A DSP-Based Remote Control Laboratory

Darko Hercog, *Student Member, IEEE*, Bojan Gergič, *Member, IEEE*,
Suzana Uran, *Member, IEEE*, and Karel Jezernik, *Senior Member, IEEE*

***Abstract*—This paper presents a framework for rapid remote experiment implementation in the field of automatic control. The proposed solution is based on in-house developed embedded control hardware and two commercially available software packages. MATLAB/Simulink is used for rapid experiment control algorithm development, while LabVIEW is used for the user front-end and remote control. A combination of presented hardware and software solutions enables the rapid and easy creation of different interactive remote control experiments. Using this solution, a digital-signal-processor-based remote control laboratory for teaching purposes has been realized. This remote laboratory enables the remote users to easily interact with a set of physical control experiments through the Internet. In the friendly user interface, the remote user can change predefined system parameters and observe system response in textual, graphical, or video format. In addition, this remote laboratory includes a booking system, which enables remote users to book experiments in advance.**

***Index Terms*—Automatic control, booking system, dc motors, digital signal processor (DSP), embedded systems, engineering education, LabVIEW, MATLAB, real-time workshop (RTW), remote laboratories, simulink.**

I. Introduction

EFFICIENT learning in the engineering field requires a mixture of theoretical and practical exercises. Therefore, laboratory experiments play, and will certainly play, an important role in control-engineering education [1]. During experimental work, students become acquainted with real-world features and gain experience and knowledge, which cannot be obtained by just using simulations. Although classical hands-on laboratories are very useful and educational, they have many limitations regarding space, time, and staff costs. They are usually fully occupied, and students have to conclude their research within the time that is allotted for experimental work. The problems with traditional classical laboratories can be avoided by using remote experiments and remote laboratories. In remote experimentation, students operate with the real system, although they are not physically present in the laboratory. Such a solution presents a cost-effective way of opening up a laboratory for students 24 h a day. The remote users can conduct their experiments by accessing the laboratory when they most need it and from a remote location that is more comfortable to them. Remote laboratories are mainly used within the academic field to enhance classroom lectures, share research equipment, and supplement the learning process. In the majority of existing solutions, remote users can change system parameters, execute experiments, observe results in text or graphical view, and download the experimental results. In addition to these capabilities, some remote laboratories also include a booking system, which helps the remote users to organize their time and activities. Currently, a great deal of available remote experiments and laboratories come from the area of remote measurement, while the minority of them also covers automatic control. In this paper, only solutions from the latter are taken into account.

Following one from the first remote laboratory that was developed in 1992 by Stanford Center for Innovations in Learning, a variety of different approaches have been proposed for developing remote experiments or remote laboratories in the field of control engineering education [2]–[10] and telerobotics [11]–[14]. In many cases, authors have burned up much energy trying to find a simple solution, where only a standard Web browser is needed by the remote user, in order to perform remote experiments [2]–[6]. In some other solutions, the remote users must download a special program, thus enabling remote control [7], [8]. In the majority of cited remote control experiments, remote users can run an experiment and adjust the process or controller parameters from a set of predefined parameters. In a Web-based laboratory for a two-degrees-of freedom helicopter [2], for example, the remote user can select among four different predefined controller types. An interesting and advanced remote laboratory has been presented by the members of Siena University [3]. The Automatic Control Telelab (ACT) enables students to choose a control law, change the control parameters online, and even design their own controller, simply through the MATLAB/Simulink environment. Using ACT, the remote users can design a custom controller and reference signal on a local personal computer (PC) and, after successful simulation, upload it to the ACT server and verify it against the real process.

Although remote experiments seem to be very useful and educational, much hard work is needed when setting them up. Designers must be acquainted with different tools and technologies, such as the Web server's operation, Java programming, Common Gateway Interface script's principle of operation, and Internet communication. Because of this broad but necessary knowledge, remote experiments are rarely present as a supplement in undergraduate courses. In this contribution, a framework for rapid implementation of remote experiments in the field of automatic control is presented. Using the presented solution, remote experiments can be realized quickly, easily,

Manuscript received March 28, 2007; revised August 15, 2007. This work was supported by the European Community within the framework of the Leonardo da Vinci II Program under Project CZ/06/B/F/PP-168022.

The authors are with the Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia (e-mail: darko.hercog@uni-mb.si).

Digital Object Identifier 10.1109/TIE.2007.907009

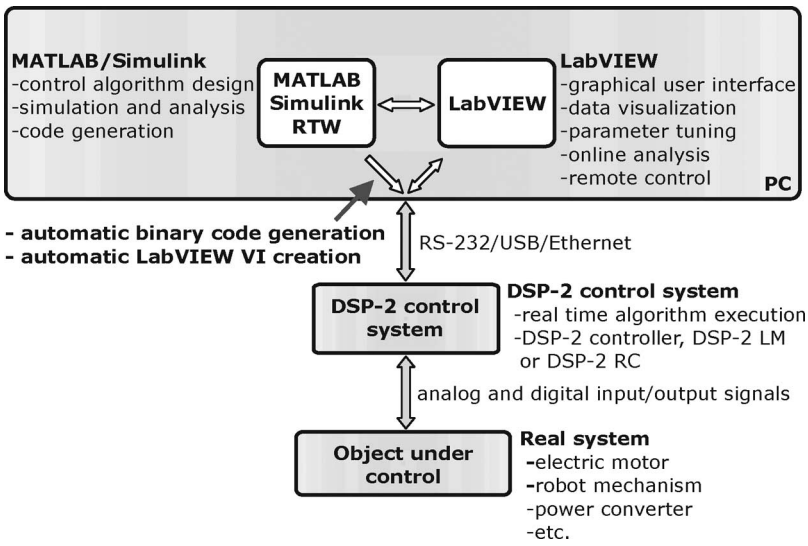


Fig. 1. DSP-2 rapid control prototyping solution.

and with only a basic knowledge of two of the electrical engineering world's leading software packages. The proposed solution (Fig. 1) is based on a DSP-2 control system's hardware that was developed by the Faculty of Electrical Engineering and Computer Science (FERI), University of Maribor; software packages MATLAB/Simulink from MathWorks Inc.; and LabVIEW from the National Instruments Company. MATLAB, Simulink, and Real-Time Workshop (RTW) are used for control algorithm development, simulation, offline analysis, and rapid executable code generation, while the LabVIEW serves as the user front for data visualization and parameter tuning and, via LabVIEW Remote Panels technology, for remote operation.

This paper is organized as follows: Section II presents a brief hardware description of custom-made digital signal processor (DSP)-based control systems. Section III contains an overview of the rapid control prototyping support and data visualization solution for the mentioned DSP-2 control systems. This section includes an overview of a Simulink library, which enables easy programming of DSP-2 control systems using Simulink, and a short description of the LabVIEW toolkit for the DSP-2 control systems. Section IV contains an overview of the realized "DSP-based remote control laboratory" [15]. Section V describes example remote applications, which are available on a remote laboratory Web page. Finally, conclusions and future work are stated in Section VI.

II. DSP-2 Control Systems

All DSP-2 control systems are based on a DSP-2 controller [16] that was developed at FERI. The key components of the DSP-2 controller are the Texas Instruments (TI) TMS320C32 floating-point processor, which is used for control algorithm execution; the Xilinx field-programmable gate array of the Spartan family, which implements the pulsewidth modulator (PWM); and onboard peripheral interfaces. In addition, the DSP-2 controller contains all the necessary peripherals, for ac and dc motor control i.e., analog-to-digital and digital-to-analog converters, a three-phase PWM, an optically isolated digital input/output (I/O), interface for the incremental encoder, random access memory, Flash read-only memory, and Controller Area Network (CAN) chip. Two different types of control systems have been developed based on this controller. These control systems are mainly used during education processes and are briefly described in the following sections.

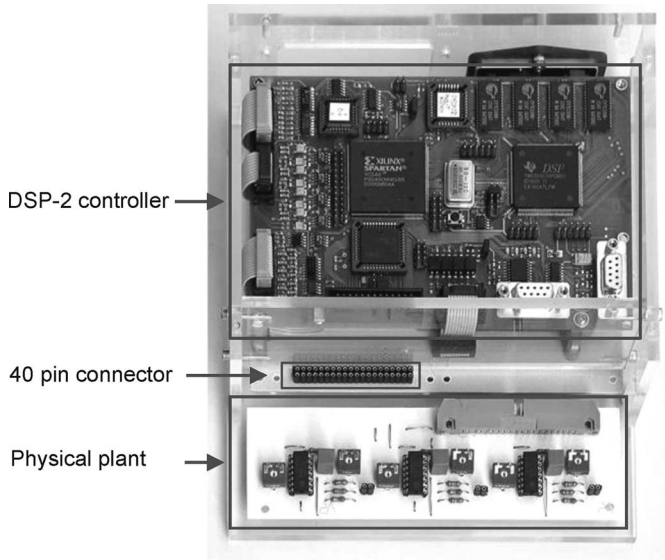


Fig. 2. DSP-2 LM.

A. DSP-2 Learning Module (DSP-2 LM)

The DSP-2 LM that is shown in Fig. 2 was developed from a desire to offer students a powerful and universal learning system. The learning module is composed of the DSP-2 controller and an additional board, where the power supply and expansion connector take place, for important DSP-2 I/O signals. The DSP-2 LM is versatile, light and small, handy, and an easy-to-use learning system. In combination with a laptop computer,

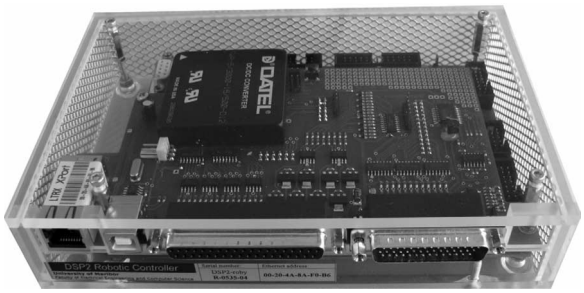


Fig. 3. DSP-2 RC.

it represents a mobile rapid control prototyping system that is appropriate for hands-on experiments or in-class demonstrations. The developed learning module is plant flexible, because a variety of in-house developed plants or plants from different manufactures can be connected to the module through its 40-pin expansion connector (Fig. 2). DSP-2 LMs are used during different control courses that were performed at FERI, as well as in a DSP-based remote control laboratory, as described in this paper.

B. DSP-2 Robotic Controller (DSP-2 RC)

The DSP-2 RC [17], as shown in Fig. 3, is another control system that is based on the DSP-2 controller. The DSP-2 RC is composed of the DSP-2 controller and a DSP-2 add-on robotic board. The DSP-2 RC contains all the necessary peripheral for four-axis robot control, i.e., this system has 16 digital inputs, eight digital outputs, four analog inputs/outputs (± 10 V), and four incremental encoder interfaces. The DSP-2 RC can be connected to the development computer via USB or Ethernet connection.

III. Code Generation and Data Visualization Solution for DSP-2 Control Systems

Code generation and data visualization solution for DSP-2 control systems are based on two well-known commercially available software packages, i.e., MATLAB/Simulink and LabVIEW.

A. Rapid Control Prototyping Support for DSP-2 Control Systems

In the desire for rapid control prototyping support of DSP-2 based control systems using MATLAB/Simulink, the so-called "DSP-2 Library for Simulink" [18] has been developed. This DSP-2 library is a Simulink add-on toolbox, which contains a set of Simulink device driver's blocks (Fig. 4) for all available I/O ports of DSP-2 control systems (DSP-2 controller, DSP-2 LM, and DSP-2 RC). The DSP-2 library contains blocks for analog I/O, digital I/O, and incremental encoder and blocks for communication between a PC and the DSP-2 system. In combination with the Simulink, RTW, and TI Code Composer, the DSP-2 Library for Simulink enables developers to model control or signal processing applications in the Simulink block diagram environment and, after successful simulation, verify the designed algorithm on one of the DSP-2 control systems that are connected to the physical system. The code generation tool chain includes an RTW for automatic ANSI C code generation from Simulink models and TI Code Composer for binary executable code generation from an automatically generated C code. All the DSP-2 device driver blocks are supported by an RTW Embedded Coder; therefore, the generated code is highly optimized in performance and space. More information about the DSP-2 Library for Simulink and code generation process for DSP-2 systems can be found in [18] and [19].

B. Data Visualization and Parameter Tuning Solution

A stand-alone program DSP Terminal [19], [20] and a LabVIEW toolkit named ComVIEW [21] have been developed to serve on-the-fly data visualization and parameter tuning tasks for the DSP-2 control systems. The ComVIEW toolkit enables rapid LabVIEW user-interface creation for the selected Simulink model. When the DSP-2 embedded target is selected in the Simulink model and LabVIEW with the mentioned toolkit is installed on the development PC, a LabVIEW virtual instrument (VI) is automatically created from the ComVIEW template VI during the binary code generation (Fig. 1). Afterward, the created VI can be fully modified by the end user. 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 control system, functions for transmitting and receiving messages between the development PC and the DSP-2 system, and other low-level routines.

During VI creation, numerical controls and indicators are automatically added to the ComVIEW template front panel, where the number of controls and indicators depends on the number of DSP-2 global variables blocks that are used in the Simulink model. Links between DSP variables and VI front panel objects are established programmatically using the DSP Connection Manager window (Fig. 5). This window appears on the PC immediately after the downloadable binary code starts executing on the DSP-2 control target. Using mouse clicks, the user can create links between the VI front panel indicators and the DSP-2 output variables, and links between the VI front panel controls and the DSP-2 input variables 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 control system. Whenever the controls on the VI front panel are changed, LabVIEW automatically downloads them via a communication link to the DSP-2 system. At the same time, all the retrieved DSP-2 output variables are read from the PC communication port (serial, USB, or Ethernet) and displayed on an appropriate numerical indicator. In addition, ComVIEW also provides scope capabilities. In the scope mode, a small portion of the code running on the DSP-2 controller handles data acquisition (DAQ) and storage management. The selected DSP-2 global variables are, first, captured and then stored in the temporary controller memory. After that, the captured data are transferred to the VI running on the PC and graphically presented in VI graphs. For those variables to be captured, ComVIEW menu options enable setting

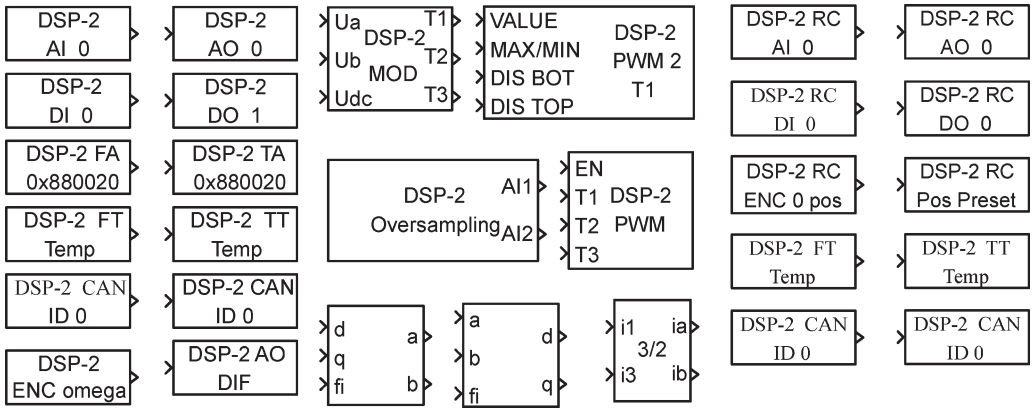


Fig. 4. Simulink device driver block sets for DSP-2 control systems.

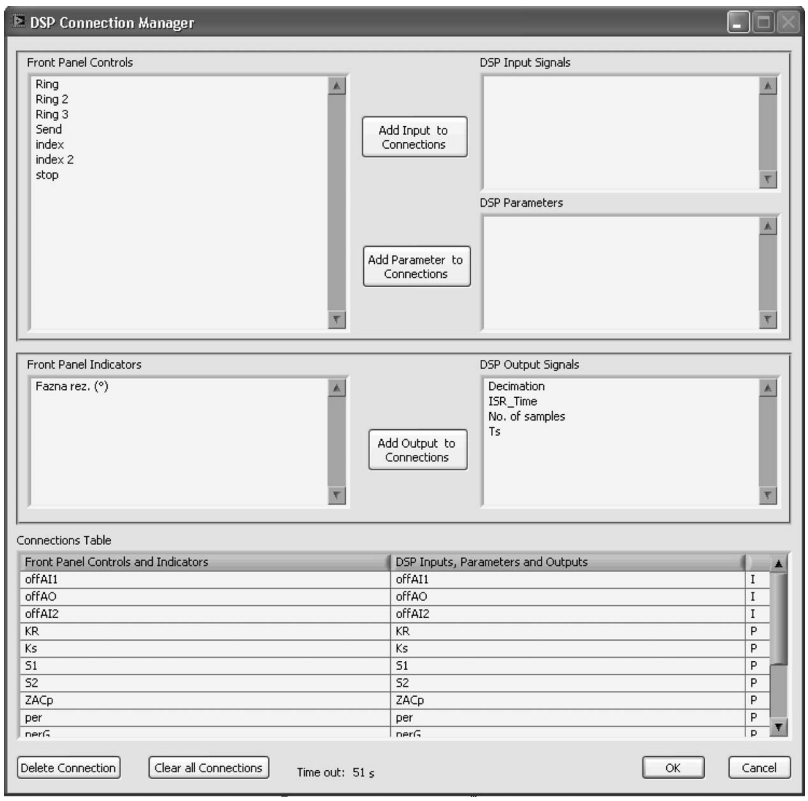


Fig. 5. DSP connection manager window.

the number of samples and decimation and trigger parameters including selection of the trigger signal, trigger level, slope, and the number of presamples.

IV. DSP-Based Remote Control Laboratory

The hardware and software components that were described in the previous sections provide a framework for a rapid remote control experiment development process. Using this solution, a DSP-based remote control laboratory has been successfully realized. This remote laboratory, which is available in [15], is composed of DSP-2 control systems, laboratory PC, and an additional server for laboratory Web pages and a booking system (Fig. 6). The laboratory Web pages and a booking system could be placed on the laboratory PC; however, in this solution, they are placed on the separate server, because they are part of the faculty course management system. DSP-2 control systems are connected to the laboratory PC, which is, in turn, connected to the Internet (Fig. 6). The control systems implement a control algorithm that was developed using Simulink and, through the analog and digital I/O signals, drive the real process. At the same time, the VI for individual experiments and the LabVIEW server are run on the laboratory PC for the purpose of enabling remote control. Individual VI performs data exchange between the DSP-2 control system and the laboratory PC, while the LabVIEW server enables remote operation of this VI. Virtual instruments are published on the Web using the LabVIEW built-in Web Publishing Tool. When using this tool, LabVIEW

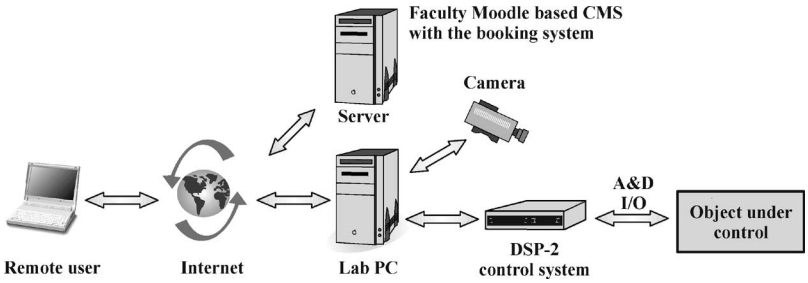


Fig. 6. Block scheme of remote laboratory operation.

front panels can be quickly and effortlessly published on the Web without additional programming [22]. Once the VI is published, anyone on the Web, with proper permission, can access and control an experiment using the standard Web browser. When a remote viewer enters an appropriate Uniform Resource Locator (URL) address, the LabVIEW front panel appears in the Web browser. Once the user has been granted control, the graphical user interface controls become active, and running the LabVIEW application is like running the application from the local environment. During remote experiment execution, the remote user can adjust the controller parameters and observe process output values in textual or graphical mode. LabVIEW Remote Panels include an algorithm that allows only one user to take control over the remote experiment. When one client is controlling the remote experiment, the other remote clients are able to monitor the actions of the controlling client. When another client requests control, the controlling client is notified that control time has now become limited. Once timeout occurs or the controlling client releases control, application control is automatically switched to the requesting client [22].

A. Booking System

Although Remote Panels include an algorithm to prevent control access conflicts by different remote users, the current solution does not provide booking possibilities. The presented remote laboratory introduces a booking system based on the Moodle course management system (www.moodle.org). Moodle is a widely adopted software package for producing Internet-based courses and websites. The original booking system for Moodle was developed within the Leonardo da Vinci MARVEL project [23]. In addition to the original booking system, some modifications have been made [24], which enable easy booking creation for those remote experiments that were developed using the LabVIEW Web Publishing Tool. Experience has shown that a booking system is very useful, because it helps the students to organize their time and activities. In the DSP-based remote control laboratory, each experiment Web page contains a *run/book this experiment* link. After selecting this link and log-in process, a new Web page with the booking table appears (Fig. 7).

In this window, the remote user can book an experiment for 1 h, and then, when the reserved time slot becomes the current 1-h time slot, the user gains a valid link (red arrow in Fig. 7) to the remote experiment Web page.

	Tue August 14	Wed August 15	Thu August 16	Fri August 17	Sat August 18	Sun August 19	Mon August 20
0:00		RC oscillator					
1:00							
2:00		RC oscillator				RC oscillator	
3:00				RC oscillator			
4:00							
5:00							

Fig. 7. Booking timetable.

B. Visualization of Experiments

A very important part of live experiments is visual feedback [25]. Visual feedback can be added to a remote laboratory in many different ways. A simple solution would be to integrate the live images into the front panel of a remote experiment and then use Remote Panels. Unfortunately, this solution would slow down the program execution significantly, because the images are sent uncompressed over the Internet. A better solution would be to embed the live webcam images in the Web page that was created by the LabVIEW Web Publishing Tool. This solution requires a separate software package for implementing a remote experiment computer as the image host and creating the code that is necessary to add the live images to the existing LabVIEW-created HTML code [26]. Another solution uses a separate application for remote experiment visualization. One such application is the Microsoft Windows Media Encoder 9 Series, which could be downloaded free of charge from Microsoft's home page. The Microsoft Windows Media Encoder 9 Series is a powerful tool for converting live video into compressed Windows Media streams that are then viewed by remote users using a Windows Media Player. The weakness of this solution is the long delay between image acquisition and image display.

The solution that was proposed in [27] is based on client–server architecture and sends compressed images over the Internet with only a short delay. The server application sends Joint Photographic Experts Group (JPEG) compressed images

Fig. 8. Front panel of client application.

to the client application, which decompresses and displays the received images to the remote experiment user. These applications are implemented as separate LabVIEW VIs, which require a free LabVIEW runtime engine installed on the client's computer. A LabVIEW runtime engine is also necessary, in order to operate remote experiments using Remote Panels. The software toolkits and drivers that are necessary for developing the proposed applications are included in the National Instruments Academic Site License, as used by many academic institutions worldwide. Images are acquired with digital cameras complying with the 1394 Trade Association's Industrial and Instrumentation specification for Digital Cameras (IIDC) and NI-IMAQ for an IEEE 1394 Camera's driver, which enables simultaneous acquisitions from more than one camera and direct settings of camera attributes. Unibrain's Fire-i Color Digital Cameras, complying with IIDC, were chosen, because of good picture quality and reasonable price. The server application sends the table of active cameras with corresponding URLs and experiment titles to the client application, where the student can choose the camera from the list of friendly experiment titles (Fig. 8). A camera list and JPEG compressed image data are sent over the Internet by using a LabVIEW shared variable.

It is important to minimize the processing power for remote experiment visualization, particularly, when the same server computer is also used for remote laboratory experiments. The processing power requirements increase rapidly with image size and frame rate. For most remote laboratory experiments, it is enough to have an image size with 320×240 pixels or less and a frame rate with less than 5 frame/s. In this case, the server application for remote experiment visualization uses less than 10% of the processor power on today's average personal computer. The proposed codec is based on JPEG compression and, therefore, cannot achieve such compression rates and quality as the Microsoft Windows Media Video 9 codec, which uses efficient moving estimation, but it has a very short delay when compared to the combination of the Microsoft Windows Media Encoder 9 Series with a Windows Media Player.

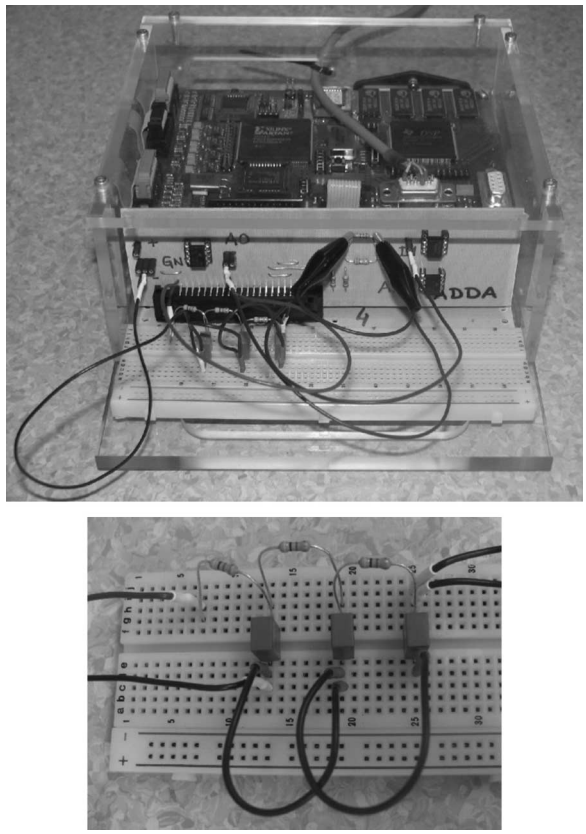


Fig. 9. DSP-2 LM with *RC* circuit on the breadboard.

C. DSP-Based Remote Control Laboratory Key Features

In addition to the booking system and the remote experiment visualization solution, the presented remote laboratory also has some other features that make it unique.

- Powerful control hardware: The described laboratory is based on DSP hardware. The majority of currently available remote control laboratories are based on PC with DAQ cards. In such architecture, the sample rates exceed 1 ms, and consecutively, the control of the rapid dynamic systems (with a time constant that is lower than 10 ms) is impossible. A DSP-based remote control laboratory, in contrast, enables much lower sample rates. In a cascade dc motor control experiment (available online in [15]), for example, an algorithm takes approximately 80 μs, and the sampling time is set to 200 μs.
- Rapid remote experiment development process: The described hardware and software solution enables a rapid remote experiment development process. Using this solution, a remote experiment can be quickly developed with only a basic knowledge of MATLAB/Simulink and LabVIEW programming languages. Experience reveals that a remote experiment development process, using this solution, takes less than 1 h.
- Plant flexibility: DSP-2 control systems are plant flexible; therefore, different custom or commercially available real plants can easily be connected to them. When a DSP-2 RC is used, remote control of a four-axis mechatronic device can be easily achieved.

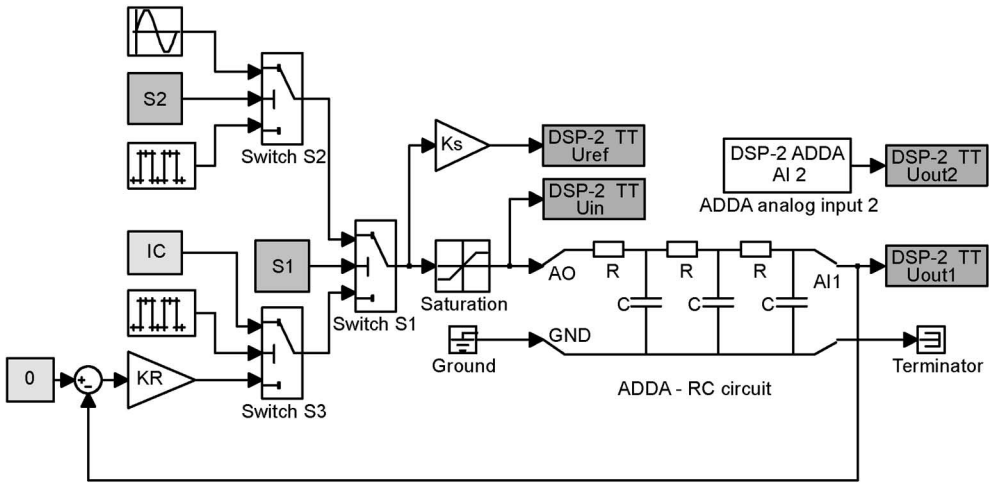


Fig. 10. MATLAB/Simulink block scheme for *RC* oscillator.

- Easy interface: Simplicity of use is essential to establish an interface that can be used by everyone [28]. All available experiments in the presented remote laboratory have a simple graphical user interface; therefore, the user can focus his/her energy on interacting with the experiment.
- Degree of interactivity: There are basically two kinds of interactivity: 1) pseudobatch and 2) online [29]. Pseudobatch means nonimmediate response from the time that the process is initiated. The online degree of interactivity advances continuously and dynamically, and the user obtains the results in the form of a continuous flow of numerical values or graphics, steadily developing in each sampling period [29]. The presented remote laboratory has the online degree of interactivity. Remote users can iteratively adjust parameters and observe the results during one continuous interactive session (1-h time slot), instead of a series of discontinuous batch sessions.
- Safety: A remote experiment needs to be safe; therefore, security issues are one of the most important aspects to be considered when developing remote applications, particularly, when moving parts are involved. Security includes the security of control systems and objects under control, and the security of the information system from malicious attacks. In the presented solution, the first two security problems are solved by preventing commands from outside the allowed range from being issued to the controller through the Web interface (all front panel controls are limited). In addition, when the user changes or remote user closes the connection to the remote experiment (closes the Web browser, for example), all front panel controls are automatically reinitialized to their default values. In the case of a cascade dc motor control experiment (available online in [15]), for example, the motor is automatically turned off. The second problem is solved by restricting access to remote experiments only to authorized users.

D. Drawbacks

The main drawback of the presented remote control laboratory lies in the fact that the LabVIEW runtime environment must be present on the remote client's work stations. In this case, platform independence is not guaranteed, even if the LabVIEW runtime is available for many operating systems.

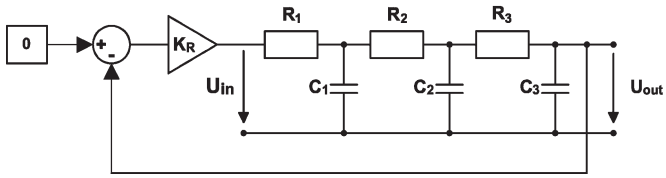


Fig. 11. *RC* oscillator control loop.

V. Example Applications

Until now, the following remote experiments (available online at [15]), based on the presented hardware and software solution, were developed:

- *RC* oscillator;
- speed control of dc motor;
- cascade control of dc motor;
- teleoperation of mechatronic device.

The *RC* oscillator and speed control of dc motor experiments are used during an introductory "Control systems I" course, while the cascade control of dc motor is used in the "Servo systems" course. In the following sections, the *RC* oscillator and cascade control of dc motor experiments are described in more detail.

A. RC Oscillator

The *RC* oscillator is a Web-based version of interactive controller design and experiment [in the following Web-based Interactive Controller Design and Experiment (WICDE)]. The objectives of WICDE are given as follows:

- to teach students control design;
- to minimize the gap between control theory and practice, by teaching control implementation;
- to show students how to learn by Web and how to use it;
- to support learning by doing.

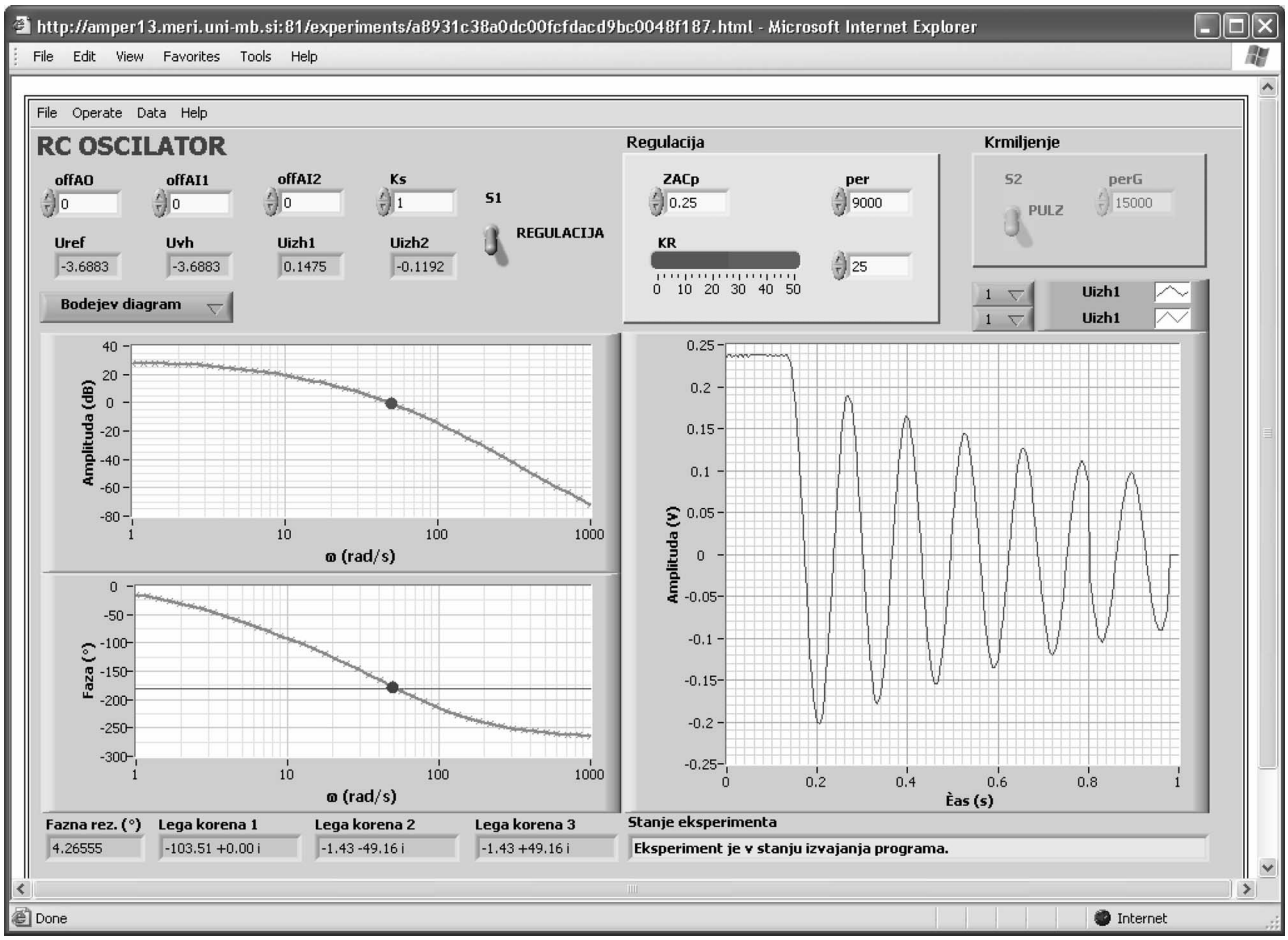


Fig. 12. Bode plot design with interactive experiment ($K_R = 25$).

The WICDE was designed to be available to a broad range of our students. Therefore, it was designed with minimum software requirements from the student prospective. To perform the WICDE experiment, a standard Web browser and "LabVIEW runtime engine" are needed. Unfortunately, the assumption of minimum student's software requirements sets an undesirable limitation on the implementation of "Learning through doing." This limitation means that students cannot build their own experiment but can only vary the parameters of the already prepared experiment.

An *RC* oscillator experiment is implemented using a DSP-2 LM and a breadboard with an *RC* circuit (Fig. 9). The resistor and capacitor values of the *RC* circuit are

$$R_1 = R_2 = R_3 = R = 47\ \text{k}\Omega$$
$$C_1 = C_2 = C_3 = C = 1\ \mu\text{F}.$$

A control loop of the *RC* oscillator is shown in Fig. 11. The mathematical model (transfer function) of the *RC* circuit is given by

$$F_{\text{RC}}(s) = \frac{U_{\text{out}}(s)}{U_{\text{in}}(s)} = \frac{1}{(R{\cdot}C)^3 \cdot s^3 + 5 \cdot (R{\cdot}C)^2 \cdot s^2 + 6 \cdot (R{\cdot}C) \cdot s + 1}. \quad (1)$$

The MATLAB/Simulink scheme that was implemented for the WICDE *RC* oscillator is shown in Fig. 10. The open-loop and feedback controls of the *RC* circuit are combined in one Simulink block scheme. Switch S1 is used to select either open-loop or feedback control. When open-loop control is selected, switch S2 is used to select the step or the sinusoidal input to the *RC* circuit. When feedback control is selected, switch S3 takes action. This switch selects between the initial condition (IC) input and feedback with gain K_R, for input into the *RC* circuit. Therefore, two phases of the *RC* oscillator response are observed. The capacitors of the *RC* circuits are charging to the value IC when the IC input is selected. This phase is called the "charging phase." When feedback with gain K_R is selected, the *RC* oscillator response is observed. This phase is called the "*RC* oscillator relaxation phase." Signals Uref, Uin, Uout1, and Uout2 from the Simulink block scheme can be observed. Online tuning of K_R (*RC* oscillator feedback gain), IC (the value to which capacitors should charge), and the period of the pulse generator switching between the charging and the relaxation phase is possible. The variation of these input parameters and the *RC* oscillator graphical response is mainly delayed, because of graph data buffering on DSP-2 system, while a minor part of the whole delay contributes Internet communication and communication between the DSP-2 system and the laboratory PC. This delay does not affect the RC oscillator control loop (Figs. 10 and 11),

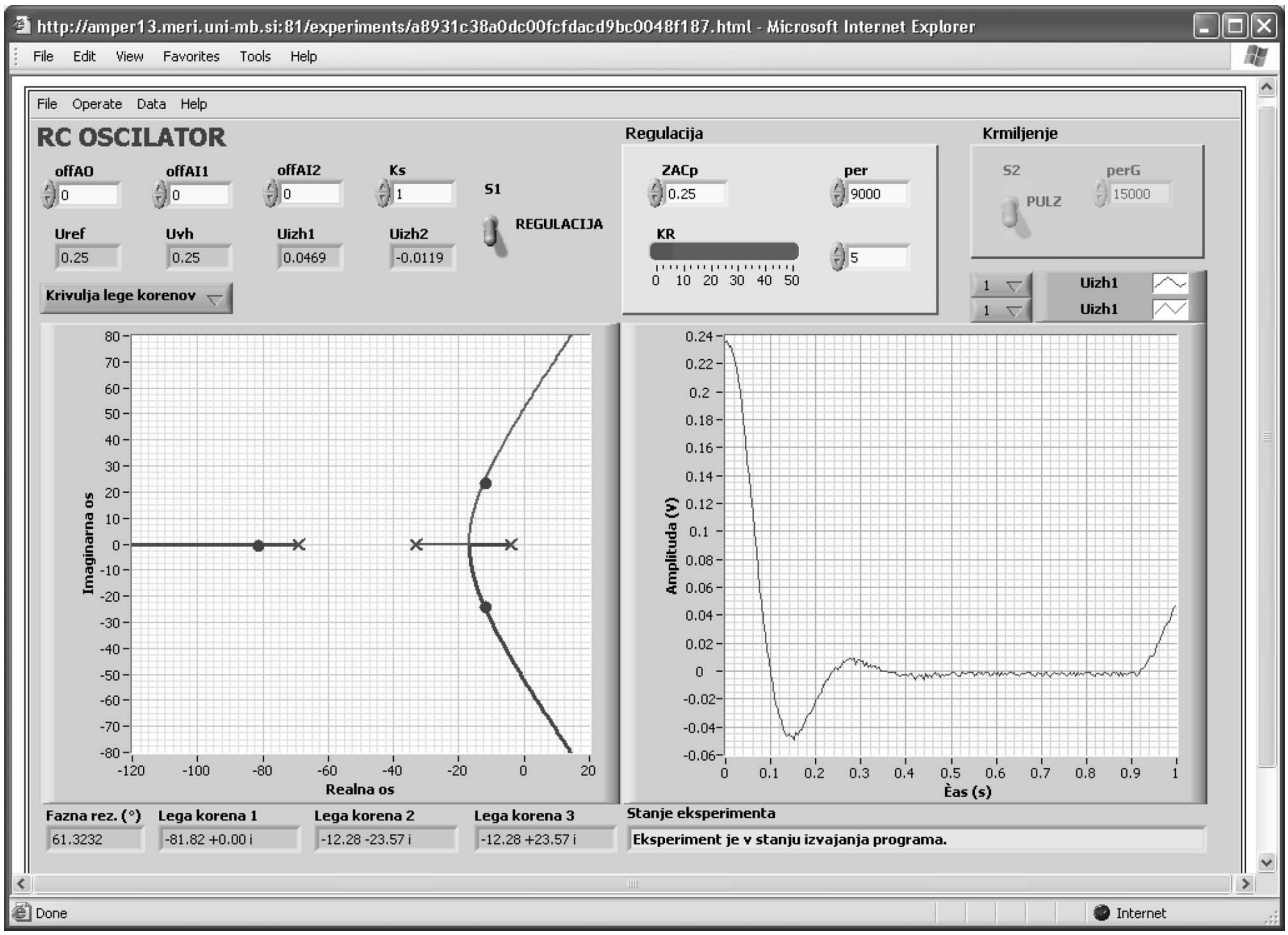


Fig. 13. Root locus method design with interactive experiment ($K_R = 5$).

because the control algorithm is executing on DSP-2 (Fig. 9) in real-time.

The *RC* oscillator Bode plot or Root Locus plot can be observed interactively, together with the *RC* oscillator experimental response. Bode plot or Root Locus design is selected with the button on the VI front panel (Figs. 12 and 13). Bode and Root Locus plots are calculated by the MATLAB script server on the basis of the *RC* circuit transfer function that was given in (1). Controller design parameters, such as phase margin, crossover frequency, and the actual roots of the control loop, are clearly marked in the Bode and Root Locus plots.

Considering learning in the class, then WICDE is suitable only as a means for demonstrations that are done by the lecturer. The reason for this is the limitation of resources (one device–one user) that is inherently given by every remote experiment. However, on the other hand, WICDE excellently supports individual assignments for students and learning by doing over the Web. One form of individual assignments for students is homework. During this school year, we have designed a homework task to be performed using the *RC* oscillator experiment and give it to students taking part in the introductory control course at our faculty.

Short description of homework task: Use WICDE to design a P controller with gain K_R using the Bode plot for the *RC* oscillator.

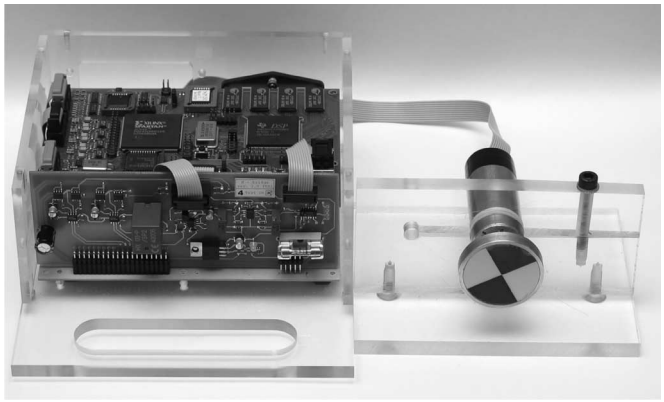


Fig. 14. DSP-2 LM with the H-bridge and dc motor.

An *RC* oscillator is used for the generation of sinusoidal signals and, therefore, design gain K_R for the margin of stability. Verify your Bode plot design by using the time response experiment. Observe the Bode plot and time responses of the *RC* oscillator at $K_R = 3.2$, 10, 25, and 40, and determine the phase margin at given gains. Compare the results that were obtained from the WICDE with those that were obtained using Web sisotool or M-file application for MWS, and explain the reasons for deviations, if any. Using WICDE, find out how dependent the overshoot of the *RC* oscillator, time response, and phase margin in the Bode plot are on gain K_R.

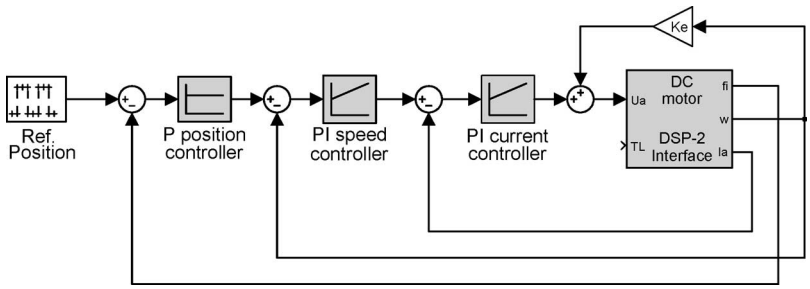


Fig. 15. DC motor control algorithm in MATLAB/Simulink.

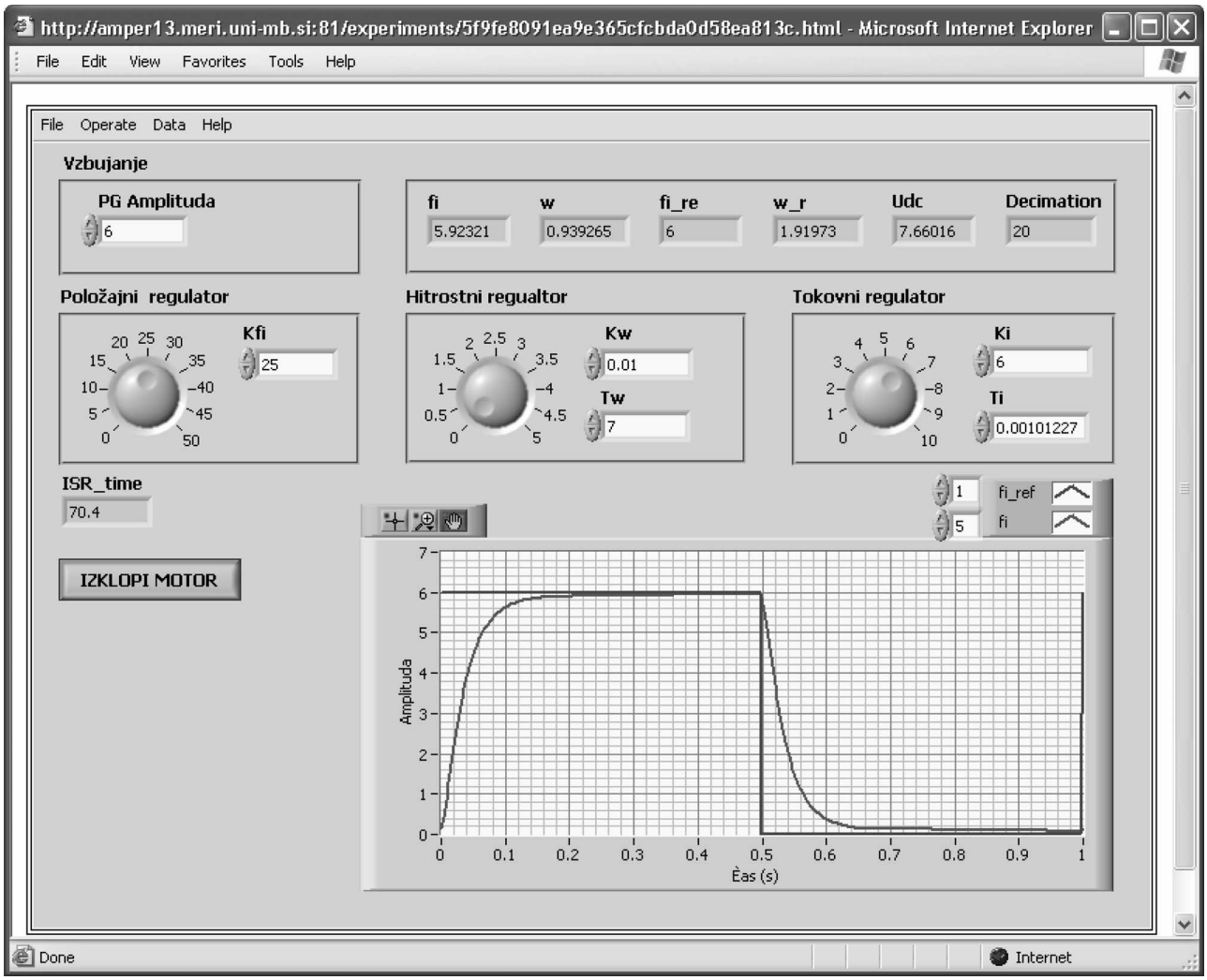


Fig. 16. Remote dc motor experiment interface in a standard Web browser.

B. Cascade Control of DC Motor

In this experiment, the dc motor setup is used. This setup is composed of DSP-2 LM, H-bridge, and a commercially available dc motor (Fig. 14). Fig. 15 presents a Simulink model of dc motor cascade control. The Simulink model contains current, speed, and position control loops; a position reference generator; and the DSP-2 DC motor interface subsystem. This subsystem contains an algorithm for armature current measurement, current offset compensation, calculation of the speed and position of the motor shaft, speed measurement filtering, PWM signal generation, and dc-link voltage measurement. The DSP-2 DC motor interface subsystem is realized using DSP-2 blocks and the Simulink built-in blocks. Additional information about this Simulink model can be found in [19].

During remote experimentation (Fig. 16), the remote user can start/stop the dc motor; set position reference; and adjust current, speed, and position controller parameters. It can also reinitialize all VI controls to the default values and observe experimental data in a numerical or graphical view.

VI. Conclusion

In this paper, the hardware and software solution for a rapid remote experiment development process and a DSP-based remote control laboratory has been presented. The presented solution is based on an architecture that can be easily adapted to different remote control experiments. Using the presented hardware and software solution, the remote experiment developers do not need to waste time publishing remote experiments on the Internet; therefore, they can direct their energies to experiment control algorithm design. In contrast to some existing remote experiment solutions, where only the "pure" Web browser (browser with no additional plug-in programs) is needed by the remote user, this solution requires a free "LabVIEW runtime engine" program on the client computer in order to perform experiments.

The presented remote laboratory is currently being utilized in "Control systems I" and "Servo systems" undergraduate courses. Available remote experiments are used throughout the course homework assignments, in which students are required to solve an actual control problem using techniques that were learned in class and verify their designs on the actual system through the Internet. Discussions with students indicate that the use of the remote control laboratory has improved their understanding of the course material.

Although only two experiments are described in this paper, the remote laboratory paradigm can easily be applied to a wide range of experiments. Currently, a variety of different remote control experiments are under development. These experiments will offer an interactive teaching and demonstration facility to students over different applications in the field of automatic control.

Acknowledgment

This work has been performed within the project "E-learning Distance Interactive Practical Education (EDIPE)." The opinions expressed by the authors do not necessarily reflect the position of the European Community, nor does it involve any responsibility on its part.

References

[1] P. Horacek, "Laboratory experiments for control theory courses: A survey," *Annu. Rev. Control*, vol. 24, pp. 151–162, 2000.

[2] C. C. Ko, B. M. Chen, J. P. Chen, J. Zhang, and K. C. Tan, "A Web-based laboratory on control of a two-degrees-of-freedom helicopter," *Int. J. Eng. Educ.*, vol. 21, no. 6, pp. 1017–1030, 2005.

[3] M. Casini, D. Prattichizzo, and A. Vicino, "The automatic control telelab—A Web-based technology for distance learning," *IEEE Control Syst. Mag.*, vol. 24, no. 3, pp. 36–44, Jun. 2004.

[4] J. Sanchez, S. Dormido, R. Pastor, and F. Morilla, "A Java/MATLAB-based environment for remote control system laboratories: Illustrated with an inverted pendulum," *IEEE Trans. Educ.*, vol. 47, no. 3, pp. 321–329, Aug. 2004.

[5] A. R. S. Castellanos, L. Hernandez, I. Santana, and E. Rubio, "Platform for distance development of complex automatic control strategies using MATLAB," *Int. J. Eng. Educ.*, vol. 21, no. 5, pp. 790–797, 2005.

[6] B. Duan, K. V. Ling, H. Mir, M. Hosseini, and R. K. L. Gay, "An online laboratory framework for control engineering courses," *Int. J. Eng. Educ.*, vol. 21, no. 6, pp. 1068–1075, 2005.

[7] J. Henry and C. Knight, "Modern engineering laboratories at a distance," *Int. J. Eng. Educ.*, vol. 19, no. 3, pp. 403–408, 2003.

[8] T. Kikuchi, S. Fukuda, A. Fukuzaki, K. Nagaoka, K. Tanaka, T. Kenjo, and D. A. Harris, "DVTS-based remote laboratory across the Pacific over the gigabit network," *IEEE Trans. Educ.*, vol. 47, no. 1, pp. 26–32, Feb. 2004.

[9] R. Pastor, J. Sanchez, and S. Dormido, "A XML-based framework for the development of Web-based laboratories focused on control systems education," *Int. J. Eng. Educ.*, vol. 19, no. 3, pp. 445–454, 2003.

[10] A. Valera, J. L. Diez, M. Valles, and P. Albertos, "Virtual and remote control laboratory development," *IEEE Control Syst. Mag.*, vol. 25, no. 1, pp. 35–39, Feb. 2005.

[11] R. Šafarič, M. Debevc, R. M. Parkin, and S. Uran, "Telerobotics experiments via Internet," *IEEE Trans. Ind. Electron.*, vol. 48, no. 2, pp. 424–431, Apr. 2001.

[12] R. Marin, P. J. Sanz, P. Nebot, and R. Wirz, "A multimodal interface to control a robot arm via the Web: A case study on remote programming," *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1506–1520, Dec. 2005.

[13] H. Temeltas, M. Gokasan, and S. Bogosyan, "Hardware in the loop robot simulators for on-site and remote robotics education," *Int. J. Eng. Educ.*, vol. 22, no. 4, pp. 815–828, 2006.

[14] C. S. Tzafestas, N. Palaiologou, and M. Alifragis, "Virtual and remote robotic laboratory: Comparative experimental evaluation," *IEEE Trans. Educ.*, vol. 49, no. 3, pp. 360–369, Aug. 2006.

[15] *DSP-Based Remote Control Laboratory Web Page*. [Online]. Available: http://remotelab.ro.feri.uni-mb.si

[16] M. Čurkovič, *DSP-2 User's Manual*. Maribor, Slovenia: Fac. Electr. Eng. and Comput. Sci., Univ. Maribor, 2001.

[17] M. Čurkovič and D. Hercog, *DSP-2 Robotic Controller User's Manual*. Maribor, Slovenia: Fac. Electr. Eng. and Comput. Sci., Univ. Maribor, 2006.

[18] D. Hercog, *DSP-2 Library for Simulink User's Manual*. Maribor, Slovenia: Fac. Electr. Eng. and Comput. Sci., Univ. Maribor, 2006. ver.2.0.

[19] D. Hercog and K. Jezernik, "Rapid control prototyping using MATLAB/Simulink and a DSP-based motor controller," *Int. J. Eng. Educ.*, vol. 21, no. 4, pp. 596–605, 2005.

[20] E. Urlep, *DSP Terminal User's Manual*. Maribor, Slovenia: Fac. Electr. Eng. and Comput. Sci., Univ. Maribor, 2005. ver.1.5.

[21] D. Hercog, *ComVIEW: LabVIEW Toolkit for DSP-2 Control Systems*. Maribor, Slovenia: Inst. Robot., Fac. Electr. Eng. and Comput. Sci., Univ. Maribor, 2006.

[22] National Instruments, *LabVIEW User Manual*, 2006.

[23] J. M. M. Ferreira and A. C. Cardoso, "A Moodle extension to book online labs," *Int. J. Online Eng.*, vol. 1, no. 2, pp. 1–4, 2005.

[24] D. Hercog, B. Villaca, B. Gergič, and M. Terbuc, "Moodle booking system for remote experiments," in *Proc. Int. Symp. Remote Eng. Virtual Instrum.*, 2006, pp. 1–8.

[25] C. Salzmann, D. Gillet, and P. Huguenin, "Introduction to real-time control using LabVIEW with an application to distance learning," *Int. J. Eng. Educ.*, vol. 16, no. 3, pp. 255–272, 2000.

[26] Webcam Corp., *Webcam 1-2-3*. [Online]. Available: http://www.webcam123.com

[27] B. Gergič and D. Hercog, "Virtual instruments for remote experiments visualization," in *Proc. Int. Symp. Remote Eng. Virtual Instrum.*, 2006, pp. 1–6.

[28] S. E. Poindexter and B. S. Heck, "Using the Web in your courses: What can you do? What should you do?" *IEEE Control Syst. Mag.*, vol. 19, no. 1, pp. 83–92, Feb. 1999.

[29] S. D. Bencomo, "Control learning: Present and future," *Annu. Rev. Control*, vol. 28, no. 1, pp. 115–136, 2004.

Darko Hercog (S'03) received the B.Sc. degree in 2001 from the Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia, where he is currently working toward the Ph.D. degree in electrical engineering.

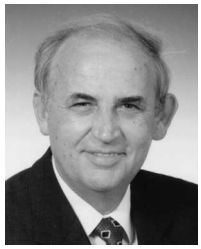
His research interests include real-time systems, digital control implementation, rapid control prototyping, remote control, and virtual instrumentation.

Bojan Gergič (M'95) received the B.Sc., M.Sc., and Ph.D. degrees from the University of Maribor, Maribor, Slovenia, in 1993, 1996, and 2000, respectively, all in electrical engineering.

Since 1993, he has been an Assistant Professor with the Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include virtual instrumentation, digital signal processing, machine vision, and data compression.

Suzana Uran (M'95) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of Maribor, Maribor, Slovenia, in 1985, 1989, and 1998, respectively.

Following the receipt of the B.Sc. degree, she became a Researcher in the Faculty of Electrical Engineering and Computer Science, University of Maribor, where she is currently an Assistant for Robotics and Automation. Her research interests include robot force/position control, telerobotics, sliding-mode control, robot dynamics, robot kinematics, and mechatronics design.



Karel Jezernik (SM'04) received the B.Sc., M.Sc., and Dr.Eng. degrees in electrical engineering from the University of Ljubljana, Ljubljana, Slovenia, in 1968, 1974, and 1976, respectively.

In 1976, he joined the University of Maribor, Maribor, Slovenia, and in 1985, he became a Full Professor and the Head of the Institute of Robotics, Faculty of Electrical Engineering and Computer Science. He is a Consultant on industrial servo control systems and other control and computer applications. His research and teaching interests include automatic control, robotics, power electronics, and electrical drives. His current projects in these areas are high-precision tracking control in machine tools, DD robots, and robust torque control in EVs.