"Lab@Home": An Internet-Based Real Laboratory for Distance Control Education

N. Sepehri , S. Onyshko, W. Lehn, R. Song , Z. Zheng

The University of Manitoba
Winnipeg, Manitoba, Canada R3T-5V6

The Faculty of Engineering at the University of Manitoba completed the development of software drivers and interfacing programs to establish a prototype remote control laboratory station. The station, which can be operated from a distance, is called "Lab@Home" and is becoming a part of core control courses in the Faculty of Engineering. This paper presents the motivation behind this initiation as well as step-by-step development of the test station. Typical results obtained by students are also provided.

1. Introduction

The motivation for the construction of this station comes from the fact that the control laboratories at the University of Manitoba are heavily utilized by as many as 200 students each year. At the present time, we are unable to provide the students enough exposure to the equipment due to the increased number of students as well as limited access time (e.g., the lab can not be open after 6:30 pm, a period in which students may have more time to do the experiments on their own time).

Lab@Home will allow the students to work with the real system by remotely operating it at any time and from any places, therefore, increasing the exposure to the experimental facility. Compared to other valuable attempts to develop a virtual laboratory for control education, whereby the students deal with virtual (simulated) systems, via Internet, here we allow the students to run a real system remotely which is believed to be more effective.

Lab@Home is an interactive setup in which the students can study PID control implementation and tuning issues for typical first- and second- order systems. The goal is to perform actual experiments and relate the findings to the theoretical analyses. The test station allows the students to perform a "do-and-see" approach anytime and from the comfort of their home and thus, they will not be restricted to certain laboratory access times. This gives the students a chance to redo the experiments many times, any time and from any remote computer system. In short, the remote laboratory allows more exposure of the students to an experimental test rig.

"Proceedings of 2002 American Society for Engineering Education Annual Conference & Exposition

Lab@Home is a fully instrumented DC electric motor interfaced to a high-performance computer, which receives position and speed data from the motor and returns control signals to it. A web camera and a microphone are used to transmit sounds and images of the motor in operation. At the same time, on-line plots of the results are shown. This will allow the students to have a feeling of presence.

An interactive program has also been written and all variables of interests are stored and can be graphically shown. These variables include angular rotation, angular speed, motor current, control signal and contribution from proportional (P), derivative (D) and integral (I) control actions. With this setup one can study:

1. open-loop response of a DC motor,
2. PID position or velocity control,
3. effect of sampling time and,
4. effect of stick-slip friction or signal saturation on control performance.

The organization of this paper is as follows. First the descriptions of the test station, which include hardware design and software tools, are detailed. Next, typical exercises that have been performed by the students are outlined. Finally, conclusions are provided with some discussion on possible use of the test station towards distance engineering education.

2. Description of the System and Screen Shots

As illustrated in Fig. 1, several hardware devices are hooked up together to make the Lab@Home remote control test station. The system to be controlled is a DC motor which is equipped with an optical encoder and a tachometer for measuring the rotational angle and velocity, respectively. The motor is connected to a computer running Windows 2000 [1] (the computer is hereafter called 'server') via two interfacing cards: (*i*) a DAS16-F data acquisition board, equipped with A/D and D/A units, to read motor current (as voltage across a 1Ω resistor) and the output of the tachometer, and to transmit the control signal, and (*ii*) a Keithley-M5312 quadrature encoder board which reads the increments in the rotational angle transmitted from the optical encoder attached to the motor shaft. A video camera and a microphone are also connected to the server, which are responsible for sampling video and audio data. The user requires only a personal computer (the computer is hereafter called 'client') running Windows 95 or higher with Internet connection.

The Lab@Home software package has been designed in client/server architecture with multi-user support capability using Visual C++ [2].

2.1 Server

There are three main tasks for the server: (*i*) to transfer the video and audio stream to clients, (*ii*) to communicate and process connection requests from clients, and (*iii*) to control the motor and sample the output signals. The schematic connection between server and DC motor is illustrated in Fig. 2. Task (*i*) is fufilled by Microsoft NetMeeting [3]. The other two tasks are performed simultaneously; thus, the software on the server side is programmed using multithreading. There are two threads running: the main thread performs task (*ii*), and the second one performs task (*iii*).

Once started, the main thread keeps listening to incoming connection requests. Each connected client will be put into a queue. The user of the first client operates the equipment for a specified time

(adjustable on the server). The other users will have to wait until the one(s) ahead of them run out of time or quit the experiment. Dead links are detected and are removed from the queue automatically. The main thread is also responsible for transmitting data for "on-line display" on the user's monitor.

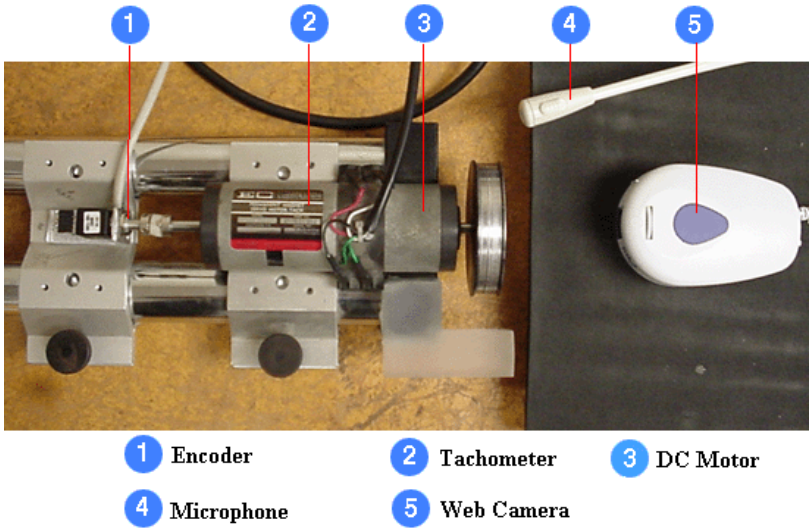

Fig. 1 General view of the experimental device.

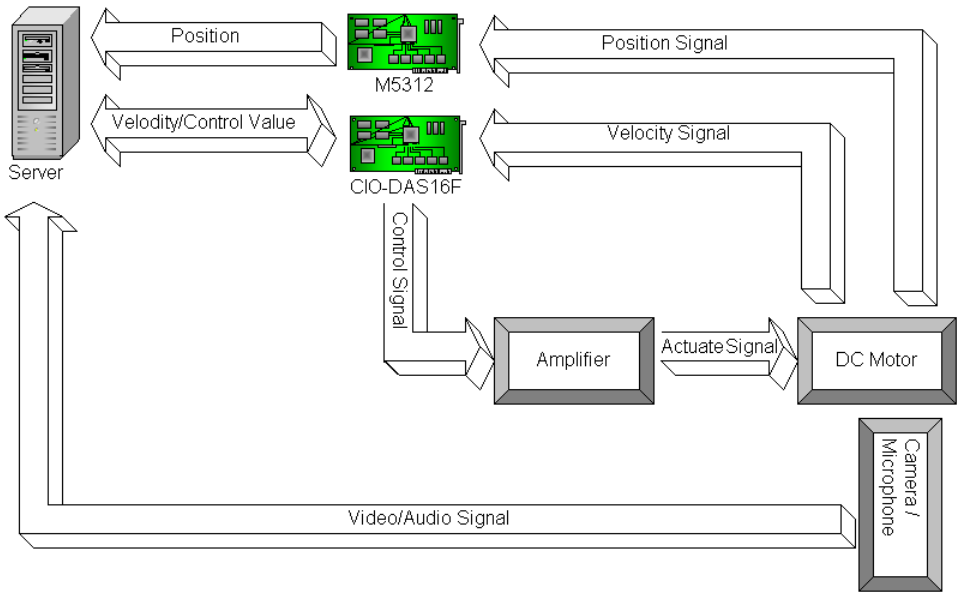

Fig. 2 Hardware configuration of sensory and control.

The second thread is in charge of the motor control and data collection. When the user starts an experiment, this thread is generated and is configured with the arguments such as control type, control parameters, sampling frequency and duration defined by the user. It samples the outputs from the device, and calculates and transmits the control signal. Once the experiment is completed or is terminated, this

thread will notify the main thread. The main thread will then send a message to prompt the user to save the data file. If the user decides to save the data, the data file will be transferred from the server to the corresponding client. The data file on the server will be overwritten during the next experiment.

In order to support the "on-line display", the main thread should constantly receive up-to-date data from the second thread. For this purpose, the second thread writes the current data to a specific memory block and the main thread reads them out. Therefore, this memory block is to be accessed by the two threads. In order to prevent a scenario whereby the main thread begins its reading before the second thread finishes its writing, the memory block is programmed to be 'thread-safe', meaning that whenever a thread performs an operation on it, the other thread can not access it.

2.2 Client

The client is the user's computer. It communicates with the server, collects user's inputs and processes the experimental results. The interface is a user-friendly control panel consisting of five areas: a menu bar, a status bar, an information board, a display area and a central control panel (see Fig. 3).

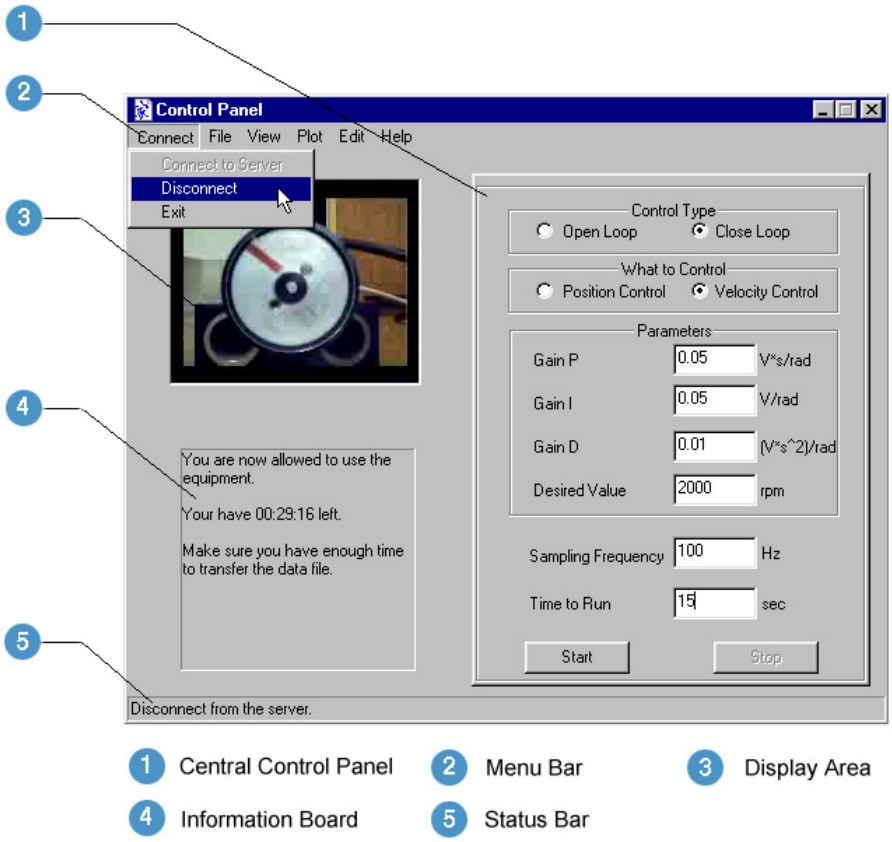

Fig. 3 Screen shot of control panel.

The menu bar allows direct access to all functions: connect to or disconnect from the server, open/save data files, show/hide on-line display of real-time experimental responses and many more. The status bar

displays brief explanations as the user's mouse moves over the menu commands. This is to help the user become familiar with the functions quickly. The information board provides the user with the server's status. During the waiting mode, it shows the number of users ahead and the approximate waiting period. During the experiment, it displays the remaining time for the user to complete the experiment.

When it is time for the user to do the experiment, she/he is notified by a sound. At the same time the front view video of the DC motor is shown on the display area, which is achieved using an embedded NetMeeting component object [4-6]. In the central control panel, the user firstly selects the control type and then fills the required control parameters in the corresponding fields. When the user clicks the 'Start' button, the parameters are checked. If any invalid parameter is detected, an error message prompts the user to make a correction. After the validation, these parameters are sent to the server for the experiment. The user can interrupt the experiment at any time by clicking the 'Stop' button.

The client also has a built-in strip chart display (an ActiveX control [7]) which acts as a real-time response monitor and a plot generator. Fig. 4 shows a screen shot of the on-line display. Provisions have been built into the program to allow the user to scroll, zoom, change the settings or save the plots in an Enhanced Meta File (EMF) format which can later be easily embedded in any document.

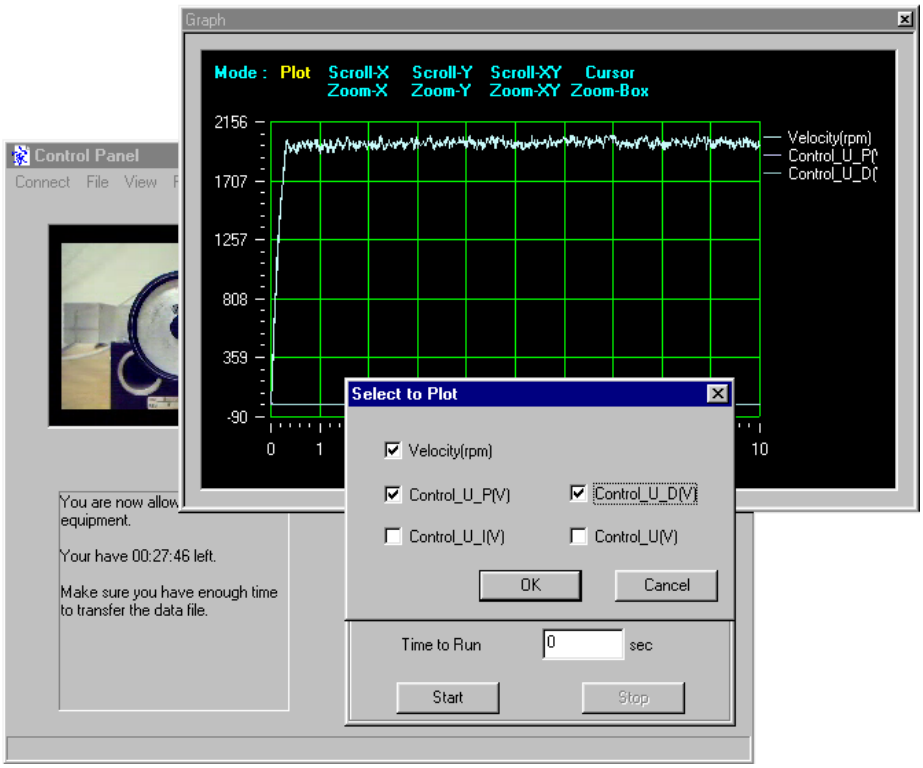

Fig. 4 Screen shot of on-line display system.

2.3 Communication between Server and Client

For client and server to understand each other, the information exchanged between them is packed in a predefined format to form a message. Each message consists of two parts, header and body. The header

gives the receiver a hint on how to unpack the message. For example, a typical message that is sent from client to server to perform an experiment is shown below:

5
10
10000
1
90
0.1
0.3
0.01

where 5 is the message header indicating that the user initiates an experiment. The remaining part of message is the body that consists of the following parameters: sampling period, $T_s = 10$ ms, duration of experiment, $T = 10000$ ms, type of control action (in this example '1' indicates position control), desired set point, $\theta^d = 90^o$, and the values of the proportional, integral and derivatives gains $K_p = 0.10$ V/rad , $K_i = 0.30$ V/(rad s) , $K_d = 0.01$ V/(rad/s), respectively. The messages are embedded into IP Internet Protocol (IP) packets and then delivered to the receiver through the Internet using TCP/IP protocol.

3. Sample Experiments and Results

In this section, typical laboratory experiments, performed by students, are presented. In order to assist the students to make the most out of Lab@Home, we provide some suggestions without restricting them from testing their own ideas.

3.1 Open-Loop Velocity Test

In this set of tests, the students are encouraged to apply various motor voltages to the DC motor and observe the responses. In particular, they can observe how the motor current changes with input voltage. They can: (*i*) relate the steady-state motor current to the disturbance in the form of friction, (*ii)* investigate the nature of the friction (viscous versus dry), and (*iii*) determine the sampling period from the stored data and relate it to the one they specified. Fig. 5 shows motor speed and current responses to a 2-Volt input signal. The students are expected to relate their observations to the theoretical results discussed in the class and explain the differences.

3.1 Closed-Loop Velocity Control

The students are encouraged to first apply proportional control and find the best performing gain (according to their judgement). They are then asked to: (*i*) increase the value of the proportional gain and study the effect of the control signal saturation on the response, (*ii*) reduce the sampling frequency and study the response for a given proportional gain, and (*iii*) add an integral control to see its effect on the response. Figs. 6 and 7 show typical results. Fig. 6 shows the response to a proportional control, which resembles a typical first-order system response. The effect of adding an integral control action is shown in Fig. 7; it is seen that the steady-state error is eliminated. The use of an integral control action, however, would increase the system's order by one. Depending on the gains and the damping present in the system, the transient response may assume many different forms. The students can further explore this point.

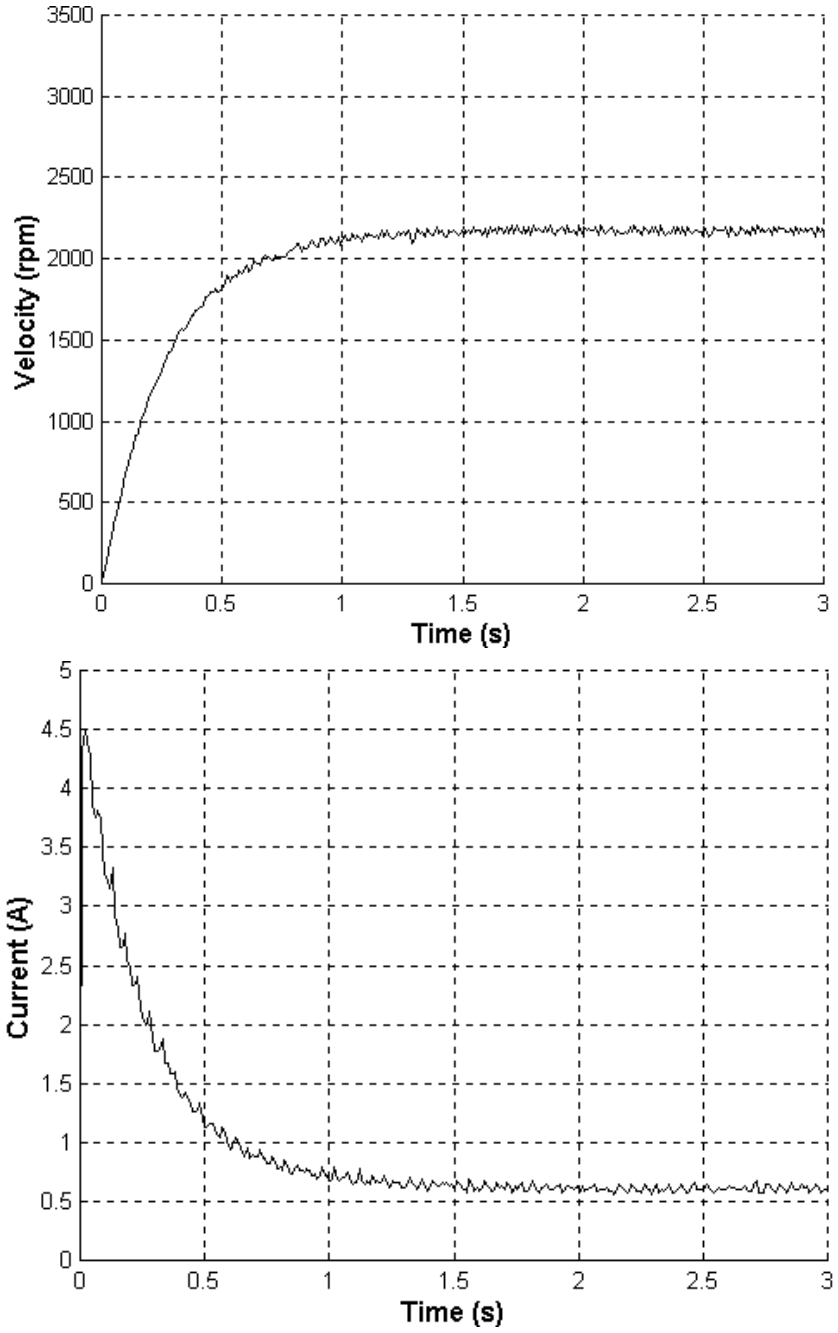

Fig. 5 Motor speed and current responses to a 2-Volt input signal.

3.3 Closed-Loop Position Control

For this set of tests, the students are asked to apply and study the response of a PID control scheme to positioning the DC motor. In particular, they are encouraged to: (*i*) experimentally investigate the effects of increasing the proportional gain, or adding a derivative control action on the response, (*ii*)

exploring the source(s) of steady-state position errors, and (*iii*) observe the effect of adding an integral

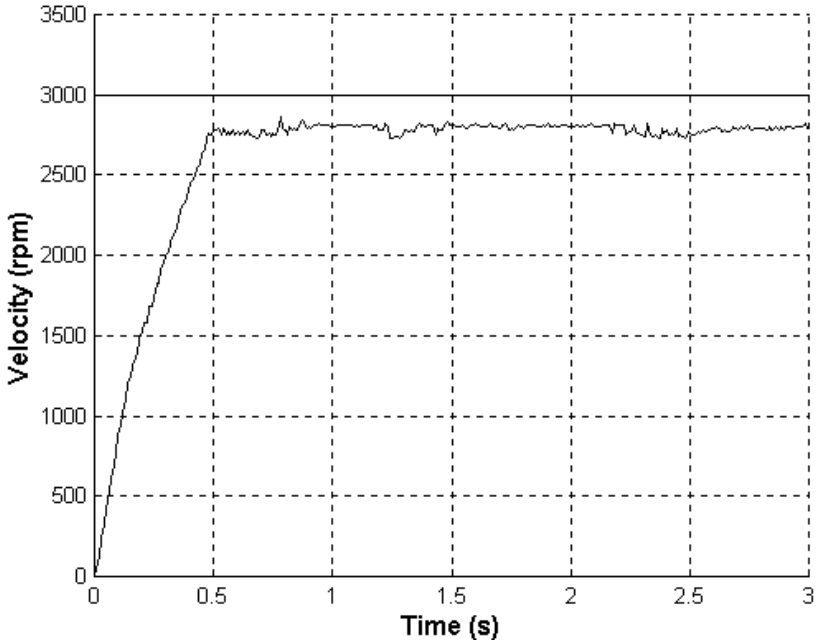

Fig. 6 Proportional velocity control response;
set-point= 3000 rpm, K_p=0.1 V/rad; sampling period = 10 ms.

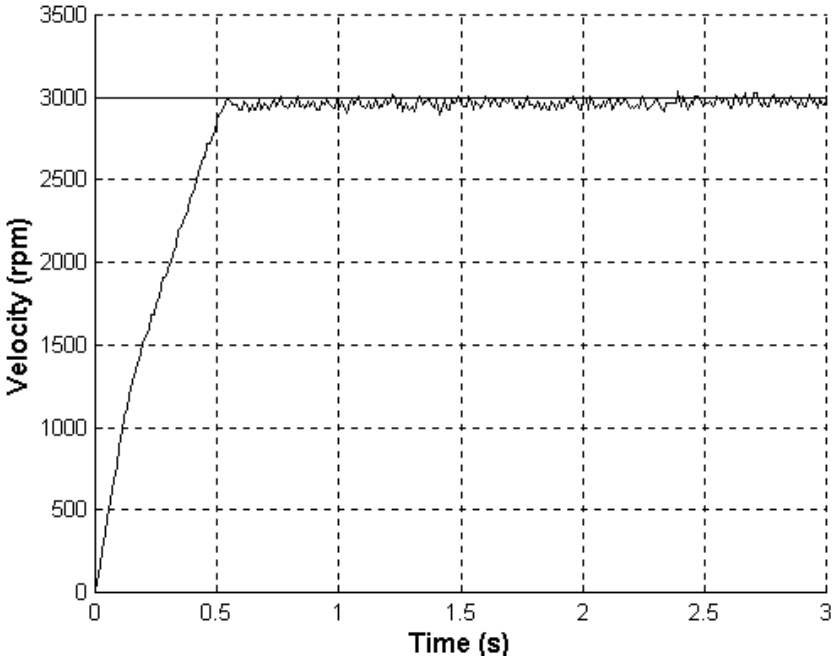

Fig. 7 Effect of adding an integral control action to the system in Fig. 7; K_i=0.03 V/(rad s).

control action. Figs. 8, 9 and 10 show typical results. Fig. 8 shows that with a proportional control scheme, the overall system possess characteristics of a second-order system. The non-zero steady-state error is due to the disturbance introduced by stick friction acting on the motor shaft. By repeating the same experiment, the students observe that the amount of steady-state error (characterized by the stick friction) can vary. Fig. 9 shows the stabilizing effect of adding a derivative control action (reduced overshoot) on the transient behavior. It is also apparent that the addition of derivative control action did not improve the offset; the dry friction inherently present in the system precludes the possibility of achieving the desired position. Further experiments could be performed to observe the effect of adding an integral control action. Ideally, integral control eliminates steady-state errors. The students can easily observe that this can not be always achieved in practical systems. Particularly, the students observe the interesting phenomenon of hunting about the desired position, which can happen when stick and slip frictions have different magnitudes. Fig. 10 shows a typical result.

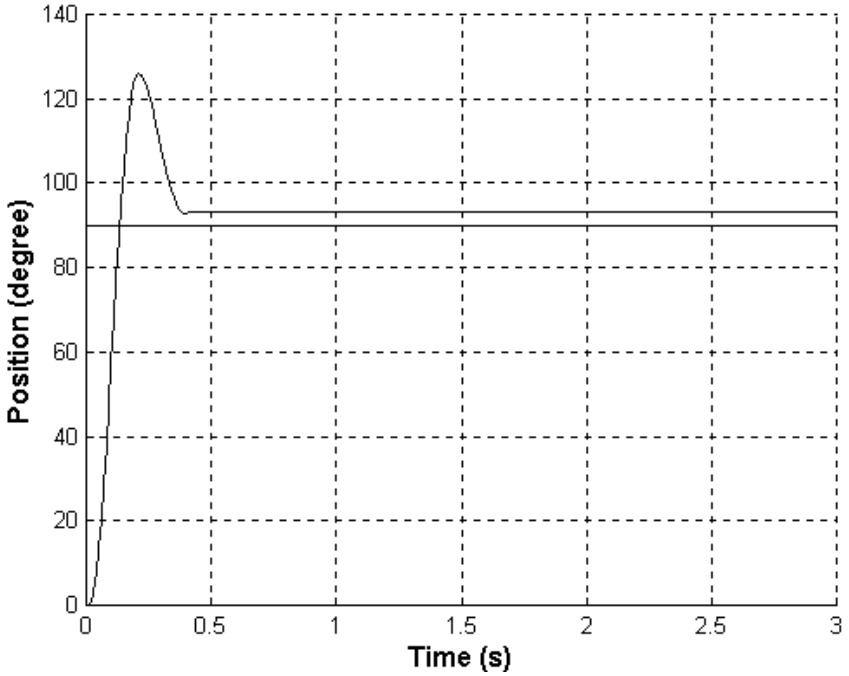

Fig. 8 Proportional position control response;
K_P=0.5 V/rad ; desired angle=90°; sampling period = 10 ms.

4. Concluding Remarks

In this article, we have described the development of a laboratory test station, which can be operated remotely. The test station is intended to be used for control engineering education. It runs 24 hours without attendance and is easily accessible and operational through the Internet connection. Furthermore, the system is totally interactive and user friendly.

The hardware used in this station consists of a DC motor equipped with an optical encoder and a tachometer for measuring rotational angles and speeds, a Pentium-III computer, a PC camera, a DAS16-F analog-to-digital board and a quadrature encoder board. The software package has been

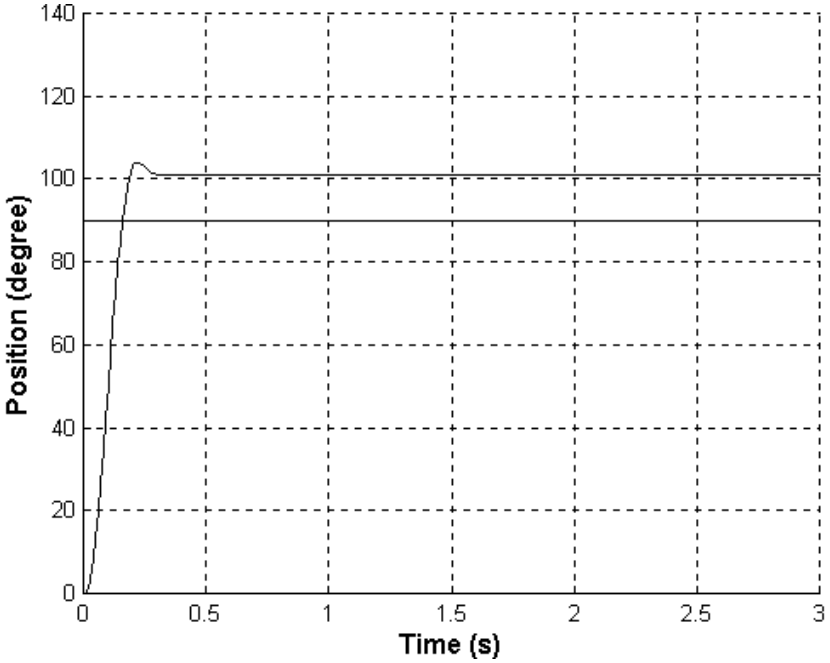

Fig. 9 Effect of adding a derivative control action to the system in Fig. 10; K_d=0.01 V/(rad/s).

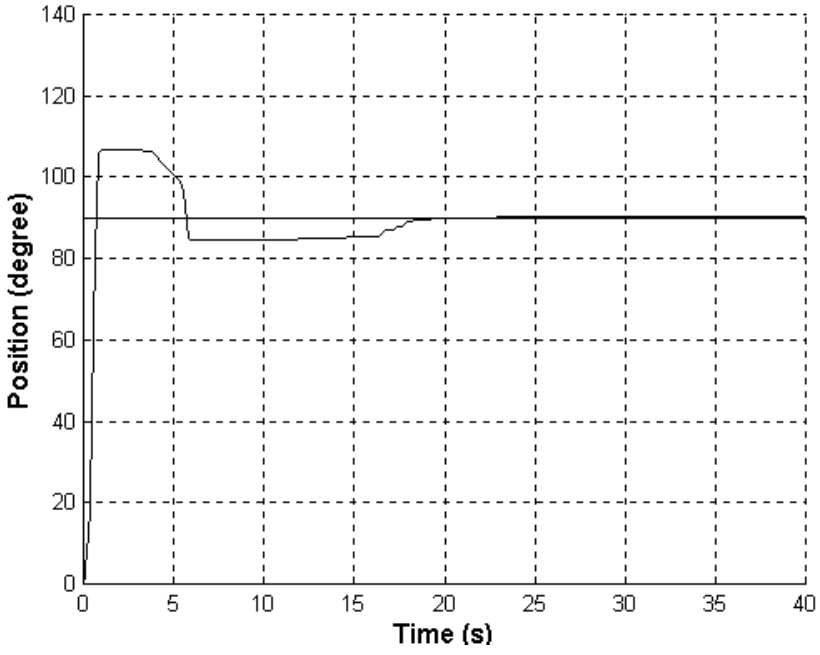

Fig. 10 Hunting due to the existence of dry friction in a PID position control response; K_P=0.1 V/rad, K_d=0.01 V/(rad/s), K_i=0.3 V/(rad s), sampling period = 10 ms.

developed in-house, using the visual C++ programming environment and has incorporated component object model, multi-thread programming, socket programming and ActiveX control. The Lab@Home station is expected to enhance the control laboratories in the Faculty of Engineering at the University of Manitoba. Considering the large number of engineering students who are enrolled in core control courses, this innovative approach to teaching is very promising. Particularly, the new facility allows the students to perform real experiments any time and from any remote computer system; thus, they are not restricted to certain laboratory access times. This initiation is also in-line with our continuous effort to promote and upgrade the level of engineering education at our university and at the same time to improve the curriculum to meet the requirement for future engineers. Furthermore, there is a potential for the facility to serve other institutions across Canada.

References

1. *Inside Microsoft Windows 2000*, Microsoft Corporation, 2000.

2. *MSDN Library Visual Studio 6.0 Release*, Microsoft Corporation.

3. Summers, R., *Official Microsoft NetMeeting 2.1 Book*, Microsoft Press, 1998.

4. *The Component Object Model Specification*, Version 0.9, Microsoft Corporation, 1995.

5. *Windows NetMeeting 3 SDK*, Microsoft Corporation, http://www.microsoft.com/windows/ NetMeeting/Authors/ SDK default.ASP, 1999.

6. Eddon, G. and Eddon, H., *Inside COM+ Base Services*, Microsoft Programming Series, 1999.

7. *Iocomputer Software; Getting Started Guide*, http://www.iocomp.com/, 2001.

Biography

NARIMAN SEPEHRI is a Professor in the Department of Mechanical and Industrial Engineering and is the corresponding author for this paper (email: nariman@cc.umanitoba.ca).

S. ONYSHKO is a Professor in the Department of Electrical and Computer Engineering.

W. LEHN is a Professor in the Department of Electrical and Computer Engineering.

R. SONG and Z. ZHENG are M.Sc. graduate students , both from the Department of Mechanical and Industrial Engineering.