A MATLAB-Based Control Systems Laboratory Experience for Undergraduate Students: Toward Standardization and Shared Resources

Warren E. Dixon, *Member, IEEE,* Darren M. Dawson, *Senior Member, IEEE,* B. T. Costic, and Marcio S. de Queiroz, *Member, IEEE*

***Abstract*—This paper seeks to begin a discussion with regard to developing computer aided control system design (CACSD) tools to promote undergraduate controls laboratory development. The advocated CACSD design tools are based on the popular, commercially available MATLAB environment, the Simulink toolbox, and the Real-Time Workshop toolbox. This paper describes how these tools can be utilized to address several issues that are confronted by control systems educators including: standardization, budget constraints, and limited resources. Specifically, by confronting the standardization issue, the following advantages will be realized for laboratory development: 1) the required computer hardware will be low cost; 2) commercially available plants from different manufacturers can be supported under the same CACSD environment with no hardware modifications; 3) both the Windows and Linux operating systems can be supported via the MATLAB based Real-Time Windows Target and the Quality Real-Time Systems (QRTS) based Real-Time Linux Target; and 4) the Simulink block diagram approach can be utilized to prototype control strategies, thereby, eliminating the need for low level programming skills. The advantages related to standardization of the CACSD design tools will enable educators to confront the additional budget constraint and limited teaching resources issue by facilitating: 1) the sharing of laboratory resources within each university (i.e., between departments); 2) the development of Internet laboratory experiences for students (i.e., between universities); and 3) the initiation of an Internet-based archive of laboratory tutorials and Simulink files for in-house developed plants and commercially available plants.**

***Index Terms*—Control systems laboratory, Internet-based control, real-time control, Simulink.**

Manuscript received July 3, 2001; revised January 9, 2002. This work was supported in part by the Eugene P. Wigner Fellowship Program of the Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC, for the U.S. Department of Energy (DOE) under Contract DE-AC05-00OR22725 and in part by the U.S. DOE Environmental Management Sciences Program Project ID 82797 at ORNL, and in part by U.S. National Science Foundation Grant DMI-9457967, ONR Grant N00014-99-1-0589, a DOC Grant, and an ARO Automotive Center Grant.

W. E. Dixon is with the Engineering Science and Technology Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6305 USA (e-mail: dixonwe@ornl.gov).

D. M. Dawson is with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634 USA (e-mail: ddawson@ces.clemson.edu).

B. T. Costic is with Boeing Satellite Systems, El Segundo, CA 90009 USA (Bret.T.Costic@boeing.com).

M. S. de Queiroz is with the Department of Mechanical Engineering, Louisiana State University, Baton Rouge, LA 70803-6413 USA (e-mail: dequeiroz@alpha2.eng.lsu.edu).

Publisher Item Identifier S 0018-9359(02)05049-5.

I. Introduction

BECAUSE of the multidisciplinary nature of the field, a consensus exists among control systems educators that laboratory experiences are particularly important with regard to the teaching of control systems [1]. Unfortunately, recent studies have revealed a lack of formal experimental control education in many universities. Specifically, a control systems report card from industry [1] showed relatively low ratings for engineering graduates in attributes, such as laboratory and hands-on experiences. The Accreditation Board for Engineering and Technology (ABET) 2000 criteria have also recognized that a well-developed laboratory component is a key for preparing a modern technological workforce. In addition, the recent National Science Foundation (NSF)/Control Systems Society (CSS) workshop on control education [2] acknowledged the importance of laboratory experiences with regard to exposing students to broader design issues that range from problem specification to hardware implementation and economic considerations. To be more specific, the NSF/CSS workshop report [2] forwarded the following statement as one of its primary recommendations: "*Promote control systems laboratory development … and make experimental projects an integral part of control education for all students. …*" Unfortunately, despite the fact that ABET, NSF, and most faculty agree that the control laboratory experience is important, a strong emphasis on laboratory-based education is not commonplace among academic institutions (as evident by the low ratings of the hands-on abilities of engineering graduates reported in [1]).

Given the above impetus, control systems educators have recently been investigating how technological advances can impact laboratory-based education. That is, advances in hardware and software technologies have generated much discussion with regard to the nature and development of an undergraduate control systems laboratory experience [3]–[6] and with regard to remote access to the control systems laboratory [7]–[10]. In summary, recent publications regarding control systems education point out that because of the advent of high-speed, low-cost, real-time computing platforms, the development of control systems laboratory hardware is now becoming cheaper and more accessible. Moreover, developments in automated code generation allow users to create real-time code from graphical, control system simulation software (e.g., MATLAB/Simulink).

These tools enable educators and students to focus on control system design, implementation, and evaluation rather than on time-consuming, low-level programing (i.e., real-time programing that is required to interface with control plants is often beyond the scope of undergraduate control courses). In addition, a variety of educational/research plants are now commercially available from different vendors that capture the multidisciplinary nature of the field (e.g., robot manipulator, inverted pendulum, magnetic levitation, water tank, pH control rig, helicopter, ball and beam, dc motor). However, despite the recent interest in incorporating technological advancements into control systems laboratory courses, a control laboratory experience for a typical undergraduate is not commonplace. This fact may be credited to a number of issues; however, in this paper, the following barriers are addressed: 1) lack of standardized hardware/software; 2) budget constraints; and 3) limited teaching resources.

This paper describes computer aided control system design (CACSD) software tools and how they can be applied to overcome the aforementioned obstacles that impede undergraduate control systems laboratory experiences. Specifically, the development of a set of standardized CACSD software tools are described that allow a student to prototype controllers for a variety of manufacturers' supplied plants using a student-friendly Simulink/Real-Time Workshop (RTW) front-end. Based on the advantages of confronting the standardization issue, potential solutions to the budget constraint and limited teaching resource issues are provided. Specifically, some future directions are described with regard to control system laboratory development that will improve faculty productivity by fostering cooperation among various academic departments and institutions, and with regard to developing new material for control systems education. For example, the creation of a future Internet-based archive of laboratory tutorials and Simulink files for inhouse developed plants and commercially available plants is discussed. Moreover, some possible technical directions that can be pursued with regard to Internet laboratory experiences for students who attend institutions that do not have direct access to control equipment are also highlighted.

This paper is focused solely on CACSD design tools that use a Simulink/RTW interface and do not require a digital signal processing (DSP) board. The reason that this paper is targeted only at CACSD tools that use a Simulink/RTW interface is that, based on conversations with leading manufacturers of undergraduate control equipment, it seems that the future undergraduate laboratory experience will be MATLAB/Simulink-based. The reason for excluding DSP-based tools is that a DSP-based architecture cannot be easily interfaced with current plants (i.e., the CACSD must be able to interface with commercially available plants without requiring hardware modifications). Moreover, DSP-based control architectures tend to be excessively expensive and complicated when compared to a non-DSP, personal computer (PC) based solution. Hence, based on the refined class of CACSD tools that are examined in this paper, software environments such as Advanced Realtime Control Systems, dSPACE, or the laboratory design discussed in [11] will not be examined. This paper also excludes the Extended Real-Time Toolbox, since it does not seem to guarantee some measure of hard real-time performance. Information regarding the outstanding products made by Opal-RT has also been omitted, since it is highly unlikely that an undergraduate laboratory would be constructed around a sophisticated product that utilizes two separate PCs as the hardware platform as well as two different operating systems (i.e., QNX and Win32). This paper is organized as follows. In Sections II–IV, the issues of standardization, budget constraints, and limited resources are addressed, respectively. Concluding remarks are given in Section V.

II. Standardization Issue

In this section, issues related to the standardization of the computational platform and the hardware/software interface are described. As a means to overcome the standardization issues related to the hardware/software interface, a set of CACSD tools is advocated. A discussion is provided that demonstrates how the CACSD tools have been utilized at Clemson University, Clemson, SC, and comments are provided regarding the application of the CACSD to inhouse developed experimental plants (e.g., the pendubot [12]), cost comparisons, and real-time performance.

A. Computational Platform

One of the first standardization issues to be addressed by control educators is the computational engine. Fortunately, because of factors such as cost, ease-of-use, and compatibility, a consensus is slowly forming among control educators regarding the use of the PC (excluding a DSP board) as the standard computational engine. The feasibility of using a PC without requiring an add-on DSP board for control applications seems to have occurred sometime around the 1993 timeframe as a result of the innovative work by Quanser; however, it took some time for many control engineers to become comfortable with the concept. In fact, only until around 1997 did the use of standard PC hardware (i.e., without the requirement for a DSP), in conjunction with high level software language tools, become a more widely accepted method for implementing sophisticated control strategies in real-time. Although the standard PC hardware is becoming a standard among control educators, questions regarding the standardization of a software front-end and educational plant hardware are still open (this is mainly fueled by educational plant manufacturers who often use custom software front-ends and unique hardware interfaces as a means to ensure profit share).

B. Hardware/Software Interface

Since the PC is becoming the standard computational engine for control systems laboratories, the biggest obstacle to standardized control systems laboratories is the differences in various hardware/software interfaces. For example, while Quanser has been at the forefront of developing a standard front-end for laboratory experiments based on widely utilized educational tools such as Simulink, other equipment manufacturers have slowly embraced this concept by developing plants that utilize proprietary hardware/software components. That is, each laboratory experiment manufacturer typically utilizes a different software environment, interface hardware, and I/O

board. As a result, undesirable hardware and/or software modifications are often necessary to develop a common laboratory testbed that can be utilized among experiments developed by different vendors because of the incompatibility resulting from the vendor-specific hardware/software interface. To alleviate this problem, Educational Control Products (ECP) and Feedback recently started marketing products with common front-end software environments that exploit the use of Simulink, Real-Time Windows Target (RTWT) and Real-Time Linux Target (RTLT), and general I/O boards (to facilitate a common hardware interface). Recently, Quanser developed the WinCon software environment to provide a Simulink/RTW front-end for Win32 operating systems (www.microsoft.com); unfortunately, WinCon cannot be readily used with plants sold by other manufacturers (e.g., ECP, Feedback, Mechatronic Systems) without back-engineering their electronic interfacing for use with a specific I/O board or writing device drivers for other I/O boards. This retrofit-based approach often results in a barrier for many departments because it requires a certain level of programing and/or electronics expertise that simply may not exist. In addition, a homegrown retrofit-based approach is often time consuming and unreliable, often requiring additional staff to be employed to provide technical assistance. One approach that some control educators have taken to overcome the aforementioned compatibility issues is to design the control systems laboratory using plants from only one manufacturer (e.g., [3]). However, this approach limits the educational experience to the plants supplied by one manufacturer and does not allow for the flexibility of rotating between a wide range of experiments by various manufacturers or the development of inhouse experiments. Moreover, this approach excludes the development of inhouse testbeds which many educators have developed as a result of: 1) the incompatibility among the leading educational control equipment manufacturers, 2) the disillusionment with some of the commercially available CACSD front-ends, or 3) the interest in examining a plant that is more challenging from an educational and/or research point of view.

C. Standardizing the CACSD

To hurdle the hardware/software standardization obstacles that impede the development of a control systems testbed that incorporates experimental plants in a plug-and-play manner, a software environment is required that provides a low-cost, standardized interface for commercially available plants and/or inhouse developed plants. A hierarchical CACSD environment that meets these requirements is composed of five design tools including: MATLAB, Simulink, RTW, RTLT, and RTWT. Each of these software components can be executed on standard PC hardware running on the Linux (see www.linux.com) or Win32 operating systems. Fig. 1 illustrates the hierarchical structure of the CACSD environment with interfaces to the user and a physical plant. Since MATLAB, Simulink, RTW, RTLT, and RTWT are the components of the CACSD environment, a brief description of each component is given as follows (see also [13]).

MATLAB [14] is a software environment that allows problems and solutions to be expressed in familiar mathematical notations. Numerous toolboxes and other software packages

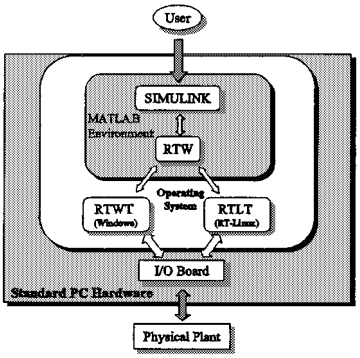

Fig. 1. The CACSD environment.

have been developed for MATLAB to facilitate a variety of engineering and educational tasks such as algorithm development, modeling, simulation, data analysis, visualization, engineering graphics, and application development (including graphical user interface (GUI) development). Simulink [14] is a software package that works in conjunction with MATLAB for modeling, simulating, and analyzing dynamic systems through an intuitive block diagram-based GUI that utilizes various block-set libraries to incorporate preconfigured blocks and connectors by simple drag and drop operations. Scopes and other display blocks allow the user to view the simulation results while the simulation is still running. RTW [14] is an automatic C language, code generator for Simulink that generates C code directly from the Simulink models and automatically constructs a file that can be executed in real-time in various environments. The block diagram interface of Simulink coupled to the RTW code generator allows the user to concentrate on the modeling and control issues as opposed to programing issues (although if a faculty wanted to include programing skills in some laboratory exercises, these skills could be incorporated through the development of S-function[1] blocks).

RTLT [15] is a software package that provides a user with the ability to implement a Simulink block diagram on a standard PC in hard real-time (i.e., provide a deterministic response). Specifically, RTLT is a set of source files, device driver libraries, a template makefile, and a MEX-file [16] interface that uses RTW to automatically generate and compile C code on a PC running RT-Linux from a user-defined Simulink block diagram. During the execution of a Simulink block diagram, RTLT captures sampled data from one or more input channels (e.g., A/D channels, digital lines, and encoder lines) and then provides the data to the block diagram model. The Simulink block diagram model then processes the data accordingly. RTWT [14] is a Win32-based software package that has similar capabilities as RTLT. Specifically, RTWT merges the power of Simulink block diagrams and the C code conversion ability of RTW into one package that is able to implement a control algorithm on Win32 operating systems.

[1]As stated in [14], an S-function is a description of a Simulink block that is encapsulated in a programming language. For example, S-functions can be written in MATLAB, C, C++, Ada, or Fortran. S-functions that are not developed using MATLAB script files can be compiled as MEX-files [16] using the MEX utility described in the MATLAB Application Program Interface Guide and are dynamically linked into MATLAB when needed.

To illustrate the advantages of this CACSD software environment, an example laboratory exercise (stabilization of an inverted pendulum manufactured by ECP) was performed utilizing RTLT. Specifically, after some collaborative efforts with QRTS, a software driver for the ECP I/O board was developed that facilitates control prototyping with a Simulink/RTW front-end. A simple Simulink block diagram was then developed for a proportional derivative controller that forced the inverted pendulum to track a square wave reference signal. The Simulink user interface tools were then used to tune the control gains to achieve the desired response.

The main advantages observed from this process were that the control experiment was implemented in real-time using a low-cost standard PC, and the executable was generated from a Simulink block diagram; hence, low-level programing skills were not required. Motivated by these advantages, other experiments were performed on various plants from various vendors to determine if the CACSD could be used to overcome the standardization problem. Specifically, new Simulink files were developed, and similar experiments were performed using the other plants from ECP (e.g., the Servo Trainer, Rectilinear, and Torsion experiments). This same process was repeated with Feedback plants (e.g., the Helicopter, Magnetic Levitation, Modular Servo, and Pendulum experiments), and a Quanser plant (e.g., the Inverted Pendulum experiment). To illustrate that the above solution to the standardization problem is not limited to the Linux operating system, software drivers were developed in collaboration with QRTS to develop a software interface for RTWT (i.e., a Win32 operating system solution). This approach allowed all of the previous Simulink files developed under the Linux operating system to be reused for controlling the ECP plants, Feedback plants, and the Quanser plant under a Win32 operating system. That is, by using the same Simulink block diagrams with the I/O blocks replaced by appropriate S-function blocks that were developed in collaboration with QRTS, the same experiments were implemented on a PC operating under Windows 98 using RTWT (i.e., the CACSD does not require different operating modes to be selected; rather, the user simply selects the proper I/O block from a library for the Simulink block-diagram). Explicit details regarding the hardware interface procedures, developed Simulink block-diagrams, and experimental results were incorporated in various laboratory instruction manuals that can be freely downloaded from the manuals link of QRTS (see www.qrts.com).

In addition to illustrating the compatibility of the CACSD with inhouse developed plants, experiments were also performed with the Mechatronic Systems, Inc. Pendubot. By leveraging off the previous experiences, a controller was prototyped for the Pendubot under RTLT with the ServoToGo I/O board in a few hours. Because of the availability of the QRTS developed software interface for the ServoToGo I/O board, it would be a trivial matter to run the same experiment under RTWT. Since QRTS and Quanser both supplied solutions for the use of a generic multifunction I/O board for both the Win32 and Linux operating systems (i.e., QRTS supports both the MultiQ and ServoToGo I/O boards while Quanser supports the MultiQ and Keithley–Metrabyte I/O boards), it seems that the standardization issue has been solved by the advocated CACSD even for inhouse developed plants (provided a generic I/O board is utilized).

To interface with the various plants from the different equipment manufacturers and on different operating systems, collaborative efforts were pursued with QRTS, Quanser, Mechatronic Systems, ECP, and Feedback to develop various device drivers. Since the detailed results from these efforts would require an extensive and narrowly focused discussion, these details have been omitted; however, these efforts have resulted in product changes by QRTS, Mechatronic Systems, ECP, and Feedback, to facilitate the standard CACSD environment so that other educators do not have to pursue similar efforts. These product changes include: 1) incorporation of the device drivers in the source code (e.g., QRTS now incorporates the device drivers in the RTLT product, provides S-functions for interfacing with various I/O boards for RTWT, and provides laboratory manuals for examples of how to use the products based on the collaborative efforts), 2) new front-end environments (e.g., ECP and Feedback), and 3) new hardware interface capabilities (e.g., Mechatronic Systems). These product changes were the result of CACSD software manufacturers and manufacturers of educational products that realize the benefits of standardization for the educator and for the manufacturer (i.e., with a common software interface, the competition between manufacturers reduces to the quality and innovation of the physical plant and the associated documentation regarding the use of the product).

Some of the main advantages and disadvantages (with regard to the standardization issue) that were experienced when using the CACSD are summarized below:

1) Standardization of the software interface via Simulink.
    a) *Advantages*: The advantages of having a standardized software interface include: 1) **reduced development time**—the authors did not have to develop their own custom GUI, learn how to use the custom software interface that was supplied by the various plant manufacturers, or be familiar with low-level real-time programing skills; 2) **improved student familiarity**—all of the students that were involved in testing the CACSD with different plants were either familiar with the Simulink interface or quickly became familiar because of the intuitive block-diagram structure; 3) **improved software flexibility**—new experiments (e.g., adaptive nonlinear model-based experiments) could be quickly and easily prototyped (proprietary interfaces are often inflexible and restrictive to customization); 4) **improved operating system support**—the transition from Linux to Windows 98 was seamless; and 5) **improved plant selection**—the choices regarding different educational plants was reduced to which manufacturer developed the best physical plant since the software interface was common.
    b) *Disadvantage*: The user may be limited by the capabilities of Simulink (e.g., the data plotting features) unless time and expertise is available to develop

custom "replacement" blocks (e.g., a user could write code to develop a new data plotting block).

2) Standardization of the hardware interface.
   a) *Advantages*: The advantages of having a standardized hardware interface include: 1) **reduced development time**—a general multipurpose I/O board (e.g., the MultiQ or the ServoToGo) was used to interface with the various manufacturer and inhouse plants; 2) **improved hardware flexibility**—the authors were not constrained to use a specific company's hardware interface (i.e., the manufacturer's I/O board was replaced with an existing I/O board); and 3) **improved plant flexibility**—plants developed inhouse and by multiple manufacturers were easily controlled using a general multipurpose I/O board.
   b) *Disadvantage*—If a user wants to use the I/O board that the educational plant manufacturer provides, device drivers may have to be written or purchased to interface with RTW in the desired operating system; however, as described previously, this disadvantage has recently been mitigated by changes made by various companies.

D. Cost Comparison

For comparison purposes, the cost of the WinCon solution for Win32 operating systems is approximately $1277 US per seat, while the cost of the RTWT solution is approximately $150 US per seat. (All of the price quotes in this paper are calculated based on a classroom kit pricing structure for less than 25 copies and are subject to change.) The cost of a fully supported SimuLinux solution is approximately $527 US per seat, while the cost of a fully supported RTLT solution is approximately $682 US per seat. Based on the above pricing structure, it seems that RTWT may become the real-time computation engine of choice for undergraduate laboratory instruction. That is, while WinCon, SimuLinux, and RTLT have some advantages over RTWT (e.g., RTWT has no guaranteed measure of real-time performance), the pricing scheme may outweigh the disadvantages for undergraduate laboratory instruction. In addition, since MathWorks provides software interfaces for many generic I/O boards that can be used with inhouse developed plants, and several vendors of commercially available plants (e.g., Feedback and ECP) are providing RTWT software interfaces for their equipment, it seems inevitable that RTWT will become the standard real-time engine for undergraduate control laboratories.

E. Real-Time Performance

There is a consensus skepticism about the real-time capabilities of RTWT since no information has been published that ensures some measure of real-time performance of RTWT under Win32 operating systems. To examine the real-time performance of RTWT from a practical point of view, some relatively sophisticated, trajectory tracking, control experiments were recently completed with RTWT. Specifically, the same control experiments were performed using both RTLT and RTWT for a six degree-of-freedom PUMA 560 robot manipulator. After comparing the link position tracking performance for both RTLT and RTWT, no distinguishing differences could be determined. Since RTLT ensures real-time performance by using a hard real-time extension of Linux, it seems that the real-time performance of RTWT may not be a practical issue (especially given the pricing structure as described in the previous section). Perhaps, WinCon (with the appropriate extensions), SimuLinux, and RTLT with their guaranteed hard real-time performance and other advantages will remain attractive alternatives for the control researcher or industrial user who demands hard real-time performance as well as a Simulink/RTW front-end.

III. Budget Constraint Issue

In this section, the use of shared laboratories between various departments and various universities is described with regard to easing the issue of the budget constraint. Included in this discussion is a description of the advantages and disadvantages of a shared laboratory. Finally, a discussion is provided regarding how the advocated CACSD environment can be utilized to overcome some of the obstacles associated with an Internet-based shared laboratory.

A. Shared Laboratories Within a University

Currently, it is quite common for engineering departments (e.g., electrical, mechanical, aerospace, chemical) to simultaneously offer undergraduate control system courses. These courses, although sharing some common theoretical content, are properly adapted to the technical needs of their respective engineering fields [6]. Because of the multidisciplinary nature of control, it seems natural to develop educational control labs that are shared among engineering departments. In addition, the existing paradigm of individual departmental laboratories seems difficult to sustain as a result of the high cost of laboratory equipment (i.e., the plants, oscilloscopes, voltmeters, actuators, sensors, computers, I/O boards, etc.) and the increasing demands on faculty time [6]. That is, funds can be saved because the field of control systems is multidisciplinary in nature. As noted in the NSF/CSS workshop [2], shared laboratories have several financial and pedagogical advantages. For example, shared laboratories: 1) avoid the duplication of equipment, and hence, enable the more efficient use of resources; 2) increase the exposure of students to the multidisciplinary nature of the field; and 3) encourage interaction of faculty and students across disciplines. One recent implementation of the shared laboratory concept that can serve as a model for other universities is the experience instituted in the College of Engineering, University of Illinois at Urbana-Champaign. Specifically, an integrated network of laboratories was designed to service all controls-related courses in the College of Engineering. A detailed description of this experience can be found in [6]. Other educators who have investigated the development of synergistic multidisciplinary "mechatronic" laboratories can be found in [17]–[23].

B. Internet Laboratory Concept

Taking the shared laboratory paradigm a step further, the controls community is also starting to witness a trend toward the development of Internet-based labs [7]–[10]. The idea is to develop laboratory experiments that can be accessed and controlled remotely over the Internet. The primary motivating factor of the Internet laboratory concept is to enhance the accessibility of laboratory facilities for instructors and students. An Internet laboratory experience can be used to accommodate students whose schedules may not conform to the traditional laboratory model or students who require more time to complete laboratory work. The Internet laboratory concept also provides an experimental experience for instructors and students at universities that may lack the inhouse resources. Typical components of an Internet laboratory include [7]: 1) a physical plant to be controlled; 2) a control server computer that computes the control algorithm and handles actuator/sensor signals to/from the plant and all communication with the remote user; 3) a controlling client computer that allows a remote user to operate the plant; 4) an Internet connection to link the client computer to the server computer (e.g., TCP/IP protocol); and 5) audio, video, and/or animation to give the remote user a sense of telepresence in the laboratory.

C. Obstacles Associated With the Internet Laboratory Concept

Although the use of the Internet may save funds with regard to providing a controls laboratory experience for undergraduates, there are some obstacles that impede the development of an Internet-based lab. As described previously, the operation of Internet laboratories requires that the remote user connect to the server computer via a client computer and an Internet connection. Once connected, most of the recently developed remote labs only allow users to send set point commands to the physical plant and perhaps alter the control gain (i.e., the controller structure remains fixed). This arrangement is very restrictive since the student cannot design and test his/her own controller. Ideally, an Internet laboratory should allow the student to design his/her own controller (e.g., allow a student at one university to implement a linear lead-lag controller, while allowing a student at another university to implement a nonlinear controller), upload it to the server computer, and test it on the actual plant. In this scenario, three issues need to be carefully addressed. First, the server computer should have the ability to detect and avoid problems (e.g., mistakes when a user uploads an "unsafe" controller that results in an unstable system or saturated amplifiers). One potential solution of this problem is to incorporate a "simulate first" criterion, where the student's controller must pass a simulation test before the controller is implemented on the actual plant. This option will also free the plant resource for other remote students. Faulty controllers will not occupy the plant, allowing more access for students who have designed successful controllers. Second, the Internet laboratory system should only allow the experiment to run for a preset (by the host) duration. This requirement is necessary because some students may want to run an experiment for excessive periods of continuous time (because of excitement over success, waiting for a lab teaching assistant to check that the experiment is successful, etc.), blocking other remote students from accessing the plant. Potentially, if a host experiment becomes excessively popular among remote institutions, a scheduling procedure (potentially at a cost to the remote institution) may have to be incorporated. Third, to the greatest extent possible, the Internet laboratory system should avoid requiring the installation of special software on the client computer since compatibility problems may arise and discourage the student from making the effort necessary to get the experiment working. Some Internet-based robotic systems work using a web browser as the human interface for the remote computer system [8]. Although this solution eliminates the need for downloading specialized software, it limits the prototyping of new control strategies. Another aspect requiring further investigation is that, because of Internet traffic and bandwidth, one must take care in developing a system to provide telepresence features that augment the Internet laboratory experience. Previous Internet-based robots (e.g., [24]) have only given visual feedback through a web browser of the robot's status which is updated every 5–10 s. This slow visual update detaches the end user from a feeling of "being there." That is, it seems that the present speed of the Internet requires some sort of hybrid approach that provides a limited "low-resolution" live video of the experiment followed by a "high-resolution" downloadable version of the video.

D. Internet Capabilities of the CACSD

As explained previously, RTLT and RTWT are components of the advocated CACSD software environment that allow the user to implement a Simulink block diagram in real-time on standard PC hardware, using the RT Linux/Win32 operating systems. Presently, RTLT provides Internet-based control capabilities out of the box. Specifically, RTLT's Internet capability is achieved through the use of the X Window system, which implements a protocol for network-based windowing. Specifically, the user can log into a RTLT PC, using telnet or rlogin, and display an xterm (an X Windows client) at the user's workstation. MATLAB can then be started in the xterm, thereby, allowing the user to: 1) create/edit a Simulink block diagram; 2) compile the Simulink block diagram using RTW; and 3) execute the compiled code in real-time. The user may monitor data signals at the remote PC or workstation using the Simulink scope.

The performance of the current Internet capability of RTLT is acceptable on a local area network; however, because of network traffic, this solution is not practical for use over the Internet. The use of X Windows to remotely display a real-time plot, such as the Simulink scope, consumes much more bandwidth than simply sending decimated log data to the remote user workstation. In addition, the Internet experience (see Fig. 2) will be more realistic to the user if: 1) live streaming video of the experiment is provided as the experiment is operating; 2) a high quality 30 fps version is provided when the experiment is over; and 3) a live virtual reality (VR) model is animated as the experiment is operated (i.e., this animation would be directly connected to the actual plant outputs). The VR animation would allow the student to examine the system from any viewpoint, something not possible with a simple fixed camera video. In addition, the ability to synchronize the video and VR playback with plots of signals

logged during the control would be very useful, since this capability would allow the experimenter to correlate the behavior of the physical plant with the variables being controlled. Note that this scheme is in contrast to simply having a student perform a local closed-loop computer simulation, because the student can see and hear the experiment execute and, despite the simplicity or complexity of the experiment, the student can make a connection that a real system exists and not just a computer simulation. The VR playback is simply a means to allow a more detailed view of the experiment (which is more interesting once the student makes the connection that he/she is really in control of a physical system). Although Win32 operating systems do not inherently allow remote access, as does Linux, similar functionality can be achieved on a Win32 platform running RTWT by installing additional software. One possible option is the use of free Internet resources such as Virtual Network Computing (see www.uk.research.att.com/vnc/).

In essence, the use of remote operation has the advantage of: 1) reducing costs by sharing laboratory equipment; 2) allowing users to have greater oversight of the control implementation; and 3) allowing access to facilities 24 h/day. Although the benefits of remote operation are monumental, there are also drawbacks to such activity. Any type of computer system that allows free access is vulnerable to hacking. To maximize security, users can be forced to use local copies of Simulink to create models, which should then be uploaded to the Internet Laboratory PC. All interactions between the Internet Laboratory PC and the user's workstation can then be implemented through communication protocols. (This method limits what users are able to do on the Internet Laboratory PC.) In addition to security concerns, there is no guarantee that the user's code is error free. For example, the user's code may contain syntax errors, undefined variables, or calculation errors that may result in an unstable closed-loop system (i.e., excessive voltage may be commanded and/or violent oscillations may occur). To address these issues, the community needs to investigate using a switching control strategy that detects situations in which the user's controller is determined to be "unsafe." If an unsafe control situation is detected, the safe controller is switched on, and the user is notified that his/her controller has failed; hence, system robustness is assured while allowing maximum flexibility for the user. One also needs to ensure that all Internet experiments are self-resetting, so that the system will be able to reboot itself and resume operation without local human intervention.

Remark 1: The use of remote laboratory technology is meant to provide students with laboratory experience on systems that would otherwise be inaccessible, to facilitate student enrollment by individuals with schedules that do not conform to the typical laboratory schedule, and to enable rapid prototyping of control designs without time-consuming hardware/software development and interfacing. Although remote technology can compliment a student's laboratory experience, it is important to stress that it should not be used as a substitute for "hands-on" experience. That is, traditional "hands-on" hardware/software development and interfacing experiences of a traditional curriculum should not be completely replaced with a remote laboratory experience and should still remain as a component of a control systems laboratory.

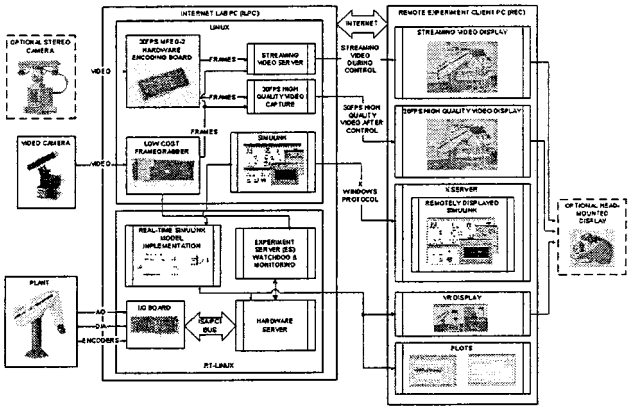

Fig. 2. Internet control laboratory setup.

IV. Limited Teaching Resource Issue

One of the possible reasons that engineering graduates showed relatively low ratings in laboratory skills in the recent control systems report card from industry [1] could be that either control systems laboratories are not commonplace or that the control systems laboratory experience is limited by: 1) outdated and/or uninteresting material; 2) complicated interfaces that inhibit proficiency among the laboratory assistants and students; and 3) no quality laboratory manuals to clearly connect the classroom theory to the experimental exercises. One reason for these possible shortcomings is the time that an educator would be required to invest. In this section, some specific time consuming tasks that are required to develop or update a control systems laboratory are examined. Then, based on some of the advantages of standardizing the CACSD, some potential mechanisms for reducing the investment of teaching resources are presented.

A. Currently Required Resource Investments

The development of a new laboratory is often time consuming for faculty because: 1) educational plants must be built or purchased; 2) the appropriate software/hardware requirements for each plant must be satisfied; 3) the hardware must be set up and tested; 4) the dynamic model of the plant must be developed; 5) the control law must be designed and tested; and 6) a manual must be written or purchased that instructs the student on how to construct and implement the controller. While purchasing educational plants from commercial companies can eliminate some of the above obstacles, educators are often not satisfied with the instructional material that is purchased with these "canned" plants. From an instructional perspective, the main problem is the lack of sufficient modeling and control design details. For example, most physical plants have a nonlinear dynamic behavior. Thus, it is common to linearize the nonlinear plant dynamics about certain operating conditions to enable the design and testing of the linear controllers that are often taught in class. However, in many manuals, the nonlinear dynamics are not provided and/or the linearization procedure is not carefully explained. In addition, more sophisticated control techniques, such as sliding mode control and adaptive control, are rarely discussed. Therefore, faculty must spend valuable

time, money, and potentially extra support staff to overcome the aforementioned shortcomings.

B. Reducing Resource Investments

By standardizing the CACSD environment, educators will be able to use common resources to address the limited teaching resource issue and to improve the quantity and quality of teaching material for undergraduate control systems laboratory development. For example, based on a common Simulink front-end for the CACSD, an Internet-based repository can be proposed that would serve as an archive for tutorials and Simulink files developed by educators for commercially available and inhouse-developed plants. Faculty who desire to incorporate new experiments in an existing laboratory, or for faculty that are just beginning to establish a new control systems laboratory, the repository will be an aid that can be used to reduce the teaching resources that would typically be required. Some resources already exist for this purpose; however, their scope has been limited by the lack of a common CACSD. Specifically, there are currently several excellent archives of laboratory instructional material, such as the NEEDS database originated by the Synthesis Coalition (www.needs.org), and several university archives and control education tools, including the University of Michigan, the University of Tennessee at Chattanooga, John Hopkins University, University of Texas at Dallas, and the California Institute of Technology. (Links for each of the university sites are provided at www.qrts.com/relatedsites/controlrelated.shtml.)

As indicated by the number of archives and web-based tools listed previously, the concept of a web-based repository as a means to offset teaching resources and enhance the educational process is not new; however, the characteristics of a repository facilitated by a common CACSD are quite unique and possibly better suited to some faculty and student needs. Specifically, this new repository could provide a tutorial introduction to a broad class of multidisciplinary experiments, detailed descriptions of the system dynamics, descriptions of the different types of control strategies that can be utilized, descriptions of the mechanical and electrical hardware components and interfacing, and a library of controllers in the form of Simulink files. From a review of literature and Internet resources, it does not seem that a repository of educational resources seems to be this comprehensive, especially to the extent of downloading actual Simulink files to perform the experiment (e.g., after searching the NEEDS database for Simulink files, none could be found). The potential advantages of a new Internet-based repository made possible by the advocated CACSD are that: 1) the time and effort that individual faculty must devote to design a new control systems laboratory is greatly reduced by the shared resources previously described; 2) shared ideas from a collection of world-wide educators will result in an influx of innovative tutorials; and 3) greater insight will be provided with respect to the modeling and control of various plants.

To facilitate the development of an Internet-based repository, the development of prototype tutorials is in progress at Louisiana State University for several plants including: 1) the Feedback magnetic levitation experiment; 2) an ECP ball-and-beam experiment; 3) a Quanser rotary inverted pendulum; and 4) a Quanser flexible-link robot. The Internet-based tutorials for the plants listed above will concentrate on modeling the plant using principles of mechanics and electromagnetics and then using linear and nonlinear techniques to control the system. Many of the resources that were created through collaboration with QRTS are available now through the previously mentioned *manuals* link from www.qrts.com.

Remark 2: Some educators may be concerned that too much information could be provided to the student (especially since the actual Simulink files can be obtained) and that programing, even at the level of Simulink block-diagram construction, is a necessary skill that students should develop through laboratory exercises. Although a valid concern, the philosophy behind the proposed repository is that by providing the student and the laboratory instructor with better information regarding the nonlinear dynamics, linearization techniques, and several example Simulink block-diagrams, the student is enabled to explore more advanced control concepts, using the base Simulink block-diagram as an example that can be appropriately modified. Moreover, the Simulink block-diagrams are constructed using MATLAB functions where the source "C" code is inaccessible by the user. Hence, the potential to include the development of device drivers for I/O interfaces as part of the educational experience is not eliminated by the extensive nature of the repository.

V. Conclusion

In this paper, the standardization of CACSD software tools is discussed for undergraduate control laboratory development. Specifically, the proposed CACSD advocates the use of MATLAB compatible products to standardize the execution of controllers in real-time, using standard, low-cost PC hardware. To illustrate the standardization concept, details were given on how the Simulink/RTW CACSD was utilized for a specific example undergraduate control experiment. A description of how the CACSD was applied to a variety of other educational plants on various operating systems was provided.

Based on the standardization of the CACSD, a discussion illustrated how the budget constraint issues and the limited teaching resource issues could be addressed. Specifically, to address the issue of reducing the cost associated with control systems laboratory development, some comments were provided with regard to using an Internet-based system, including several advantages, disadvantages, and possible new avenues to overcome several obstacles to an Internet-based experience. In view of the limited teaching resources, the advantages of a standardized CACSD were described in terms of a new Internet-based repository containing laboratory tutorials and Simulink block diagrams that might be used to prevent faculty from spending valuable time on repeating work that has already been done by other faculty. Future efforts will target implementing an Internet-based repository and remote laboratory experience. Based on these efforts, the authors will examine the effects of communication bandwidth, changes in the quality of education from the instructor and student perspective, course content, and course management issues.

REFERENCES

[1] N. A. Kheir, K. J. Astrom, D. Auslander, K. C. Cheok, G. F. Franklin, M. Masten, and M. Rabins, "Control systems engineering education," *Automatica*, vol. 32, no. 2, pp. 147–166, Feb. 1996.

[2] P. Antsaklis, T. Basar, R. DeCarlo, N. H. McClamroch, M. Spong, and S. Yurkovich, "Report on the NSF/CSS workshop on new directions in control engineering education," *IEEE Control Syst. Mag.*, vol. 19, pp. 53–58, Oct. 1999.

[3] V. Kapila, M. S. de Queiroz, and A. Tzes, "A multidisciplinary undergraduate real-time experimental control laboratory," in *Proc. Amer. Control Conf.*, June 2000, pp. 3980–3984.

[4] S. K. Agrawal, "Undergraduate control education: An ME perspective," in *Proc. Amer. Control Conf.*, June 1999, pp. 983–986.

[5] J. Apkarian and A. Dawes, "Interactive control education with virtual presence on the Web," in *Proc. Amer. Control Conf.*, June 2000, pp. 3985–3990.

[6] M. W. Spong, "Control education crossing department boundaries," in *Proc. Amer. Control Conf.*, June 1999, pp. 992–996.

[7] H. H. Hahn and M. W. Spong, "Remote laboratories for control education," in *IEEE Conf. on Decision and Control*, Dec. 2000, pp. 895–900.

[8] H. Hirukawa and I. Hara, "Web-top robotics," *IEEE Robot. Automat. Mag.*, pp. 40–45, June 2000.

[9] J. Overstreet and A. Tzes, "An Internet-based real-time control engineering laboratory," *IEEE Control Syst. Mag.*, vol. 9, pp. 19–34, Oct. 1999.

[10] M. Shor, "Remote-access engineering educational laboratories: Who, what, when, where, why, and how?," in *Proc. Amer. Control Conf.*, June 2000, pp. 2949–2950.

[11] Y.-C. Chen and J. Naughton, "An undergraduate laboratory platform for control system design, simulation, and implementation," *IEEE Control Syst. Mag.*, vol. 20, pp. 12–20, June 2000.

[12] M. W. Spong and D. J. Block, "The Pendubot: A mechatronic system for control research and education," in *IEEE Conf. on Decision and Control*, Dec. 1995, pp. 555–556.

[13] W. E. Dixon, D. M. Dawson, B. T. Costic, and M. S. de Queiroz, "Toward the standardization of a MATLAB-based control systems laboratory experience for undergraduate students," in *Proc. 2001 Amer. Control Conf.*, Arlington, VA, June 2001, pp. 1161–1166.

[14] The Mathworks Inc.. [Online]. Available: www.mathworks.com.

[15] Z. Yao, N. P. Costescu, S. P. Nagarkatti, and D. M. Dawson, "Real-Time Linux Target: A MATLAB-based graphical control environment," in *IEEE Conf. on Control Applications*, Sept. 2000, pp. 173–178.

[16] D. C. Hanselman and B. Littlefield, *Mastering MATLAB 6: A Comprehensive Tutorial and Reference*. Englewood Cliffs, NJ: Prentice-Hall, 2001.

[17] K. Craig, "Is anything really new in mechatronics education," *IEEE Robot. Automat. Mag.*, pp. 12–19, June 2001.

[18] J. Wikander, M. Torngren, and M. Hanson, "The science and education of mechatronics engineering," *IEEE Robot. Automat. Mag.*, pp. 20–26, June 2001.

[19] R. Siegwart, "Grasping the interdisciplinarity of mechatronics," *IEEE Robot. Automat. Mag.*, pp. 27–34, June 2001.

[20] D. G. Alciatore and M. B. Histand, "Integrating mechatronics into a mechanical engineering curriculum," *IEEE Robot. Automat. Mag.*, pp. 35–38, June 2001.

[21] K. Nagai, "Learning while doing: Practical robotics education," *IEEE Robot. Automat. Mag.*, pp. 39–43, June 2001.

[22] R. R. Murphy, "Competing' for robotics education," *IEEE Robot. Automat. Mag.*, pp. 44–55, June 2001.

[23] J. J. Zhu, "Discussion of 'undergraduate control education: A ME perspective'," in *Proc. Amer. Control Conf.*, June 2001, pp. 987–991.

[24] R. Simmons, J. L. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan, "Lessons learned from Xavier," *IEEE Robot. Automat. Mag.*, pp. 33–39, June 2000.

Warren E. Dixon (S'94–M'00) received the B.S. degree from the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, the M.E. degree from the Department of Electrical and Computer Engineering, the University of South Carolina, Columbia, and the Ph.D. degree from Clemson University, in 1994, 1997, and 2000, respectively.

After completing his doctoral studies, he was selected as an Oak Ridge National Laboratory Eugene P. Wigner Fellow, where he currently works in the Robotics Group of the Engineering Science and Technology Division. His main research interest has been the development and application of Lyapunov-based control techniques for mechatronic systems. He has authored recent articles in areas including: mobile robots, visual servoing, fault detection, adaptive/robust/learning control, amplitude limited control, output feedback control, underactuated systems, and nonholonomic systems.

Darren M. Dawson (S'89–M'90–SM'94) received the B.S. and Ph.D. degrees, both in electrical engineering, from the Georgia Institute of Technology, Atlanta, in 1984 and 1990, respectively.

From 1985 to 1987, he was a control engineer with Westinghouse, Pittsburgh, PA. In July 1990, he joined the Electrical and Computer Engineering Department, Georgia Institute of Technology, where he currently holds the position of Centennial Professor. His research interests include nonlinear control techniques for mechatronic applications such as electric machinery, robotic systems, aerospace systems, acoustic noise, underactuated systems, magnetic bearings, mechanical friction, paper handling/textile machines, flexible beams/robots/rotors, cable structures, and vision-based systems. He also focuses on the development of realtime hardware and software systems for control implementation.

B. T. Costic received the B.S., M.S., and Ph.D. degrees from the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC.

He is currently an Attitude Controls Engineer with Boeing Satellite Systems, El Segundo, CA. His research interests include the development and application of Lyapunov-based control techniques for space-based systems, attitude control for spacecraft, magnetic bearings, and adaptive autobalancing.

Marcio S. de Queiroz (S'94–M'98) received the B.S. degree in electrical engineering from the Federal University of Rio de Janeiro, Brazil, the M.S. degree in mechanical engineering from the Pontifical Catholic University of Rio de Janeiro, Brazil, and the Ph.D. degree in electrical engineering from Clemson University, Clemson, SC, in 1990, 1993, and 1997, respectively.

From August 1997 to August 1998, he was a Post-Doctoral Researcher in the Rotating Machinery and Controls Laboratories, University of Virginia, Charlottesville. From September 1998 to May 2000, he was a Visiting Assistant Professor with the Department of Mechanical Engineering, Polytechnic University, Brooklyn, NY. In June 2000, he joined the Department of Mechanical Engineering, Louisiana State University, Baton Rouge, where he is currently an Assistant Professor. His research interests include nonlinear control of electromechanical, mechanical, and aerospace systems, and the control of distributed parameter systems.