Control and robotics remote laboratory for engineering education

R. Šafarič, M. Truntič, D. Hercog and G. Pačnik

University of Maribor, Faculty of electrical engineering and computer science, Maribor, Slovenia

***Abstract*—The new tools for education of engineering emerged and one of the most promising is a remote rapid control prototyping (RRCP), which is very useful also for control and robotics development in industry and in education. Examples of introductory remote control and simple robotics courses with integrated hands on experiments are presented in the paper. The aim of integration of remote hands on experiments into control and/or robotics course is to minimize the gap between the theory and practice to teach students the use of RRCP and to decrease the education costs. Developed RRCP experiments are based on MATLAB/Simulink, xPC target, custom developed embedded target for DSP-2 controller and LabVIEW virtual instrument.**

***Index Terms*—Education of control and robotics, rapid prototyping, remote engineering**

I. INTRODUCTION

Recently much attention has been focused on modern control education in Engineering. A leading idea to all educators was given in [1]: "*Educators must have an open attitude towards new technologies. They should sensibly incorporate new technological development to avoid the risk of teaching the students of today, how to solve the problems of tomorrow, with the tools from yesterday.*" Nowadays, two new tools: the web (Internet) and the rapid control prototyping have a great impact on control and robotics systems used in industry and education. The Web influences the industry because it enables supervision and teleoperation of devices (cost reduction). The rapid control prototyping (RCP) influences the industry because it saves time needed for a development of control approaches for different devices and therefore reduces control development costs for 30 to 40%.

Both mentioned tools also have a great impact on control and robotics education. The impact of the Web was extensively discussed in [1]. Different designs of Web based control labs are presented in [2]. The Web enables more flexible delivery of teaching materials, distance education, new visualization possibilities, interactivity and cost reduction. The impact of RCP is more limited to control implementation. RCP frees control implementation of particular implementation details, like coding the controller algorithms in C language for computers, and therefore speeds up implementation of control approaches. This performance makes RCP suitable for hands on experimental learning of control.

In the paper, an example of a simple remote DC-motor controller for teaching the basics of control for students of Mechatronics specialization and more complicated 6 D.o.F robot arm controller for teaching of basics of robotics with integrated remote web based and a real hands on intelligent teach pendant for robotics arm remote experiments are presented. We called it a remote rapid control prototyping. The concept with the remote computer, internet, an executive server and the lab is shown in the Fig. 1. The integrated experiments are structural remote lab exercises and are not a replacement for laboratory exercises, nor the project based learning of students.

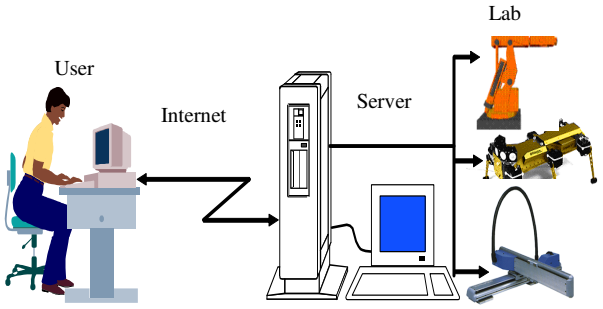

Figure 1. The concept of RRCP in the educational process

One of the first successful web based robotic projects used predominantly for educational purposes were the Mercury project [3], Telegarden project [4] and The university of Western Australia's Telerobot experiment [5] where users are required to manipulate objects in the workspace of the robot arm. The users view of the work-cell space is limited to a sequence of static images captured by cameras located around the workspace. Problems with the static picture can be avoided by using video technology but it is clear that such an approach needs a high speed network to achieve on-line control of the robot arm. Data transmission times across the WWW depend heavily on the transient loading of the network, making direct tele-operation (e.g. the use of cameras to obtain a robot arm position feedback) unsuitable for time critical and dangerous (e.g. collision between a robot arm and the environment) interactions.

Rather than allowing the users to interact with the physical resources directly, as in the previously mentioned examples, our robotics experiment requires users to configure the experiments using a simulated

representation (e.g. a virtual robot arm and its environment) of the real-world apparatus where is also done the check of the possible collision between the robot arm and its static (predictable) environment [6]. This collision detection test is done by so called Intelligence interface between the real and the virtual world. The configuration data (only few bytes of reference position for all axis) is then downloaded to the real work-cell, for execution on the real device, before returning the results to the user once the experiment is complete (few bytes of actual position for all axis).

The virtual robot arm and environment model is used, instead of cameras, to minimize the data transmission time through the network, so network speed is no longer a critical issue and solves the problem of robot arm collision in the real world with a predictable environment.

II. TELEOPERATION OF MECHATRONIC DEVICES USING DSP-2 RCP SYSTEM

DSP-2 Rapid Control Prototyping (RCP) system is based on two well-known commercially available software packages i.e. MATLAB/Simulink and LabVIEW and a custom-made hardware i.e. embedded DSP based motor controller [7]. MATLAB, Simulink and Real-Time Workshop (RTW) are used for control algorithm development, simulation, offline analysis and rapid executable code generation. While this code is executing on the embedded DSP-2 controller and through the analog and digital I/O lines drives the real process, LabVIEW virtual instrument (VI), running on the PC, is used as a user front end. LabVIEW VI provides the ability for online parameter tuning, signal monitoring, online analysis and via *Remote Panels* technology also teleoperation.

A. *Hardware and software components*

DSP-2 roby system (Fig. 2) is composed of DSP-2 controller [8] and DSP-2 add-on robotic board. The key components of DSP-2 controller are floating point digital signal processor (DSP), used for control algorithm execution, and the Xilinx FPGA, which implements peripheral interfaces. DSP-2 roby system contains all the necessary peripheral for 4 axes robot control i.e. this system has 16 digital inputs, 8 digital outputs, 4 analog inputs/outputs and 4 incremental encoder interfaces.

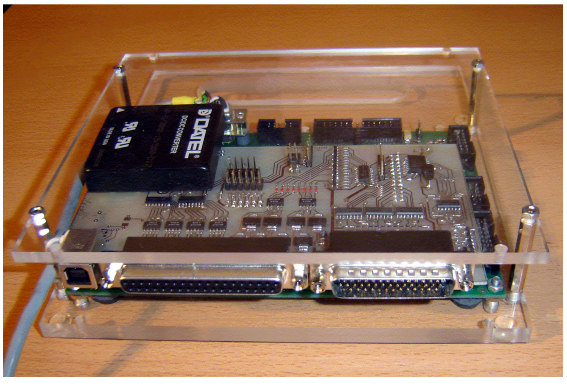

Figure 2. DSP-2 roby system

The "DSP-2 Library for Simulink" [8] is Simulink add-on toolbox, which provides rapid control prototyping (RCP) support for DSP-2 system (DSP-2 controller and DSP-2 roby system). DSP-2 library contains a set of device driver's blocks for all available I/O ports of DSP-2 system including blocks for analog I/O, digital I/O, incremental encoder and blocks for communication between PC and DSP-2 system. Simulink, Real-Time Workshop and the DSP-2 Library for Simulink enable developers to model applications in the Simulink block-diagram environment and, after successful simulation, quickly verify designed algorithm with the DSP-2 controller or DSP-2 roby system on the real mechatronic device.

In addition to rapid code generation, a LabVIEW virtual instrument (VI) named "ComVIEW" has been developed to serve on the fly data visualization and parameter tuning tasks for the DSP-2 system (DSP-2 system is connected to the PC using RS-232 serial connection). When the DSP-2 target is selected in the Simulink model, a LabVIEW virtual instrument is automatically created from the ComVIEW template VI during the binary code generation. A ComVIEW template contains an empty front panel and a fully functional block diagram. The block diagram implements functions for VI initialization, executable code download to the DSP-2 system, functions for transmitting and receiving messages between the PC and the DSP-2 system, and other functions. During VI creation, numerical controls and indicators are automatically added to the VI front panel template, where the number of controls and indicators depends on the number of DSP-2 communication blocks used in the Simulink model. Links between DSP signals and VI front panel objects are established programmatically using the DSP Connection Manager window. This window appears on the PC immediately after the downloadable binary code starts executing on the DSP-2 target. All information about the DSP-2 input signals, DSP-2 parameters and DSP-2 output signals, as well as all information about VI scalar controls and the indicators of front panel window, appear in the afore mentioned window. Using mouse clicks, the user can create links between the VI front panel indicators and the DSP-2 output signals, and links between the VI front panel controls and the DSP-2 input signals or DSP-2 parameters. When these links are set, a communication link is established between the VI running on the PC and a code executing on a DSP-2 controller. Whenever the controls on the VI front panel are changed, LabVIEW automatically downloads them via RS-232 serial connection to the DSP-2 controller. At the same time, all arrived DSP-2 output signals are read from the PC serial port and displayed in an appropriate numerical indicator. In addition, ComVIEW provides scope capabilities. In the scope mode, a small portion of code running on the DSP-2 controller handles data acquisition and storage management. The selected DSP-2 signals are, firstly, captured and then stored in the temporary controller memory. After that, the captured data is transferred to the PC.

"Remote Panels" is a LabVIEW add-on toolkit developed by National Instruments that enables viewing and controlling of LabVIEW VI's over the Internet. Using this toolkit, the LabVIEW VI can be published

on the internet with no additional programming. Afterwards, the virtual instrument can be remotely observed or controlled by using standard web browser. The remote user can fully access the user interface that appears in the web browser and consecutively has complete control of the remote application. Other users can point their web browser to the same URL to view a remote experiment. To avoid confusion, only one client can control the application at a time, but that control can pass easily among the various clients at run-time.

B. Teleoperation of mechatronic device

The hardware and software components described in the previous section provide an open framework for remote rapid control experiment development. A mechatronic set-up, composed of the DSP-2 roby system, two axes mechanical mechanism with the DC motor in each axe and the DC motors amplifier is presented on the Fig. 3.

A DSP-2 roby system, connected to lab PC through the serial port, implements a control algorithm developed using Simulink (Fig. 4) and through the analog and digital I/O signals, drives two axes mechatronic device (Fig. 3). ComVIEW VI and the LabVIEW server are run on the same lab PC for the purpose of enabling teleoperation of described system. ComVIEW VI performs communication between lab PC and the DSP-2 system, online DSP-2 signals monitoring and parameters tuning, while the LabVIEW server enables remote operation of the ComVIEW VI. Remote users, connected to the server through the internet, must have a 'LabVIEW Run-Time Engine' installed on the personal computer, in order to perform remote experiments.

During remote operation (Fig. 5), the remote user can change position reference signal (constant, sine, square), adjust PID position controller parameters for each axe, observe internal DSP-2 signals in numerical or graphical view, select which DSP-2 signals will be captured and shown in a graph and select the trigger parameters (trigger signal, trigger level, trigger slope, number of pre-samples). The remote user can also send teleoperation results via an email. The e-mail attachment contains experimental results in a format appropriate for further offline analysis in MATLAB.

III. REMOTE ROBOTICS LAB

The remote laboratory approach is based on the concept that it provides a working facility for off-line programming of actual working robot in a remote environment and hands-on training. It is desirable that the robot simulation should be capable of being executed through any standard WWW browser application, e.g. Netscape Navigator, Microsoft's Internet Explorer etc. and VRML browser, e.g. Cortona or Cosmo Player etc. Standard browsers for the VRML 97 language don't incorporate collision detection between shapes in the virtual world. The physical test equipment includes: a WWW network server, a network layer, a robot workcell, and remote user computers. Because the adopted control strategy does not provide the remote user with immediate feedback from the actual work-cell, it is desirable that some kind of collision detection between the virtual robot and the virtual environment is created to prevent, or to predict, robot collisions in the real world. This

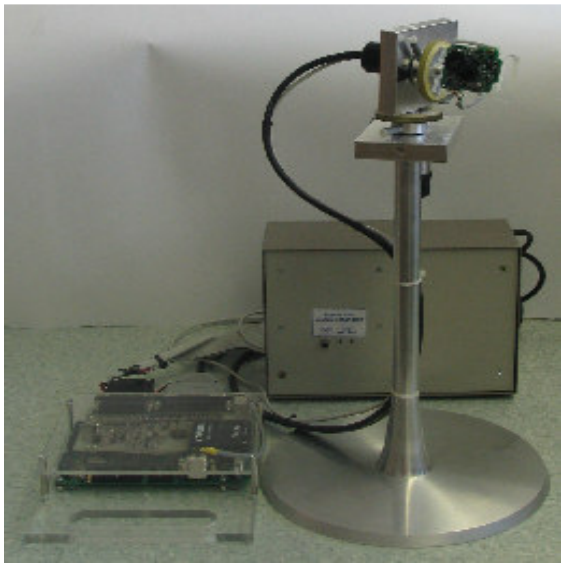

Figure 3. Mechatronic set-up

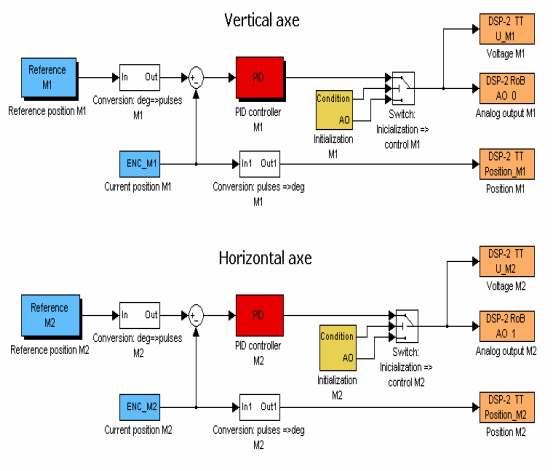

Figure 4. Control algorithm for two axes mechatronic device

problem has been solved by building JAVA oriented collision detection software [11] to assure platform independent approach. It may be solved with a use of the complete browser and collision detection software [5] in the C++ language which has capability to decrease the execution time of the complete software and to increase animation speed as was done in [4] and [5] but it loses the platform independence.

The physical test equipment includes: a WWW network server, a network layer, a robot workcell, and remote user computers (see Fig. 6).

The robot work-cell allows Point to Point (PTP) motion of the robot. The robot data and environment are constant and are set in the VR software. The remote motion data for the robot arm mechanism are programmed and controlled, respectively, by the user.

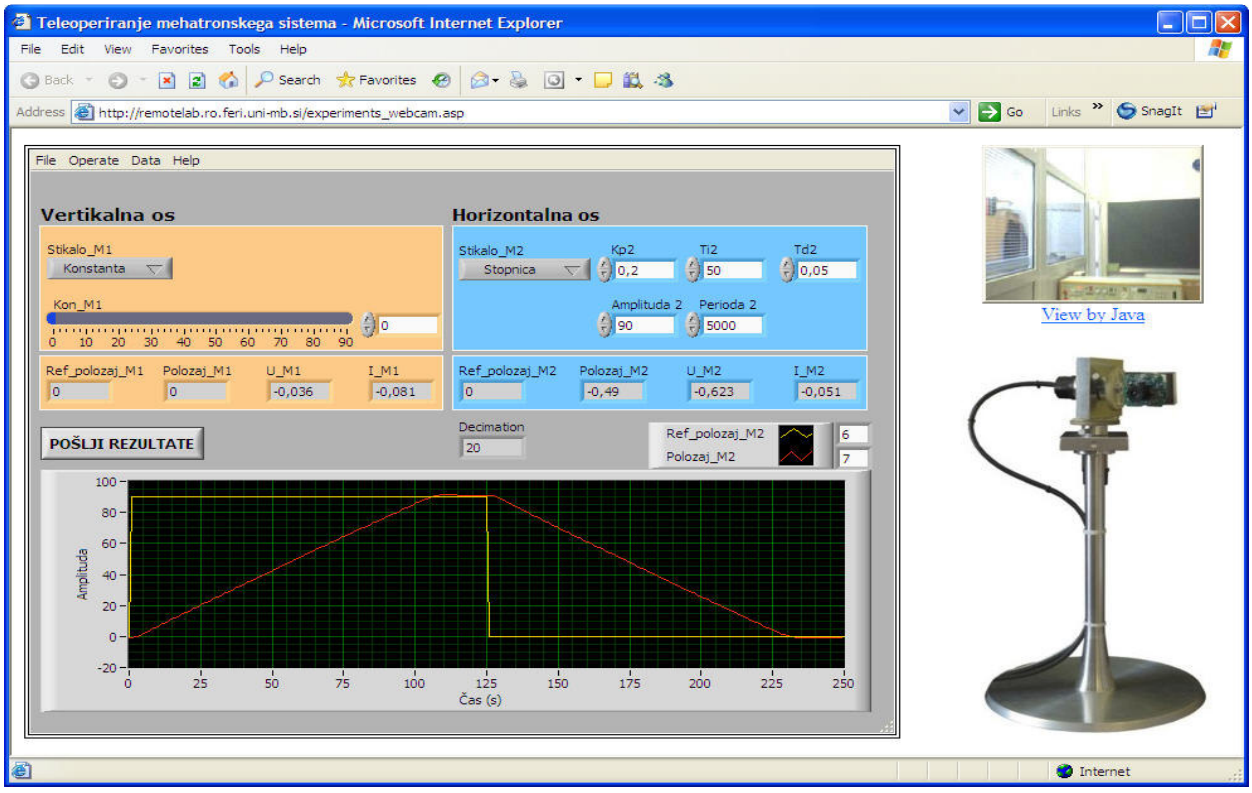

Figure 5. A remote experimental front panel in a classical web browser

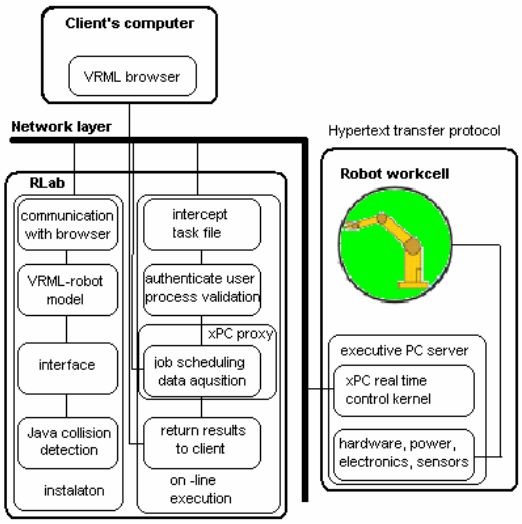

Figure 6. The robot interface between servers and a client

The work-cell, shown in Figure 7, includes the 6 axis educational robotic arm, which has P-position controllers written by MathWorks' MATLAB (ver.6.1) / Simulink / Real-Time Workshop / xPC Target fast prototyping software. The program is executed by xPC operating software on the executive PC server.

The WWW network server is responsible for processing the requests done by an external WWW

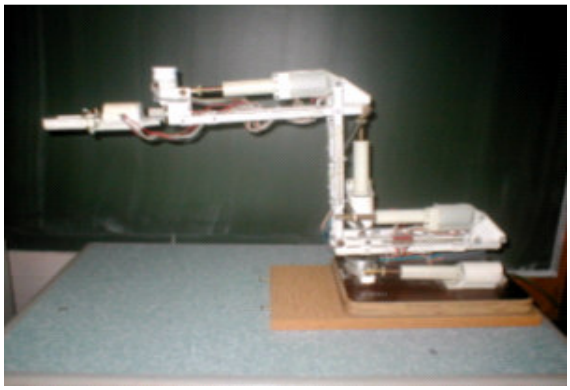

Figure 7. A robot arm

browser installed on the user's remote personal computer, delivering on-line documents and providing access to the robotic and control hardware.

IV. REMOTE RAPID ROBOT PROTOTYPING

Following the success of the earlier experiments [5] and [6], the development of an improved human computer interface, integrating the Java language and VRML language within a non-immersive desktop virtual reality environment, was undertaken in order to help improve the realism and sense of presence the user feels when programming the robot. This simulation tool allows the kinematics behaviour of the

system to be studied, and permits research into task planning, process synchronisation and the communication issues involved with the control of robotic manipulators. The additional processes which are executed by the RLab server are shown in Figure 6. Figure 10 illustrates the users view of the robot model and it's associated 'virtual' teach pendant.

The basis of a remote robot control presents a model build in MATLAB/Simulink program. As a remote use of the robot control model xPC target system is used. xPC target system is a toolbox of MATLAB/Simulink package and enables an execution of the application in an external mode.

Application of a robot control model (Fig. 8) is first created in Simulink on a host computer and downloaded on the target PC to which the robot hardware is connected. After creating the model, the user can run simulations of-line. The model must be compiled with the Real-Time Workshop and a C/C++ compiler to create executable code. The model has to be downloaded to the target PC and then run with the xPC Target real-time kernel, so the remote application can be run and tested in a real time. These applications are stand-alone applications and run on a target PC independently from the host PC. This procedure is used for rapid robot control prototyping. Functions of target applications are: transfer of application on a target PC, start and stop of application execution, change of sample and stop time, detection of processor overload and statistics of executing of the application. Application can proceed on several host computers and can be transferred on one target PC for a real-time execution on a remote controlled system. The remote running application can be accessed over MATLAB command window, direct communication with a Simulink model (external model) or with target's command web page interface. The model and the parameters of the remote application can be modified or updated with the use of MATLAB. Only the parameters of running application can be altered with the other two access point. The MATLAB interface contains functions for adjusting the parameters on target PC. By changing one or more parameters in Simulink, these new values are transferred to the target PC immediately and we can observe response of the application.

The command web page interface (Fig. 9) enables simple acquiring data from executive (target) PC server. We can execute signal logging, so that all data are stored from whole time of application execution. Values are stored in RAM on a target PC server. After the application is stopped the files can be transferred to the host PC. Alternative to acquiring signals is tracking the signals like digital scope. Data can be presented numerically or graphically.

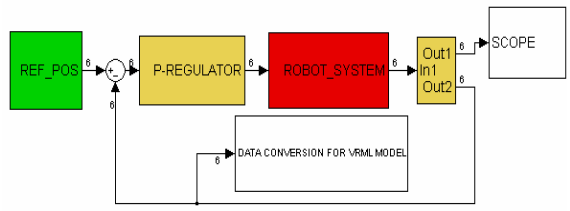

Figure 8. The MATLAB/Simulink model

V. REMOTE ROBOT CONTROL WITH RLAB

RLab is a collection of web applications, which represent a main part between a user and a robot. It is divided in four parts. RLabClient represents a user interface for guiding the robot with addition of VRML model of the robot. RLabExecutiveProxy serves for transferring data from RLabClient to the xPCproxy. On the other side of the xPCproxy is the xPC Target, on which the application for guiding the robot is loaded. Application is first created in Simulink and then it is loaded on xPC Target. xPCproxy has a web application for administrating the projects. This application enables loading, starting and deleting projects of xPC Target computer

The remote user is connected to RLab's WWW site to register the job using an on-line form. The user details are processed by a CGI program running on the server, to determine user authentication, access control and job queue status experiment. If the robot is currently in use then the users may decide to cancel their job and try again later, otherwise the file is placed in a queue for execution at a later date. In this case an acknowledgement will be returned to the user and the results stored in an on-line archive for retrieval at a later date. Real time manipulator control is achieved using a separate computer (PC) control system. However, the existing controller requires that set points for limb movement, motor drive characteristics, process status, etc. are provided as a stream of parameters from a host computer, in this case the server, to the robot's own control system, thus necessitating the development of a replacement command scheduling program written by MathWorks' MATLAB (ver.6.1) / Simulink / Real-Time Workshop / xPC Target fast prototyping software.

VI. COLLISION DETECTION IN RLAB ENVIRONMENT

In the real world, we have to pay attention to the robots' workspace, so that they do not have anything to collide into. As fast as we start controlling the robots remotely, the problem becomes even bigger, because we do not receive immediate feedback information about the collision. That is why the RLab environment has an intelligent interface with a built-in collision detection library in a virtual world of collision. Robot models in the RLab environment can be sorted into three worlds, which are closely connected. The first world is an actual remotely controlled robot. Second, the virtual – visualization world, controls the graphical side of the controlled robot world and can be used as a simulator. The third – virtual world of collision is responsible for the collision detection between the robot and the environment. All worlds together make a concluded entity of remote controller, which as precise as it can imitate and control the real robot. The cooperation of the worlds is next: there is given a command for the desired movement in the virtual - visualization world which is checked in the virtual world of collision. In the case that the desired movement is collision free it is executed in the real world.

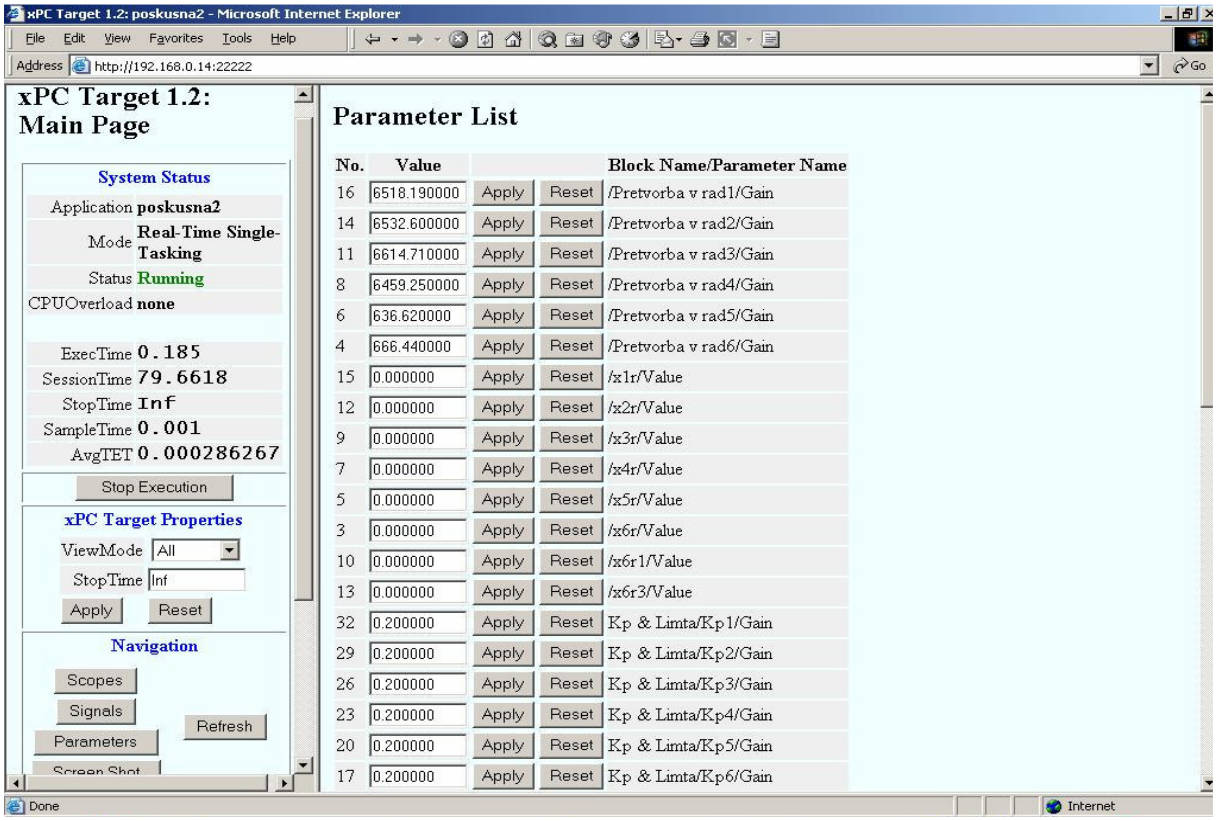

Figure 9. Web interface for controlling the application

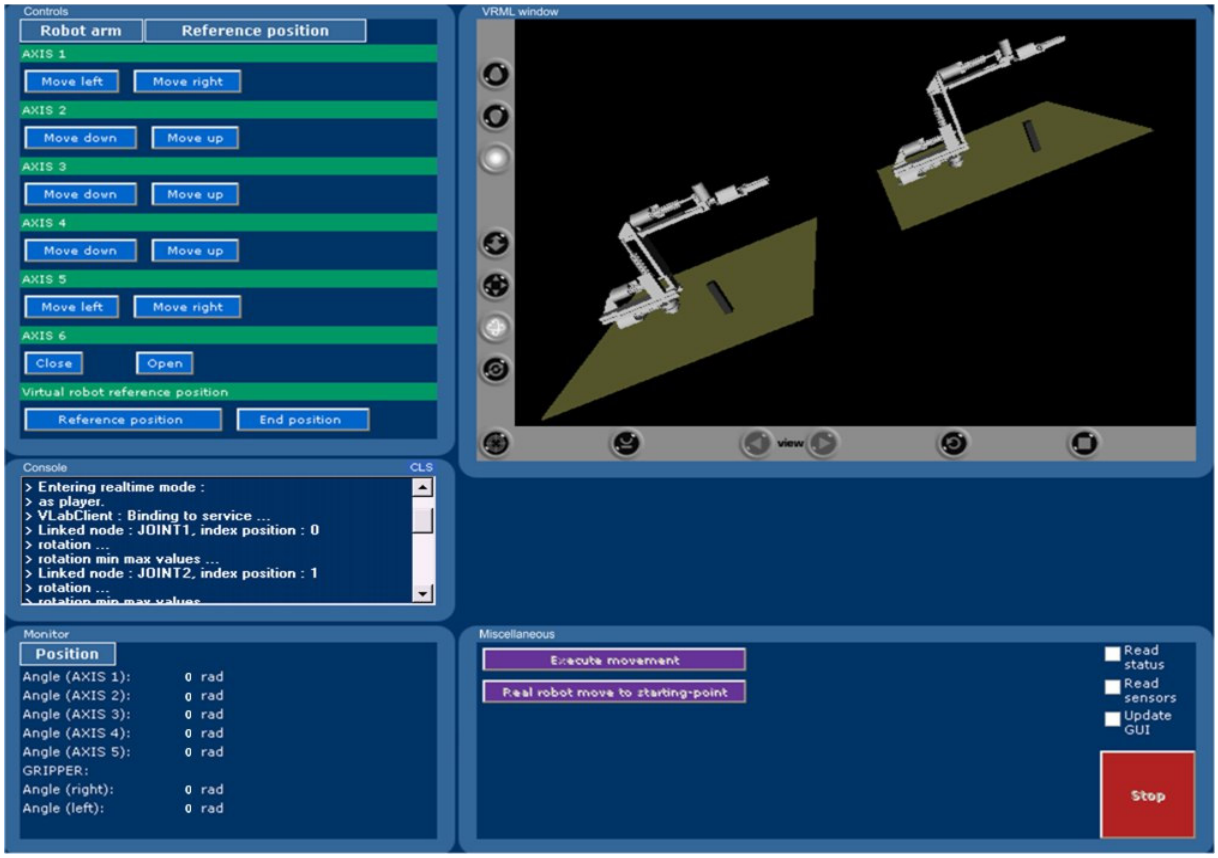

Figure 10. Virtual teach pendant and robot model

The library takes care of the collisions detection between the robot and known static barriers in its environment. Additionally, it takes care of the detection of collisions with the robot itself. The library has to detect a collision in the virtual model of the robot before we can send the command to the real robot. The library checks for possible collision with the whole robot before every desired move of a separate axle. We get feedback about the builders of the robot arm and the parts that are in collision. The library for collision detection is downloaded to the user computer at the start of the experiment, where it is executed. This principle of collision prediction in a virtual environment is used, to achieve faster response of a detected collision.

Preparation of the library for collision detection for inclusion of a geometrical configuration of a robot and its environment in the RLab environment is shown with the following steps:

1. The program for constructing the library for collision detection is opened.
2. The VRML robot model for which we want to make the library is selected. All of the transformation joints are automatically uploaded in a suitable hierarchic order in the schematic tree. The example is shown on the figure 11. The joints named in the VRML code that influence the change of the axes' position are therefore easier to find. The VRML node »JOINT1«, which controls a first robot axis rotation, is shown in the example (Fig. 11).
3. The transformation joints which define the translation and rotation of each robot axis and also define whether it is translation or rotation are chosen in the schematic tree. Defining the rotation joint of the first robot axis is shown on the Figure 11. The chosen joints of the robot are automatically linked to the buttons for manipulation of different axes, while loading into the user interface.
4. The accuracy of the geometrical model for collision detection is also defined. Two different choices of accuracy exist in our environment. The first one is not the best solution; however it works faster because of a simple geometry. The second approach is more accurate, but it is slower due to complex geometry. The slower detection speed is the result of a better robot geometrical model and additionally a larger number of triangles, which have to be checked for collision [12].
5. It is necessary to define pairs of robot components for which it is not necessary to detect the collision. Normally, these are neighbouring axis, which are linked to each other and never collide between themselves.
6. It is necessary to give the name of the library before its construction. In this way, we can dynamically upload the libraries for different robots. The library has to be compiled and uploaded into a suitable folder on the RLab server after the construction.

Basic program window for displaying the geometry and parameters preparation for the library construction can be seen on Figure 11. The »Transform JOINT1« robot node is selected on the Figure11 and also defined the rotation transformation in the lower right corner of the same figure. Methods are generated for the parameters controlling of the defined joints during the library construction. For an example: the method is generated for defining the axis rotation angle for "JOINT1". A collision detection working diagram of the RLab system is shown on Figure 12. The name of the collision detection library of the particular robot mechanism, which is uploaded on the start-up, is written in the configuration parameters of every robot. The verification of the collision is made by every axis move command (rotation and translation). A detailed description of working collision detection is next:

1. The user executes the chosen axis movement in the user interface of the RLab system.
2. The command is mediated to the collision detection library, which verifies for possible collision between robot and its environment components in the virtual geometrical model.

a. In the case when collision is detected, the execution of the position command is stopped and a warning for the detected collision (3) is shown in the console window.
b. In the case when the collision has not been detected, the execution of the command continues (4) and the virtual model of the robot (5) moves as user decides.

3. The command, which doesn't cause a collision, is stored into a buffer to be executed on the RLab server. When the user decides, the buffered commands are downloaded to the server (6) where are executed one after another. So, the collision protection in the real environment is ensured.

VII. CONCLUSIONS

This paper introduces WWW based remote rapid prototyping laboratories for control and robotics engineering students, which provides users with on-line access to the real-world hardware for remote experimentation.

The control laboratory is based on the well known MATLAB, Simulink and Real-Time Workshop (RTW) which are used for control algorithm development, simulation, offline analysis and rapid executable code generation while the LabVIEW VI provides the ability for online parameter tuning, signal monitoring, online analysis and via *Remote Panels* technology also teleoperation.

The telerobotics approach requires the user to develop tasks off-line, using their remote computing resources, before submitting the experiment to the remote laboratory server for execution on the actual device. The advantageous of the RLab system are mainly the system administration, from the system independent client application and modular architecture of the RLab system which allows relative simple and inexpensive extension of the system with new mechatronics applications. The disadvantageous of presented system concept are communication delays and the execution time of some processes, so the RLab system is not able to deal with the completely real-time teleoperating applications in a case of public internet use and poor communication connection (slow modems and ADSL…) to the internet. The communication delays should be decreased with the use of simpler text transfer protocol instead of WEB service communication technology.

REFERENCES

[1] S. Dormido, "Control Learning: Present and Future," In Proc. 15th IFAC World Congress on Automatic Control,Barcelona, Spain, 2002, pp. 314-319.

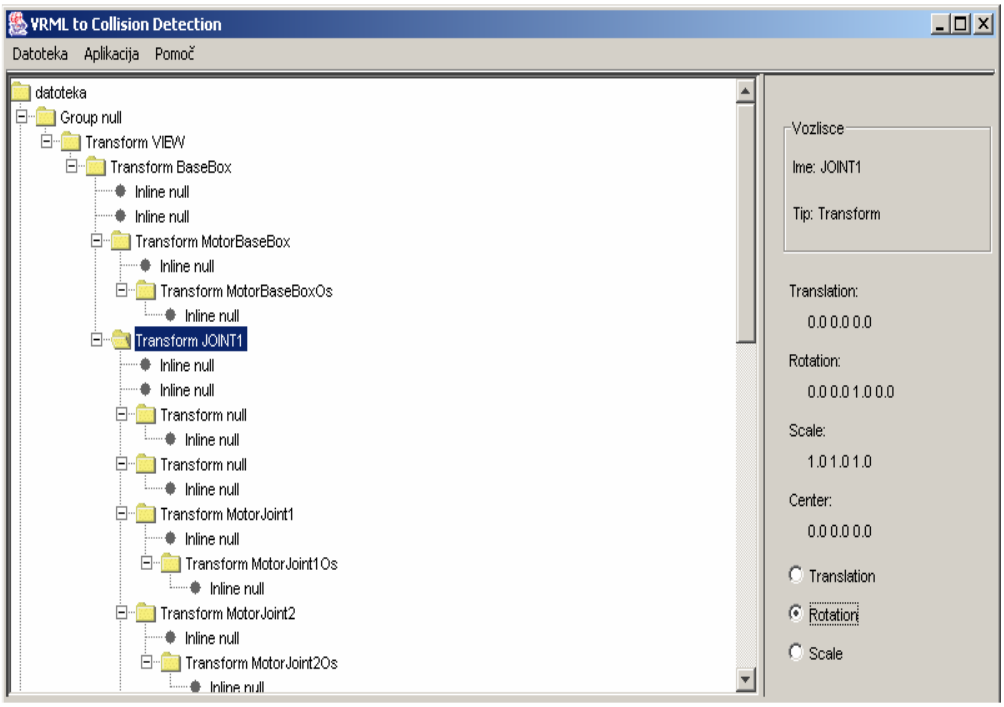

Figure 11. Collision detection library generator

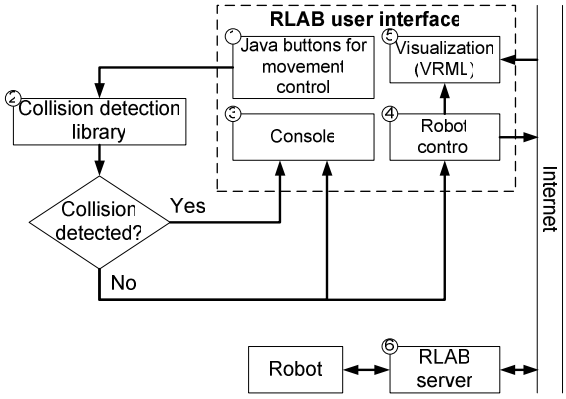

Figure 12. RLab system architecture

[2] C.Schmid, "Internet-basiertes Lernen," *Automatisierungstechnik*, vol. 51, No. 11, 2003, pp. 485-493.

[3] K. Goldberg, M. Maschna, S. Gentner, et al., "Desktop Teleoperation Via The WWW," *Proc. of the IEEE International Conf. on Robotics and Automation*, 1995, pp. 654-659.

[4] http://telegarden.aec.at

[5] http://telerobot.mech.uwa.edu.au

[6] R. Šafarič, M. Debevc, R. M. Parkin, S. Uran, "Telerobotics experiments via Internet," *IEEE transactions on industrial electronics*, 2001, vol. 48, no. 2, pp. 424-431.

[7] D. Hercog, M. Čurkovič, G. Edelbaher, E. Urlep, "Programming of the DSP2 board with the MATLAB/Simulink," Proceedings IEEE ICIT 2003, December 2003, pp. 709-713;

[8] DSP-2 web page: www.ro.feri.uni-mb.si/projekti/dsp2

[9] D. W. Calkin, R. M. Parkin, C. A. Czarnecki, "A PID Servo Control System Conducted Remotely Via Internet, " *Mechatronics*, 2002, Vol. 12, pp. 833-843.

[10] R. Šafarič, M. Debevc, R. M. Parkin, S. Uran, "Telerobotics experiments via Internet," IEEE transactions on industrial electronics, 2001, vol. 48, no. 2, pp. 424-431.

[11] R. Šafarič, D. W. Calkin, R. M. Parkin, Czarnecki, "Virtual environment for telerobotics," Integrated. computer.-aided engineering, 2001, vol. 8, no. 2, pp. 95-104.

[12] Šafarič, R.; Šinjur, S.; Žalik, B.; Parkin, R. M.: Control of Robot Arm with Virtual Environment via Internet, Proc. I.E.E.E., 2003, vol. 91, issue 3, pp. 422-429.

AUTHORS

R. Šafarič is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: riko.safaric@uni-mb.si).

M. Truntič is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: mitja.truntic@uni-mb.si).

D. Hercog is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: darko.hercog@uni-mb.si).

G. Pačnik is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: gregor.pacnik@uni-mb.si).

Manuscript received May 31, 2005.