RT – Linux Based Control of an Experimental Platform for Teleoperated Systems with 1 d.o.f.

R. Puerto, L.M. Jimenez, J.A. Alenda
Dept. of Industrial Systems Engineering. Miguel Hernandez University
Av. del Ferrocarril s/n, Elche (Alicante), Spain
{r.puerto}@umh.es

R. Aracil
DISAM, Polytecnical University of Madrid
Jose Abascal, 1. Madrid, Spain

Abstract

In this work, a control system developed under RT-Linux is presented. The system consist of an experimental platform for time-response analysis and control of a teleoperated system with one degree of freedom, controlled under different strategies and connected by means a computer network. This platform can be used for testing new strategies for remote control on teleoperated systems. The implementation core is an RT-linux module that uses the COMEDI library to execute the feedback control tasks interacting with the data acquisition board. Communication tasks between the RT-Linux controller and the client graphical user interface (GUI), are based on Java2 and JNI technologies.

1 Introduction

In a general way, teleoperation covers all those technologies that allow humans the remote operation by means two devices: a master device close to the operator in the local area, and a slave device in the remote zone. This configuration is specially important in tasks performed in dangerous or hostile environments (radioactive products manipulation, underwater or spatial works, remote surgery, etc.).

The use of tools to transport or manipulate dangerous substances has been very common from the beginning of the industrial era. But is in the middle of XX century, when the need for manipulation of radioactive elements leads to the development of more sophisticate systems for remote operation, and becomes what it is now called "master-slave systems", where a device called slave, reproduce the same movements done of the master device, who is controlled by a human operator.

The teleoperated systems [1], have been evolving since their first applications in the fourties, where the teleoperate systems were mechanical systems to manipulate radioactive substances, until the newest systems, like comercial systems (GRIPS, JPL-PUMA, ...), or submarine, space or medical applications, where human operator, can "sense" the same forces than the ones produced in the slave device work place.

The teleoperated systems are still a science in development. The present work, develops a computer application who has to be stable, robust, and independent from the operating system, where teleoperated scientists or engineers, can develop or test new control strategies suitable for teleoperation. This application had to be capable of running "real-time" tasks, and to allow supervision and control in a distributed way through a computer network.

The developed application is made of three main blocks:

- RT-Linux module: this program runs the feedback control tasks interacting with the data acquisition board. Their functionality consists of: recovery of data from acquisition board A/D channels, control action calculation for the configured control scheme, and execution of control actions through the analog output channels of acquisition board.

- Data server: this program is in charge of the communication tasks between the RT-Linux controller and the client graphical user interface (GUI). Therefore, this module receives configurations of each experiment from the client GUI and send them to the RT-Linux control module. Also, collects the experiment output data from the control tasks and send them to the client GUI. This communication is based in standard sockets library (TCP and UDP protocols) so it can be accessed from Internet.

- Client graphical user interface: this application allows the user to configure the experiment, request its execution, and recover the resultant experimental data. This software application has been developed in Java, so it can be executed in any platform with a Java2 virtual machine.

The use of RT-Linux [2], [3], as base plataform for this system, was choosed by this excellent performance as hard real – time operating system, its reliability, stability and, of course, its low cost. Also, RT-Linux has a installation, configuration and administration less complicated, that other real – time operating systems based on UNIX.

2 Types of control in teleoperation.

Control schemes in teleoperation can be classified in two main groups [1]: unilateral control (information only flows from master to slave devices) and bilateral control (information flows in both ways). Bilateral control (more robust and accurate) allows for force feedback.

The first developments of master-slave teleoperated systems were based on mechanical transmission. The implemented control was of unilateral type, or also call direct control. In this type of control doesn't exist feedback between master and slave devices. And the master doesn't have motor drived links.

The name of the unilateral comes from the notion that the control only has one direction. The slave device can be moved when the master device is moved, but not backwards. Therefore, the master generates reference signals (position or velocity) for the slave links. We can say that the slave has a control system similar to an industrial robot, but the references are generated by the master articulations, instead of a CPU. The control of the slave side is easier than the one implemented in most industrial robots, because in this case, the inverse cinematics and dynamics calculations are not needed.

The bilateral control (fig.1) appears when we need haptic[1] feedback to the operator. In this case, the slave forces are reproduced to scale in the master device, and therefore, to the user hands or arms. In this type of control, the control signals are flowing in both directions, from the master to slave, and from the slave to the master.

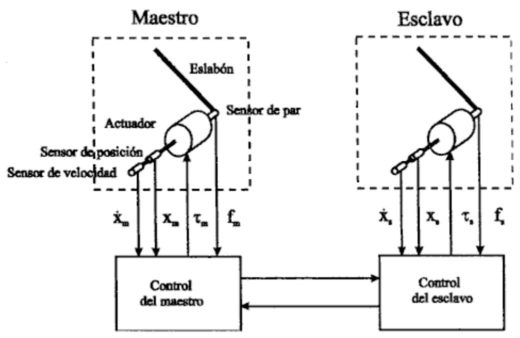


FIGURE 1: *General scheme of a bilateral control implementation. The figure shows a 1 dof teleoperated system.*

Bilateral control can be classified in several types. Basically, there are two main types of bilateral control: "position-position" and "position-force". In the first type, reference signals are the position of the devices. The master reference is the slave position, and the slave reference is the master position. That control scheme is completely symmetrical.

In the "force-position" control type, the slave has as a reference, the master position. But now, the master reference is a force signal from the slave manipulators. Usually this force feedback signal is atenuated in order to improve the stability of system. This type of control, is an asymmetric scheme. Due to this, stability problems can arise, but this method, provide us a better feedback of the slave state (slave forces).

[1]Haptic capacity: Human capacity to feel some information through the sense of the tact, or reaction forces.

To conclude this section, it is worth to comment the huge number of applications where teleoperated technology has an important role. In the previous section, we listed classical applications, like nuclear industry, submarine or space applications, but the new developments aren't less importat. New developments in construction, minery sector, or natural catastrophes, etc., where dangerous or uncomfortable works for the people, can be made by remote controlled machines.

3 RT – Linux Application.

As it has previous been stated, the objective of this work is to develop an application under a real – time operating system, which has to be robust and stable. RT-Linux complies with this specification. Also, this application must allow supervision and control from any computer conected to the network of the teleoperated system. Java language has been choosed as programming tool by its plataform independence, and network capabilities. The application can be divided in three main blocks: the driver or controller, written in C under RT-Linux; the user graphical interface application written in Java; and the data server, written with C and Java.

3.1 RT – Linux Module

The RT – Linux kernel job, is mainly the control of the master-slave devices, through the data acquisition board. In order to program the kernel application the modularity of RT – Linux is used [4], [5], [6], [7]. In this way, a kernel module implements the necessary operations to make the control: data acquisition from physical system, calculations of the control signal value, and transfer of these values to the physical system through data acquisition system. This module is made up of two RT-Linux threads (fig. 2). These threads starts their execution when the module has been installed in the kernel, and they finish when the module is extracted from the kernel. The obligatory functions "init_module()" and "cleanup_module()" intall and extract the module. Additionally, the module use mutex functions, RT-Linux FIFO, and several data structures.

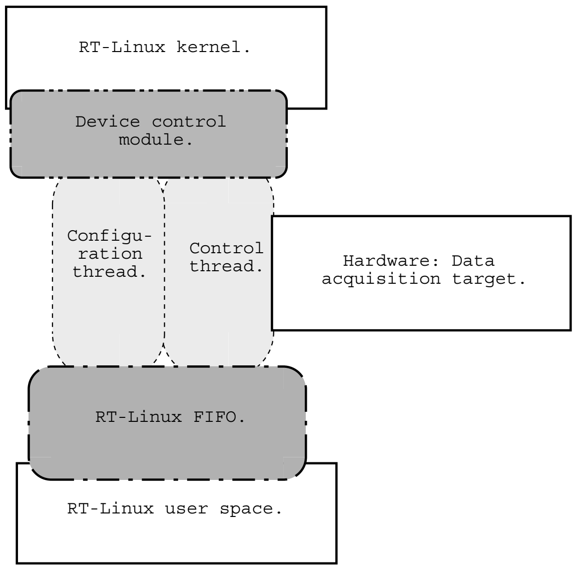


FIGURE 2: *Scheme with the processes in the RT-Linux kernel.*

COMEDI libraries are used to communicate input and output channels of the data acquisition board. "COMEDI" (COntrol MEasure Device Interface), [8], give an interface to use I/O ports, with measure devices such data acquisition boards. This interface is easier and simpler than the Linux I/O interface.

The operation sequence that follows the whole kernel module installation"init_module()" is as follows:

- Start up the mutex variable, to protect the critical section.
- Start up the data structures and arrays, to store information.
- Start up the RT-Linux FIFO.
- Launch the thread who manages the device control operations.
- Launch the thread who manages the kernel module.
- Return of the function.

When the module is extracted from the kernel, the function "cleanup_module()" undoes the installation procedure, following the next steps:

- Stop the thread in charge of the control devices.
- Stop the thread who manages the module (internal administration and configuration).
- Clean up the RT-Linux FIFO.

- Close the COMEDI device ("/dev/comedi0").
- Clean up RT-Linux mutex.

When the control thread is started, it runs the tasks of data acquisition, calculation the value of the control signals, and sending signals to the physical master-slave devices through the data acquisition system.

When this thread is launched, it makes its internal variables selfconfiguration and the Comedi functions initalizations, and after that it starts the control cycle. This main cycle implements the feedback control of the teleoperated devices, according to the user configuration. The thread, suspends itself after every control cycle, until the next clock cycle syncronously linked to the sampled time set by the user. The results of the experiment, are sent to the server through a FIFO.

The configuration thread, is in charge of the internal management of the module. This thread, after self-configuration, repeats a cycle waiting for requests of parameters configuration from de server. Upon a request the thread upgrades the module variables, and comes back to listen for new configurations. The configurations are received through a specific FIFO. The access to the modules variables is protected by an RT-Linux mutex because they are used by both threads.

By means the COMEDI libraries, six input analogic channels are read, and two output channels are writen from the acquisition system. This channels sample signals of position, velocity and current of both devices (master and slave). The output channels are used to send control signal to the devices. These control signals are calculated from input signals, and user parameters.

3.2 Data Server

The main function of the server is to connect the kernel module with the user client interfaces. Therefore, the server has to be a bridge, between the other two applications, and it has to permit the communication between those programs through a computer network.

Basically, the server does the following functions (fig. 3):

- It performs a "direct" communication with the kernel module. This is possible because both processes are in the same computer.
- It allows a conection with the client interface, through computer network using Java sockets.
- It transmits the information between the kernel module and the client.

The server is a text based application. The messages inform the user about the server state, its operations, etc. The conection with the client interface, uses two Java sockets. One of them is a TCP socket, and the other is a UDP socket. The configuration parametres are transmited by the TCP socket, and the results are sent back by the UDP socket.

The communication between user space, and the kernel space are made through RT-Linux FIFOS. At the user space, the RT-Linux FIFOS are accesed by mean C-functions. These functions are built and assembled in a dinamic library under Linux [9].

The link between the Java and C software is implemented by means JNI libraries of Java [10]. These libraries implements functions and data structures which make possible the communication between C and Java programs.

Finally, the server checks the variables that contains data about the conection whit de client and redirects this information between the RT – Linux module and the remote client.

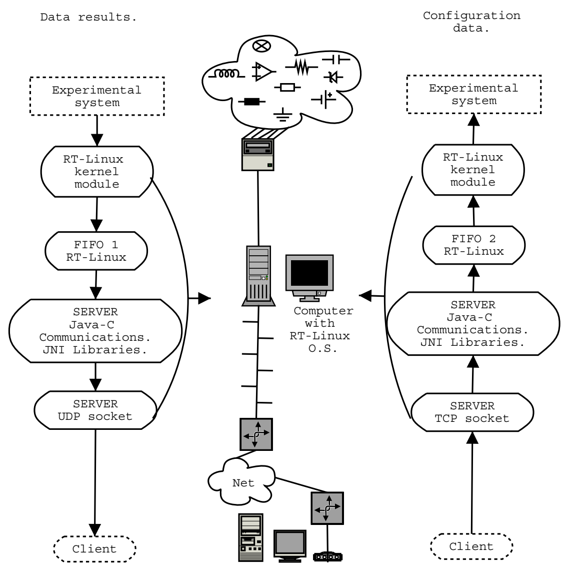


FIGURE 3: *Data server functionality.*

3.3 Client Graphical User Interface

The client graphical user interface is the application where user can introduce the configuration parameters of the experiments. This application allows to control and supervise the experiments, and save the results. The client interface has been written totally in Java, [11], [12], [13]. This feature allow us executing the client in every computer, where it is installed a Java 2 virtual machine.

In the previous sections, it has been shown that the server and the client interface, use Java sockets to communicate with each other. The server is listening for requests, while the client is who begins the communications. That procedure allow users to experiment with the physical system through a network.

The client interface is meant to be a friendly GUI. It is structured unsing tabs in order to access to different configuration or visualization windows (fig. 4). All experiments must be configured step by step using a form. According to the parameters that users have chosen, the interface show new fields to fill. This method pretends to be an easy way to configure the experiments.

Graphical components of the cient interface are programed with the Swing libraries of Java. This package allow users to have more graphical independece from the operating system. Thus, the apparence of the application is similar on different platforms.

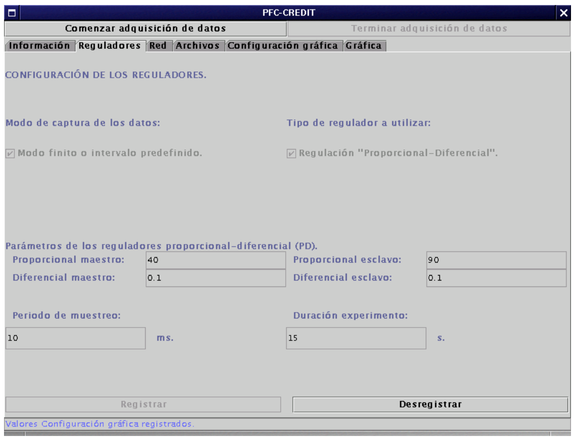
3.4 Example.

The application has been tested with a teleoperated system. This teleoperated system is a model with one degree of freedom (fig. 1). The experimental test carried out has proved the validity of this architecture showing optimal results.

Figures 4 and 5, shows the client interface. In the figure 4, we can see the window where experiments are configured. Figure 5 shows the results of the position–position controler experiment. The control strategy used in this experiment, has been PD controllers. The value of these regulators are shown in the table 1.

Parameter	Value
Master proportional	40
Master Differential	0.1
Slave proportional	90
Slave differential	0.1

TABLE 1: *Value of the PD-controllers*



CONFIGURACIÓN DE LOS REGULADORES.

Modo de captura de los datos:

[✓] Modo finito o intervalo predefinido.

Tipo de regulador a utilizar:

[✓] Regulación "Proporcional-Diferencial".

Parámetros de los reguladores proporcional-diferencial (PD).

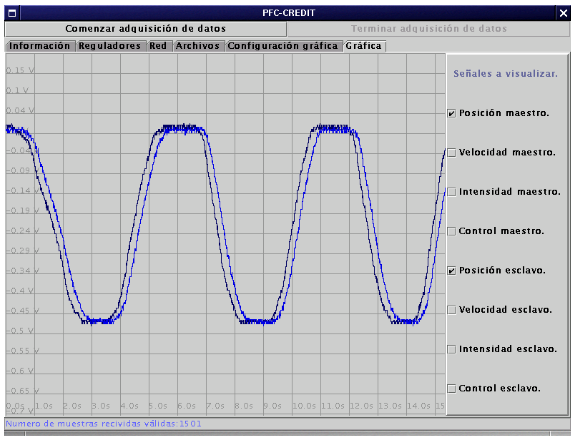
Proporcional maestro:	40	Proporcional esclavo:	90
Diferencial maestro:	0.1	Diferencial esclavo:	0.1

Periodo de muestreo: 10 ms.

Duración experimento: 15 s.

Valores Configuración gráfica registrados.

FIGURE 4: *Parameters page, to configure the teleoperated experiment (position-position).*



Señales a visualizar.

[✓] Posición maestro.

☐ Velocidad maestro.

☐ Intensidad maestro.

☐ Control maestro.

[✓] Posición esclavo.

☐ Velocidad esclavo.

☐ Intensidad esclavo.

☐ Control esclavo.

Numero de muestras recividas válidas:1501

FIGURE 5: *Position signals of the devices, teleoperated experiment (position-position)*

.

4 Conclusions and Future Work

As conclusion, in the present work a laboratory model for a teleoperated system with one degree of freedom has been build. The software application under RT-Linux has also been completed allowing the control, analysis, and development of specific application giving optimal results.

The new tasks to continue this work are improving the security, the code optimization, the shared information among the diverse blocks of the application, and more analysis features.

As we can see at the references section, one of the big points to work with RT-Linx and free software, is the amount of available documentation.

Another advantage of working with Linux, as important as the previous one, is the open source code. It is possible to consult the source code to solve any problem, or to modify the original code to our needs. These advantages are difficult to find in others operating systems.

Also to mention, its excellent performance as hard real – time operating system, stability, reliability, robustness, availability, and of course, its low cost. Finally, we can conclude that is an excellent platform to develop real time projects.

References

[1] L.F. Penin 1998, *"Control Bilateral de Robots Teleoperados. Contribuciones en Reflexin de Esfuerzos"*. PhD Thesis. Universidad Politcnica de Madrid.

[2] V. Yodaiken, M. Baradenov, "*RTLinux version two".*

[3] J.I. Ripoll, 2001, *"Tutorial de Real Time Linux"*.

[4] D.A. Rusling, 2002, *"The Linux Kernel"*.

[5] T. Aivazian, 2001, *"Dentro del ncleo 2.4"*.

[6] O. Pomerantz, 1999, *"Programacin de mdulos del ncleo de Linux"*.

[7] K.A. Robbins, S. Robbins, 1997, *"Unix Systems Programming"*, Ed. Prentice Hall.

[8] D. Schleff, F. Hess, H. Bruyninckx, 2002, *"Comedi documentation"*.

[9] A. Rubini, 1998, *"Linux device drivers"*, Ed. O'Reilly.

[10] F. Lpez, 2002, *"JNI (Java Native Interface)"*.

[11] N. Meyers, 2000 *"Java Programming on Linux"*, Ed. Prentice-Hall.

[12] A. Froufe, 2000, *"Java 2: Tutorial and User's Manual"*, Ed. Ra-Ma.

[13] Sun Microsystems, 1003, *"Online Documentation"*, http://java.sun.com