Real-Time Devices At Practical Prices: Low Cost Laboratory Projects

Ruth A. Maulucci
MOCO, inc.
344 Gannett Road
Scituate, MA 02066, USA
maulucci@mocoinc.com

John Lentz and Richard H. Eckhouse
Math & Computer Science Dept.
University of Massachusetts Boston
Boston, MA 02125, USA
eckhouse@cs.umb.edu

Abstract

It is a challenging task to develop a course in software methods which also includes a laboratory in which to experiment with the concepts. The challenge is made even more difficult when funds for such an endeavor are scarce. Practical and low-cost projects were implemented which give students hands-on experience with electronic devices while learning the software requirements of real-time systems. Students designed and built systems and wrote controlling software for a model train, floppy drive components, a balance beam, a robotic arm, and an elevator model. In all the projects, the problems are open ended so the instructor can require simple or more complicated solutions, depending on the abilities of the students and the course objectives.

1. Introduction

To practitioners in the field of real-time systems, it comes as no surprise that "…real-time systems are becoming a pervasive component of everyday life." [1] Experience with such systems for use in a variety of seemingly different fields corroborates the assertion that "…most real-time computing applications require interdisciplinary knowledge, spanning across various fields." [1] A challenging task, then, is to develop a new offering for computer science programs that creates both a course in software methods as well as a laboratory in which to experiment with and test concepts.

The Computer Science program at UMass Boston is fairly traditional with the usual set of applied and theoretical course requirements for the students. Within the applied portion of the curriculum, there is a heavy emphasis on programming, although there is no undergraduate requirement in software engineering. In fact, there is no computer engineering in the curriculum, and the one required course on computer organization and architecture focuses only on the building block approach (i.e., AND, OR, and NOT gates) to understanding the internals of a computing system. The lack of courses related to both software (software engineering) and hardware (control of electromechanical devices) changed when the first undergraduate course in real-time systems was offered a year ago. This was prompted by the desire to add more computer engineering to the curriculum so that students could gain hands-on experience with electronic devices, while gaining experience with the requirements of real-time systems. The biggest limitation was the lack of funds to purchase either software or hardware that would become part of the real-time laboratory.

The authors all have a background in real-time systems, although that background has not previously been applied in an educational environment. However, our common motivation was to develop such systems to produce real-time solutions at the lowest cost while working with typical circuits and systems found in everyday situations. Based on our experience we sought to identify the needs of the students and to invent some practical and low-cost real-time projects. Since our experience ranged rather broadly, from that of hobbyists to that of researchers in such fields as electronics, video, rehabilitation, and instrumentation, we aimed to select projects that were not only informative but also interesting.

This paper addresses one specific aspect of real-time computing, namely, that of developing low-cost laboratory projects that illustrate control issues. In what follows, we offer some insight into the specific needs of one university as envisioned by three practitioners with significantly different backgrounds.

2. Premiere Project

The first project we implemented was unplanned and unanticipated. Two students expressed a desire to know more about the electronics side of computer controlled systems. Several discussions later we decided on the idea of building the classic computer controlled train [2]. Our design was simple and was based on a store-bought HO gauge train set. To make the programming more challenging than driving one train in a clockwise or counter-clockwise direction, we purchased a second train set and some turnouts so that two trains could run at the same time but in opposite directions. The traditional layout, shown in Figure 1, was used so that one train could pass the other by using the sidings. There were eight isolated sections (numbered 1 - 8), four switches, and eight photo-diodes, shown as black dots, to detect a train passing a photodiode as it moved through a section.

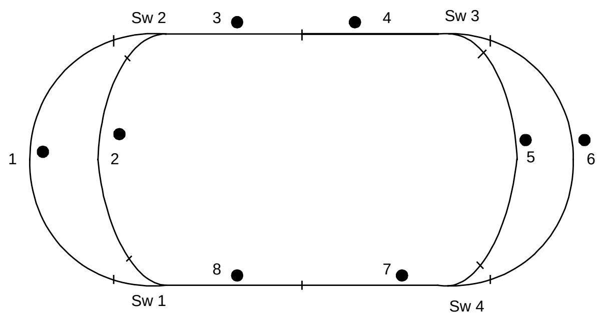

Figure 1. Traditional train set layout

While the use of a model train with such a simple layout was not particularly remarkable, it did represent a first attempt by the department to offer a course that required real-time programming. It also illustrated that it is not difficult to construct projects that are both academically stimulating and enjoyable for the students.

The design approach bears mentioning. The students who built the model train layout had no previous experience with electrical devices, including model trains as toys. Thus, they had to learn: a) what made up a model train set, and b) how to use electronic components (MOSFETs, relays, and TTL devices) to power the individual sections of the track in a forward or reverse direction, detect a train passing a photodiode, and open/close a switch. Containing costs was a simple matter since the Boston area offers many electronics stores, hobby shops, and even flea markets where parts can be readily and inexpensively obtained.

Control for the train was designed to be entirely digital. Each section of track could be powered individually (sections were electrically isolated from each other), switches could be opened or closed, and the photodiodes could be used to report when a train passed by. There was no speed control for the train and electrical power for the entire system came from the transformers supplied with the train set. Documentation was initially non-existent but soon became a reality in order to explain the design and allow other students to use the project. Although the original goal was to design the system, build the system, and write controlling software during the summer break between semesters, this proved to be a much greater effort than the students anticipated, and only the first two objectives were accomplished.

Some obvious mistakes were made in the original design. These included using two relays in parallel to power the trains in either a forward or reverse direction. The result was a four state controller where only three states (forward, reverse, and off) were allowed for each section; the fourth state must never occur because in that state a section would be powered in both directions simultaneously thereby creating a direct short of the train transformer. Had the students who created the original design also written the control software, they would have realized that this fourth state was a contributing factor to badly written software, and they might have changed their original design. Indeed, we thought that allowing this state, in principle, would demand better programming practices from the students. In fact, it did not, and after witnessing the frustration caused by the transformers overheating in the fourth state (thus impeding the testing of the software), we decided to wire the relays to prevent this state from occurring.

Another source of problems with the original design was in the operation of the turnouts. The original designers had expected the student programmers to power the turnouts to open or close, but no longer than necessary, so they would not burn out when power was left on for an extended period of time. After replacing several turnouts, we decided to modify the design by effectively pulsing the turnouts in hardware.

Other problems occurred as well, such as not latching the photodiode outputs long enough for the software to poll their state and not being able to adjust their sensitivity to ambient light. The students in the course also requested a change in the original design to show which sections were powered and in which direction. Finally, to improve isolation from the toy transformers, the controller board was powered from the controlling computer by making an extension cable from an unused Molex connector inside the computer to the controller. A separate ground between the computer and the controller was needed as well. The entire project was built for just under $300 US, including the 48-bit digital I/O board used to interface the project to the computer (see Figure 2).

From these modest beginnings, we learned that student projects to be used for continuous real-time control problems need careful supervision, a great deal of attention to documentation of the design, and the application of good engineering principles to anticipate problems and shortcomings. Nonetheless, we were able to produce a series of programming projects. These projects started with the simple objective of monitoring where a train was on the track layout and controlling the train as it moved between sections. Adding a second train required more attention to program details since most students approached this level of control by defining the entire state as something to be kept "in their heads." Years ago, it was not uncommon for a computer architect to be the keeper of a design, in his or her head, and be the leader of the team that built the final design. As computers became increasingly complex, this approach was disastrous and the concept of the architectural design document was born.

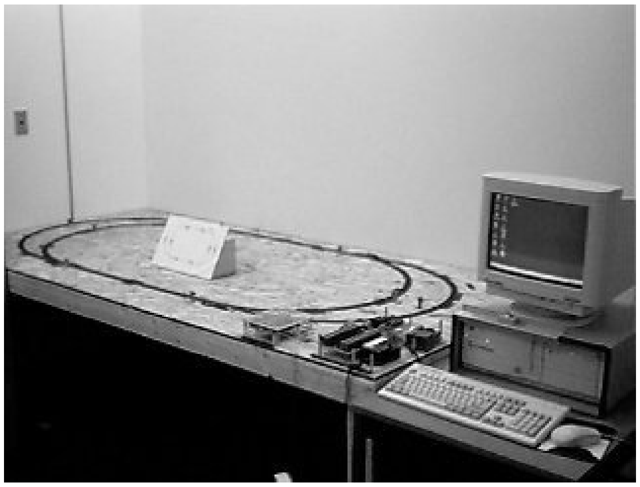
Figure 2. The train set

A similar situation arose with the controlling software for two trains running in opposite directions. Students would attempt to write their programs to grow with the increasing complexity of the problem they were solving. Most students had difficulty in solving the problem of moving two trains around the layout in opposite directions when they applied ad hoc programming using IF and CASE statements. Eventually they would get to the point where they could no longer control the trains and their solutions would lead to greatly simplifying assumptions (such as stopping one train on a siding while it was passed by another train on a different siding), or the trains would become hopelessly deadlocked, rocking back and forth between powered sections. A few students did recognize that better solutions were required to manage the control issues, and they started applying finite state automata techniques.

Students were encouraged to test their solutions when the speed of the trains was increased. While no formal examples had been given about what happens when the rate of service required is increased, this simple extension to their solutions produced a great deal of interest and a realization of the time critical nature of their code. Also, it pointed out the clear distinction between solutions which quickly collapsed as the speed increased, and those that did not.

3. Subsequent Projects

While the train control project was quite helpful in identifying real-time issues, there was a distinct lack of experience with analog signals and motor control. Because of our focus on the PC for our organization and architecture course, we had adopted a textbook [3] that included a chapter on these subjects. Incorporating this previously unread chapter into our real-time course offered a chance to discuss such concepts, but we were in need of problems and projects to gain some practical experience. Because we could obtain inexpensive ISA boards that include analog input and output, we considered how we could best use them to illustrate real-time issues.

3.1 Floppy Drives

Our discussions led to the projects we could envision by being mindful of our goal to keep things inexpensive. Our first project came about when we recognized that old 5¼ full height floppy drives can be obtained at very little, if any, cost. We picked the floppy because of its wealth of parts, including two motors and controller boards. The first motor, the head positioner, is a stepper motor and can be controlled from the parallel port by attaching five leads to the floppy controller board and externally powering it from the PC itself. The student projects can be easily set up, offering concrete examples of what the students have read in their textbooks, and the parts can be remounted in various ways (see Figure 3) to add interest to the problems to be solved. No difficulties have been encountered by students writing solutions to the problems, although there was the usual amount of puzzlement stemming from a lack of understanding of how stepper motors work. Since the floppy stepper motors and controller boards are very robust and in great supply, we disassembled one by removing the coils so that the students could easily

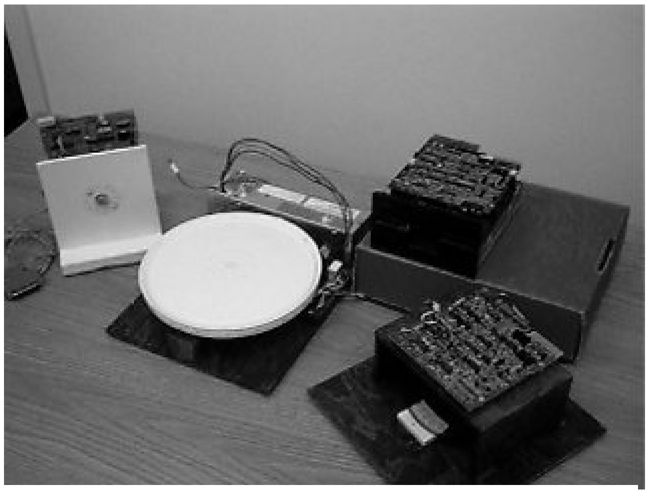
Figure 3. The floppy drive and parts

understand the mechanical construction of the rotor and stator. Total cost for such a project was a few dollars US for the wiring and connectors.

3.2 The Balance Beam

Possessing a wealth of parts, the floppy controller led to our next project -- the balance beam. Based on an earlier article by Pease [4] where an electromechanical solution was given, we implemented a similar solution using the complete drive mechanism from a Canon floppy drive. The stepper was used to raise and lower one end of a Z gauge track mounted on an aluminum channel that pivots about its midpoint. One rail of the track is powered so that 0.5 amps flows through it. A steel ball then glides from one end of the track to the other, acting as the wiper element in a potentiometer. The second rail is used to pick up the voltage from the first, and it along with the stepper motor signals (select, direction, and step) are fed to an analog-to-digital converter board. This is shown in Figure 4. We used a surplus power supply to provide five volts. However, the computer itself could be used for this purpose. Once again we have a low cost project where the only costs are the discarded floppy drive, the wood used for the base and pivot arms, the dropping resistor to limit the current, the aluminum channel, and the Z gauge track.

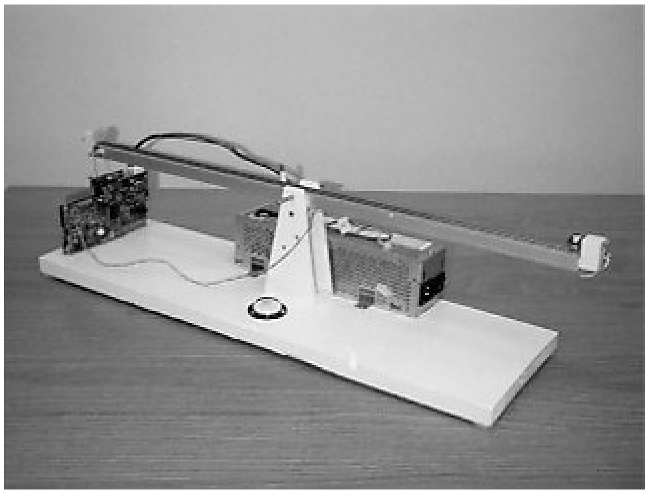

Figure 4. The balance beam

Students are currently struggling with this project but have not reported any specific problems. Simple solutions have been submitted that do not use anything other than the position of the ball on the balance beam as a means of moving the ball from one position to another. However, the students have been encouraged to implement a PID controller function in software to move the ball more quickly. As in other such projects, solutions are open ended and instructors can require simple solutions or more complicated ones, depending on course objectives.

3.3 The Robotic Arms

Another goal for this course has been to develop projects that have both open- and closed-loop controllers. We wanted to use robots to illustrate these principles but almost abandoned the idea when we realized the cost of commercially available units (several hundreds to several thousands of dollars US). We managed to purchase one device for slightly less than $200 US, which uses a serial interface and proportional control servo motors as found on remote control airplanes, boats, and automobiles. However, we felt this was still too expensive for our laboratory and were happy to discover the hobbyist market as a source of less expensive (~$70 US) toys. These toys use geared DC motors and have switch box interfaces that control the base, shoulder, elbow, wrist, and gripper motors. Our low-cost interface is once again built using MOSFETs and relays. No feedback is available with such a device but in conjunction with the more expensive robot arm, it is possible to solve the same problem with each of the two robotic arms to see the differences in performance. The project is again simple, requiring the students to write a program to move an object from one position to another, and then return the object to its original position. We have not yet found a low cost solution to providing external feedback on the position of the robot arm, but the concepts involved are clearly demonstrable. The two robots are shown in Figure 5.

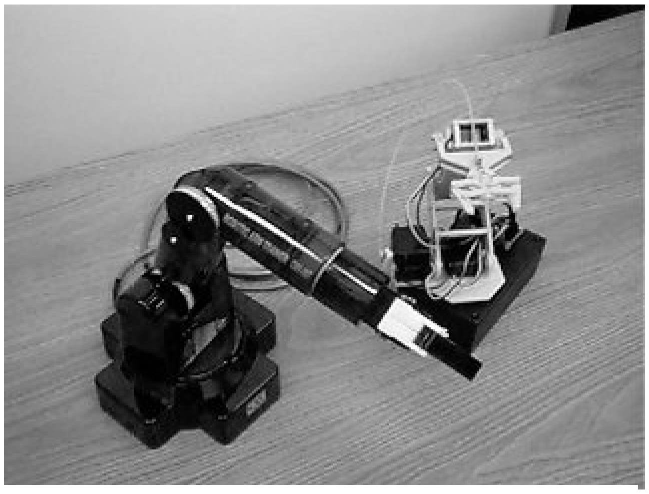

Figure 5. The robotic arms

Parenthetically, we should point out that there is another, even lower cost, alternative to a robotic arm project. The DC motor used to drive the floppy itself includes a second winding that can be used for motor speed feedback. Although we have not explored this yet, it appears that it would be quite simple to use the floppy motor and controller card to develop a project where software provides the feedback to the motor based on the feedback received from the second, or generator, winding. All the drives that we have examined include a stroboscopic pattern which can be used with an available fluorescent light to confirm visually what the software is reporting about the actual motor speed.

3.4 The Elevator

Our computer science department is located on the fifth level of the building that houses it. Students normally use the elevators to go from floor to floor. Over time they have learned the idiosyncrasies of the elevator controller when trying to get somewhere quickly. It thus seemed fitting to offer them an elevator model as a way to determine what is involved. The model, shown in Figure 6, is quite complex and offers more of a programming challenge than do the previous projects described.

Our elevator model includes three elevator cars moving between five levels. The elevators are hobby enclosures driven by DC motors taken out of old toys. A limit switch is used at the top level to prevent the motor from breaking the thread that draws the elevator up. If a program tries to move the elevator beyond the top level, the limit switch is engaged and does not allow the motor to activate. An override switch is then used to force the motor to lower the elevator. The position of the elevator at each level is sensed by a LED-phototransistor pair. At present the elevators use digital control for both movement and level sensing. Since we have recently added some digital-to-analog boards to our laboratory equipment, we will use them to control elevator speed and will attempt to use our analog-to-digital boards to allow more accurate positioning of the elevators at each level.

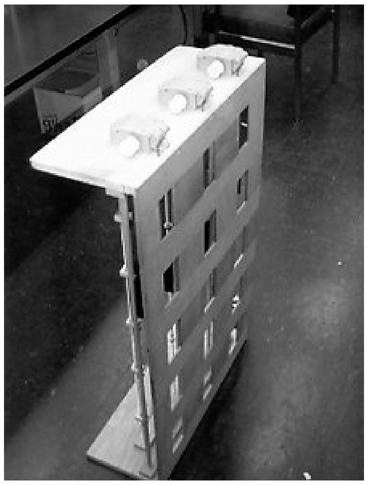
Figure 6. The elevator

The programming task is to accept real-time requests from different levels. The elevators are to be moved from one level to another, in a way that minimizes service request/fulfillment times. As before, there are limitations to such low-cost projects, and the most obvious one in this case is that if an elevator is sent downward to the lowest level without stopping, the DC motor will eventually run to the end of its thread and the elevator will start to move upward.

4. Conclusions

The open-endedness of all these low-cost projects allows students or student groups to submit solutions based on their abilities. We set a minimum standard of performance for each project and then offer extra credit for submissions that exceed the standard. We also encourage students and groups to compete with each other by offering small rewards based on the quality and uniqueness of the solutions presented during a gathering of the class where the students and groups demonstrate their projects.

The projects have acted as a catalyst for student projects in other courses. Our graduate software engineering students have used the train as a base for building a fail-safe programming project that does not allow one train to enter a section of track that is occupied by another train, and to provide a minimum number of empty sections between trains. Our balance beam is a contender for a project in a course on fuzzy logic. Interestingly, this was the basis for the original balance beam project documented in [4].

While low cost real-time devices save money, there is another practical side to developing them. With the increased enrollments in real-time systems courses, it is necessary to have many laboratory stations even when students work in groups. Low cost devices allow replication and offer the additional benefit that spares are available should catastrophes occur. At the same time they demonstrate to students how relatively easy it is to use computer control on every day objects. This has led to students developing additional projects for their own use or as new projects for our real-time laboratories.

5. References

[1] Halang, W. et al. "Real-Time Computing Education: Responding to a Challenge of the Next Century." *Real Time Programming 1997*, Pergamon, Oxford, 1997, pp. 121-125

[2] Grason, J. and Siewiorek, D., *Computer*, Vol. 7, No. 12, 1975, pp. 73-81.

[3] Sargent III, M. & Shoemaker, R. *The Personal Computer from the Inside Out, 3rd ed.*, Addison-Wesley, Reading, MA, 1995.

[4] Pease, R. "What's All This P-I-D Stuff, Anyhow?" and "What's All This Ball-On-Beam-Balancing Stuff, Anyhow?" *Supplement to Electronic Design*, August 4, 1997, pp. 49-56 and pp. 61-64.