ADDING A BLOCK-DIAGRAM INTERFACE TO A LOW-COST PLATFORM FOR FEEDBACK CONTROLS EXPERIMENTS

Ryan Krauss[1] **and Jeff Croxell**[2]
Southern Illinois University Edwardsville, Edwardsville, IL
[1]*email: rkrauss@siue.edu,* [2]*email: jcroxel@siue.edu*

ABSTRACT

Feedback controls courses can be greatly enriched by experimental demonstrations and projects. These experiments help make an abstract subject more concrete, capture students' attention, and motivate them to learn more about the theoretical concepts behind a system's response. However, incorporating experiments in controls courses can be very difficult. Feedback controls experiments must be performed in real-time in order to preserve performance and insure stability. Guaranteeing that control actions occur at hard real-time intervals on the order of 1 millisecond is difficult to do with most computer operating systems. Commercial systems for performing feedback controls experiments can cost as much as $2000 per station. This paper presents a low-cost system that uses the combination of a microcontroller and a PC running a standard operating system to create a real-time control system. The system costs less than $200 per station and is capable of real-time control with update rates of 500-1000 Hz. The microcontroller enforces the real-time execution of the control law and handles all analog-to-digital and digital-to-analog conversion. No specialized hardware is required other than the microcontroller. Real-time communication between the PC and microcontroller provides plenty of data for the students to use in system identification and controller tuning/debugging. This system frees students from all of the low-level details associated with programming the microcontroller, allowing them to work entirely in Python on the PC, making the system easy-to-use as well as affordable. A block-diagram interface is being developed that will make it easier for students to program the system.

1. INTRODUCTION

Feedback controls courses can seem abstract and mathematically intensive. These courses are sometimes taught from a purely theoretical perspective. Experimental projects and demonstrations can greatly enrich graduate and undergraduate controls courses (Bernstein 1999, Bernstein 1998). Such projects can deepen students understanding and increase their interest as they see connections between mathematical models and the experimental responses of physical systems.

Incorporating experimental projects and demonstrations in controls courses can be difficult, especially for professors in smaller engineering programs. The hardware and software necessary to perform real-time feedback controls experiments can be quite expensive and difficult to set up. Control actions need to occur at hard real-time intervals on the order of 1 millisecond. This is difficult to ensure with most computer operating systems. Solutions to this problem typically involve specialized hardware and a real-time kernel or a real-time operating system.

1.1. Commercial Solutions

Control prototyping systems are commercially available from The Mathworks, dSpace, Quanser, Labview, Vissim, and others (The Mathworks Inc.(a) , The Mathworks Inc.(b) , dSPACE GmbH , Quanser , National Instruments, Inc. , Visual Solutions, Inc.). The cost for these systems can exceed $2,000 per lab station. Many papers have been presented that discuss using these systems in undergraduate controls courses. These systems seem to enrich the courses in which they are used (Shiakolas & Piyabongkarn 2003, Kamis, Topcu & Yuksel 2005, Salzmann, Gillet & Huguenin 2000).

1.2. Open-Source Solutions

The combination of RTAI-LAB with Scilab/Scicos can form a Linux-based, completely open-source control prototyping environment (Bucher & Dozio 2003, Dozio & Mantegazza 2003, Bucher & Balemi 2005). Users can create control systems using a block-diagram interface and the software automatically generates code for the real-time target environment. Most RTAI-LAB based solutions still require a data acquisition board for the target computer. These can run in the neighborhood of $500.

An interesting approach has been presented that combines Scilab/Scicos with FLEX microcontroller boards (Evidence). These boards run roughly $200 and this approach creates an open-source microcontroller-in-the-loop system.

1.3. A Microcontroller-Based Solution

The solution in the literature that is most similar to the work in this paper is that of Jack and Blauch (Jack & Blauch 2004, Jack & Blauch 2005). Their approach uses a microcontroller and avoids the cost of any additional hardware, such as data acquisition boards. The microcontroller board that they use costs roughly $100. Students in their course learn to program the microcontroller in C and implement control algorithms directly on the microcontroller. Their approach does not include any real-time communication between the microcontroller and the PC.

2. SYSTEM DESCRIPTION

This paper presents recent improvements to a unique microcontroller-in-the-loop (MIL) system that has been created at SIUE. The system uses the combination of a microcontroller and a PC to form a real-time control system. The microcontroller ensures that the control actions are performed at hard real-time intervals. The microcontroller also handles all digital-to-analog and analog-to-digital conversions, reading the sensor signals and sending the actuator command voltages. The control law is actually written in Python on the PC. This allows the user to quickly and easily modify the control law without having to reprogram the microcontroller. Students using the system do all of their programming in Python on the PC; they do not have to do any C programming and they do not have to program the microcontroller. An overview of the MIL system is shown in Figure 1.

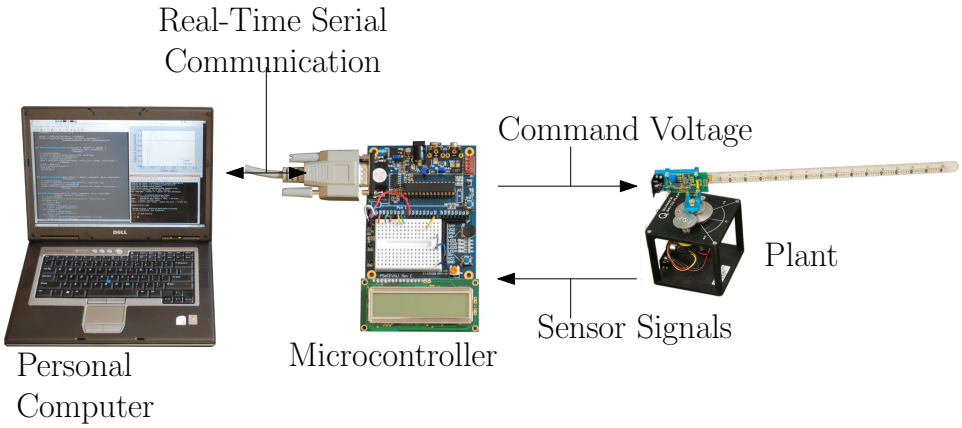

Figure 1: Overview of the microcontroller-in-the-loop (MIL) system.

This system has several strengths that make it a good fit for educational use, particularly at smaller schools with limited lab budgets. The system was designed to be both low-cost and easy to use. Insulting the students from low-level details of programming the microcontroller in C is a key part of making it easy to use. The only hardware necessary to build a control system using the MIL approach is the microcontroller board. While the approach could use almost any microcontroller, development so far as focused on the 8-bit Programmable System on Chip (PSoC) from Cypress Microsystems (Cypress Microsystems). General purpose PSoC development boards cost $140, but if this approach were pursued by more engineering programs, customized boards could be developed with an estimated cost of less than $100.

In addition to being low-cost and easy-to-use, the MIL system is also fully open and flexible. Students in more advanced courses can learn more about the system by digging into the source code. The use of Python on the PC allows great flexibility. Examples of this might include running large numbers of tests and saving the data from each test as part of statistical analysis or system identification. It is also straightforward to run many tests while varying certain parameters and saving the results of those tests.

3. PILOT TEST SURVEY RESULTS

The MIL system was used in a first undergraduate course on feedback controls in the spring 2010 semester. A control design competition was held in which the students were expected to minimized the 1% settling time for a DC motor given a step change in desired angular position. The DC motor system mimicked one joint of a robot. The competition rules penalized the students' settling times if they exceeded 10% overshoot or if the response failed to settle within 1% of the desired final value. There were two parts to the competition. In the first part, the students were required to use either a PI, PD, or PID controller. In the second part, the students were allowed to use any control algorithm they could come up with. Some of the students developed algorithms that used gain scheduling and others used command shaping to reduce integrator wind-up.

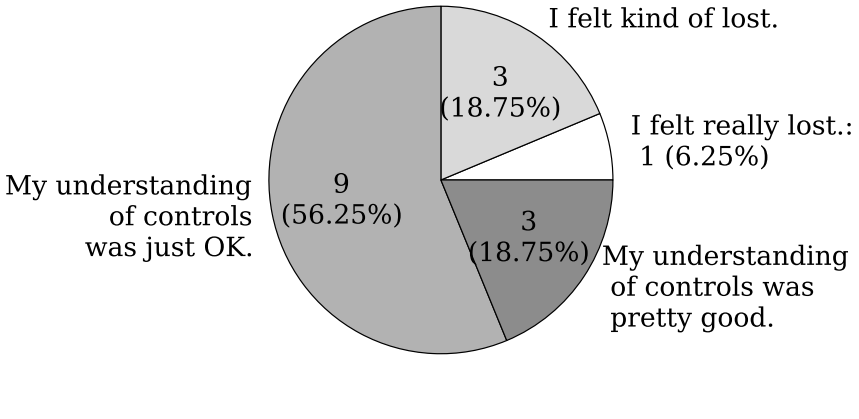

Figure 2: Answers to survey question #1: How would you rate your understanding of controls before your began working on the projects?

After the second competition, the students were given a survey concerning the experience and what they felt they learned from it (formal assessment using pre/post testing was not performed). Of the twenty three students taking the course, sixteen completed the survey. The survey results are presented in Figures 2-8.

Comparing Figures 2 and 3 shows that the students felt like the projects increased their understanding of controls. Similarly, Figures 4-6 show that the students felt like the projects increased their understanding of PID control. Specifically, Figure 6 shows that twelve out of sixteen students either agree or strongly agree that the projects increased their understanding of PID controls. The students felt like they learned something, even if student perception of learning does not guarantee learning actually occurred. Along the same lines, Figures 7 and 8 show that only one student thought the projects were a waste of time while thirteen students either agreed or strongly agreed that the projects enriched the course.

Perhaps the most important survey results are the answers to question 7: While working on the projects, what did you spend most of your time doing? A primary goal for the design of the MIL system is to free students from low-level programming details so that they can focus on control design concepts. The survey results are shown in Table 1. Only one student felt like he or she spent the majority of his or her time programming.

These survey results are encouraging and show that the students feel like working on projects using the MIL system is deepening their understanding of controls. It also seems like the MIL system has successfully freed the students from focusing on low-level programming details. The results of more a formal assessment of student learning while using the MIL system will hopefully be presented in a future paper.

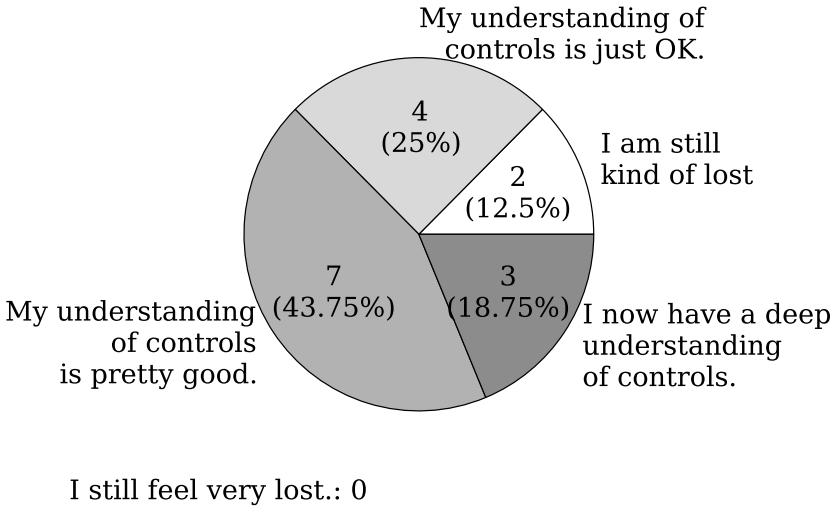

Figure 3: Answers to survey question #3: How would you rate your understanding of controls after working on the projects?

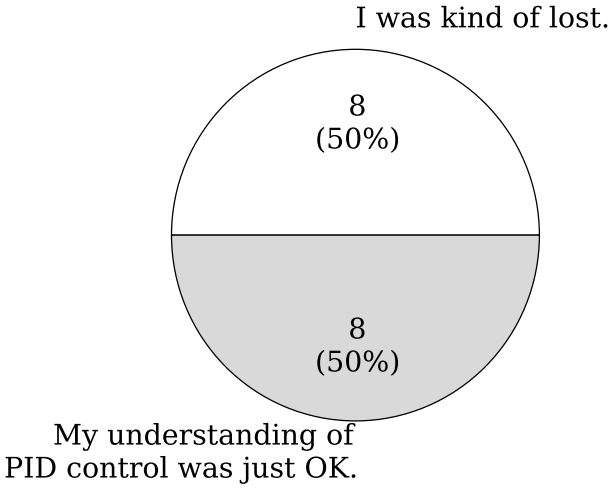

Figure 4: Answers to survey question #2: How would you rate your understanding of PID control before you began working on the projects?

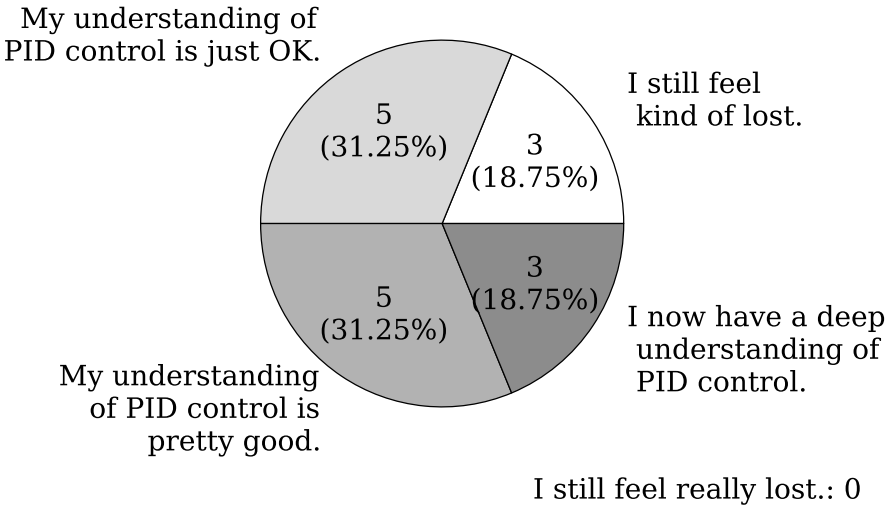

Figure 5: Answers to survey question #4: How would you rate your understanding of PID control after working on the projects?

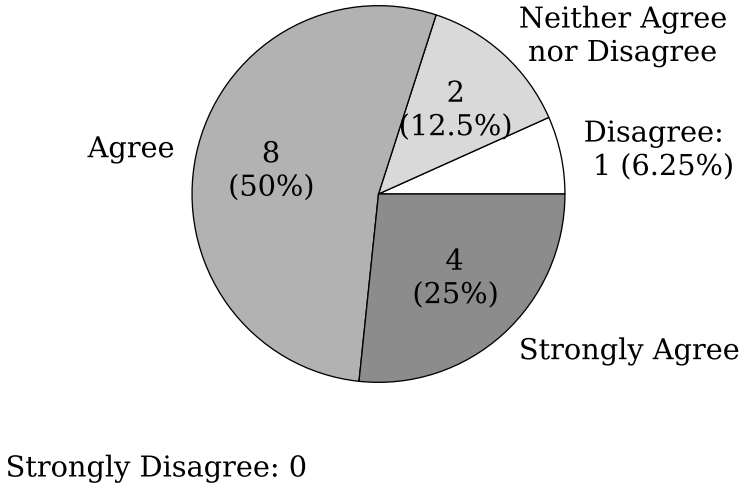

Figure 6: Answers to survey question #6: Working on the projects increased my understanding of PID control.

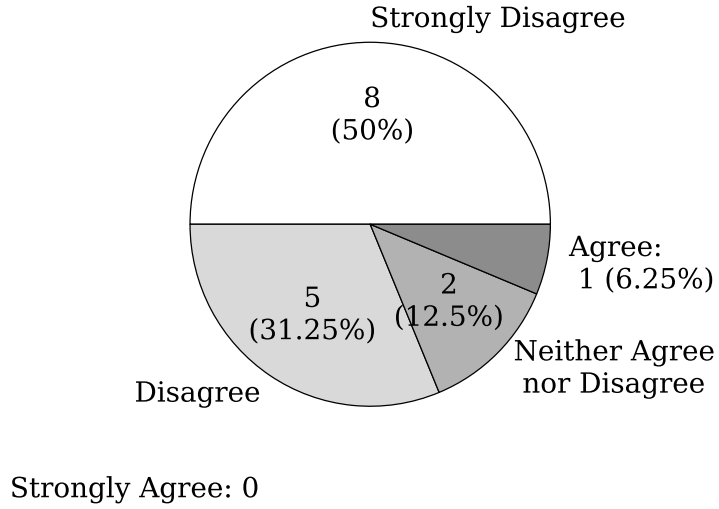

Figure 7: Answers to survey question #11: I learned nothing from these projects. They were a waste of time.

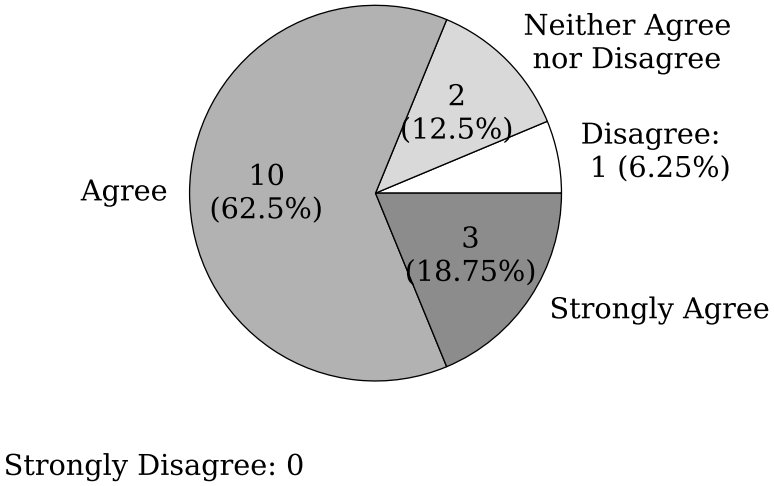

Figure 8: Answers to survey question #12: These projects enriched the course.

Table 1: Answers to survey question #7: While working on the projects, what did you spend most of your time doing?

Answers	Number of Students
Debugging	0
Programming	1
Control design, analysis, and simulation	4
Running experiments and tuning your controller	11

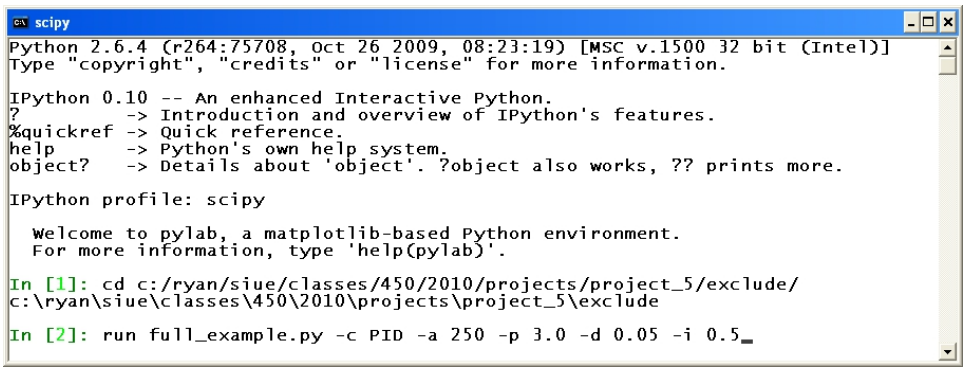

Figure 9: An example of using a DOS-like command window to run a SISO feedback control experiment.

4. RECENT WORK: A BLOCK-DIAGRAM INTERFACE

4.1. Interface Needs and Design

The MIL system discussed in sections 2 and 3 has been used in several courses and has been well-received by the students. However, the user interface needs significant improvement to make the system easier to use and allow it to be applied to a broader range of control applications. The MIL system successfully prevents the students from ever having to reprogram the microcontroller and it allows them to work entirely in Python on the PC. Nonetheless, students are still required to program using a text editor and execute their code in a DOS-like command window. The MIL system also effectively limits undergraduate students to work with single-input/single-output (SISO) systems with unity feedback.

An example of using the MIL system to run a SISO feedback control experiment is shown in Figure 9. Command-line flags allow the students to easy switch between different controllers (using the $-$c flag) and change the parameters of their controllers (for example using the $-$p, $-$i, and $-$d flags to set the PID gains).

When using the MIL system in undergraduate courses, the students are given template code

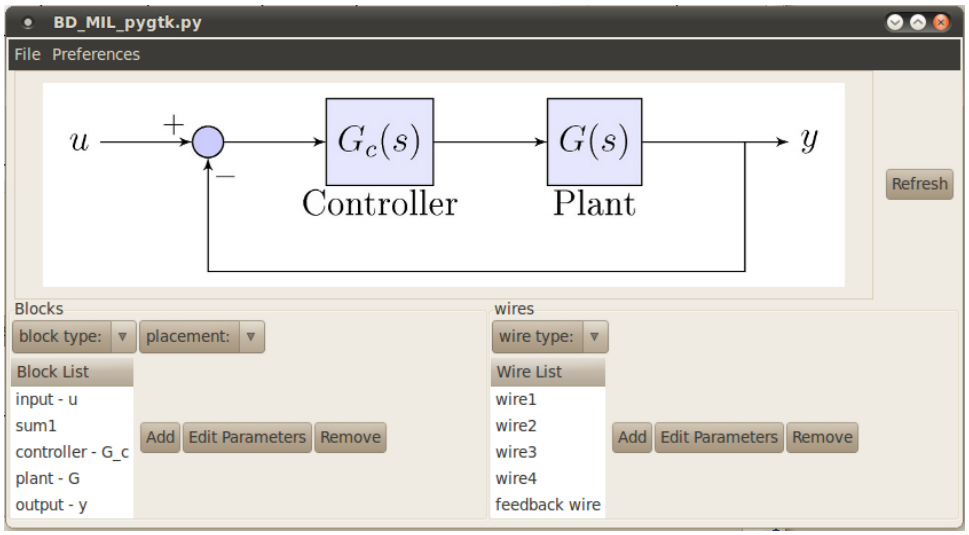

Figure 10: Prototype of a block-diagram based GUI that is being developed for the MIL system.

with examples of how to run open-loop tests and tests using proportional control. The students are encouraged to only edit the code that implements the controller block. This effectively limits them to unity feedback of SISO systems. While the author has used the MIL system to perform experiments on a system with two outputs and two feedback loops, it would be very difficult for most users to do this. If the students were asked to create more complicated control systems using the existing software, it would likely frustrate them and they could potential break the equipment. The students would have to be responsible for converting their block diagrams into text-based computer code that may lack the clear block structure.

Work is underway to create a graphical user interface (GUI) for the MIL system. This GUI will make it easy for users to create arbitrary control systems using a block-diagram interface. It will also allow them to edit the transfer function or other code associated with each block from within the GUI. The software will also auto-generate Python code for the entire control system, with classes and methods associated with each block of the system. This auto-generated code should be executable, but it should also be relatively easy for the users to edit.

4.2. Additional Features

The interface will allow users to easily specify which blocks correspond to physical plants when running experiments. These blocks will have models associated with them that are used in simulations along with input/output specifications to be used when running experiments. The software will allow the user to effortlessly toggle between running simulations and experiments and overlay results from both.

While it will be possible to run experiments with the push of a button on the GUI, the software will also auto-generate Python code that the user can edit. This will give the user greater flexibility concerning how the code is executed. Many tests could be run inside of a loop for statistical or

tuning purposes. It will also be possible to systematically vary system parameters and save data after each tests. A report of all the different tests could then be auto-generated.

5. ONGOING WORK

Along with the develop of the block-diagram interface, work is also underway to increase the digital update rate of the MIL system. Until recently, the MIL system could perform reliably with digital update rates of around 500 Hz, but this was only possible in Windows. A graduate student just finished working on a thesis project that got the system working in Linux with update rates of 1000-1200 Hz.

6. CONCLUSIONS

Performing real-time feedback controls experiments can be difficult and expensive. Commercial systems for control prototyping can exceed $2000 per lab station. A microcontroller-in-the-loop (MIL) system has been created that is both low-cost and easy-to-use. Students work entirely in Python on the PC without ever having to program the microcontroller. Real-time communication between the PC and the microcontroller allows significant amounts of data to be stored and used for tuning and debugging the control system. Survey results from using the system in a first undergraduate controls course are presented. The MIL system was well received and the students felt like working on projects while using the system increased their understanding of controls. A block-diagram user interface is being created to make the MIL system more flexible and easier to use.

REFERENCES

Bernstein, D. (1998), Control experiments and what I learned from them: a personal journey, *Control Systems Magazine, IEEE* **18**(2), 81–88.

Bernstein, D. (1999), Enhancing undergraduate control education, *Control Systems Magazine, IEEE* **19**(5), 40 –43.

Bucher, R. & Balemi, S. (2005), Scilab/Scicos and Linux RTAI - A unified approach, In: *Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on*, IEEE, pp. 1121–1126.

Bucher, R. & Dozio, L. (2003), CACSD under RTAI Linux with RTAI-LAB, In: *Fifth Real-Time Linux Workshop, Valencia, Spain*.

Cypress Microsystems
URL: *http://www.cypress.com*

Dozio, L. & Mantegazza, P. (2003), Real time distributed control systems using RTAI, In: *Object-Oriented Real-Time Distributed Computing, 2003. Sixth IEEE International Symposium on*, IEEE, pp. 11–18.

dSPACE GmbH
URL: *http://www.dspace.com/*

Evidence
URL: *http://www.evidence.eu.com/content/view/175/216*

Jack, H. & Blauch, A. (2004), A Modeling and Controls Course using Microcontrollers, In: *Proceedings of the 2004 ASEE Annual Conference and Exposition*.

Jack, H. & Blauch, A. (2005), A Modeling and Controls Course using Microcontrollers, In: *Proceedings of the 2005 ASEE Annual Conference and Exposition*.

Kamis, Z., Topcu, E. & Yuksel, I. (2005), Computer-Aided Automatic Control Education With a Real-Time Development System, *Computer Applications in Engineering Education* **13**(3), 181–191.

National Instruments, Inc.
URL: *http://www.ni.com/realtime/*

Quanser
URL: *http://quanser.com/*

Salzmann, C., Gillet, D. & Huguenin, P. (2000), Introduction to Real-time Control using LabVIEW with an Application to Distance Learning, *Int. J. of Engineering Education* **16**(5), 372–384.

Shiakolas, P. & Piyabongkarn, D. (2003), Development of a Real-Time Digital Control System with a Hardware-in-the-Loop Magnetic Levitation Device for Reinforcement of Controls Education, *IEEE Transactions on Education* **46**(1), 79–87.

The Mathworks Inc.(a)
URL: *http://www.mathworks.com/products/simulink-coder/*

The Mathworks Inc.(b)
URL: *http://www.mathworks.com/products/xpctarget/*

Visual Solutions, Inc.
URL: *http://www.vissim.com/products/addons/vissim/real-timepro.html*