# FOSTERING INTERDISCIPLINARY ACTIVE AND DISCOVERY LEARNING

**Carl W. Steidley, Patrick Michaud, R. Stephen Dannelly, Holly Patterson-McNeill**

Texas A&M University Corpus Christi
and
Lewis Clark State College

## ABSTRACT

In this paper we describe a student-centered laboratory developed by the Department of Computing and Mathematical Sciences at Texas A&M University – Corpus Christi (A&M-CC) and partially supported by a National Science Foundation CCLI Program grant (DUE-9950839). This laboratory is utilized by students, freshmen to graduate level, as an active learning laboratory. These are students from several disciplines including computer science, mathematics, geographical information systems, engineering technology, and educational technology.

The laboratory is centered around a computer-controlled model railroad system and a range of simple to sophisticated robotic platforms. The laboratory equipment and layout encourage "near-peer" teaching activities such as presentations and group projects. The student projects move beyond basic theory verification by requiring students to practice higher-level thinking, and students are able to physically observe the results of their own computational solutions to problems. The student projects encourage students to reorganize knowledge, understand computation in the context of larger systems, and discover the connections among several disciplines.

## BACKGROUND

Our Computer Science program has an enrollment of approximately 340 undergraduates. Although the under-graduate program produces graduates who are able to attend graduate school, many choose to join industry. Our graduates are highly proficient at developing software to solve real problems. The approach we use to achieve this result is to emphasize the design and writing of software in a range of areas.

It was our belief that the effectiveness of many of our courses could be greatly improved by placing students in a setting where they could actively extend the concepts being taught into tangible realities [1, 2]. Therefore, we implemented a new laboratory, the Real-Time Lab, equipped with a model railroad system and five networked Linux-based control computers, two ActivMedia Pioneer class and two ActivMedia AmigoBot autonomous mobile robots, and 10 LEGO MindStorms robot kits.

The Real-Time Lab was initially used to support the Systems Analysis and Design, Introduction to Software Engineering, Systems Programming, and a graduate-level summer special topics course on graphical user interface design using the X Window system. The first use of the lab was during spring 2001. But, once sufficient interface applications and libraries were developed by our students, we have since used the Real-Time Lab in our Artificial Intelligence courses, our senior capstone design project

*Proceedings of the 2003 ASEE Gulf-Southwest Annual Conference*
*The University of Texas at Arlington*

course, Operating Systems courses, and the graduate-level Mathematical Modeling course. We have also used the Real-Time Lab as an important part of tours given to prospective students.

A model railroad component was selected for five reasons. First, both discrete-state (modeled by state machines or Petri nets) and continuous-state (described by transfer functions) control problems can be modeled. Second, a sensor-instrumented model railroad is versatile and enables students to work on a large variety of simple to complex projects. Third, it is not difficult to obtain the necessary equipment; model railroads are easily available in our community and many other locations. Fourth, it is virtually impossible to think of a more inexpensive option with the same capabilities. Finally, we believe that the model railroad stimulates interest throughout the campus and the local community. Students are enthusiastic about a course that will teach them to use the computer to control a model railroad [5, 6, 7].

We selected the ActivMedia Pioneer mobile robots as the principal robotic platform for the laboratory for several reasons. First, it is a sturdy, proven platform [4]. Second, the robot control software runs a reactive planning system with a fuzzy controller, behavior sequencer, and deliberative planner, all of which are easily interfaced with the C language. Third, it has seven sonar transducers to provide object detection, range information, and feature recognition. Fourth, it has an optional versatile, high-performance vision system. Fifth, each of its drive motors includes a high-resolution encoder for precise position sensing.

The AmigoBot mobile robots are referred to as the "cousin of the Pioneer" by ActivMedia. This robot has a similar architecture and utilizes much of the same software, making the AmigoBot a natural choice in order to reduce robot-to-robot communication difficulties. However, the AmigoBot is a small robot weighing 8 pounds and measuring 11 inches by 13 inches.

Finally, we chose the LEGO Mindstorms robot as an introductory robotic platform for the laboratory for several reasons. This product line is a breakthrough in technological toys. With plastic gears, pulleys, beams, bricks, axles, connector pegs, and other building elements students can create mechanical contraptions that would make Leonardo DaVinci jealous. Combine these mechanisms with motors, sensors, and a programmable LEGO brick, the Robotic Command Explorer (RCX), and the student's creation can run autonomously, interacting with and responding to the student and its environment including other robots.

## RAILROAD SETUP

The Real-Time Lab's model railroad system has two main parts: the track bed and the control/development computer network. The track bed is composed of approximately 100 linear feet of HO scale track with 30 electronically controlled track switches called turnouts. Sensors have been placed every three feet along the tracks. These sensors only detect the presence of an object, not the type of train car, nor its speed. This location detection system has since been augmented with a student-developed computer vision system and by a student-developed mini-GPS system. The layout will also utilize an electronically controlled model crane that is able to lift, move, and lower objects with very fine precision. These components rest on half inch paperboard that is fixed to a 30-inch high, six foot by eight foot table constructed of plywood. All track, turnouts, and sensors were purchased off-the-shelf. We believe this low cost setup provides our students with an enormous number of projects that they will find very interesting, challenging, and useful.

Five Pentium III PCs are used for software development and testing. These Linux-based computers are attached to the campus LAN and hence the Internet. This allows our students to use the system from anywhere. One computer is equipped with an electronic signaling device, known as a "command station", that controls all components of the track system. We have configured the control PC to control

the movement of five separate locomotives, each turnout, and the model crane. This main control computer is equipped with the A/D card that supplies primary location sensor data.

In addition to the computer controlled track system, we have also installed two low-cost cameras. Each camera is fixed in place to provide different views of the system. These cameras provide a web-based remote viewing system and are used for secondary location detection.

## USING THE RAILROAD SETUP IN A COURSE IN SYSTEMS PROGRAMMING

In the 2000-2001 academic year, the Department substantially completed the physical construction of the model railroad component of the Real-Time Computing Laboratory in a room formerly used as a faculty lounge. Based on the results of a student project performed as part of the Systems Analysis and Design course, it was decided to use the model railroad setup as a central component of the Systems Programming course (COSC4348) in the Spring 2001 semester.

The intent of this course is to provide a detailed study of the "system-level" components in modern operating systems such as device drivers, application support libraries, and communications between processes within a machine and over a network. The course assumes that students already have prerequisite knowledge of basic operating system and compiler principles (from COSC3346 Computer System Software); COSC4348 builds upon this prerequisite knowledge to give students hands-on experience in process management, software library implementation, and interprocess communication such as pipes, sockets, and semaphores.

The Real-Time Lab at A&M-CC was designed to be integrated into the computer science curriculum as a source of hands-on, "real-world" sorts of computing problems. In this sense, the railroad setup is highly suited to providing hands-on examples of and experience with problems that commonly face systems-level programmers. Examples of systems-programming-related problems that can be addressed using the model railroad component of the Real-Time Lab include:

- Understanding and interpreting device-level protocols for controlling hardware devices (in this case, the train locomotives and switches)
- Developing software libraries and device drivers to provide an application programming interface to the low-level devices
- Exploring issues of concurrent systems control, semaphores, and asynchronous event handling
- Developing robust systems that can gracefully deal with errors and unexpected conditions or events
- Creating client-server-based systems for remote and distributed control and
- Using and building simulations as "idealized" versions of real-world systems, and understanding the advantages and limitations of such simulations

### Project objectives

The content of the course was structured around a semester-long project to develop (1) a client/server system that would enable users to monitor and control the model railroad from across the network, and (2) a set of libraries that would enable application-level programmers to write higher-level applications that could monitor and control the model railroad. To complete the project, students worked on a sequence of assignments to design and implement the components of the client/server system and libraries. By completing the assignments, the students achieved the course objectives and gained direct experience in systems programming activities. The details of the assignments are described below.

Assignment #1: Understanding the LocoNet protocol

In the first assignment, students were expected to write a program that would read a stream of bytes coming from the model railroad decoder, parse them into valid and invalid LocoNet messages, and display the messages in a human-readable format. The students were provided with streams of binary data containing LocoNet™ messages captured from the model railroad controller, the LocoNet™ protocol manual describing the codes used in the command sequence, and examples of desired output. This assignment achieved a number of objectives:

1. It gave the students increased experience in dealing with bit-level manipulations commonly found in device protocols.
2. The students learned to parse fixed and variable-length messages from streams of bytes.
3. The students gained an understanding of data communications as a stream of bytes where message framing must be deduced directly from the data.
4. The students had to write programs that could deal gracefully with errors in the data stream such as missing, extraneous, or malformed bytes.
5. The students gained a preliminary understanding of the LocoNet protocol, which would be needed later for communicating with the model railroad itself

I/O Assignment #2: Buffered and unbuffered

Handling streams of input and output data are fundamental to any modern operating system, and systems have facilities for buffered and unbuffered I/O. In this assignment, the students were expected to write a simple program to copy files using different block sizes in buffered and unbuffered I/O, to time the results of copying large files using such programs, and to explain the differences in execution times. This assignment provided the students with a greater understanding of the basic Unix I/O facilities for reading and writing stream data.

Assignment #3: Interprocess communication using pipes

Students were then given an assignment to use pipes to communicate with a *RRsim* process using the SRCP protocol (designed by Michaud) to move a locomotive from a given location of the layout to another. The start and destination locations were chosen such that the students' programs would have to monitor the locomotive's progress and make changes to the locomotive's direction and the states of several turnouts in order to reach the desired destination. The students' programs were not required to find partial paths between the two points (This properly belongs in a Discrete Mathematics course.); rather, the students were allowed to hard code paths of their choosing into their programs. This assignment achieved a number of important objectives:

1. It gave the students experience in process control and communication using pipes.
2. The students had to deal with issues of timing, asynchronous event handling, and event-driven programming.
3. The students had to write programs to generate requests and understand responses according to a well-defined protocol.
4. Students acquired experience in process and job control in a multitasking environment.

5. The students were able to analyze the SRCP protocol to find desirable and undesirable features of a communications protocol.

Assignments #4 and #5: Design and implementation of an application programming interface library to LocoNet

For this assignment, students were to design and implement a library that would provide a set of low-level function calls to control and monitor the status of a LocoNet™-based model railroad. Students were allowed to develop their libraries in C, Perl, or Java language environments. In assignment #4, students were to design and describe the public function-call interface to their library; in assignment #5, students were to implement the library using the LocoNet™ message formats from assignment #1. These assignments provided the students with their first opportunity to write programs that directly interfaced with the physical model railroad. Objectives achieved in these assignments included:

1. The creation of a library that could be used as a component of the semester-long project.
2. The proper design and implementation of libraries, and a demonstration of the role of libraries in the compile/link/execution cycle.
3. The manipulation of data structures and state information across multiple calls to API functions.
4. Discussions of the levels of abstraction available in designing and implementing application interfaces to low-level devices.
5. Discussions of statically versus dynamically linked libraries and the effects on portability and performance.

Assignment #6: Interprocess-level communications using sockets

In assignment #6, students were to write a program similar to the one given in assignment 3 above, but using socket-level communications instead of pipes. This provided students with an introduction to interprocess communication using sockets.

Assignment #7: The railroad server

In assignment #7, the students were to complete the semester project by creating a server which would allow clients on a network to monitor and control the model railroad using the SRCP protocol. The server programs integrated the SRCP protocol (used in assignments 3 and 6), experience with socket-level communications from assignment 6, and the LocoNet library developed in assignment #5. The students were then expected to provide an in-class demonstration of their server.

## ROBOT PLATFORMS

Pioneer Mobile Robots were chosen as a component of the Real-time Lab because of their ease of use and ability to support complex and varied programming. The robots will be used in the next offering of our artificial intelligence course for the purposes of offering a unifying theme that draws together the disparate topics of artificial intelligence, focusing the course syllabus on the role artificial intelligence plays in the core computer science curriculum, and motivating the students to learn by using concrete, hands-on laboratory exercises.

In addition to the many standard problems that can be assigned in advanced courses such as artificial intelligence, we have begun to use our robots in lower-division courses. For example, we have used these robots in our sophomore-level data structures course.[1] With the appropriate introduction we believe that the use of such manipulables promotes retention and turns abstract concepts into real-world problems and solutions. [3, 4]

## A COURSE IN INTRODUCTION TO PROGRAMMING PRINCIPLES

The Computer Science program serves not only its majors but also provides electives for the graduate Educational Technology (EdTech) program. Many of the EdTech students need a programming course but have difficulty with the pacing and content of the courses for computer science majors. The EdTech students come into the class knowing basic computing literacy skills. They can send email, manage their files, and use educational development tools such as HyperStudio. Most have created web pages using a composer. They are confident at plugging in cables and installing software. The skill they lack is programming. Most of the students in this class have been public school teachers from the local school districts and from all levels--elementary through high school and adult education. All are visibly nervous about learning to program and some have been on the verge of tears.

The course uses the Lego Mindstorms Robotics Invention System to motivate the students to learn programming. Lego products are familiar to these teachers and therefore less intimidating than other mobile robots. By including these concrete mechanisms to build, decorate, and control, the element of fun has offset some of the trauma of traditional programming.[13] Although team projects are not used, the students are encouraged to cooperate. They share what works well in the construction of the robots and the programs. The instructor's role changes from 'fount of all wisdom' to resource, collaborator, and facilitator for each of the assignments. Some initial instruction is required when introducing a new language, but the students are eager to try it on their own.

### Assignment #1: Getting Started with the Acrobot

In the first exercise, the students attach the IR transmitter to the computer and download the firmware into the programmable RCX.[10] This process is not overwhelming to these students because of their technology training. But, the massive number of parts (approximately 750) ranging in size from the 4x3x2 inch RCX brick to the quarter inch bushings is intimidating. Sorting the pieces becomes necessary for the students to become familiar with all the parts and to make them touch the pieces for the first time.

After the initial immersion into the parts, the students work individually to build their first robots. Constructivist teaching using a project-based approach with minimal instruction and lots of discussion works well with these highly motivated students. Their project is explained, the books laid in front of them, and they begin. Parts are spread across all working surfaces. And slowly their Acrobot robotic inventions appear. Using the sample programs included in the text. [12] and on the Lego CD, the students teach themselves and each other the first programming language. The RCX code and environment were designed for children to be 'user-friendly' with a minimal amount of reading and lots of encouragement for experimentation. Completing the first assignment brings jubilation and a large sense of accomplishment for these students. They are really programming and seeing their robot scurry across the floor.

### Assignment #2 - Traversing a Figure-Eight

With the success of their first robots, the students are ready to tackle more complex problems. Again the program is built using the RCX code, but this time to get the robot to traverse a figure eight for a total of three times. When their programs are successful, the students translate them into a different programming language, RoboLab. RoboLab has more explicit icons for branching, for looping and for controlling motors and sensors. Because of the additional capability of the language, the students can build and program robots that increase in complexity. RoboLab also requires more structured instruction, explaining the purpose of the icons and how to 'wire' them together.

### Assignment #3 - Imaginative Creature

In assignment #3, the task is to build and program a creature. This term, the students are building a flock of birds with various behaviors, including greeting a visitor at the door, raising an alarm, and circling the room. This exercise is based on the Build It Yourself Cuckoo Bird mission. [11] During this assignment, the concept of variables is taught, using the containers in RoboLab. Although the behaviors of the birds are the same, the creativity exhibited in each bird is wonderful.

### Assignment #4 - Maze Follower

Navigating a maze is a traditional assignment in mobile robotics. The robot needs to be sturdy enough to crash into walls without disintegrating and maneuverable enough to make turns in both directions. Some students try to use the bumper car to satisfy the requirements while others use a line-following robot. No matter which approach is chosen, this solution must be programmed using NQC (Not Quite C).

Both the RCX code and RoboLab are visual languages with icons representing the program flow. NQC provides a more traditional programming environment. The students must write lines of code to accomplish the tasks that they previously did with dragging, dropping and connecting 'little pictures.' Although the program outcomes are equivalent, the new language provides some obstacles for the students, including typing and syntax. More instruction is needed to explain the relationship of the icons to the lines of code.

Sensor input translates into conditional statements. Arrows indicating loops translate into repetition statements. The text provides most of the code for either of the approaches for this robot, with only slight modifications required. Because the students understand the flow of control they need for the programs, they quickly write the new programs and then debug them. When they realize that they have written 'real' programs, they are again empowered and are very energetic in their celebrations.

### Assignment #5 - Programming in C++

Lastly, the students are required to write a program in C++, using a procedural programming approach rather than the object-oriented approach. The reason for using C++ instead of C is the very simple I/O statements of 'cin' and 'cout'. Students write several short programs, including the traditional "Hello World" program, and a program that performs a simple computation. If time and enthusiasm allow, simple programs that utilize a conditional statement and a repetition statement are included. Examples of this type of program are available in any introductory programming textbook.

This course has been very successful in changing the attitudes of these EdTech students. By the end of the course, they can visualize themselves as programmers, at least on an elementary level.

Programming is no longer a scary process; it changes to either exciting or tedious, depending on the student. But it is no longer mysterious.

## OTHER COURSES

COSC 4354 Systems Analysis and Design is the capstone design course. Students work in groups of four on projects for on-campus researchers and off-campus non-profits. The first team from this course to be assigned to develop systems for the Real-Time Lab designed and implemented a Graphical User Interface to control our ActivMedia mobile Robots. This software has since been used as components of numerous other projects.

Various homework assignments for COSC 3370 Introduction to Software Engineering also make use of the Real-Time Lab. During the design phase of the course students are given the problem "Here is a description of all the railroad hardware, now create a design for software to allow control via the Internet." Later during the testing phase of this course, students perform unit testing on various model railroad software control modules.

A student project assigned soon after those listed above was in a summer short course covering the X Window System. Each student created a graphical user interface to display the current status of the system, control train speed and direction, and control each turnout. Simple display status information included the location of each train. This required the use of the C library routines and device drivers written by earlier Systems Programming students to determine both the number of trains on the track and their lengths. (Trains may have varying numbers of attached cars.) Other information displayed included the actual and scaled speed of each train, direction of each turnout, crane status, and pre-collision warning messages. Effectively displaying this wide array of information in real-time presented an exciting challenge to the students. Effective interfaces will become modules to be used by later projects.

As a challenging problem for our Artificial Intelligence course students were asked to have an application automatically move a single train from one point on the track to another point with the minimum number of changes to turnouts. In other words, implement a version of the travelling salesman problem. This forced the students to draw on their data structures background to determine an appropriate abstract data type to represent the track.

Building on the previous problem, we have students design, implement and test algorithms to have one application automatically move a train from point A to point B, while a separate application simultaneously moves a different train from point C to point D. Students have provided two versions of solution to this problem: first, both applications running on a single machine allowing the shared resources to be controlled via semaphores; second, each application running on separate machines, requiring one of the distributed control methods described above.

As to robot use in the artificial intelligence course, the first law for great many robot projects can be stated as "Move about and don't get stuck." Jones, et al [9] describe a series of introductory robotic projects the first of which is described as in the previous sentence. They call this project "The Lewis and Clark" project. We use this and variations of several other projects described by Jones in which we require students to incrementally incorporate the sensor suite of the robots. These projects have names such as: Mouse, where students are required to program the robots to follow walls; Magellan, where students are required to program the robots to navigate a closed path; Apollo 13, where students are required to program the robots to home in on a single bright light source; Fire!, where robots are programmed to traverse a maze and extinguish a lit candle placed somewhere in the maze. (See www.trincoll.edu/~robot/ for a description of the Fire Fighting Home Robot Contest.)

## GRADUATE STUDENT PROJECTS

Moreover, this laboratory is being utilized by master's degree students to complete more advanced projects. As an example, one student is developing a "path finder" module for the navigation of one of our mobile autonomous robots. This project is utilizing a "Best First Search" algorithm to determine a path for a robot between two fixed points and a series of intermediate points in a predefined external world. The Best First Search Algorithm is a "goal directed" and "knowledge based" heuristic algorithm. That is, the points are well defined at the outset and all points of the pre-defined world are taken into consideration during the search. The value of a heuristic function is calculated at each node or location along the navigation path. This value is an estimate of the cost to move to the next location.

Another student has created an API to allow one of our robots (or robot simulator) to be controlled by a CLIPS expert system. This will enable AI students to easily create an intelligent control system without having to know details of interfacing with sonars or drive motors.

## SUMMARY

We see the Real-Time Lab as a highly effective means of turning abstract ideas into firm realities while exciting the interests of students. This inexpensive system supports the teaching of a vast array of computer science concepts. We expect that involving science students in "real-life" challenging problems in a setting that they find interesting will promote retention and aid in recruitment. [6, 8] While this paper has introduced the hardware setup and overviewed numerous potential projects, more information concerning the Real-Time Lab can be found at www.sci.tamucc.edu/trainlab.

### REFERENCES


[1.] Dannelly, R. S., "Use of a Mobile Robot in A Data Structures Course", The Journal of Computing in Small Colleges, Vol. 15, No. 3, March 2000, pp. 85-90.

[2.] Dannelly, R. S. and C. Steidley, "A Hardware Laboratory for Incorporating Computer Science Theory with Real-World Interests and Difficulties, *Proceedings of the 2000 ASEE-GSW Annual Conference*, CD-ROM Session 61, Paper C1, March, 2000.

[3.] Kumar, D. and L. Meeden, "Robots in the Undergraduate Curriculum", The Journal of Computing in Small Colleges, Vol. 13, No. 5, May 1998, pp. 105-112.

[4.] Kumar, D. and L. Meeden, "A Robot Laboratory for Teaching Artificial Intelligence", SIGCSE Bulletin, Vol. 30, No. 1, March 1998, pp.341-344.

[5.] Steidley, C., "An Undergraduate Laboratory for Real-Time Software Systems Development", *Journal of Computing in Small Colleges*, Vol. 11, No. 7, April, 1996, pp. 1-5.

[6.] Steidley, C., "A Laboratory for Real-Time Software Systems Development", Computers in Education Journal, Vol. VI, No. 2, April-June 1996, pp. 10-14.

[7.] Steidley, C. and M.M. Asoodeh, "Developing An Interdisciplinary Hardware Laboratory With CIM Capabilities", *Proceedings of the 1996 Annual Conference of the American Society for Engineering Education inaugural CDROM edition*, Session 1620, June 1996.

[8.] Steidley, C. and M.M. Asoodeh "Developing an Interdisciplinary Computer Science/Industrial Technology Laboratory", *Proceedings of the 1996 Annual Conference of the Gulf-Southwest Section Conference of the American Society for Engineering Education*, March 1996,pp. 366-370.

[9.] Jones, Joseph L., Anita M. Flynn, Bruce A. Seiger, *Mobile Robots: Inspiration to Implementation*, A.K. Peters, Ltd., Natick, MA, 1999.

[10.] Baum, Dave. *Dave Baum's Definitive Guide to LEGO Mindstorms*, Apress, Emeryville, CA, 2000.

[11.] Build-It-Yourself Toy Laboratory. *Mission #6 - Remote Control Mobile, Cuckoo Bird Designs.* Available from http://www.build-it-yourself.com/ (Cited 2/22/02).

[12.] Erwin, Benjamin. *Creative Projects with Lego Mindstorms*, Addison Wesley, Boston, 2001.

[13.] Patterson-McNeill, Holly and Binkerd, Carol. "Resources for Using Lego Mindstorms", *The Journal of Computing in Small* Colleges, Vol. 16, No. 3, pp. 48-55, March 2001

CARL STEIDLEY is Professor of Computer Science and Chair of Computing and Mathematical Sciences. His interests are in the applications of artificial intelligence, real-time computing, and robotics. His most recent extra-university research and development appointments have been with NASA Ames Research Center, Oak Ridge Natl. Labs, and Electro Scientific Industries in Portland, OR

PATRICK R. MICHAUD is a Professor of Computer Science and the Director of the Division of Nearshore Research at Texas A&M University-Corpus Christi. He has been principal investigator or co-principal investigator for many research activities. His primary research interests are in software development methodologies, environmental information systems, and the impacts of the Internet.

R. STEPHEN DANNELLY is Associate Professor of Computer Science at Texas A&M University – Corpus Christi. Dr. Dannelly's interests are in software engineering, environmental modeling, and involving undergraduates in research. Dannelly's most recent funding has come from NASA, NSF, and the U.S. Army.

HOLLY PATTERSON-MCNEILL is Associate Professor of Computer Science at Lewis-Clark State College in Lewiston, Idaho. Her interests include computer science education, especially cooperative learning and the application of robotics to motivate learning, and human-computer interaction, especially usability issues and emergent communities in cyberspace.