DISTRIBUTED PLATFORM FOR THE CONTROL OF THE WIFIBOT ROBOT THROUGH INTERNET

L. Payá, A. Gil, O. Reinoso, M. Juliá, L. Riera, L.M. Jiménez

Departamento de Ingeniería de Sistemas Industriales,
Miguel Hernández University. Avda. de la Universidad s/n.
Ed. Torreblanca. 03202, Elche (Alicante), Spain.
E-mail: {lpaya, arturo.gil, o.reinoso, luis.jimenez}@umh.es

Abstract: This paper presents a distributed platform composed of a team of mobile robots so that they can be controlled by the students through Internet. With this platform, a remote environment is created so that the students can carry out different experiments over several available equipments in the lab, with a flexible schedule. This way, the student can design and test different control algorithms and experience over the mobile robots in the laboratory. Currently, there are 3 kinds of robots available; each one is accessible by the students for monitoring and supervision tasks and also for control tasks using a common interface of communication, in a transparent way from the point of view of the user. *Copyright © 2006 CEA-IFAC*

Keywords: Co-ordination, mobile robots, distributed control, laboratory education, remote control, educational aids

1. INTRODUCTION

In engineering education, it is essential to provide the necessary resources that allow the student putting into practice all the knowledge acquired during the theoretical study of the different subjects. In this sense, students are highly motivated to make use of and benefit from the resources available in remote environments through Internet (Stafford, 2005). Traditionally the element that makes possible this fact is the laboratory, where the student can make the necessary training to become accustomed with the tools and equipments he will find in real situations. However it requires the physical presence of the students and the professors in a pre-established timetable, and the available time is also limited. Besides, the cost of the setting and maintenance of the laboratory is usually high, so, the contents of the practical lessons will be strongly conditioned by the number of available equipments.

The use of Internet-based techniques supposes several advantages. The students can carry out the training lessons in a remote way, accessing the laboratories in a free timetable, from their own house, and disposing of all the time they require to accomplish the aims of the lessons. Also, this system will allow the professor to take into account not only the final results obtained but also the work the student has actually carried out to evaluate the practices. Even, this system could make possible that several educative centres share their equipments and so, the expenses of establishment and maintenance of them (Berger and Topol, 2001).

This paper presents a platform, developed so that the students can make use of the mobile robots available in the laboratory through Internet. Therefore, it makes possible they carry out the practical lessons with this equips out of the pre-established timetables and from any place they prefer.

Many works have been made until this moment about the use of the available equipment through Internet. This way, in the 'control education' area, different 'remote labs' have been developed. Cassini, *et al.*, (2003) present a remote laboratory of automatic control. This lab allows the user to design his own controller by means of the *MATLAB/Simulink* environment, and to test it on the plant through a user-friendly interface. Pastor, *et al.*, (2003) describe a framework that allows Web-based labs focused on learning control systems. A similar remote lab is presented in *RECOLAB* (Jiménez, *et al.*, 2005), where a feedback servomotor is used to test the controller designed by users in a remote environment. Also, several remote labs using robots in the remote environment have been developed. Candelas, *et al.*, (2003) present an educational virtual laboratory for training in robotics that allows as the simulation of a robot arm as the tele-operation of the equivalent of a real robot. Also, Thamma, *et al.*, (2004) describe a client/server architecture for the remote control of a manipulator robot. It makes use of Java programming using *TCP* and sockets, acceding the server using any web browser. Khamis, *et al.*, (2003) present the interface and the architecture '*Developer*' to control and tele-operate the *irB21* robot. At last, Siegwart et Saucy, (1999) present and discuss several applications using mobile robots through Internet at EPFL in Lausanne.

In the proposed platform, the distinguishing feature consists on having a heterogeneous team of different mobile robots that the student can use to make and test the developed algorithms.

On the other hand, in addition of testing the algorithms over one of the available robots, the user can test its interaction with the rest of them carrying out a monitoring of the remaining robots as they are running a task. Therefore, the user can interact with the implemented algorithms that are running in any of the robots available in the platform.

The remainder of the paper is organized as follows. In the next section, the developed platform for the communication with different kinds of robots is described. Section 3 shows how it has been applied to a specific type of robot (*WiFiBot*). In section 4, the use of the system is described, from the point of view of the final user. Al last, conclusions of this work are exposed.

2. PLATFORM FOR THE COMMUNICATION WITH THE ROBOTS

The coordination of a team of mobile robots requires, in general, a system that allows the communication between the different members of the team. Then, if the robots have to act in collaboration to carry out a complex task, the information they must interchange is composed, usually, by the current values of its sensors. As an example, when several robots are exploring an environment, the interchange of the information each robot knows is very important in order to carry out the exploration in an efficient way and to create a topologically correct map of the environment.

Then, having a heterogeneous team of mobile robots, that have different hardware architecture and sensorial capabilities, the objective is to create a common communication layer that allows the access to each element of the system in a easy and transparent way to the user.

For the correct functioning of the system, it is necessary to know which elements are connected in each moment and which services provides each of them. This function is carried out by an *Identification Server*.

It is also important the representation of the state of each element that is connected to the network. (E.g. pose of each robot and measure of the sensors). In this case, a graphical interface has been designed to maintain a list of the members of the team that are active, named *Supervision Application*. This application is based on *Java JDK 1.4.2* and shows the most relevant information of each member. It has been chosen *Java* due to independence it offers respect to the Operative System.

Another interesting feature is the flexibility respect to the programming language used. In this case, *C/C++* is required to program the services in the mobile robots, and *JAVA* to develop the supervision tool.

To sum up, the differences between robots and the data that must be communicated have originated the necessity of having a common communication architecture. The chosen option is based in the CORBA standard.

2.1 Communication Architecture for distributed applications CORBA.

CORBA (OMG, 1995) (Common Object Request Broker Architecture), provides an object-oriented methodology well defined for the implementation of distributed applications, and it constitutes the reference model used in the general design. Two fundamental features of *CORBA* are:

(a) Separation between interface and implementation. The objecs interfaces are specified in a language specially defined with this goal, *IDL* (Interface Definition Language) that is part of the *CORBA* standard. *IDL* isolates the interface of the object from its implementation, offering more portability and interoperability this way.

(b) Independence of the location: An object offers services through its operations and attributes, which are visible through its interface. The clients can require the accomplishment of certain operations over an object with independence of its location. The element that allows this transparency in the location and access to the objects is *ORB* (Object Request Broker).

The components of *CORBA* architecture allow that a client application can request for the accomplishment of operations over a *CORBA* object, which resides in a server. The implementations of the objects that are in the server side, define the methods that implement the *IDL* interfaces. These objects can be implemented in different programming languages.

The *CORBA* standard has been used in similar tasks of communication (Utz and Stulp, 2004), resulting efficient to interchange information in a team of heterogeneous robots in *RoboCup F2000* league.

2.2 *Developed model.*

A *CORBA* based architecture of communication has been developed. The communication between the components of the system is shown of Fig. 1.

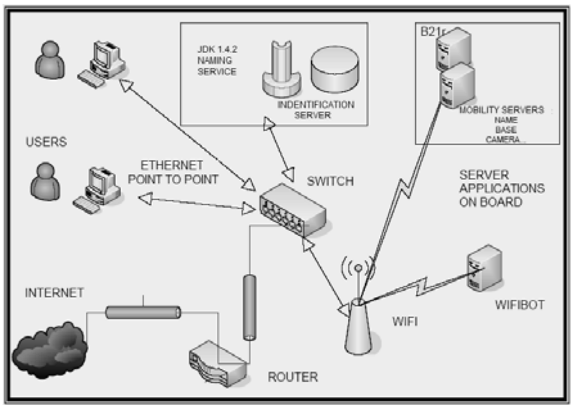

Fig.1. Components needed for controlling the robot in a remote way.

<u>Identification Server:</u> The main function of this application is to provide updated information about the active servers (robots) in the system. This information includes identity, state, and all the *IOR* references that are necessary. Each *IOR* reference allows us to identify in a univocal way an object in our system so that its services can be acceded in a transparent way.

<u>Robots servers:</u> These servers are in charge of managing the services that each robot offers. This way, their objective is to solve the high level requirements that arrive and run the code that implements each service (e.g. reading odometer values, or capturing camera image). This element is created using *C++* with *ORBacus 4.3.0* software.

<u>Supervision Application:</u> Its goal is to request, periodically, the services of the robots that are active in the system. This application is deeply described in section 4.

The identification server and the supervision application were created using *Java JDK 1.4.2*, taking profit of the interoperability that exists between *ORBacus* and *Java JDK.*

3. APPLICATION TO THE WIFIBOT ROBOT

3.1 *Description of the system.*

The *WiFiBot* is a low-cost robot characterized by its flexibility, which allows its use in different environments and situations. Currently, there are two *WiFiBot* models: *SC* and *4C.* The first one includes a *BECK* controller *(SC12),* with a real time operating system (RTOS) and several interfaces may be used, such as *Ethernet, RS232, I2C* and *WiFi* access through an *ASUS 330* wireless bridge. The second has an *'Access Cube'* from 4G-Systems (*MIPS AMD* Processor, 400 MHz, 64 Mb RAM and 32 Mb Flash Memory). It includes *RS232*, *USB*, *I2C*, *Ethernet* and *Wlan* ports, so it is not necessary an independent wireless bridge. Both of them include an Ethernet camera, with pan-tilt movements in the *4G* case, two infrared sensors situated in the front part with a reach of 1.2 meters and four optical encoders with 300 sectors that allow the measure of the speed of each wheel independently. To extend its possibilities, we have included in both models a PC-104+ board (P4M 1.4 GHz, 512 Mb RAM, 2 Gb Flash Disk), with *Linux Debian* operative system. The four motors can be controlled in open or closed loop. The use of these robots is totally transparent for the students in the remote environment having into account that the developed classes allow using them with the same platform despite its differences. Fig. 2 shows both models of the *WiFiBot.*

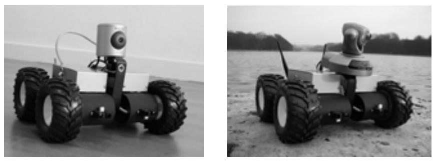

Fig. 2. WiFiBot. (a) SC and (b) 4G models.

3.2 *Implemented services.*

<u>Server.</u> The robot is provided with a server program for the *SC12* controller and a *Windows* client application that acts as an interface to control and monitor the robot. In order to include the kinematics and odometer, some changes have been made on the server. Functions to calculate kinematics and odometer, and a timer to update the odometer, have been incorporated. Also, the protocol has been changed to text mode in order to be used by any kind

of *telnet* client to manage the robot. The methods implemented in the server are summarized in Fig. 3. As the infrared sensor does not present a linear behaviour, a new function *getir* has been implemented to return the distance already linearized.

Client. A *C++* client class has been developed so that the communication is transparent for the user, who has just to develop the algorithm to control the robot. Through it, the user can program any of the functions that the implemented server offers.

```
class wbotctrl{
public:
   wbotctrl();
   int connection(char *servername, int serverport);
   int setv(int v, int w);
      // Sets the linear and steering velocities.
   int getpos(int *x, int *y, int *a);
      // Returns the position and orientation of the robot.
   int reset();     // Resets the odometer to zero.
   int setpos(int x, int y, int a);
      // Sets the odometer to certain values.
   int startodo();     // Activates the odometer.
   int stopodo();     // Stops the odometer.
   int getenc(int *enc1, int *enc2, int *enc3, int *enc4);
      // Returns the current values of the 4 encoders.
   int getbat(int *bat);     // Returns the battery level.
   int getir(double *ir1, double *ir2);
      // Returns the value of the two infrared sensors.
   int getirn(int *ir1, int *ir2);
   int getv(int *v, int *w);
      // Returns the linear and steering velocities.
};
```

Fig. 3. Methods implemented in the client class.

On other hand, the robot includes a camera that provides 640x480 *jpg* images. This camera incorporates a server to which the client has to connect to make the acquisition of the image. The client has to create a socket with the *IP* and port required, and send a request to the camera, using *HTTP* protocol. It consists on a *GET* request, followed by the name of the requested file, the protocol and two return characters. As the camera requires user authentication, a line with the login and password, coded in base-64, must be added. The file asked can be *'video.cgi'* or *'image.jpg'*. It is recommended to require the first one, so that, continuous images are sent, and it will work quicker. A client class has been implemented, including an independent thread that is in charge of the image reception and other methods to obtain the last image in *jpg* format.

Multi-thread bridge server. This server runs on the *PC* board mounted on the robot, and allows several programs to access simultaneously the services that the *SC12* provides. It has been implemented using the same protocol than the *SC12* so that the same client class can communicate with the *SC12* server and the bridge server, with only specifying the new *IP* and port numbers. In fact, the bridge server makes use of this class to link with the *SC12*.

The advantage of this multithread server is that several clients can access the low-level services that the robot offers. For example, a program to monitor the robot can be running simultaneously with the program that controls it. However, when a client is controlling the robot, the rest of clients who want to connect could just monitor it.

The different threads run in parallel, sharing the same space in memory. This server establishes a link with the *SC12* using the client class and a socket remains opened waiting for possible clients. If a new client accesses the system, it is created an independent thread to attend this client, as shown of Fig. 4. To avoid errors in the communication with the *SC12* produced by simultaneous request of several clients, a mechanism of synchronization must be used. This server has been built using a mechanism of mutual exclusion (*mutex*), that allows blocking the fragments of code where the request to the *SC12* are carried out, so, only a thread can be running this fragment of code at the same time. The rest of threads are blocked until the first thread finishes the request process.

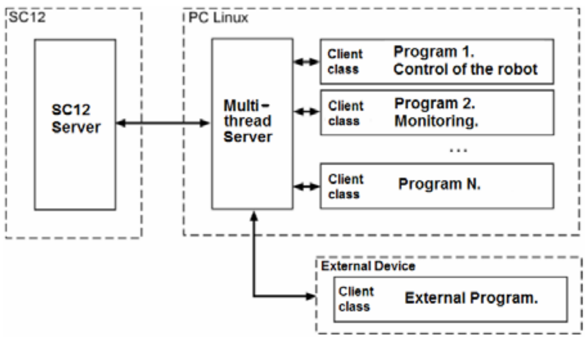

Fig.4. Multi-thread server. It allows several programs to run simultaneously on the server.

To conclude, the layer model developed for the implementation of the communication platform based in *CORBA* in the *WiFiBot* is shown on Fig. 5.

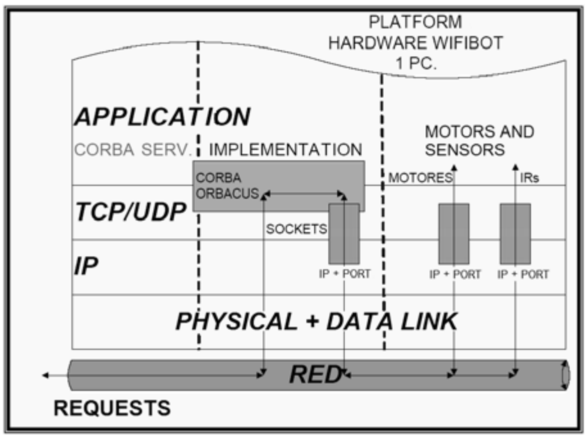

Fig.5. Layer model developed for the implementation of the platform in the WiFiBot.

4. USE OF THE SYSTEM

A *Java GUI* application, using the *MVD* model has been developed to allow the remote access to the robot. The graphical interface has been developed using the *javax.swing* classes. The system can generate events based on three reasons: when the client interacts with the graphical components, by a demon timer that synchronizes the processes and by a thread that, using *CORBA* requests to the identification server, controls the presence of the robots in the system. This application acts as a *CORBA* client, so it is capable to demand services that act over the actuators and sensors of the robots.

When the client application, named *ClienteTop*, begins, it shows a hierarchy tree that contains the available robots and its interfaces. First of all, the client has to configure the necessary data for the Naming Service (*IP* and port number of the name server, and time step to synchronize the requirements).

After this, the user has to activate the command *FindRobots*, which looks for the active robots and assigns them a graphical support if it exists. If no robot is available at the moment, it remains awaiting for a new connection. Fig. 6 shows the global appearance of the application. At the top of the panel, there are some controls to move the robot manually. With the arrow buttons, linear and steering speeds can be increased or decreased, and the central button stops the robot. The step is configurable. At last, the current lineal and steering speeds are shown. Down, there is a *javax.swing.JTabbedPane* component with several lapels. When *WiFiBot* is selected, the panels (shown on Fig. 7 and Fig. 8) selectable are:

<u>Odometer panel:</u> The user can reset the odometer of the robot or set it to a pre-determined value. Also, the odometer data can be continuously read and saved in the file the user chooses.

<u>Infrared panel:</u> The current distances are shown and it can be configured to be continuously written in a file.

<u>CameraWEB Panel:</u> The user can visualize the images captured by the Web Camera. When *'stop'* button is pressed, the camera gives up capturing and the last image can be saved in a *jpg* file.

<u>Algorithm Panel:</u> The user can download trough this lapel the program he has implemented to control the robot so he can test different control algorithms. First of all, the user has to look for the *C++* source file to be sent. Once it is localized, when pressing the button *'Compile'*, the file is transferred and compiled in the remote server (the user has not got the necessary libraries to compile it in his computer) and a window appears to the client, showing him the results of the compilation. If there is no error in compilation, when pressing the button *'Run'*, the uploaded algorithm will begin running. At any moment, the user can stop execution and stop the robot using the button *'Stop'*.

Currently, the system is at the disposal of the students so that they can connect to it during a pre-established period. The student needs a validation for accessing the system.

This way, the students can build algorithms of basic reactive control. Reading the measures of the sensors and the camera, and taking into account the kinematics of each robot, the student must program the behaviour of the robot so that it accomplishes a determined task (e.g. following a pre-recorded route). During the navigation of the robot, all the data can be saved in several text files so that the student is able to get the graphical evolution of the necessary variables, such as trajectory, speed, etc.

Three robots are available in the lab by the moment. (WifiBot, B21r and Eyebot). Taking into account the fact that the developed platform allows controlling them in a common and transparent way, and the platform also allows communication between robots, the student can program algorithms to carry out tasks that suppose collaboration between the robots (e.g. formation maintenance).

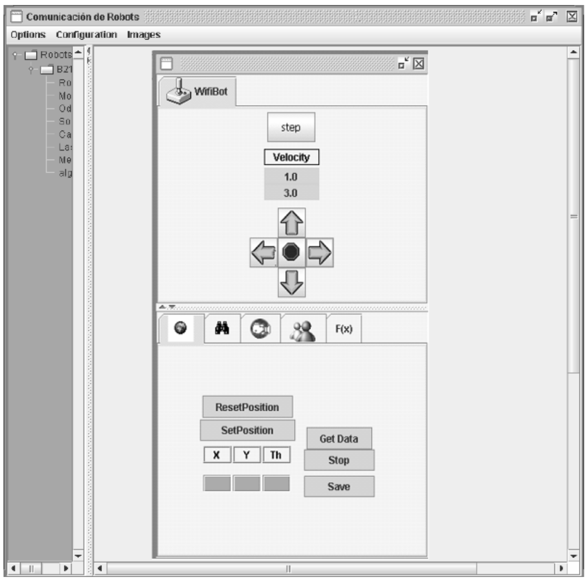

Fig.6. Interface developed for the WiFiBot.

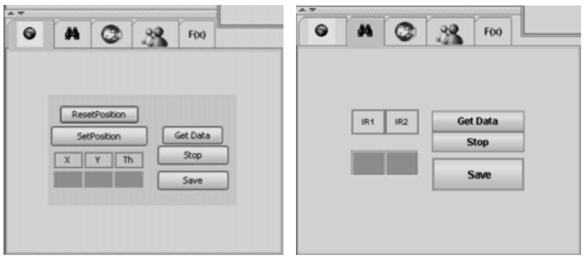

Fig.7. Panels for the monitoring of the WiFiBot. (a) Position and (b) Infrared sensors.

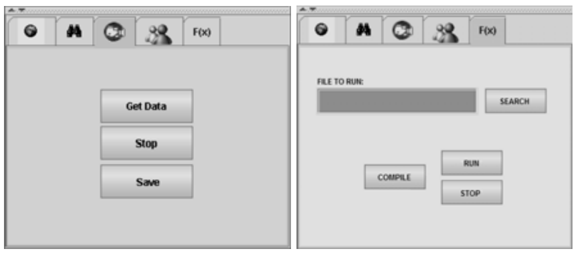

Fig.8. Panels for monitoring and control of the WiFiBot. (a) Image capture from the camera. (b) Download of a new control program to the robot.

5. CONCLUSIONS

A distributed platform for the communication with and between the members of a team of heterogeneous robots has been presented.

This platform allows the monitoring and control of these robots, in a remote way, through Internet and in a transparent way for the user, who manages with all robots using the same graphical interface. Besides, it possibilities to carry out tasks that imply collaboration between robots, what needs sharing information between the members of the team.

The system implemented is currently being used for the accomplishment of practices of mobile robotics by engineering students. The students can test the algorithms they design over the real equipments and can access all the services the robots offer, taking profit of the advantages the education through Internet has.

As an example, the applications developed on one of the robots (*WiFiBot*) are deeply described. Similar applications have been developed on the rest of the robots, and they should be defined on all the robots that we want to belong to the platform so that the user can monitor and control them in a transparent way.

ACKNOWLEDGEMENTS

This work has been supported by Ministerio de Educación y Ciencia through project DPI2004-07433-C02-01 ‘Herramientas de Teleoperación Colaborativa. Aplicación al Control Cooperativo de Robots’.

REFERENCES

Berger, K. A. and Topol, M. T. (2001). Technology to enhance learning. Use of a web site platform in traditional classes and distance learning. *Marketing education review.*, **vol. 11**. pp 15-26

Candelas, F. A., Puente, S. R., Torres, F., Ortiz, F. Gil, P., Pomares, J. (2003). A virtual laboratory for teaching robotics. *International Journal of Engineering Education*, **vol. 19, No 3**. pp. 363-370

Casini, M., Prattichizzo, D., Vicino, A. (2003). The automatic control telelab: a user-friendly interface for distance learning. *IEEE Transactions on Education.* **vol. 46, No 2**. pp. 252-257

Jiménez, L. M., Puerto, R., Reinoso, O., Fernández, C., Ñeco, R. (2005). RECOLAB: Laboratorio remoto de control utilizando Matlab y Simulink. *Revista Iberoamericana de Automática e Informática Industrial,* **vol. 2, No 2**. pp 64-72

Khamis, A., Rivero, D. M., Rodríguez, F., Salichs, M. (2003). Pattern-based architecture for building mobile robotics remote laboratories. *Proceedings of the IEEE International Conference on Robotics and Automation,* **vol. 3,** pp 3284-3289

OMG, Object Management Group. (1995). Common Object Request Broker: Architecture and Specification. Revision 2.0.

Pastor, R., Sánchez, J., Dormido, S. (2003). A XMLFramework for the Development of Web-based Laboratories focused on Control Systems Education. *International Journal of Engineering Education,* **vol. 19, No 3**. pp. 445-454

Siegwart, R. and Saucy, P. (1999). Interacting mobile robots on the Web. *Workshop Proceedings of the IEEE International Conference on Robotics and Automation*

Stafford, R. F.. (2005). Understanding motivations for Internet use in distance eduction. *IEEE Transactions on Education,* **vol. 49, No 2**. pp. 301-306

Thamma, R., Huang, L. H., Lou, S. J., Diez, C. R. (2004). Controlling robot through Internet using Java. *Journal of Industrial Technology,* **vol. 20, No 3**

Utz, H., Stulp, F., Moeld, A. (2004). Sharing belief in teams of heterogeneous robot. *Proceedings of RoboCup-2004 symposium,* pp. 508-515