sub areas, one to change the control parameters, one to change the set-point and one to set the external disturb. In the present version, control parameters consist of controller's proportional, derivative and integral gains. Any parameter can be modified either by entering a numerical value in a field or by incrementing/decrementing its value through a pair of buttons.

The second area is located at the bottom of the experiment window and shows the graphical evolution of a selected system variable.

Finally, the third area on the upper-right corner of the window is used to display the images acquired from a web cam located near the physical device. Such a feature allows the user to monitor in real time the actual evolution of the system, 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 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 to avoid occupying the lab for a long time so allowing several students to run their own experiments. 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.

4 Software architecture:

The software architecture proposed for our environment is based on two main concepts: the laboratories and the experiments. A laboratory is a container for the experiments that can be added or removed from it. A laboratory has to maintain a list and a brief description of all its experiments and has to route the users to the correct experiment.

The experiment is the abstraction of the trial that a student can perform on a physical device. His main tasks are to handle the physical resources and to communicate with the remote clients.

In the next sections we present the main characteristics of the software architecture. We first consider the simplest case in which we have a laboratory with a single experiment. Then, we present the software architecture from a more abstract point of view and we will describe a more complex environment where a laboratory has to handle more experiments.

4.1 Basic details:

The software architecture of the virtual lab is organized into three layers. This approach increases flexibility and modularity, since it allows sharing the responsibility of the system among the different layers and allows decoupling between different components.

Thanks to this solution, the clients do not know about the business logics on the servers and are free to change, improve or modify the graphics without affecting the server layer. In addition, the real-time system (layer 3) is completely decoupled from the server (layer 2), which is essential to guarantee the timing behavior of the control application. In this way, the real-time system runs in a well known environment and is not affected by the unpredictable behavior of internet latencies introduced by the TCP/IP communication protocol.

The three layer architecture of the system is depicted in Figure 6.

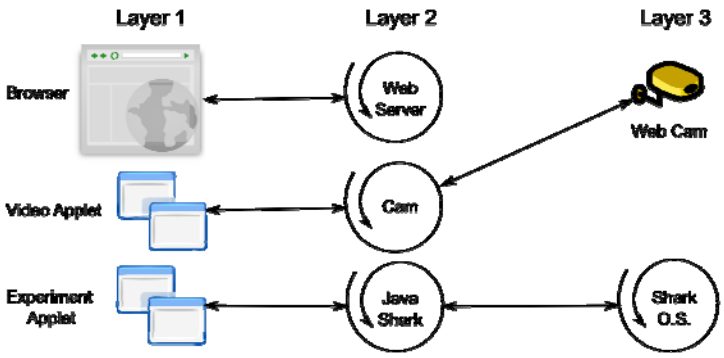

Figure 6: Three layer architecture of the system

The Browser contains the applets and the spirals represents the servers waiting for or handling connections.

The interaction between layers works as follows. When the browser gets connected with the Web Server through the internet, it asks for permission to begin the experiment and the Cam and Java/Shark servers starts the communication. The Cam server starts sending the images captured from the web cam to the Video Applet on the client and the Java/Shark server starts sending experimental data from the Shark real-time kernel to the Experiment Applet. The communication continues until the experiment is ended by the user or until a timeout notification is raised by the system. An appropriate message is displayed when the experiment cannot be initiated.

The purpose of the first layer is to handle the user graphical interface, as also done in [12]. This layer consists of a browser that contains two applets: a Video Applet and an Experiment Applet. The Video Applet receives the images from the

Cam server and displays them. The Experiment Applet handles the parameters set or changed by the user and sends them to the Java/Shark server. Moreover, it receives the experimental data from the Java/Shark server and displays them in real time. These two applets have been integrated in the same frame, as can be seen in Figure 5, but for the sake of clarity we consider them separately.

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), which sends the images from the webcam to the client; and a Java/Shark server, which manages the communication between the client and the Shark real-time kernel.

This layer is responsible for connecting the user to the laboratory via web. To achieve this goal, we needed a stable, reliable, responsive and, possibly free, web server. After evaluating different solutions, we selected the Apache web server. Besides meeting all our needs, this server is open source and includes a very useful module, called Tomcat. Tomcat is a servlet container and provides a pure Java HTTP web server environment for Java code to run. Tomcat allows the possibility to build customized servers and it has been used to push the images acquired from web cam to the clients.

The Java/Shark server is the most important component of the architecture. His main goal is to handle the communication between the clients and the underlying real-time subsystem. Among the different tasks, it handles the user access logic to the experiment, checks the experiment timeout and manages all the service communications. A scheme of the threads involved in the Java/Shark server is illustrated in Figure 7.

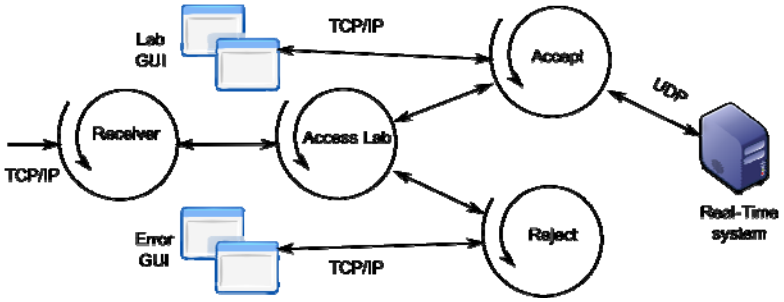

Figure 7: Threads involved in the Java/Shark server

In Figure 7 the spirals represent the threads that compose the server, the little windows named Lab GUI and Error GUI represent the Graphical User Interfaces shown to the users, and the Tower PC on the right represents the computer running the Shark real-time kernel handling the control application.

This server is a modular component that has two main blocks: the Receiver and the Access Lab thread. The Receiver thread is a daemon that listens to a socket for

incoming requests. The Access Lab is the heart of the system and handles all the experiment access requests. It knows the experiment status, knows whether the experiment is free or busy, and notifies the users with appropriate messages. If the experiment is free, it starts two threads, one for enabling the communication between the client and the real-time system, and one for enabling the communication between the real-time system and the client. For the sake of simplicity, only the Accept thread is shown in Figure 7. If the lab is busy, the Access Lab starts the Reject thread that warns the user and closes the communications.

As shown in Figure 7, the communication between the real-time system and the Accept thread is done through the UDP protocol, which is more suitable for implementing control applications with real-time constraints.

Finally, we have the web cam server that enables the students to see the experiment scene in real-time. Different solutions have been considered to implement such a functionality, keeping in mind that our objective was to minimize the bandwidth while keeping good speed and image quality. The implemented solution is called server push and represents a good compromise between a streaming solution, where there is a continuous flow of images, and the snap-shot solution, where the client, from time to time, look for new images. According to this method, the server periodically sends, with a configurable period, the image acquired from the web cam. Selecting a suitable push period, it is possible to reach a good tread off between bandwidth consumption and video speed.

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.

For this layer, we decided to adopt the Shark real-time operating system, which supports either hard and soft real-time applications and has a modular component-based interface for the specification of scheduling algorithms and resource management protocols [13].

The network communication protocol is a crucial aspect for achieving a predictable behavior in a real-time system, due to high variability of packet delays in the underlying network. The Shark kernel supports the UPD/IP stack protocol. The low-level driver has been implemented to meet real-time system requirements avoiding unpredictable delays during dispatching and receiving of network packets. The goal has been reached solving two main problems: the interrupt handling and the mutual exclusion on the network card. The first problem has been solved handling the network card interrupt with a soft aperiodic task, whereas the second problem has been solved using a priority inheritance protocol to access shared resources [14].

Shark implements may different device drivers that enable the application to interact with a lot of physical devices. For our experiments, the serial port communication driver and the National Instruments 6025E board driver have been used.

4.2 Advanced details:

In this section we present how the system works when a laboratory contains more than one experiment.

Different solutions have been considered. In a first instance we thought to exploit a hybrid solution using a standard web server that allows remote users to connect to the laboratory adopting a technology, like CORBA or RMI, to handle the system back-end. The advantage of this solution is that it would hide low-level details and would simplify the development of the network services. On the other hand, this solution is not optimal to transmit data streaming and introduces a certain complexity to the system. In the last years web technology became more dynamic, thus we preferred to use a solution completely based on the web. It is illustrated in Figure 8.

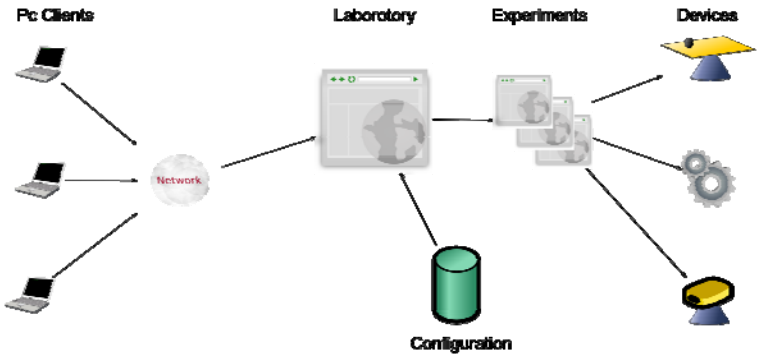

Figure 8: Advance software architecture overview.

On the right we have the remote PC clients and the Internet cloud. In the middle, the big rectangle and the staked rectangles represent web pages. The cylinder represents the classical database and the physical devices are illustrated on the right.

A remote PC connects to the laboratory through the main web page, labelled Laboratory. This page lists and describes all the available experiments and the user can choose the desired experiment. When an experiment is selected, the user is redirected to the web page of the experiment, named Experiment in Figure 8. Here the Java/Shark server is started and the interaction between system and client is the same as described in the previous section.

Before a laboratory is set, it is necessary to start a configuration session where any experiment is described and registered with it. The start-up operations are

handled by a script, which reads the configuration database and checks the availability of all the components of the system. Then, the main page is dynamically generated.

The system also defines a privileged user, called configurator user, who can modify the configuration on the fly through a protected web page and, for example, can put some experiments off line and some others on line for maintenance purposes.

This architecture is scalable and it is possible to make a hierarchical structure where a server is used as a high level gateway that routes the user towards the selected remote laboratory, which can implement many different experiments. In this way it is possible to have many remote laboratories geographically located in different places. It is also possible to use a more sophisticated and dynamic technology introduced by web 2.0, but this last idea has not been tested yet.

5 Hardware architecture:

The Hardware architecture reflects the software organization and consists of three different layers. In this section we illustrate the main characteristics of the hardware components, focusing on the structure of the connections and on the relations that occur among physical components.

A scheme of the hardware architecture is illustrated in Figure 9.

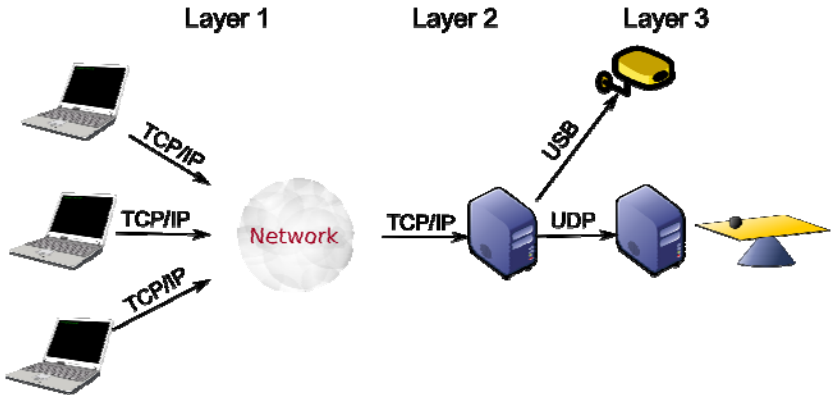

Figure 9: Hardware architecture

The users are represented on the left hand side, whereas the big cloud in the middle represents the Internet connections. The two tower PCs denote the two servers (one acting as a network interface and the other supporting the real-time control software). The server dedicated to the network interface also manages the web cam. On the right, we considered a sample control application consisting of a ball and plate balancing device.

The PC clients use a TCP/IP protocol connect to the internet, but can be based on any kind of platforms, like x86 architectures or ARM architectures, and any kind of operating system, like Windows, Macintosh, Unix, or GNU/Linux, as long as Java technology is supported.

The server system consists of two computers: one dedicated to the communication with the clients and one acting as a gateway for the control experiments. The server on the second layer is also responsible for the system security. In our work, this server is also directly connected with the web cam, but this is not strictly required, since different servers could be used to handle the video transmission.

The real-time computer at the third layer is not visible from the external world. It communicates with the communication server using a UDP connection on a dedicated local network. This computer runs the Shark real-time kernel [9] 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.

All the servers consist of x86 machines. The web cam we used is a Logitech quickCam, which has been selected because its driver is available for many operating systems, like Windows, Macintosh and Linux. Moreover, it is provided with a free programming tool kit that allowed us to modify its behavior. It has a CMOS sensor, manual focus, 640 x 480 frame resolution and a 30 fps frame rate.

The multi function data acquisition device we used is a National Instruments PCI-6025E board. This board is equipped with eight 24-bit input ADC lines, two 24-bit output DAC lines, eight Input/Output programmable digital lines, and two 24-bit counter/timers. The heart of this board is the application-specific integrated circuit DAC-STC. We have used the library provided with the Shark operating system to exploit the board functionality [15].

6 Experiments:

The virtual environment presented in this chapter allows the implementation of two basic components: the laboratories and the experiments. A laboratory is a container for experiments, whereas an experiment is a specific procedure for using a physical device available in the laboratory.

In order to test our system, we implemented a laboratory that contains two experiments: a controller for a ball balancing device and a position controller for a web cam.

6.1 Ball balancing device experiment:

The specific experiment we decided to develop is a real-time controller for 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 10.

Figure 10: The ball and plate balancing device

The graphical interface on the client allows the user to modify the parameters of a PID controller. In addition, it is possible to visualize the system through a web cam and follow the temporal evolution of the error in a graphic diagram (see Figure 5). Finally, the user can also introduce a disturbance on the plate to verify the stability of the system and to measure the response times required to bring the error below a certain desired value.

The rotating plate is made of a squared plexiglass surface with a small fence of the borders that prevents the ball to fall out during motion. 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 six periodic tasks, three are hard real-time tasks and the remaining three are soft real-time tasks. We have used the Cyclical Asynchronous Buffer technique (CAB) [16] to share some data that can be concurrently accessed by the tasks and we have used the semaphores technique for other mutually exclusive resources.

The Cyclical Asynchronous Buffers (CAB) are useful to handle the communication among periodic tasks with different periods. A CAB provides a one-to-many communication channel and in each instant it contains the most recent message. The message is not consumed by the receiver and is updated only when overwritten by the producer. In this way many clients can read the same message simultaneously. On the other hand, an internal buffer mechanism prevents the producer from being blocked until it has finished to write the new message.

The task control scheme used for the application is illustrated in Figure 11.

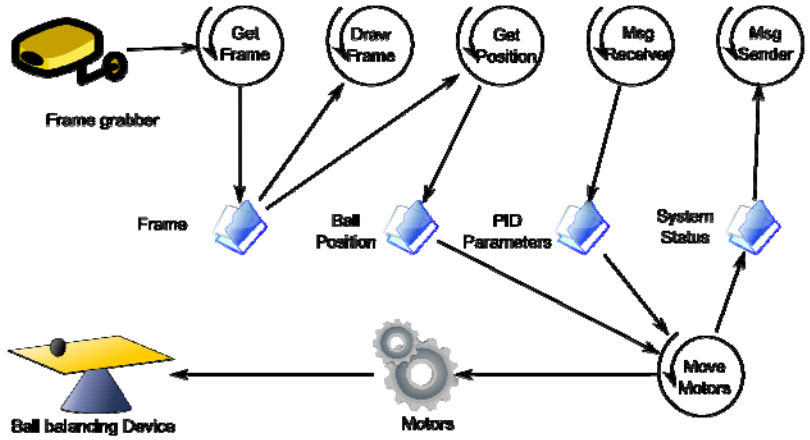

Figure 11: Tasks control scheme

In Figure 11, the spirals represent the tasks composing the real time application under Shark, and the folders, in the middle of the figure, represent the shared resources. The figure also illustrates the camera, the ball balancing device and the servomotors for moving it.

The tasks named Get_Frame, Get_Position and Move_Motors are hard periodic tasks scheduled according to the Earliest Deadline First (EDF) [17] scheduling algorithm.. The other tasks, namely Draw_frame, Msg_Receiver and Msg_Sender, are soft periodic tasks that are scheduled using the Constant Bandwidth Server (CBS) [18]. The Shark operating system also allows changing the scheduling algorithms and the tasks parameters during the initialization process of the real time application. In this way it is possible to test the behaviour of the control application with different algorithms and/or timing constraints.

Figure 11 illustrates how the different tasks interact to control the ball balancing plate. Get_Frame is a hard periodic task used to interact with the camera: it opens the communication with the frame grabber, gets the image and closes the communication; then, it puts the image in a Cyclic Asynchronous Buffer.

The Draw_frame is a soft periodic task. It gets the image from the CAB and calculates statistical information on the running tasks. All data are printed on the screen of the computer hosting the real-time application. This information is only used for debugging purpose and is not accessible to the remote users.

Get_Position is a hard periodic task. It 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. After the computation, this information is stored in the Ball Position shared buffer.

Move_Motors is a hard periodic task. A PID control algorithm is implemented in this task and the information stored in the Ball Position and the PID parameters buffer are used by the control algorithm to calculate the new position of the two servo motors that move the ball balancing plate. This information and the ball position are subsequently stored in the System Status shared buffer.

Msg_Receiver is a soft periodic task. It opens a UDP connection and periodically reads the incoming messages from the network. The received messages are parsed, formatted, and finally put in the PID parameters buffer.

Msg_Sender is a soft periodic task. It opens a UDP connection and periodically reads the data stored in the System Status buffer. Then parses and formats these data and send them through the network to the upper levels.

6.2 Rotating Web cam experiment:

The second experiment we developed is a real-time controller for a rotating web cam that moves according to user commands to monitor the environment. The system consists of a servo motor attached to a web cam that has to be controlled to get the required angular position to reach the desired view. A picture of the system is illustrated in Figure 12.

Figure 12: Controlled Cam Structure

The graphical interface on the client is very similar to the one presented in the previous experiment and allows the user to modify the set point parameters of a PID controller (see Figure 5). The acquired images are displayed on the screen and the data showing the difference between the starting angular position and the actual position are tracked. In this experiment, the error command is disabled and, unlike the previous experiment, the set point command is enabled.

The motion of the servomotor is transmitted to the web cam throughout an aluminium structure. Figure 12 illustrates the structure holding the servomotor and the web cam.

The servomotor is an Hitachi HS-805BB controlled by a PWM (Pulse With Modulation) signal generated by the multi function data acquisition device of the PCI-6025E board. To protect the data acquisition device from extra currents, an electronic circuit has been developed to connect the servomotor to the board.

The control application is developed under the Shark operating systems and consists of three periodic tasks, as shown in Figure 13. The figure illustrates how the different tasks interact to achieve the goal. We have used the semaphores technique to access mutually exclusive resources.

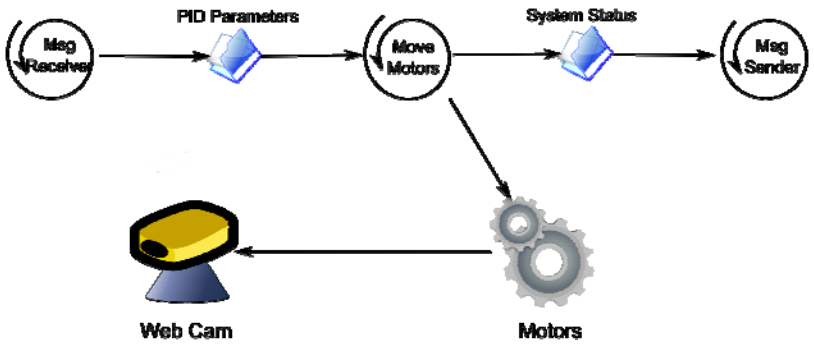

Figure 13: Web cam controller

Move_Motors is a hard real-time periodic task scheduled using the Earliest Deadline First scheduling algorithm. It implements a PID controller and uses the information stored in the PID parameters buffer to calculate the new position of the motor, which is then stored in the System Status buffer.

Msg_Receiver and Msg_Sender are soft real-time periodic tasks scheduled using the Constant Bandwidth Server. They work as the equivalent tasks described in the previous experiment. Both tasks open a UDP connection, but Msg_Receiver receives the messages from the user to modify the experiment set point, whereas Msg_Sender sends the experiment status to the upper levels for monitoring purposes.

7 Conclusions

In this chapter 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 experiments we have implemented consists in controlling a two-degrees-of-freedom plate balancing device for keeping a ball in a desired position and in controlling a rotating web cam to monitor a remote environment.

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 tasks 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.

Bibliography

1: S Hsu, B Alhalabi (2000) A Java-based Remote Laboratory for Distance Education. http://www.ineer.org/Events/ICEE2000/Proceedings/papers/MD2-1.pdf. Accessed 16 Aug 2008.

2: A Alhalabi, M K Hamza, S Hsu, N Romance (1998) Virtual Labs VS Remote Labs: Between Myth Reality. http://www.cse.fau.edu/~bassem/CADET/FLHEC1998-Deerfield.pdf. Accessed 16 Aug 2008.

3: B Alhalabi, D Marcovitz, K Hamza, S Hsu (2000) Remote Labs: An Innovative Leap in Engineering Distance Education. http://www.cse.fau.edu/~bassem/Publications/Pub-35-C-ACE2000-Australia.pdf. Accessed 16 Aug 2008.

4: K Hamza, B Alhalabi, M Marcovitz (2000) Remote Labs!. http://polaris.cse.fau.edu/~bassem/CADET/AACE2000-SanDiego.pdf. Accessed 16 Aug 2008.

5: A Cervin, D Henriksson, B Lincoln, J Eker, K E Årzén (2003) "How Does Control Timing Affect Performance?". IEEE Control Systems Magazine, 23:3, pp. 16--30, June 2003.

6: M Casini, D Pratichizzo, A Vicino (2002) Automatic Control Telelab: un Laboratorio Remoto per l'Ingegneria dei Controlli. BIAS Automazione e Strumentazione:142-145.

7: J Schwarz, A Polze, K Wehner, L Sha (2000) Remote Lab: A Reliable Tele-Laboratory Environment. International Conference on Internet Computing: 55-62

8: A Bicchi, A Coppelli, F Quarto, L Rizzo, F Turchi, A Balestrino (2001) Breaking the Lab's Walls: Tele-Laboratories at the University of Pisa. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, pages 1903-1908, 2001.

9: C Salzmann (2005) eMersion. http://lawww.epfl.ch/page13172.html. Accessed 16 Aug 2008.

10: C Rohrig, A Jochheim (1999) The Virtual Lab for controlling real experiments via Internet. In Computer Aided Control System Design. IEEE Computer Society Washington, DC, USA.

11: P Gai, L Abeni, M Giorgi, G Buttazzo (2001) A New Kernel Approach for Modular Real-Time Systems Development. In ECRTS '01: Proceedings of the 13th Euromicro Conference on Real-Time Systems. IEEE Computer Society Washington, DC, USA.

12: D Sadoski, S Commella-Dorda (2003) Three Tier Software Architectures. http://www.sei.cmu.edu/str/descriptions/threetier_body.html. Accessed 16 Aug 2008.

13: P Gai, L Abeni, M Giorgi, G Buttazzo (2001) "A New Kernel Approach for Modular Real-Time Systems Development". IEEE Proceedings of the 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, June 2001.

14: P Gai, G Buttazzo, L Palopoli, L Albeni, G Lipari, G Lamastra, A Casile, M Giorgi (1998). S.Ha.R.K. User Manual Volume II.

15: T Facchinetti, (2000) Realizzazione di un sistea robotico per il tiro a volo di bersagli mobili. Master's thesis, Università degli Studi di Pavia, 2000.

16: G Buttazzo (1997) Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Kluwer Academic Publishers, Boston, 1997.

17: C L Liu, J W Layland (1973) Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM: 46-61.

18: L Abeni, G Buttazzo (2004). Resource Reservation in Dynamic Real-Time Systems, Real-Time Systems, Vol. 27, No. 2, pp. 123-167.