Proceedings of the International Conference on Manufacturing Systems – ICMaS
Vol. 4, 2009, ISSN 1842-3183

University POLITEHNICA of Bucharest, Machine and Manufacturing Systems Department
Bucharest, Romania

SOFTWARE AND HARDWARE ARCHITECTURES OF THE INTERNET-BASED ROBOTIC SYSTEMS FOR REMOTE OPERATION

Alexandru DORIN, Adrian Florin NICOLESCU, Stelian POPA

Abstract: *The virtual reality applications are spread in many fields and this technology becomes increasingly used. This paper accomplish a survey between different software and hardware solutions developed by different universities which allow to the human operator to simulate in virtual environment the work of a real industrial robot and carry out operations with the real robot through remote operation.*

Key words: *teleoperation, telerobotics, virtual reality, remote laboratory, Internet.*

1. INTRODUCTION

In virtual reality the user can interact with a computer simulated environment. This environment can be a simulation of a real world, for example, a solder in the combat training field, or it can differ dramatically from reality, as in a computer simulation games. The impact of this technology is growing and the fields in which virtual reality is used are becoming more and more diversified. In robotics the workspace of an industrial robot can be transposed in virtual environment and the industrial robot can be teleoperated. Some of the benefits of working primary with the virtual environment are: lower costs, lower time consumption and safety [1].

Using the Internet, it may be remote operated too the industrial robots in their working environment, while we are standing very comfortable in our houses or from the offices, far away from the physical location of the robots. The process is working like this:

- from a personal computer it is accessed a web server over the Internet, using a web browser;
- the web server respond and the web browser is loading a graphical user interface (usually a JAVA applet) which have at least two parts: a simulation of the robot and its work environment and a virtual teach pendant;
- using the virtual teach pendant we control the moves of the industrial robot by sending commands to the web server, which communicate with the robot controller (Java applet web browser > web server > application server > robot controller > robot). The feedback is send vice versa (robot > robot controller > application server > web server > web browser > Java applet) and the graphical interface is updating; a visual feedback from digital filming cameras can also be used;

For the moment, the widest practicability is found in a concept named e-learning. Using teleoperation, the students can test their theoretically knowledge while sharing the same laboratory. A summary of such a scenario is described in the following.

The student opens a web browser and in the address bar taps the internet address of the web server (for example http://address.com or http://192.222.111.3). The server responds and the web browser is loading a page in which the student has to login (that's because the access is not granted for every visitor and only the qualified persons have access to it (i.e. students, teachers, administrators etc); after login, a graphical user interface (GUI) is loaded into browser. The student introduces a series of commands (or program) and then sends it to the web server which sends it to the application server; in here the commands are compiled and tested to see if these are legal moves that the robot can make; if every thing is ok, the student get a message and he can save his work or he can ask to test his program on the real robot. To test it on the real robot, he will receive a limited period of time.

2. RELATED WORK

2.1. Robolab

At University of Alicante, Spain, it was developed a flexible virtual and remote laboratory for teaching Robotics named Robolab. This system is used for e-learning and it permits the students to work with a simulation of and industrial robot. Robolab is more flexible then other systems because it offers the ability of managing different robots and it can include new robot models and equipment or other kinds of passive objects in the workspace, without the need of changing the user-interface and the system's architecture [2, 6, 7].

Except the industrial robot and its controller, all the other components from Robolab can be bought from commerce and it doesn't require a big investment. Figure 1 shows a representation of hardware components included in the Robolab. The main server is a personal computer (PC) and has multiple roles: it is acting like web server, manage the user's accounts and access. The teleoperation servers are other PCs that compile user's imputed commands from client (a standard PC) and then send it to the robot's controllers. They also receive a feed-back. The video server streams a real visual feedback to the user [2, 6, 7].

To access and operate the Robolab (Fig. 2) the user need a personal computer connected to the Internet, a web browser, Java and Java 3D runtime libraries in-

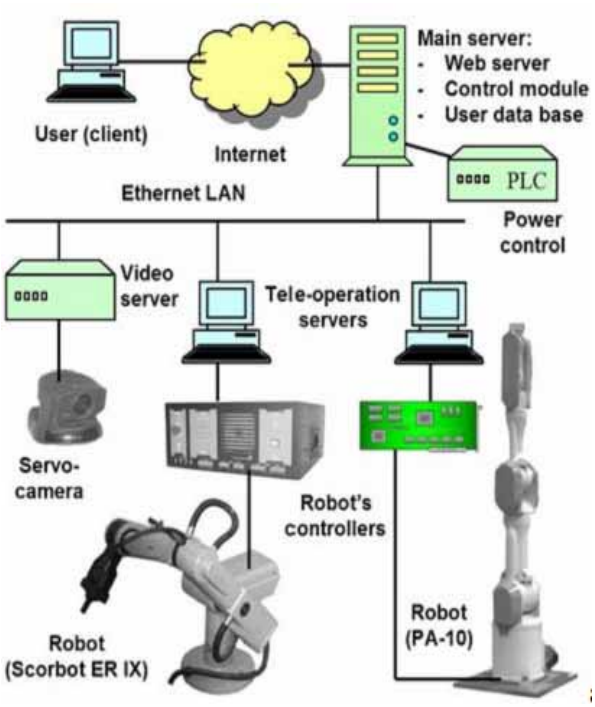


Fig. 1. Equipment architecture of Robolab system.

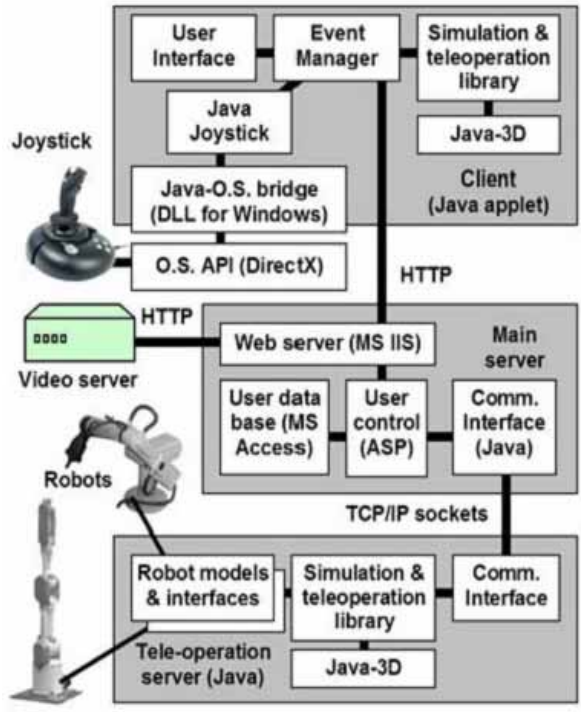
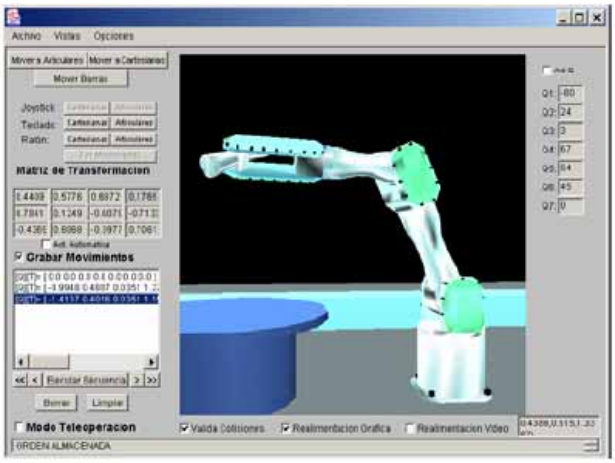


Fig. 2. Software architecture of Robolab system.

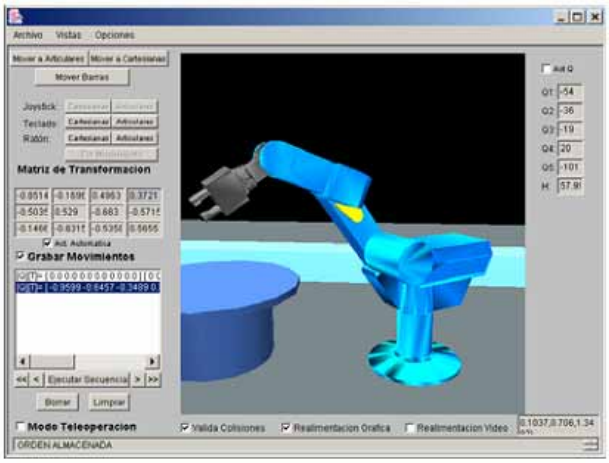
stalled. Using the web browser, the user (a student) access the “main server” and the page containing the Java client-applet, with the graphical user interface, is loaded (Fig 3) [2, 6, 7].

After accessing the "main server" the Java client-applet is loaded into the browser and he sees the graphical user interface. The graphical interface has more parts, every one of them displaying different information.

The transformation matrix with the position and the orientation of the end tool, the history of the entered commands, the values of the robot joints and other con-



a



b

Fig. 3. View of the user interface of Robolab.

trols are displayed in the left part, while in the center part of the interface appears a simulation of a Mitsubitshi PA-10 robot (Fig. 3.a) and a Scorbot ER-IX robot (Intelitek) (Fig. 3b) [2, 6, 7].

In this interface the user can load different scenarios and other objects from a local file.

To control the moves of the robot, the user can use a keyboard, a mouse or a commercial joystick for games. A force-feedback joystick can be used but this kind of joysticks, made specifically for Robotics, is an expensive device so the authors decided to improve the user interface with the ability to use regular, game joysticks. A bridge between Java library and Microsoft DirectX API is made, because Java itself can’t access the joystick functions directly (Fig 2) [2, 6, 7].

In case of collision or contact of the robot arm, the system can transmit this information to the joystick and the user feels a sensation of resistance. The Robolab system uses two sources of information for tracking contacts and collisions:

- a force sensor which can be placed into the robot end tool;
- the simulation engine from the Java applet;

The feed-back is performed in two ways:

- a online video stream from the robot workspace;
- a continuously updated virtual representation based on information received from the teleoperation server.

The advantage of the second option is the need of a lower bandwidth, this solution being more suited with slow Internet connections.

The main feature included in the Robolab system is the flexibility in changing the robot model used in the simulation or in adding new robots to be teleoperated in the laboratory. The library of classes created for modelling robots, which is based on Java 3D, facilitates the specification of new robot models and their inclusion in the system. In addition, the user interface of Robolab is very user-friendly, and the graphical simulation very realistic [2, 6, 7].

2.2. Internet-based Robotic System for remote operation

Internet-based Robotic Systems for Teleoperation developed at University of Essex, United Kingdom, combines the network technology with intelligent mobile robots. To achieve fully autonomous operation, the developed system will use cooperative learning control. Figure 4 shows the system configuration of the proposed cooperative Internet robots in which the agent-based approach is adopted. More specifically, console agents are resident in the client site, a supervisory agent runs in the server site, and a number of coordinating agents are embedded into individual mobile robots [3].

The main focuses for the performed research was the realization of some of the following features:

- a uniform interface for easy integration of different robots into the system's framework;
- an intuitive user interface and adequate feedback;
- a low-cost and easily extendable system for the addition of more complex functionality.
- cooperative behaviours to implement complex tasks that can not be implemented by single robot;
- a high degree of local intelligence to deal with the problems caused by low bandwidth and transmission delay of the Internet.

The authors at the first stage have had as main issue to design and build a basic telerobotic system framework, i.e. a test-bed for testing theories and ideas on the teleoperated mobile robots. In this way, the Internet users, such as researchers and students, could control the mobile robots to explore the Laboratory remotely.

The configuration of the current system hardware is shown in Fig. 5 [3]. The host computer communicates with the mobile robot via a radio modem connected to a serial port. The video signal is captured by the frame

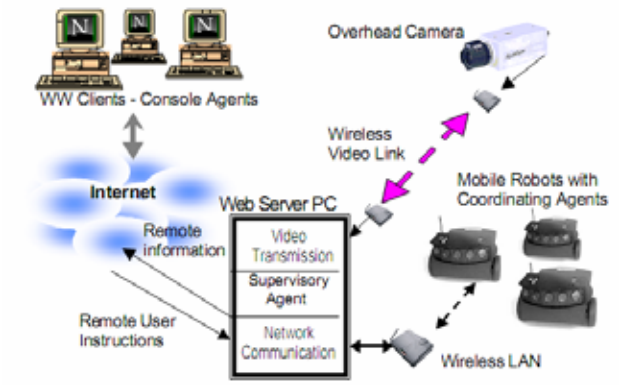


Fig. 4. System Configuration of Cooperative Internet Robots.

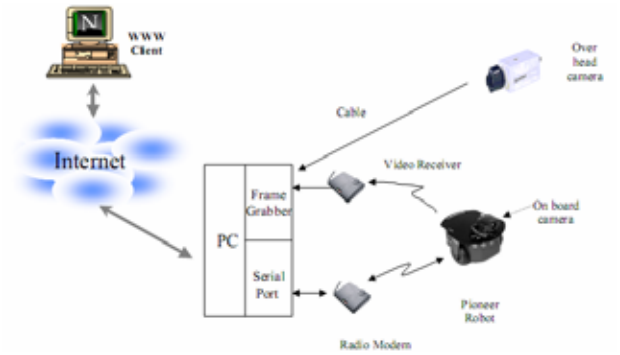


Fig. 5. System Configuration of Internet based Robotic System.

grabber that is based on the bt848 chipset. The host computer is connected to the network with a standard Ethernet card.

The Pioneer mobile robot produced by ActivMedia is powered by two reversible DC motors coupled to two wheels with a diameter of five inches (12.5 cm) and equipped with eight ultrasonic sensors in which one is on each side and other sensors are forward facing. The data produced by these sensors is used to build a global map of the robot's environment, which is displayed at the client site. An on-board camera, connected to the server though the video transceiver, is placed on the front-top of the mobile robot in order to give the user a clear view of the environment in front of the robot. Another overhead camera is available to feedback a global view of the test site to the remote user.

The web server program used is Apache HTTP web server working on the Windows 98 platform [4]. The whole system consists of several independent modules for custom service, and each of them includes a server-side program and client-side applets. These modules are the robot control module, the visual feedback module and the virtual representation module. The Java Servlet in the web server (Apache) handles the normal communication between the clients and server, as shown in Fig. 6 [3].

The robot control module commands the mobile robot. The control program of the Pioneer robots is implemented in C++, so it is necessary to build an interface to the Java program. Therefore, JNI (Java native Interface) is used to interface a DLL (Dynamic Link Library) file implemented with C++. At this stage, into the modules, no intelligence was integrated and only and it includes only some basic motion commands like: forward, recede, change speed and direction.

Once the system starts, the Java program runs continuously while it receive commands sent from the client and controls the movement of the mobile robot using a radio modem connected to the serial port. The control of

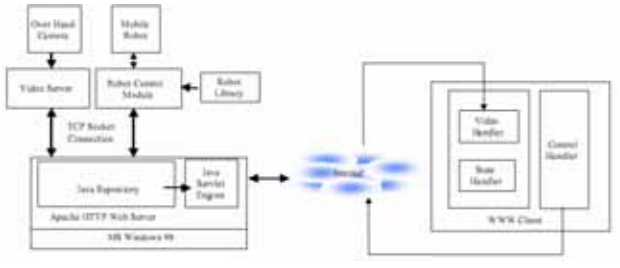


Fig. 6. Software structure.

the robot is made by only one user at a time, while the other users wait in a queue until the current operations are finished.

The Java program will feed back the robot information and the sonar readings to the clients every 100ms. In order to reduce the transmission time, all information was combined to form a string shown in Figure 6, and sent to all the clients connected to the server. This string will be interpreted at the client side to display an environment map and the necessary robot status [3].

The continuous and steady image stream feedback from the robot site is necessary when the Internet users control the mobile robot at the client site. Moreover, the image quality should be good enough to provide as much information as possible about the remote site.

Most other projects use server push technology, where the video was made up from a stream of still images, and sent by a Java program to a Java applet via a socket, and interpreted by the applet in either GIF or JPEG format. In this system, the images are captured from the frame grabber based on the bt848 chipset and compressed to JPEG format by software implemented in C++.

Then, these images are sent from the image server to the web server through a socket. The Java program streams these JPEG images to all the clients that are connected to this web server at a fixed interval. On the client side, the Java applet will recreate the image when it receives an entire frame and displays it.

The user interface is designed with the intention of making it easy for researchers and students to interact with the mobile robot. A simple interface is designed to provide as much information as possible for teleoperation. This user interface consists of several Java applets. It can work on any web browser that supports Java. On-line instructions are supplied with this console.

The control panel is made up of four direction buttons initially. The user can directly control the mobile robot by clicking the direction button on the control panel or by using the keyboard for fast and complex control such as change of speed or set fixed speed. The image display applet shows the visual feedback in a continuous jpeg image with 280x210 pixels at 24-bit colour depth. The virtual environment map applet displays some basic information about the mobile robot and the test site by analyzing the data feedback from the mobile robot. The user can find the obstacles near the robot and the trajectory, the current position and the speed of the mobile robot. With this simple user interface, one user can control the movement of the mobile robot from the web browser with the visual feed back and a virtual representation map. The other users only have the visual feedback and a virtual map at the same time, and have to wait in queue until the first user logout at this stage [3].

2.3. An Experiment in Internet-Based, Human-Assisted Robotics

At the Department of Electrical Engineering and Computer Science from Case Western Reserve University in Cleveland, Ohio was constructed a robotic test facility for evaluating the prospect of internet-based supervisory control of semi-autonomous systems. The system consisted of a robotic arm, two cameras, a PC controller, and a Web server.

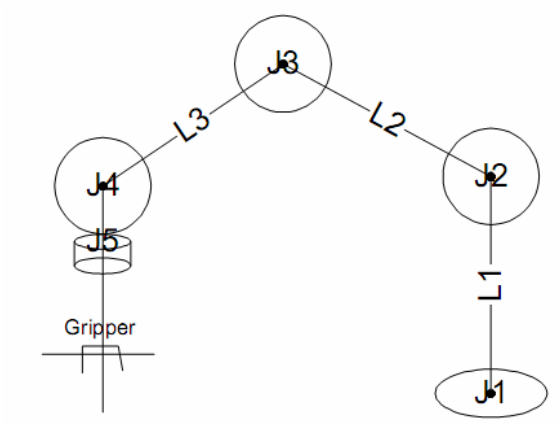


Fig. 7. Robot Kinematics.

Fig. 8. Robot's side view.

The robot used in the experiment was a low-cost educational robot that had been retrofit for open-architecture control. While this robot had limited workspace, payload, speed, and precision, this choice was attractive in terms of safety, which is a significant consideration in remote control. The robot had five degrees of freedom in a serial kinematic chain (Fig 7), similar to popular industrial designs (Fig. 8) [5].

The robot was interfaced at the torque level to an analog output board within a PC control computer. Incremental encoders on the joints were interfaced to encoder counters within the I/O card hosted by the PC. The PC had a Pentium 133 processor, which was relatively slow, but adequate for this investigation.

The actuators consisted of the joint motors and the robot's gripper. The sensors included motor encoders, an optical distance sensor, a gripper-mounted black-and-white camera, and a color Web camera.

The control software on the PC ran within the environment of the QNX operating system, a real-time operating system derived from Unix and commonly employed on x86 machines. QNX supports real-time multitasking; i.e., multiple processes can be prioritized and scheduled to run independently, emulating parallel execution. Multi-tasking was employed in our system to run concurrently several control and communication proc-

esses. Communications among the separate processes is coordinated through semaphores. Semaphores were used to drive processes at fixed rates and as a mutual exclusion mechanism. We used an existing priority-driven real-time infrastructure that divided processes into low and high levels. The low-level processes were simple but required execution at high frequency and at high priority. In contrast, the high-level processes required more computation but did not require frequent execution. The controller software ran on the QNX PC, was written in C, and was compiled with the Watcom compiler that is standard in QNX 4.

The experimental system used an Apache Web server to run as the front-end for user interaction. In turn, the Web server communicated with the QNX controller and relayed user commands to the robot. There is no version of the Apache server for the QNX operating system, and so we installed the Web server on a separate PC running Windows NT. The NT machine connected to the robot controller through Ethernet. In general, the Web server and robot controller could be installed on the same or on different computers. A single-machine installation is characterized by reduced hardware requirements and by fast communication between the Web server and the controller. The separation of front- and back-end hosts can result in legacy with existing platforms and can lead to higher system scalability and security.

The Web server initiated robot operation by invoking a CGI script that ran at the Web server side. In turn, the script embedded TCP connectivity to relay instructions to the robot. Furthermore, the scripts gathered feedback from the robot and passed it on to the user. As a result, CGI scripts provided a module to communicate and pass information back and forth between the server and the controller. CGI scripts were exposed to remote users by creating a form within an HTML document and inserting the script URI as the form action.

CGI is an acronym that stands for "Common Gateway Interface" and refers to the protocol used to pass arguments from the Web server to the script. Internally, the script could be written in any programming language but, in practice, CGI scripts are typically written in Perl.

The Apache server had a handler for running CGI scripts as threads that are part of the Web server process: when the server invoked a CGI script, a new thread was created within the Web server process to execute the script. An earlier version of the system did not use such a module and was running CGI scripts as separate processes. The robot reacted very slowly to users' commands due to the overhead for forking additional processes to execute the scripts. When CGI scripts were wrapped in the same server process, the robot reaction time improved dramatically.

The Web interface was written in HTML. The interface had feedback buttons for a user to transmit commands to the Web. The commands would be relayed to the robot controller for execution. Radio buttons, drop down menu, and normal buttons were used. The radio buttons forced the user to select from a list of valid choices so that the return string would not be an empty string. The drop down menu stored choices for the users.

Image-based servo control were accomplished through the actual image from the camera. The frame grabber took a snapshot of the robot's view and the Web server encoded the saved image as part of the HTML display. The image was a black and white JPEG picture and it projected the robot's point of view. Such pictures were used as an input means. The user could click on the actual picture on the Internet browser, and this action would invoke recording the actual pixel X and Y coordinates.

These coordinates were then delivered to the Web server and a CGI script relayed the data from the web server to the robot controller. Image-space coordinates were translated into robot-space coordinates, transparent to the operator, to command the robot to move to the selected location.

The buttons and the servo control were the input elements; however the user also needed feedback from the system. There were two forms of feedback: images from the robot's viewpoint (with the gripper-mounted camera) and a wider-angle side view from a Web camera (a WebCam). The first camera was mounted on the robot's arm and its frame was refreshed only after the robot arm completed a movement. In practice, the frame was refreshed at the behest of a CGI script spawned by the server when the arm movement completed. Since the picture was delivered only at the end of the arm movement, the user would see a "busy" icon during the movement. A second camera was mounted near the robot with a side view of the robot and its workspace and provided a real-time view of the robot environment. The camera used the Webcam32 surveyor software that acted as a real-time streaming video server to capture and deliver the images to a remote user in real time. Webcam32 was a separate server that worked independently from the Apache server [4], but ran on the same NT machine as the Web server. To display the video stream, the client used a Java applet, i.e., a Java byte code executable that was dynamically downloaded into a browser over the Internet. The applet continuously downloaded video streams from Webcam32, displaying the video at a rate of about 1 frame per second.

The human supervisor interacted exclusively with the Apache Web server and with the Webcam32 video server. In practice, the operator could use any of the commercially available Web browsers to supervise the robot and download video streams. When the user clicked on a button or selected an option from a drop-down menu, the user choice was sent to the Web server in a standard HTTP request. The server would then trigger a CGI script that communicated with the robot controller through a TCP connection. Meanwhile, the Webcam32 server would continuously push video frames to the Java applet running on a Java Virtual Machine within the browser, presenting the video feed to the user.

The client site used widespread off-the-shelf components, such as Java-enabled Web browsers, so that the human interface to the robot required no specialized software or hardware beyond what is already commonly available.

Furthermore, the use of standard components shifted the software design process from the internals to the interface between humans and machines [5].

3. ACTUAL DEVELOPMENT STATUS OF OWN WORKS

3.1. SCARA robot for wafer manipulation virtual prototyping and programming [8]

For wafer manipulation in atmospherical environment, a SCARA-type robotic manipulator with four degrees of freedom has been developed, first as a virtual prototype using the CATIA software. Figure 9 illustrates a perspective view of the designed atmospherically operating robot with internal components exposed.

The robotic arm is sustained by a support that includes the partial assembly generating first (rotation) degree of freedom for the robotic arm system, and a vertical extending column (second degree of freedom for the robotic arm system), as shown in figure 9. The base rotation is performed through a spur gears and timing belt system and the vertical translation is performed through a ball-screw system.

The robotic arm system also includes three elements linked by rotary joints (through timing belt systems): first and second links of the articulated arm respectively the end-effector orientation system. The last element of the robotic arm system is a (vacuum operating) ceramic end-effector special dedicated for wafer manipulation.

Following virtual prototyping of the atmospherically robotic arm system, physical (real) robot operational unit has been made in cooperation with Japan and China private companies. As well, the robotic arm system information unit has been designed and its prototype manufactured in a Romanian private company (Fig. 10a,b). Finally, a software package for robot teach-in and offline programming and simulation (Figs. 11, 12, 13, and 14) has been developed in partnership with Romanian private company.

The ROBOPACK software is designed for command, control and programming of industrial robots performing wafer manipulation operation. This software package includes:

- an user friendly main graphical interface in English language (Fig. 11);
- advanced 3D simulation/visualization window (Fig. 12);
- an editor for the robot's application teach-in and off-line programming using an originally developed robot language (Fig. 13);
- a virtual teaching pad needed in teac-in command of the robot and key-points teaching (Fig. 14);

Fig. 9 – SCARA-type robot designed for wafer manipulation in atmospheric environment.

a b

c

Fig. 10. The robotic arm system: a) atmospherical operating robot unit; b) robot informational unit (the dedicated robotic controller); c) full system (including the notebook running the simulation and programming software).

Fig. 11. The main graphical interface.

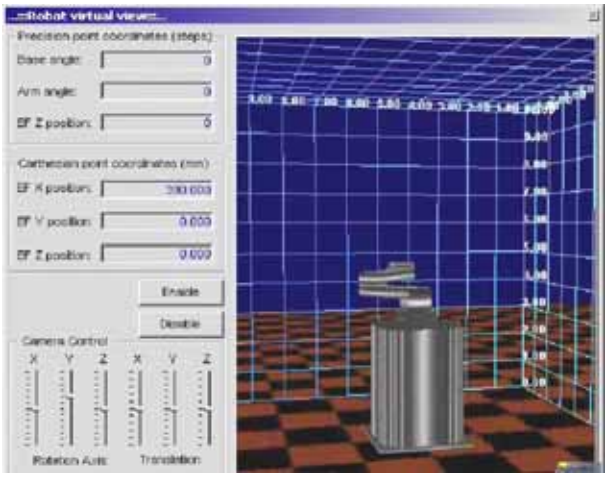


Fig. 12. 3D simulation/visualization window.

The virtual teaching pad can directly control the robot's movements, being able to command movement on each axis of the robot, to set movement direction, movement speed and acceleration values, to memorize key-points, etc.

During manual control and teach-in, the visual interface of virtual teaching pad (Fig. 14) works with real information supplied by the position transducers mounted on each robot axis.

Similarly, the advanced 3D visualization / simulation software (Fig. 12) is able to realize the virtual simulation of a program based on:

- programming information for the case of offline programming and simulation procedures, as well as
- real robotic system information supplied by the position transducers mounted on each robot axis, in case of robot manual control, teach-in or real task / program executions.

As result, the overall programming and simulation software allows a real time visualization of robot movements in the two simulation procedures and real task / program execution.

The programming language is useful for both robot teach-in programming and offline programming. It includes user friendly instructions and commands, usually to be selected from predefined pull-down menu, and inserted directly to the program, and as well the capability to directly edit specific commands / instructions / introducing numerical values by direct typing.

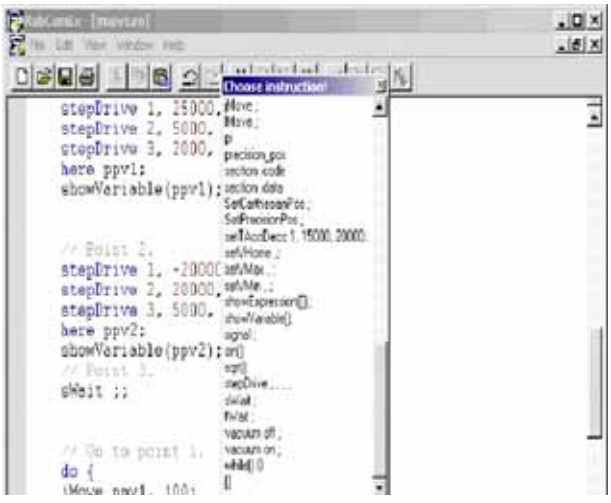


Fig. 13. The editor for the robot's application programming.

Fig. 14. The virtual teaching pad.

For complex programming task the advanced 3D visualization / simulation software can be easily configured for including in the work scene peripheral equipment or different other objects that need to be correlated with robot functionality.

4. CONCLUSIONS

Based on the analysis previously made, we noticed that all the remote system have some similarities.

The common hardware structure is composed by a remote computer, a main server, the robot with its controller and cameras used for visual feed-back.

The main server can have multiple roles: web server, controller module, database server and stream server. The software installed is open source: Apache for web server, MySQL for database management, Java for graphical user interface.

Own works achievement showed in section 3 of the paper illustrate good preliminary approach on the way to start developing a fully remote robotic operating system (in terms of specifically developed virtual environment for robot off line simulation as well as own programming language and software modules).

Considering all above, the hardware and software architecture for a completely remote laboratory operation intended to be developed by ourselves may described by below mentioned features.

Hardware system may include a remote computer, running Windows or UNIX based operating system.

The only things this computer needs is a web browser capable of running Java applets, Java platform installed and a connection to the Internet (Java applets are preferred because it is browser and operating system independent, leaving the user the ability to use its own web browser and operating system – Windows, Linux, FreeBSD etc).

The Java applet will be split in more areas. In the main area a simulation of the entire workspace of the robot will be displayed while in the left part will be displayed a virtual teach pendant. The transformation matrix

with the position and the orientation of the end tool appears also in the left side.

The simulation can be displayed in to modes: a test simulation, based on user's inputs from the virtual teach pendant or a real simulation of the robot based on the feed-back received from the robot controller and the sensors from the laboratory. In the second case, it is very important to update with accurate data. On slow internet connections this could be a very big problem and to solve it we shell integrate an application to test the internet speed.

Using HTTP (Hypertext Transfer Protocol) protocol the browser communicates with the web server located in the laboratory. On the web server is running Apache HTTP Server which permits communication on port 80. The Apache server has another role: it makes the connection between requests received from the remote computer and the controller module which communicate directly with the robot controller. The controller module will be developed using C++ programming language and it will run as an Apache module. The controller module also tests the imputed commands or programs sent by the user and in case the commands are illegal. If everything is ok, the user can save the work as a scenario or he can upload the commands to the robot controller (if the robot is free of job it will immediately execute the commands, otherwise the controller module will put the execution on a waiting queue).

A visual feed-back is also needed, so a couple of cameras will capture images from the laboratory and will transmit it to the stream video server.

The access to the main servers will be granted bases on a username and a password and multiple levels of accounts will be available. Only the system administrators are able to create users accounts. All this information and the user's sessions with their logs and command histories will be stored using the MySQL database engine.

To increase the security of the whole site, a firewall will be used, so the connection to the server will be allowed from certain IP's classes. In the start, a single PC can play many rolls: web server, database server, firewall, and stream server and controller module. In a production environment it is best to separate the rolls, every roll running on its own computer, except the web server and the controller module (to decrease the application latency).

REFERENCES

[1] *** http://en.wikipedia.org/wiki/Virtual_reality

[2] Candelas, F.A. et al. (2006). *Flexible virtual and remote laboratory for teaching Robotics*, FORMATEX, Badajoz, Spain.

[3] Huosheng Hu, Lixiang Yu, Pui Wo Tsui, Quan Zhou. *Internet-based Robotic Systems for Teleoperation*, International Journal of Assembly Automation, Vol. 21, No. 2.

[4] *** A*pache web Server* http://www.apache.org

[5] Ngai, L., Newman, W, Liberatore, V.*An Experiment in Internet-Based, Human-Assisted Robotics*.

[6] Candelas, F. A., Torres, F., Puente, S., Pomares, J., Segarra, V., Navarrete, J. (2004). *A Flexible Java Class Library for Simulating and Teleoperating Robots*, Proc. 11th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2004), Salvador de Bahía, Brasil, 5–7 Abril.

[7] Torres, F., Candelas, F.A., Puente, S.T., Pomares, J., Gil, P., Ortiz, F.G. (2006). *Experiences with Virtual Environment and Remote Laboratory for Teaching and Learning Robotics at the University of Alicante*, International Journal of Engineering Education (Special Issue on Robotics Education), 22, 4, 766.

[8] Nicolescu, A.F., Enciu, G., Ivan, A., Avram, G.C., Marinescu, D.A. (2009). *Wafer manipulating robots – design, programming and simulation*, 2nd WSEAS International Conference on Visualization, Imaging and Simulation (VIS'09), Baltimore, USA, November 7–9.

Authors:

PhD, Eng, Alexandru DORIN, Professor, University "Politehnica" of Bucharest, Department of Machines and Manufacturing Systems,

PhD, Eng, Adrian Florin NICOLESCU, Professor, "Politehnica" University of Bucharest, Department of Machines and Manufacturing Systems,

E-mail: afnicolescu@yahoo.com,

Eng, Stelian POPA, PhD Student, "Politehnica" University of Bucharest, Department of Machines and Manufacturing Systems