Towards the Standardization of a MATLAB-Based Control Systems Laboratory Experience for Undergraduate Students*

W. E. Dixon[1], D. M. Dawson[2], B. T. Costic[2], and M.S. de Queiroz[3]
[1]Robotics and Process Systems Division, Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37831-6305
[2]Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634-0915
[3]Department of Mechanical Engineering, Louisiana State University, Baton Rouge, LA 70803-6413

E-mail: dixonwe@ornl.gov, Telephone: (865) 574-9025

Keywords: Undergraduate Education, Real-Time Control, Simulink

"The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. DE-AC05-96OR22464. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes."

To appear in the IEEE American Control Conference, June 25 - June 27, 2001, Arlington, VA

* This research was performed in part by a Eugene P. Wigner Fellow and staff member at the Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725 and is supported in part by the U.S. NSF Grants DMI-9457967, DMI-9813213, EPS-9630167, ONR Grant N00014-99-1-0589, a DOC Grant, and an ARO Automotive Center Grant.

Towards the Standardization of a MATLAB-Based Control Systems Laboratory Experience for Undergraduate Students*

W. E. Dixon[1], D. M. Dawson[2], B. T. Costic[2], and M. S. de Queiroz[3]
[1]Robotics and Process Systems Division, Oak Ridge National Laboratory, P.O. Box 2008-6305
[2]Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634
[3]Department of Mechanical Engineering, Louisiana State University, Baton Rouge, LA 70803-6413
email: dixonwe@ornl.gov; ddawson, bcostic@ces.clemson.edu; dequeiroz@alpha2.eng.lsu.edu

Abstract

This paper seeks to begin a discussion with regard to developing standardized Computer Aided Control System Design (CACSD) tools that are typically utilized in an undergraduate controls laboratory. The advocated CACSD design tools are based on the popular, commercially available MATLAB environment, the Simulink toolbox, and the Real-Time Workshop toolbox. The primary advantages of the proposed approach are as follows: 1) the required computer hardware is 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. It is believed that the above advantages related to standardization of the CACSD design tools will facilitate: 1) the sharing of laboratory resources within each university (i.e., between departments) and 2) the development of Internet laboratory experiences for students (i.e., between universities).

1 Introduction

Due to 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 [15]. Unfortunately, recent studies have revealed a lack of formal experimental control education in many universities. Specifically, a control systems *report card* from industry [15] showed relatively low ratings for engineering graduates in attributes such as laboratory and hands-on experiences. Engineering accreditation guidelines (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 NSF/CSS workshop on control education [3] 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 [3] 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…."*

Since ABET, NSF, and most faculty agree that the control laboratory experience is important, why is it so difficult to build and maintain an undergraduate control laboratory? As in most problems related to standardization of hardware or software, we believe that the answer to this question is multipart. In our opinion, around 1997 the use of standard PC hardware (*i.e.*, without the requirement for a DSP), in conjunction with high level software language tools, became a more widely accepted method for implementing sophisticated control strategies[1] in real-time. We believe that feasibility of using standard PC hardware for control applications actually occurred sometime around the 1993 timeframe; however, it took some time for many control engineers to become comfortable with the concept. The use of standard PC hardware is an important concept because it reduces the cost of experimental development; moreover, it standardizes the computational engine. Second, while Quanser has been at the forefront developing a Simulink/Real-Time Workshop based front-end for standard PC hardware, other equipment manufacturers have slowly embraced this concept. Specifically, Quanser has pursued the use of Simulink/Real-Time Workshop with standard PC hardware since 1993; however, Feedback, Educational Control Products, Shandor, Kentridge Instruments, Extra Dimension Technology and many other educational plant manufacturers have not developed a Simulink/Real-Time Workshop front-end. That is, many of these companies have developed proprietary hardware and software for their plants; hence, the standardization of control laboratory equipment is made difficult due to the differences in the hardware and software components used by the various manufacturers.

To address these issues, we discuss the obstacles to standardization of a typical undergraduate control laboratory. Specifically, we describe the development of the necessary Computer Aided Control System Design (CACSD) software tools that allow a student to prototype controllers for a variety of manufacturers supplied plants using a Simulink/Real-Time Workshop front-end. In addition, we discuss some future directions with regard to control system laboratory development that will improve faculty productivity by fostering cooperation among academic institutions with regard to developing new material for control systems education. We also point out some possible technical directions that can be pursued with regard to Internet laboratory experiences for students who do not have direct access to control equipment at their university.

Before proceeding with the rest of paper, we need to stress that the concepts described in this paper provide only one possible avenue for addressing the current deficiencies with regard to the undergraduate control systems laboratory experience. The reader should note that we only discuss CACSD design tools that use a Simulink/Real-Time Workshop interface and do not require a DSP board. The reasoning for these restrictions is simple. First, based on our conversations with leading manufacturers of undergraduate control equipment, we believe that the future undergraduate

* This research was performed in part by a Eugene P. Wigner Fellow and staff member at the Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725 and is supported in part by the U.S. NSF Grants DMI-9457967, DMI-9813213, a DOC Grant, and an ARO Automotive Center Grant.

[1] The word sophisticated is used to highlight the possible use of nonlinear terms in a controller in contrast to a standard linear controller.

laboratory experience will be MATLAB/Simulink-based. Second, since we are constraining our CACSD tool development to be back-fitable (*i.e.*, we require that existing commercially available plants be usable with no hardware modifications), it does not seem possible to easily accomplish this back-fit goal with a DSP-based architecture. Third, we believe that DSP-based control architectures tend to be excessively expensive and complicated when compared to a PC-based solution. Hence, for these reasons, we will not discuss software environments that require the use of DSP boards for real-time control such as: ARCS [1], the laboratory design discussed in [5], or dSPACE [6], (for further information regarding systems that use DSP technology, see [1], [5], [6], and the references within). We have also decided not to include any discussion on the HUMUSOFT [13] product, Extended Real-Time Toolbox, since it does not seem to guarantee some measure of hard real-time performance. Specifically, when we questioned HUMUSOFT engineer Jan Houska about the real-time performance of the Extended Real Time Toolbox, the following answer was given "… if you want hard real-time performance for anything including data processing, please look at the Real-Time Windows Target by The MathWorks. It uses the same real-time technology as RT Toolbox does (we have developed it for The MathWorks) but, using Real-Time Workshop, it moves the data processing to compiled code that is able to run in the kernel." We have also decided to not include information about the outstanding products made by Opal-RT [18]. The reason for this omission is simple. We think 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 Windows).

2 Undergraduate Control Systems Laboratory Development Issues

Recent advances in hardware and software technologies have generated much discussion with regard to the undergraduate control systems laboratory experience [2], [14], [26]. Specifically, due to the advent of high-speed, low-cost, real-time computing platforms, the development of control systems laboratory hardware is now becoming more accessible. Moreover, developments in automated code generation allow users to create real-time code from graphical, control system simulation software such as MATLAB/Simulink. This tool enables educators and students to focus on control system design, implementation, and evaluation rather than on time-consuming, low-level programming. In addition, a variety of educational/research plants are commercially available from different vendors [7], [8], [9], [17], [21], [22]. These plants 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, *etc.*). Despite the availability of the many software tools and the variety of available plants, it is fair to say that the existence of a control laboratory experience for a typical undergraduate is not commonplace. This fact might be credited to any of a number issues; however, in this paper, we will discuss and address the following barriers: 1) lack of standardized hardware/software and 2) budget constraints.

3 The Standardization Issue

Background: From our point of view, one of the main obstacles with regard to developing an educational undergraduate control laboratory is the lack of standardization among educational control products. Currently, each manufacturer of laboratory experiments utilizes a different software environment, interface hardware, and I/O board. As a result, undesirable hardware and/or software modifications are often necessary to adapt a plant to vendor-specific software. For example, manufacturers such as Educational Control Products (ECP) [7] and Feedback [9] have a variety of well-designed plants, but until recently no Simulink/Real-Time Workshop front-end was provided; hence, if a student desired to change the control algorithm, he/she was required to learn a proprietary low-level software language. To alleviate this problem, Feedback and ECP recently started marketing products for Real-Time Windows Target and Real-Time Linux Target that require no hardware modifications. To provide a Simulink/Real-Time Workshop front-end, Quanser [21] developed the WinCon software environments for the Windows[2] operating system. Unfortunately, WinCon cannot be readily used with plants sold by other manufacturers (*e.g.,* ECP [7], Feedback [9], Mechatronic Systems [17], *etc.*) without back-engineering their electronic interfacing for use with Quanser's I/O board or writing device drivers for I/O boards not manufactured by Quanser. This retrofit-based approach often requires a certain level of programming and/or electronics expertise that simply may not exist in some academic departments. In addition, a homegrown retrofit-based approach is often time consuming and unreliable.

Due to the incompatibility among the leading educational control equipment manufacturers, many control educators around the world have been prompted to spend precious time developing their own experimental testbeds. This decision may be motivated by their disillusionment with some of the commercially available CACSD front-ends or the interest in examining a plant that is more challenging from an educational and/or research point of view (*e.g.*, the so-called Pendubot developed at University of Illinois at Urbana-Champaign [25]) than custom-made plants. Here again a compatibility issue arises since in-house plants cannot be easily interfaced with some of the commercially available CACSD software. Another 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.*, see [14]). 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 in-house experiments.

Proposed Solution to the Standardization Problem: To hurdle the obstacles that impede the development of a standardized control systems laboratory, a software environment is required that provides a low-cost, standardized interface for commercially available plants and/or in-house developed plants. In this section, we describe a CACSD environment that meets these requirements. The CACSD environment is composed of five design tools including: MATLAB, Simulink, Real-Time Workshop (RTW), Real-Time Linux Target (RTLT) and Real-Time Windows Target (RTWT) that are structured in a hierarchical manner[3] as shown in Figure 1. Each of these software components can be executed on standard PC hardware running on the Linux or Windows operating systems. Figure 1 illustrates the hierarchical structure of the CACSD environment along with interfaces to the user and a

[2] Quanser recently developed a Simulink/Real-Time Workshop, called Simulinux-RT, for the Linux operating system.

[3] We note that it should be possible to use Quanser's products to replace RTWT or RTLT; however, one would need to develop or purchase the corresponding hardware driver interface.

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.

MATLAB is a software environment [16] that allows a user to easily integrate computation and visualization tasks. The main advantage of MATLAB lies in that problems and solutions are expressed in familiar mathematical notations. Due to the fact that numerous toolboxes and other software packages have been developed for MATLAB, it has become the tool of choice for computation, algorithm development, modeling, simulation, data analysis, visualization, engineering graphics, and application development (including graphical user interface (GUI) development).

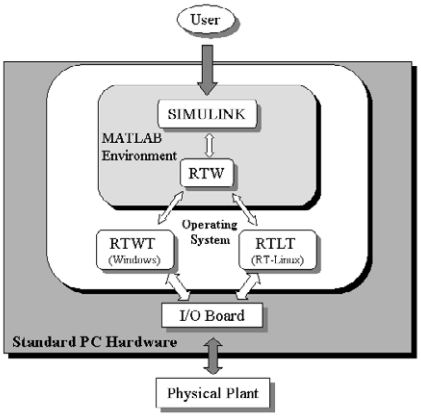


Figure 1. The CACSD Environment

Simulink is a software package [16] for modeling, simulating, and analyzing dynamic systems in the MATLAB environment. Simulink supports both linear and nonlinear systems that are modeled in continuous time and discrete time. The Simulink GUI is used to create block diagram models. Various block-set libraries provide pre-configured blocks and connectors that can be incorporated into a model by simple drag and drop operations. Different types of sources in these libraries allow the user to apply different inputs. After the model is defined, the user can simulate the response of the system by selecting the appropriate time integration method. Simulink also allows for on-line parameter tuning in order to assess the change in system response. Scopes and other display blocks allow the user to view the simulation results while the simulation is still running.

RTW is an automatic C language code generator [16] for Simulink, which runs within the MATLAB environment. RTW generates C code directly from the Simulink models and automatically constructs a file that can be executed in real-time in various environments. In conjunction with RTW, Simulink provides a powerful front-end for developing executable code without requiring a large amount of computer skills. That is, 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 programming issues.

RTLT is a software package that gives the user 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 interface that uses RTW to automatically generate C code from a user-defined Simulink block diagram. The C code is first generated and compiled on a PC running RT-Linux. A target for running the generated code is then built on the same PC. 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, *etc.*) using standard I/O boards. RTLT then provides the data to the block diagram model. The Simulink block diagram model then processes the data accordingly. RTLT then outputs the processed data via one or more output channels (*e.g.*, D/A channels). A custom Simulink block library and four different hardware I/O board drivers are also provided. The user can also observe the behavior of any signal during or after the real-time run via the Simulink Scope blocks. If the user builds the Simulink code in the external mode, the user can perform on-line parameter tuning during real-time execution.

RTWT is a Windows-based software package that 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. It has the ability to run Simulink models under Windows 95/98 or Windows NT 4.0 in real-time on standard PC hardware. It allows the user to tune control parameters while the real-time model is running. The Simulink Scope can be used to monitor the system outputs in real-time; whereas, the data archiving ability can be used to collect the run time data in a MAT-file format for later analysis and visualization in MATLAB. RTWT provides a good alternative for those users that would like to keep a complete The Mathworks, Inc. solution (the makers of Simulink, RTW, and RTWT).



Figure 2. ECP Inverted Pendulum Experiment

Case Study for Resolving the Standardization Problem: To illustrate the advantages of the CACSD software environment described in the previous sections, we utilized RTLT along with a typical undergraduate experiment (*i.e.* the inverted pendulum shown in Figure 2) manufactured by ECP to perform an example laboratory exercise. Specifically, we first worked with QRTS to develop a software driver[4] for the ECP I/O board that facilitates control prototyping with a Simulink/Real-Time Workshop front-end. We then developed a simple Simulink block diagram 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 (see Figure 3).

Based on above experience, it became clear to us that the standardization problem could be resolved if this process could be repeated with other plants made by other manufacturers. The main advantages of this approach are 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,

[4] This software extension of RTLT is now marketed by QRTS, and it allows ECP plants to be controlled with a Simulink/Real-Time Workshop front-end with no hardware modifications

low-level programming skills are not required. To demonstrate that this approach could be utilized in conjunction with other plants, we also created new Simulink files and performed similar experiments using the other plants from ECP (*e.g.,* the Servo Trainer, Rectilinear, and Torsion experiments). We then repeated the same process 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, we then worked with QRTS to develop a software interface for RTWT (*i.e.*, a Windows operating system solution). This approach allowed all of our previous Simulink files developed under the Linux operating system to be reused for controlling the ECP plants, Feedback plants, and the Quanser plant under the Windows operating system. That is, by using the same Simulink block diagrams with the I/O blocks replaced by appropriate S-function blocks supplied by QRTS, we were able to implement the same experiments on a PC operating under Windows 98 using RTWT. The reader is referred to [20] for further details with regard to downloading the Simulink files and the experimental results.

Compatibility Issues with In-House Developed Plants: A potential compatibility issue may arise if an in-house plant cannot be interfaced with the Simulink/Real-Time Workshop front-end. Fortunately, QRTS and Quanser both supplied solutions for the use of a generic multifunction I/O board for both the Windows and Linux operating systems. Specifically, QRTS supports both the MultiQ and ServoToGo I/O boards under both RTLT and RTWT while Quanser supports the MultiQ and Keithley-Metrabyte I/O boards under WinCon and SimuLinux. Both the MultiQ and ServoToGo I/O boards are excellent products that include a wide range of functionality (see [20] for more information with regard to functionality of these I/O boards). To illustrate the ease in which an in-house developed plant can be supported, we collaborated with Mechatronic Systems, Inc. [17] and QRTS to develop the necessary hardware/software interface for the Pendubot [20]. By leveraging off our past experience, it took us one day to prototype a control for the Pendubot under RTLT with the ServoToGo I/O board [20]. Due to 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.

4 The Budget Constraint Issue

Shared Laboratories within a University: The use of shared laboratories may offer some relief with regard to the budget constraint issue. That is, leveraging off of the fact that the field of control systems is multidisciplinary in nature can save funds. As such, it is quite common for engineering departments (*e.g.*, electrical, mechanical, aerospace, chemical, *etc.*) 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 [26]. Due to 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 due to 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 [26]. As noted in the NSF/CSS workshop [3], shared laboratories have several financial and pedagogical advantages.

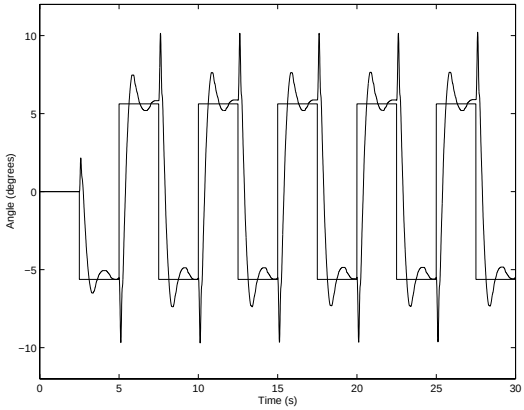


Figure 3. Pendulum Tracking Error (Desired Position vs. Actual)

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 this idea that can serve as a model for other universities is the experience instituted in the College of Engineering of the 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 [26].

Internet Laboratory Concept: Taking the shared laboratory paradigm a step further, the controls community is also starting to witness a trend towards the development of Internet-based labs [10], [12], [19]. The idea is to develop laboratory experiments that can be remotely accessed and controlled over the Internet. The primary motivating factor of the Internet laboratory concept is to enhance the accessibility of laboratory facilities for instructors and students. That is, 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 in-house resources. Typical components of an Internet laboratory include [10]: 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 as well as 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) experiment audio, video, and/or animation to give the remote user a sense of telepresence in the laboratory. To address some of these issues related to Internet control, Quanser has recently developed WebLab [4], a graphical interface to WinCon, which offers distributed control, tuning, and visualization of control systems through the Internet via a web page based environment.

Obstacles Associated with an Internet-based Control Lab: While 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 labs 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 [12] 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 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, upload it to the server computer, and test it on the actual plant. In this scenario, two 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). Second, 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 [11]. Although this eliminates the need for downloading specialized software, it limits the prototyping of new control strategies. Another aspect requiring further investigation is that, due to 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 such as Xavier [23], have only given visual feedback through a web browser of the robot's status which is updated every 5-10 seconds. 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.

A New Internet Control Lab Experience: As explained previously, RTLT and RTWT are software environments that allow the user to implement a Simulink block diagram in real-time on standard PC hardware using the RT Linux/Windows 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 Real-Time Workshop, 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.[5]

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. That is, 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 Figure 4) will be more real 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 experimenter 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. Although the Windows operating system does not inherently allow remote access, as does Linux, similar functionality can be achieved on a Windows platform running RTWT by installing additional software. One possible option is Virtual Network Computing that can be downloaded for free from *http://www.uk.research.att.com/vnc/*.

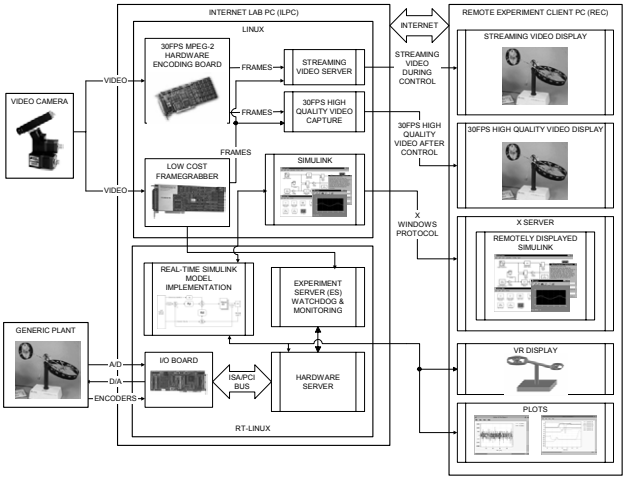


Figure 4. Internet Control Laboratory Setup

Some Solutions to Problems Associated with an Internet-based Control Lab: 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 hours per 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

[5] It is important to note that the Internet control experience provided by RTLT does not require the user to have any MATLAB products running at the remote machine (*i.e.*, the remote machine only utilizes an X server).

self-resetting, so that the system will be able to reboot itself and resume operation without local human intervention.

5 Cost Comparison

In writing a paper like this, we also need to mention some issues related to cost and real-time performance. For comparison purposes, we first note that the cost of a Quanser solution for Windows NT would run about $1,227 dollars per seat while the cost of a Mathworks solution would be about $150 dollars per seat (All of the price quotes in this paper are calculated based on the Mathworks classroom kit pricing structure for less than 25 copies). We also note the cost of a fully supported Quanser SimuLinux solution would be about $527 dollars per seat while the cost of a fully supported QRTS RTLT solution would be approximately $682 dollars per seat. Based on the above pricing structure, we believe that RTWT will become the real-time computation engine of choice for undergraduate laboratory instruction. That is, while WinCon, SimuLinux, and RTLT have some advantages over RTWT, it seems that it will be very difficult for any third party company to compete with Mathworks' pricing scheme as far as undergraduate laboratory instruction is concerned. In addition, since Mathworks provides software interfaces for many generic I/O boards that can be used with in-house 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.

6 Real-Time Performance

With regard to real-time performance, we were initially very skeptical about the use of RTWT. This skepticism was due to the fact that we could not find any information regarding how Mathworks ensures some measure of real-time performance under Windows 98 and Windows NT. We should note that we really do not know how WinCon accomplishes real-time performance under Windows 98; however, we note that Quanser ensures real-time performance under Windows NT with the VenturCom software extensions (see [28]). To examine the real-time performance of RTWT from a control point of view, we have recently completed some relatively sophisticated robot control experiments with RTWT. Specifically, we have performed the same control experiments using both RTLT and RTWT for a six degree-of-freedom robot manipulator. We achieved the same performance (*i.e.*, the performance measured by the link tracking error) for both RTLT and RTWT. Since we know that RTLT provides very good real-time performance by using a hard real-time extension of Linux, we are becoming less skeptical about the use of RTWT. Perhaps, WinCon (with the VenturCom extensions), SimuLinux, and RTLT with their guaranteed hard real-time performance and other advantages[6] will remain attractive alternatives for the control researcher or industrial user who demands hard real-time performance as well as a Simulink/Real-Time Workshop front-end.

7 Conclusion

In this paper, we discussed the standardization of CACSD software tools for undergraduate control laboratory development. Specifically, the proposed approach 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 feasibility of the approach, we discussed the development of a Simulink/Real-Time Workshop front-end for a specific ECP plant. We then described how other commercially available plants could be back-fitted with no hardware modifications. To address the issue of reducing the cost associated with control laboratory development, we presented some new concepts with regard to using Internet-based laboratory experiments.

[6] WinCon, SimuLinux, and RTLT possess several advantages over RTWT (*e.g.*, WinCon has superior plotting features in comparison to the Simulink scope); however, a discussion of these advantages was deemed beyond the scope of this paper.

8 References

[1] Advanced Realtime Control Systems, Inc. http://www.arcsinc.com.

[2] S. K. Agrawal, "Undergraduate Control Education: An ME Perspective", *Proc. of the Am. Cont. Conf.*, pp. 983-986, June 1999.

[3] 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 Cont. Sys. Mag.*, Vol. 19, No. 5, pp. 53-58, Oct. 1999.

[4] J. Apkarian and A. Dawes, "Interactive Control Education with Virtual Presence on the Web", *Am. Cont. Conf.*, pp. 3985-3990, June 2000.

[5] Y.-C. Chen and J. Naughton, "An Undergraduate Laboratory Platform for Control System Design, Simulation, and Implementation", *IEEE Cont. Sys. Mag.*, Vol. 20, No. 3, pp. 12-20, June 2000.

[6] dSpace Inc., http://www.dspaceinc.com.

[7] Educational Control Products, http://www.ecpsystems.com.

[8] Extra Dimension Technologies, http://www.xdtech.com.

[9] Feedback, Inc., http://www.fbk.com.

[10] H. H. Hahn and M. W. Spong, "Remote Laboratories for Control Education", *IEEE Conf. on Dec. and Cont.*, pp. 895-900, Dec. 2000.

[11] H. Hirukawa and I. Hara, "Web-Top Robotics", *IEEE Rob. & Auto. Mag.*, pp. 40-45, June 2000.

[12] http://chem.engr.utc.edu.

[13] HUMUSOFT, http://www.humusoft.com.

[14] V. Kapila, M. S. de Queiroz, and A. Tzes, "A Multidisciplinary Undergraduate Real-Time Experimental Control Laboratory", *Proc. of the Am. Cont. Conf.*, pp. 3980-3984, June 2000.

[15] 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.

[16] The Mathworks, Inc., http://www.mathworks.com.

[17] Mechatronic Systems, Inc., http://www.prairienet.org/msi.

[18] Opal-RT, http://www.opal-rt.com.

[19] J. Overstreet and A. Tzes, "An Internet-Based Real-Time Control Engineering Laboratory", *IEEE Cont. Sys. Mag.*, Vol. 9, No.5, pp. 19-34, Oct. 1999.

[20] Quality Real-Time Systems, http://www.qrts.com.

[21] Quanser Consulting, Inc., http://www.quanser.com.

[22] Shandor Motion Systems, http://www.shandor.com.

[23] R. Simmons, J. L. Fernandez, R. Goodwin, S. Koenig, J. O'sullivan, "Lessons Learned from Xavier", *IEEE Rob. & Auto. Mag.*, pp. 33-39, June 2000.

[24] L. Sha, "Dependable System Upgrades", *IEEE Real Time Systems Symp.*, Pozan, Poland, Dec. 1998.

[25] M. W. Spong and D. J. Block, "The Pendubot: A Mechatronic System for Control Research and Education", *IEEE Conf. on Dec. and Cont.*, pp. 555-556, Dec. 1995.

[26] M. W. Spong, "Control Education Crossing Department Boundaries", *Proc. of the Am. Cont. Conf.*, pp. 992-996, June 1999.

[27] F. C. Teng, "Implementation of Real-Time Fuzzy Logic Controller using MATLAB Based Software: A Review", *6th Int. Conf. on Cont., Auto., Rob. and Vision*. 5-8, Dec. 2000.

[28] VenturCom, Inc., http://www.vci.com.

[29] Z. Yao, N. P. Costescu, S. P. Nagarkatti, and D. M. Dawson, "Real-Time Linux Target: A MATLAB-Based Graphical Control Environment", *IEEE Conf. on Cont. App.*, pp. 173-178, Sept. 2000.