Robotic Design Studio: Exploring the Big Ideas of Engineering In a Liberal Arts Environment[1]

Franklyn Turbak[2]

Robbie Berg[3]

Final version of a paper to be published in
Journal of Science Education and Technology

Submitted January 21, 2002

Suggested Running Head: Robotic Design Studio

[1] This is a significantly expanded version of an extended abstract that appears in (Stein, to appear).
[2] *Corresponding Author:* Computer Science Department, Wellesley College, 106 Central St., Wellesley, MA 02481; fturbak@wellesley.edu; (781) 283-3049.
[3] Physics Department, Wellesley College, 106 Central St., Wellesley, MA 02481; rberg@wellesley.edu

ABSTRACT

In this paper we argue that it is important to introduce liberal arts students to the essence of engineering. Toward this end we have developed *Robotic Design Studio*, a course where students learn how to design, assemble, and program robots made out of LEGO® parts, sensors, motors, and small embedded computers. The course has no prerequisites and has attracted students from a wide range of backgrounds. The course culminates in an exhibition where students show off the robots that they have designed and built. These creative projects tie together aspects of a surprisingly wide range of disciplines. *Robotic Design Studio* represents an alternative vision of how robot design can be used to teach engineering in a way that is more inclusive and provides more room for artistic expression than contest-centered formats. A web site with detailed descriptions of student projects and all other course materials can be found at: `http://cs.wellesley.edu/rds`.

Keywords: Robots, design, engineering, liberal arts, undergraduate education

INTRODUCTION

Traditionally, engineering courses have had little or no place in a typical liberal arts education. A standard argument is that engineering does not belong in a liberal arts curriculum because it is too practice- and detailed-oriented. The purpose of liberal arts education is to give students the necessary set of intellectual tools to live fulfilled lives, not to give a narrow professional training.

> *The purpose of the [liberal arts] college was partly defined in contradistinction to other forms of education: a [liberal arts] college education was supposed to be broad rather than specific, "liberal" rather than professional, relevant but not "narrowly vocational". (Reuben, 1996)*

This separation between engineering and a liberal education goes back a long way: the use of word "liberal" to describe this kind of education dates to the ancient Greeks. They conceived of the liberal arts as an education for "free men", namely those who had the luxury of pursuing ideas and thoughts without the burden of having to do something as mundane as making things (Farrington, 1949).

Today, most people view engineering as characterized by discipline-specific knowledge and skills studied at the university level by a small handful of specialized technicians. But we argue for a broader perspective in which the essence of engineering is viewed as a fundamental component of a liberal arts education for the following reasons:

- *Constructionist Learning*: At its core, engineering is about designing and building solutions to problems. Educational research based on constructionist theories of learning has shown that people's richest learning experiences often occur when they are engaged in creating, designing, and making personally meaningful artifacts (Papert, 1980; Resnick and Ocko, 1991; Papert, 1994; Kolodner *et al.*, 1998). Engaging students in designing and building activities helps them to become better learners and problem solvers – key goals of a liberal arts education. Yet, in large part because of the absence of engineering, design-based learning is largely missing from the liberal arts curriculum. Interestingly, artistic disciplines such as sculpture, painting, music, and writing tend to have more constructionist activities than the sciences, where a large part of the laboratory experience typically involves duplicating experiments designed by others.

- *Big Ideas of Engineering:* Every field has its “big ideas”: key concepts necessary for understanding the field that often shed light on other disciplines as well. There are many “big ideas” in engineering (*e.g.*, iterative design, real-world constraints, tradeoffs, feedback, complexity management techniques) that are important for understanding not only classical engineered systems but also for understanding social systems and the natural world. Exposure to these ideas, which are typically lacking in a liberal arts education, gives students a new conceptual framework for understanding a wide range of disciplines.

- *Interdisciplinary Nature:* Any given engineering task almost always involves solving problems in multiple disciplines, typically including not only math and the natural sciences but also human factors, sociology, economics, politics, and art. An oft-touted advantage of a liberal arts education is that it encourages students to cross traditional barriers between disciplines and make connections between them. Engineering problems are an natural source of interdisciplinary activities.

- *Technological Literacy*: An important goal of a liberal arts education is to allow students to understand and appreciate the modern world and to be able to make informed decisions about critical issues. In today’s world we constantly interact with a vast array of often intimidating and mysterious technological objects. When students become designers and builders of technology, rather than passive consumers, much of the mystery and intimidation vanishes (Resnick *et al.*, 2000). The notion that technological literacy should be a core component of a liberal arts curriculum was a cornerstone of the Sloan Foundation's influential New Liberal Arts Program (Koerner, 1981; Goldberg, 1990).

The issue of teaching engineering in a liberal arts environment has achieved significant national attention recently due to the development of a new engineering department and degree program at Smith College (Sippel, 1999). We take this development, in addition to wellestablished engineering programs at liberal arts colleges such as Swarthmore College and Trinity College, as strong evidence that engineering is compatible with the liberal arts. Our focus in this paper is on the vast majority of liberal arts schools that do not, and are never likely to, have an engineering program. We believe that, quite separate from whether a college offers professional engineering training, it is vitally important to expose all liberal arts students to some form of engineering for the reasons given above.

Motivated by this belief, our aim is to help change the image and the reality of engineering, transforming engineering into a subject that can be learned and enjoyed by all liberal arts students. We set out to develop an introductory engineering experience at our home institution, Wellesley College, a undergraduate liberal arts college for women. We designed our course guided by the following goals:

- *Accessibility:* Because we believe that all of today's liberally educated students should have an understanding of the big ideas of engineering, we wanted our course to be accessible to all students, regardless of background. In order to capture the interests of a diverse student body, many of whom do not initially consider themselves to be "interested in engineering", we wanted the course to have multiple entry points -- *i.e.*, to be appealing to students from a wide range of disciplines and to offer appropriate challenges and rewards both for students with little or no technical background as well as those with significant experience in science and math.

- *"Do" Engineering, Not Just Study It:* In order to capture the essence of engineering, we wanted to engage our students in the hands-on process of constructing their own engineered artifacts. An alternative approach is for students to study existing engineered artifacts, such as refrigerators, washing machines, and automobile subsystems to see how they work, perhaps in conjunction with studying texts such as (Macaulay, 1988) or (Norman, 1990). This is the approach taken in the course described in (Henderson *et al.*, 1994), where the goal was to overcome the anxiety and intimidation experienced by women engineering students concerning mechanical and electrical devices; such a course could be adapted to the liberal arts setting. While this alternative approach yields valuable insights about engineering and can have a significant hands-on component (indeed, we include some of these sorts of activities in our course), it lacks the key constructionist learning benefit of engineering and does not engage students in essential aspects of the engineering process, such as the iterative design cycle or problem-solving in the context of real-world constraints.

· *Creativity of Expression:* Influenced by constructionism's emphasis on building personally meaningful artifacts, we wanted our course to focus on activities that encourage students to express themselves creatively. We figured that this would not only help to attract students from the humanities but would also give students in the sciences a rare chance at such expression in a science-related course. This goal is somewhat in conflict with typical engineering situations, in which a particular problem must be solved within the constraints of certain resource bounds. Although much creativity may be required to solve the problem within the constraints, there is typically little latitude for an engineer to change the nature of the problem to allow a greater sense of artistic expression. In our course, we take a broader view of engineering in which practical aspects are sometimes relaxed in favor of a creative, artistic viewpoint.

Inspired by the successful use of robot-based engineering experiences in a variety of settings (Martin, 1994; McCartney, 1996; Jones *et al.*, 1998; Beer *et al.*, 1999; Druin and Hendler, 2000; Martin, 2000; Stein, to appear), we decided to try to adapt this approach to a liberal arts environment. The absence of a formal engineering program at Wellesley (or even much of a presence of engineering anywhere in our school's curriculum) makes this setting significantly different from most of the universities and colleges that had previously implemented robot building experiences. This led us to develop a course which is in some important ways quite different from courses developed elsewhere. These differences, and the rationale behind them, are the main focus of this paper.

In pursuit of our goals, we developed *Robotic Design Studio*, an intensive laboratory course in which students are first introduced to the basics of robotics and then work in groups to design, implement, and exhibit their own robotic creations. The name of the course reflects a conscious effort on our part to draw a parallel between the creative design experience in our course and that offered in studio art courses.

In many ways, *Robotic Design Studio* has exceeded our wildest expectations. Our course has no prerequisites and over the last seven years has been taken by over 150 students with a very wide range of backgrounds. Representing 40 different departmental majors and often coming without any prior programming or mechanical building experience, our students have created robots that surprise and delight us with creativity and ingenuity. The course has had high visibility and has generated excitement not only among Wellesley College students but also among the greater Wellesley College community and at other liberal arts colleges as well. In fact, faculty at several other liberal arts colleges are following our lead by adapting the *Robotic Design Studio* course to their home institutions.

The rest of the paper is structured as follows. We begin by describing the *Robotic Design Studio* course in more detail, focusing on the activities undertaken by the students and the materials used in the course. Next, we show how our course exposes students to some "big ideas" of engineering. We then discuss the rationale behind and the consequences of our decision to have the course culminate in an exhibition rather than a competition – something that sets our course apart from almost all other organized robotic activities. We conclude by presenting a few examples of the sorts of robotic projects that students build in our course and discussing our experience in the course.

ROBOTIC DESIGN STUDIO

Course Structure

Robotic Design Studio typically has an enrollment of 20 to 30 students and meets for 12 or 13 four-hour sessions over a three-and-a-half week January session. We have found that our course works well in this intensive short-term mode. Students have few other obligations during this time, and are able to focus considerable energy on the course – a fact that is especially important during the last days of their final project, when many students spend large fractions of the day (and night) working on their robots. The faculty load is also lighter at this time, allowing us to spend significant time mentoring the students outside of the advertised class times. An important practical matter is finding a sufficiently large laboratory to house 30 students, a dozen computers, and a large supply of building materials and still have enough room for students to build and test their robots. Each January, we commandeer a large classroom and turn it into an interim robotics laboratory for the month, often using nearby hallways and classrooms as overflow space. We have considered teaching the course during a regular semester, but would need to address our lack of a dedicated robotics laboratory in order to do this.

The first six or seven sessions of the course are organized around a series of challenges and focus mainly on introducing the students to robot programming and mechanical and structural design. For example, on the very first day of the course, we ask our students to build an interactive “kinetic sculpture" as an introduction to working with programmable bricks, sensors, and motors. Several challenges involve studying *SciBorg* – a pre-constructed LEGO® robot that follows a line – to deduce how it works and then programming it to accomplish several other tasks. After they have learned idioms for making strong LEGO® structures, the students must build an “indestructible box” that can survive a six-foot fall without breaking apart. There is also a gearing challenge that involves building a drag racer that can carry a one kilogram mass and a Handy Board, powered by a single low-torque LEGO® motor.

During the second half of the course, the focus shifts to designing and building a robot from scratch. After a series of brainstorming sessions, students divide in groups of one, two or three members to work on their final projects. When forming teams for a project, students are encouraged to choose teammates with complementary strengths. For example, an ideal team should have members with programming experience, mechanical know-how, artistic sense, and good writing and presentation skills.

The final projects are open-ended. Students have the freedom to create any robot that interests them; they are limited only by their own imaginations and the available resources. This style of project stands in stark contrast to most robot design experiences (and many real-world engineering problems), where students are required to build a robot that must perform a particular task and at the same time must often satisfy additional resource constraints (*e.g.*, be built out of a certain collection of parts, fit within a certain volume, consume a limited amount of power, *etc.*). The open-ended structure of the *Robotics Design Studio* final project supports all of our course goals: (1) it helps to attract students with wide ranges of backgrounds and interests; (2) it actively involves students in a hands-on design and implementation project; and (3) it allows students to express themselves creatively.

During the second half of the course, we present tutorials on advanced topics for students who are interested. We also teach them simple web page design so that they can document their robots with a collection of web pages that are added to an on-line museum of all robot projects.

During this time, final project teams give two informal presentations on the progress of their project in which they elicit feedback from the rest of the class.

Students keep a design journal to document their journey through the course. In the design journal, students document their approaches and solutions to the challenge problems and the evolution of their final robot project, from brainstorming to final product. The design journal encourages the students to reflect on the process of engineering as well as the final product. Digital cameras are an important tool for helping students to document their projects. We are beginning to explore the use of digital video for documentation purposes as well.

On the final day of the course students present their robots to the community in a public exhibition. The exhibition format is significantly different than the competitions that characterize most robot courses, and is discussed at further length later.

Hardware

For building their robots, students have access to an extensive, computationally enhanced set of construction materials that consist of a rich assortment of LEGO® mechanical and structural elements, motors and other actuators, various sensors, and also a number of different kinds of programmable bricks (described below). The modularity of the LEGO® parts, as well as of the sensors and actuators (which have standard plugs that connect to ports on the programmable bricks), help to lower the barriers for constructing robots and iterating designs. The fact that many students are already familiar with LEGO® parts and associate them with a playful aspect of their childhood goes a long way towards making robotics seem both accessible and fun to students regardless of their majors and backgrounds.

Although they are not hardware in the traditional sense, we have found that art and craft materials for decorating robots are essential elements of the construction materials supplied to the students. They dramatically increase the opportunities for the robot projects to have strong narrative and aesthetic components. The use of craft materials gives the robots in our course a very different "look and feel" than most other LEGO® robots build by college students that we have seen. We have found that hot glue is indispensable for mixed media creations involving LEGO® parts, sensors, actuators and craft materials.

Programmable bricks are small, portable computers capable of interacting with the physical world through sensors and actuators (Resnick, *et. al.*, 1996). The programmable brick extends the robotic construction kit, enabling students to build not only structures and mechanisms, but also behaviors. With programmable bricks, students can spread computation throughout their worlds, using programmable bricks to build autonomous robots and "creatures".

We have employed two kinds of programmable bricks (Fig. 1), both of which were developed at the MIT Media Laboratory:

- *Handy Boards:* The Handy Board is a commercially available palm-sized computer with four actuator ports, 16 sensor ports, an infrared receiver, a beeper, and a 32 character LCD display (Martin, 2000). Programs can be downloaded onto the Handy Board via an electronic tether connected to a host computer. The Handy Board can execute a downloaded program without the tether.

- *Crickets:* Crickets are a new generation of smaller programmable bricks about the size of a 9 volt battery (Martin *et. al.*, 2000). Crickets are smaller, lighter, and cheaper than their predecessors and have enhanced communications capabilities. Each Cricket has an infrared receiver/transmitter pair used for downloading programs and for communicating with other Crickets. Although a Cricket has only two actuator ports and two sensor ports, it has a bus connection that allows it to use many more devices, and inter-Cricket communication makes it possible to use several Crickets to build a sophisticated robot.

In early versions of *Robotic Design Studio*, there were very few Crickets, and almost all projects were based solely on Handy Boards. Now we have about equal numbers of Crickets and Handy Boards, and projects are shifting more towards Crickets or combinations of Handy Boards and Crickets.

The commercially successful LEGO® Mindstorms™ (RCX)[4] product introduced in 1998 was inspired by the programmable brick work at the MIT Media Lab. The availability of this product greatly facilitates adoption by other schools of the kind of robotic design activities that we have developed for our *Robotic Design Studio* course. In our course, we have not made significant use of RCX programmable bricks for several reasons: (1) the RCX uses sensors and actuators that are incompatible with the ones we used for Handy Boards and Crickets; (2) the RCX uses

[4] `http://mindstorms.lego.com/`

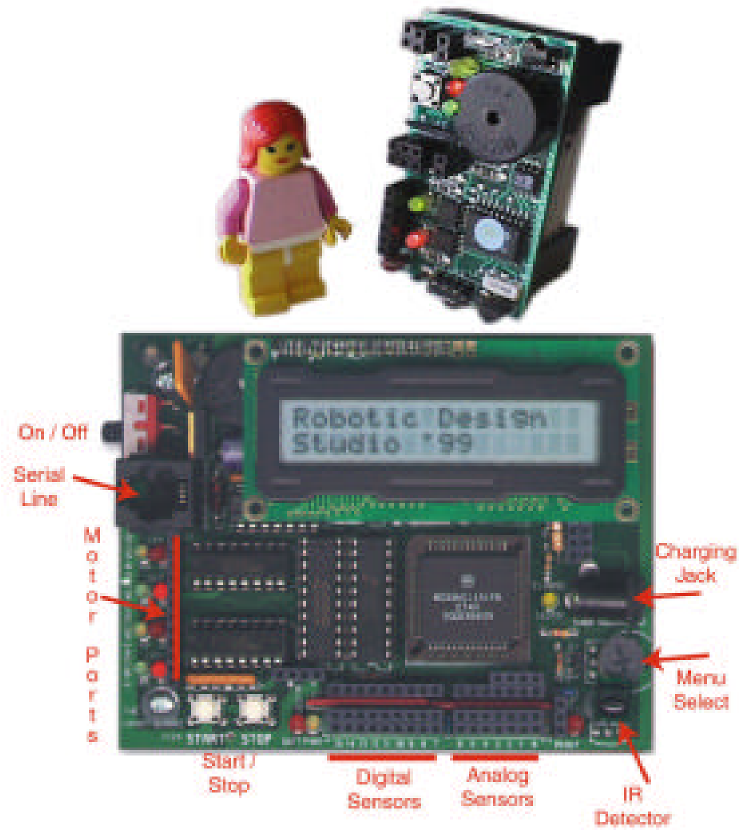

Fig. 1. Programmable bricks. The top device is a Cricket, which is about the size of a 9 volt battery. The bottom device is a Handy Board, a palm-sized robotic computer. The devices and LEGO® figure are shown at roughly the same scale.

different programming environments than the ones we use for Handy Boards and Crickets (see below); and (3) the RCX has only three actuator ports and three sensor ports. Like Crickets, the RCX does have the ability to communicate via infrared with other RCX bricks, but the lack of a bus port and the relatively large size and expense of an RCX brick makes it far less flexible than Crickets for building sophisticated robots out of multiple bricks. An advantage of the RCX is that it is physically more robust than a Handy Board because its electronics are completely enclosed in a plastic casing, but this is more important for students younger than the college-age students that we target in our course. The open architectures of the Handy Boards and Crickets make it relatively easy for students to solder together their own custom connectors and sensors.

As an example of the kind of robot that can be built with these materials described above, consider *SciBorg*, a Handy Board based LEGO® robot that we designed to introduce our students to the basics of mechanical design and robot programming. *SciBorg* is a mobile robot with two independently driven wheels, two frontward-pointing light sensors as “eyes”, two downward-pointing reflectance sensors that can distinguish between black and white surfaces, and front and back bumpers attached to touch sensors. Fig. 2 shows a schematic of SciBorg, illustrating its sensors and motors and how they are connected to the Handy Board.[5]

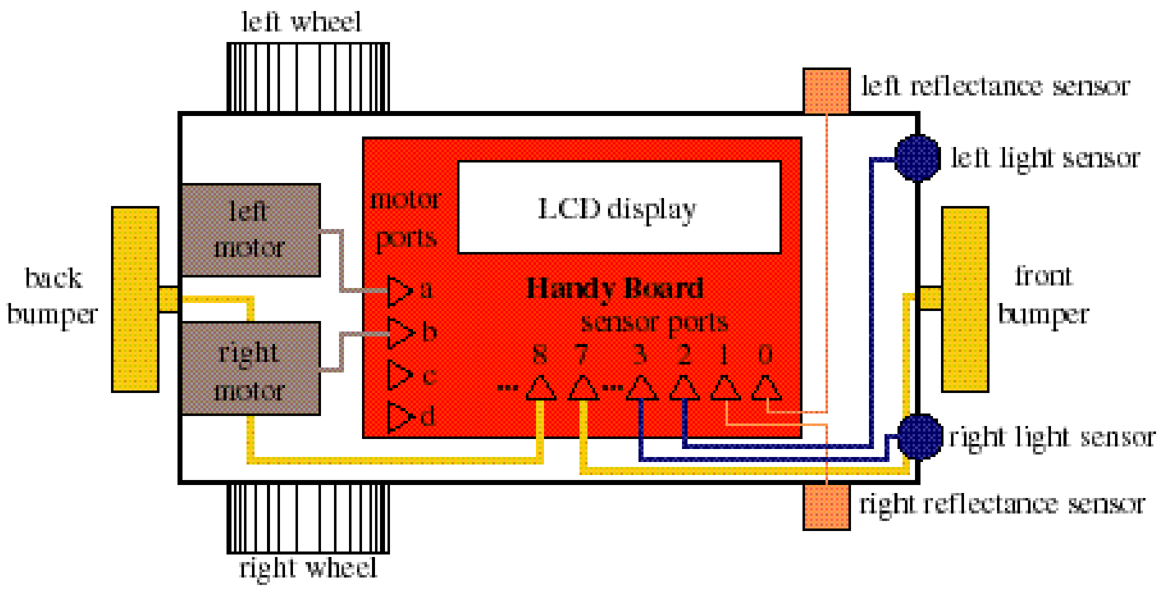

Fig. 2. Schematic of *SciBorg* (top view). It has two motors (connected to motor ports a and `b`); two reflectance sensors (connected to sensor ports `0` and `1`); two light sensors (connected to sensor ports `2` and `3`); and two bumpers (connected to sensor ports `7` and `8`).

Software

In *Robotic Design Studio*, programmable bricks are programmed using Handy Logo and Cricket Logo. These are dialects of the Logo programming language (Papert, 1980) that were developed for Handy Boards and Crickets by Brian Silverman and others at the MIT Media Lab. These versions of Logo have been extended with special primitives for obtaining sensor data, controlling actuators, and managing sequential and concurrent processes.

Fig. 3 shows a sample Handy Logo program that controls a *SciBorg* to follow a black line on a white background. The program uses *SciBorg*’s two downward-pointing reflectance sensors to detect the black line and *SciBorg*’s independently driven wheels to move forward. Each fragment of the program beginning with **`to`** and ending with **`end`** defines a new procedure, which effectively extends Logo’s vocabulary with a new verb.

[5] For pictures of *SciBorg* and instructions on how to build one, see `http://cs.wellesley.edu/rds/sciborg.html`

```
to follow-line
  go-forward
  loop [if sees-black? left-sensor
          [pivot-left]
        if sees-black? right-sensor
          [pivot-right]]
end

to go-forward
  left-wheel on thisway
  right-wheel on thisway
end

to pivot-left
  left-wheel off
  right-wheel on thisway
end

to pivot-right
  right-wheel off
  left-wheel on thisway
end
```

```
to left-wheel
  a,
end

to right-wheel
  b,
end

to sees-black? :sensor-value
   output :sensor-value > 100
end

to left-sensor
    output sensor 0
end

to right-sensor
    output sensor 1
end
```

Fig. 3. Handy Logo program for *SciBorg* line follower.

The `follow-line` procedure is the entry point to the line-following algorithm. It causes *SciBorg* to go straight until one of its reflectance sensors detects black. Once *SciBorg* has detected the black line, it follows the line by keeping it between the two reflectance sensors and forever wiggling back and forth in a “drunken walk” forward. That is, if it sees black with its left sensor, it pivots about its left wheel until it sees black with its right sensor, at which point it pivots about its right wheel, and so on.

The program in Fig. 1 employs many auxiliary procedures to clarify the structure of the program. Each one is very simple and has a carefully chosen name indicating its purpose. For instance, `go-forward` is achieved by turning both the left and right wheels on in the forward direction. Here `on` is a Handy Logo primitive for turning on the currently selected motor; `thisway` is a primitive that ensures the wheel turns in the forward direction; and `left-wheel` and `right-wheel` are user-defined procedures for selecting the actuators (in this case, motors) plugged into the motor ports named `a` and `b`, respectively. The `pivot-left` (`pivot-right`) procedure uses similar behavioral fragments to pivot forward around the left (right) wheel. The `sees-black?` procedure returns (via `output`) a truth value that indicates whether the given sensor reading is greater than the threshold for blackness (the constant `100`). The `left-sensor` and `right-sensor` procedures return the sensor readings available on sensor ports `0` and `1`, respectively.

The auxiliary procedures are not strictly necessary. It is possible to express the complete line-following algorithm using just one procedure in which all the other abstractions have been "in-lined", as shown below:

```
to follow-line
  a, on thisway b, on thisway
  loop [if (sensor 0) > 100
          [a, off b, on thisway]
        if (sensor 1) > 100
          [b, off a, on thisway]]
end
```

However, this style of programming is discouraged because it leads to code that is difficult to read and maintain. Students find the first version of the line-following program easier to understand because it reads almost like English text. Moreover, when students are asked to reprogram *SciBorg* to perform other tasks, it is helpful for them to use the first version as a starting point because it provides "vocabulary words" useful in many *SciBorg* programs. This highlights the fact that that even simple programs like `follow-line` can be used to teach essential complexity management techniques like abstraction and modularity (Abelson and Sussman, 1996).

We specifically chose Handy/Cricket Logo as the programming environments for our course in preference to some other programming environments available for Handy Boards and Crickets. Handy Boards are typically programmed using Interactive C (IC), a variant of the C programming language. While in some respects IC is more powerful than Handy Logo (*e.g.*, it supports data structures like arrays lacking in Handy Logo), its syntax and user interface are less intuitive for novice programmers. The Logo language and environment were designed to be simple enough for grade school students to use, so it is not surprising that college students find them very easy to learn. Indeed, we expect our students, even those who have never programmed before, to write simple Handy Logo programs on their very first day of class and to program *SciBorg* to perform some non-trivial tasks on the second day of class.

Other programming environments available for the Handy Board, Cricket, and RCX include graphical environments in which programs are constructed by snapping together jigsaw-like pieces. Like Logo, these environments are aimed at making programming accessible to children. However, these environments are significantly more tedious than text-based environments for writing non-trivial programs and do not have the full expressive power of Handy/Cricket Logo. We believe that Handy Logo and Cricket Logo strike a nice balance of being easy for nonprogrammers to learn while at the same time being sufficiently expressive for students with previous programming experience.

The support for concurrency in Handy/Cricket Logo deserves special mention. Specifying even simple robot behaviors often requires the concurrent composition of sequential processes. For example, suppose we want to modify the line-following *SciBorg* to also play a song every time its front bumper is pressed. This behavior can be expressed in Handy Logo as follows:

```
to follow-line-and-play-song
  when [front-bumper?] [play-song]
  follow-line
end
```

Assume that `play-song` plays a song and that `front-bumper?` returns true if the front bumper is being pressed and otherwise returns false. The **`when`** command spawns a brand new process that continually looks for `front-bumper?` to change from false to true and invokes `play-song` when this happens. The `follow-line` invocation, which generates its own control loop, executes as a separate process. The ability to concurrently compose two control loops is essential for expressing this sort of behavior in a modular fashion. Without concurrency, it would be necessary to manually interleave the two loops into a single loop – something that is extremely complex, even for expert programmers.

Concurrency is often treated as an advanced computer science topic – one often not covered until late in the undergraduate curriculum. We find it fascinating that students in *Robotic Design Studio*, many of whom have never programmed before, stumble across problems requiring concurrency on their very first day of the course, and can effectively use concurrency primitives like **`when`** within the first week of the course. This suggests (1) that robotics is a natural domain in which to teach concurrency; and (2) that concurrency is a fundamental problem solving technique that should be taught much earlier in the computer science curriculum. These points are made cogently elsewhere, particularly in (Stein, 1998).

WHAT'S THE BIG IDEA?

We should be clear what we mean when we say we want our students to learn the "big ideas of engineering." We are *not* arguing for a standard professional engineering training that emphasizes the mastery of narrow, though perhaps very practical, skills (*e.g.* C++ programming or digital circuit design). Rather, we seek to expose students to broad concepts, principles, and problem solving techniques that characterize the essence of engineering across particular disciplines.

In this section, we discuss several of the big ideas that are addressed in *Robotic Design Studio*. In early incarnations of our course, many of these ideas were only implicit. As we have come to recognize their importance, we have begun to teach many of these ideas explicitly. The value of using robot design projects as a means of learning engineering principles has been recognized by many others (Martin, 1994; McCartney, 1996; Jones *et al.*, 1998; Beer *et al.*, 1999; Druin and Hendler, 2000; Martin, 2000; Stein, to appear). Here we emphasize those principles that we believe are particularly important in a liberal arts context.

Iterative Design

The essence of engineering is imagining something (typically a solution to a problem), designing it, building it, and getting it to work. Robotics is a rich and accessible domain in which to experience this process. Furthermore, in the course of their robotic projects, students learn that engineering is an iterative process in which they continually implement, test, debug, and refine designs. This process stands in stark contrast to many traditional experimental lab courses in the sciences, where students (unlike practicing scientists) rarely have a chance to design and iterate experiments.

In *Robotic Design Studio*, students are first exposed to the design cycle in the context of several challenge problems. Solving the *SciBorg* programming challenges typically involves a fair amount of code debugging, an activity that is *terra incognita* for students without a programming background. Students typically have to iterate their “indestructible box” designs numerous times before their boxes can survive a six-foot fall intact. The drag race gearing challenge has a preliminary contest the day after it is assigned and a final contest on the day after that. This builds into the structure of the challenge an opportunity for improving the race car design between the two contests.

Students gain even more experience with the design cycle in their final robot projects. Often, the process of choosing a project requires several iterations. Initial tinkering with a project idea can suggest that it is too difficult or impractical, in which case it is necessary to simplify the project or consider another one. Once a project is deemed promising, numerous iterations may be required to solve a particular problem in the project design. Some examples of thorny problems that have arisen in our course are:

- how can a mechanical arm throw a ball?
- how can a fire-fighting robot dispense shaving cream from a can?
- how can a toy cow dispense milk?
- how can a device sort different kinds of candy?
- how can a robot play a particular note on a xylophone?

As a concrete example of the iterative nature of the design process in robotics, consider the plight of a group that had built the skeleton of a fire-breathing dragon, but did not know how to make it "breathe fire". Here is a wonderful description, written by the group members for the web pages documenting their robot, of the path they took in solving this problem:

> *Our main interest in the dragon was to have it blow flames or smoke. Fire was ruled out due to safety issues so we decided to simulate smoke, the most difficult aspect of building our dragon. First we considered making him blow bubbles or a puff of baby powder, but then, for a more authentic effect, decided to try dry ice or liquid nitrogen. We decided to use liquid nitrogen because it produced more smoke than dry ice when water was added to it. Our next problem, however, was determining how to drop the water into the liquid nitrogen when prompted. We tried making a pipe-like mechanism, having water flow through a straw into the cup of liquid nitrogen. However, making a device using a sliding gear rack to stop and start the flow of water in the straw was not effective. The straw was too hard to pinch and seal off. We realized that just pouring water by some means into the liquid nitrogen would be more effective, but we had trouble finding a container that worked well for pouring. We tried a plastic condiment cup and a balloon-like dropper without success and so ended up with the cap of Dove body wash. It was the perfect shape to be tipped and to dispense water: slightly elongated and cradle-like. The liquid nitrogen is contained in three styrofoam cups, each one placed inside the other. The plastic lid of a Starbuck's insulated cup serves as a lid for the liquid nitrogen. A piece of tubing runs from a hole in the lid upward to the dragon's mouth. (Cheng et al, 2000).*

The Real World

Engineering is something that takes place in the real world, not in a textbook. A key challenge of engineering is that it often involves the design of a complex system with interacting parts, many of which may be quite different in character. Robotics projects naturally lend themselves to design in multiple domains since most robotic projects have mechanical, computational, electrical, and artistic components, and possibly other components as well. They also serve to illustrate that the real world tends to be much messier, noisier and more unpredictable than students have come to expect from the idealized view that dominates textbooks and problem sets.

To highlight the issue of single vs. multiple domains, compare the “bin sorting” problem in computer science to the candy sorting task undertaken by one group as their final project. The bin sorting problem involves placing the elements of a collection into “bins” according to their type. In the classical computer science setting, the problem involves creating an array of lists, say, that partitions a given collection of values by some property. The focus is purely computational: developing an algorithm that solves the problem using the least amount of processor time and/or computer memory. In contrast, sorting candy into bins by type involves solving numerous problems that are idealized in the purely computational problem. For instance, determining the type of a candy is highly non-trivial and requires finding dimensions that available sensors can use to differentiate the candies. Unlike computational objects, candies have mass and volume and must be physically transported from their current position to the correct bin. How to transport a candy, how to determine when the correct bin has been reached, how to insert the candy into the bin – all these are problems in the candy sorting domain that simply are not present in the purely computational domain. Computation is also involved in candy sorting, but it is unlikely that the time and space requirements of the algorithm are primary concerns.

A classic example of real world effects arises in the context of programming *SciBorg* to “ping pong” back and forth between two walls. A common first attempt at solving this problem looks like

```
to ping-pong
  go-forward
  loop [if front-bumper? [reverse]
        if back-bumper? [reverse]]
end
```

where `reverse` is assumed to reverse the directions of both wheels. A problem with this code is that reversing the motors takes time, and during that time the bumper may continue to be depressed, in which case the motors are reversed again by the next execution of the loop. This can repeat many times, giving rise to a behavior where *SciBorg* repeatedly bangs into the same wall after backing up a very short distance.

A more robust solution specifies a different direction of motion for collisions involving different bumpers:

```
to ping-pong
  go-forward
  loop [if front-bumper? [go-backward]
        if back-bumper? [go-forward]]
end
```

In this case, if the front bumper is depressed for multiple executions of the loop, it executes `go-backward` multiple times, which has the same effect as executing it once. Yet even this approach is not immune to real-world problems. Occasionally when the front of *SciBorg* collides with a wall, the impact causes the back bumper to lift slightly, and shortly after *SciBorg* has started moving backwards, the back bumper falls back down, depresses the back touch sensor, and causes *SciBorg* to go forward again!

There are numerous environmental factors that can affect robot behavior. It is not uncommon for students to discover that a robot that works fine in the robotics lab fails to work properly in the exhibition area. Often, this is due to sensor thresholds. It is tempting to use “magic constants” for sensor threshold levels when programming a robot, like the number 100 for measuring blackness in the line-following program presented earlier. However, numbers appropriate for one environment (*e.g.*, a shaded classroom lit by fluorescent bulbs) may not be appropriate for another (*e.g.*, a sunlit lounge). For this reason, we encourage students to design their robot programs with “field settable” threshold values that they can change when they move the robot from room to room, or, even better, to use auto-thresholding techniques that measure ambient properties of a room to automatically calculate an appropriate threshold. Flash photography, halogen lamps, bright sunlight, and infrared crosstalk from other robots often disrupt the behavior of robots that depend on photocell or infrared sensors. Other environmental factors, such as the color or friction of a carpet, can greatly influence robot behavior.

The above situations exemplify a class of real-world phenomena that would simply not arise in a simulated “ideal” environment. These phenomena, which can be extremely difficult to predict and debug, occur all the time in the context of robotics. While they can be very frustrating to deal with, they teach important lessons about problem solving, unexpected interactions between subsystems, and the limits of ideal models.

Tradeoffs

Due to the multiple interacting domains and subsystems in a typical engineering task, solving one problem can create others. For instance:

· adding cross-bracing beams to an unstable structure prevents it from falling apart, but makes it too heavy;

· modifying the gear ratio of a gear train stops a car from stalling, but makes it go slower than desired;

· using extra distance sensors helps a candle-extinguishing robot align itself to a wall, but ties up sensor ports needed for detecting the candle.

Sometimes there are clever ways to satisfy what appear to be conflicting goals. However, in other cases, the conflict between goals is fundamental and it is impossible to achieve all of them. For example, there is no way to circumvent the fact of physics that the torque and angular velocity at the output of a gear train are inversely related. In such situations it is necessary to make a *tradeoff* between goals – to accept that one goal will not be met, or only partially met, so that another goal may be achieved.

Students in our course frequently encounter all sorts of tradeoffs in robotics projects. A good example of this involved a simple device developed by one group for improving communication with a student's father-in-law, who was becoming progressively blind and deaf. The purpose of the device was to be able to convey the answers to simple questions (involving "yes", "no", "maybe", and numbers) transmitted by an infrared remote control unit. The group had to make many tradeoffs involving the weight, size, physical robustness, reliability, and functionality of the device.

An important aspect of tradeoffs is that they make it clear that there is no single "right" or "best" solution to many problems; rather, each solution comes with its own benefits and drawbacks. Tradeoffs appear not only in the world of engineering but in the sphere of public policies: protecting an endangered species may negatively affect worker's livelihoods; tax money used to build a new school is not available for elder care; a mother who moves from welfare to work must find day care for her children. A goal of a liberal arts education is to educate an informed citizenry that can wrestle with these sorts of questions in an intelligent manner. Although students may study these sorts of tradeoffs in their sociology, economics, and political science classes, they do not get hands-on experience with seeing the results of making different tradeoffs. Solving engineering problems gives this sort of experience and, we believe, provides new insight into the meaning of tradeoffs.

Feedback

Control systems involving negative and positive feedback are ubiquitous in nature and in engineered devices. In negative feedback, deviations from a state drives the system back towards that state; for instance, a ball perturbed from the bottom of a trough rolls back to the bottom of the trough. In positive feedback, deviations from a state drive the system further away from that state; for instance, a ball perturbed from the top of a hill rolls further away from the top of the hill.

Robotics is an excellent area in which to learn about, and design with, feedback. A classic example is a light-seeking robot. Imagine a rotating robot with left and right photocell sensors serving as "eyes" that moves according to the following rules:

- if the right eye sees more light than the left eye, the robot rotates toward the right (clockwise);
- if the left eye sees more light than the right eye, the robot rotates toward the left (counter-clockwise);
- if the two eyes see the same amount of light, the robot does not rotate.

If a light is turned on in a dark room containing the robot , the robot will rotate toward the light. This is an example of negative feedback; if a turn towards the light overshoots the target, the rules cause the robot to turn back in the opposite direction, stabilizing its orientation towards the light. Interestingly, the very same robot can be use to illustrate positive feedback. A light-seeking robot can alternatively be view as a darkness-avoiding robot. If the robot is oriented away from the light (*i.e.*, toward the darkness) to a point where each eye detects the same amount of darkness, the robot will be still. But this is an unstable position, and any slight difference between detected light levels in the two eyes will cause the robot to turn away from the darkness.

While this example is very simple, negative and positive feedback can be used to construct robotic "creatures" with surprisingly sophisticated behaviors (Braitenberg, 1984).

In engineering, negative feedback is commonly used as a technique for making systems stable. Both negative and positive feedback are important concepts for analyzing a wide range of phenomena, such as population dynamics, financial markets, and the spread of disease. We believe that feedback is an engineering-related idea that should be in the conceptual toolkit of every liberal arts student.

Controlling Complexity

Engineers use several important techniques to combat a potentially overwhelming volume of complex detail in their work. Two of the most important tools are:

- *Abstraction:* capturing and generalizing idioms, often into "black box" entities with simple interfaces.
- *Modularity:* composing systems out of reusable mix-and-match parts.

In *Robotic Design Studio*, these techniques are evident in many domains. LEGO® pieces are extremely modular; they are carefully designed to fit together in many ways. There are numerous idiomatic assemblies of LEGO® pieces that serve as handy construction abstractions (Martin, 1995). Sensors, motors, and programmable bricks are all examples of abstraction: they are "black box" entities that can effectively be used without understanding the details of how they work. Moreover, they are modular: the standard sensor and motor connectors allow them to be plugged into the programmable bricks in a mix-and-match way. At the programming level, procedures introduce abstractions, enabling the definition of new high-level operations while suppressing implementation details. Collections of procedures (such as `go-straight`, `turn-right`, and `turn-left` in the line-following program) can be reused in mix-and-match ways, effectively defining a new level of programming language.

Once one is aware of these techniques, it is possible to see how ubiquitous they are in the modern world and how much we depend on them for day-to-day functioning. The electric power grid, the water supply, the Internet, and the telephone system are all excellent examples of important abstractions that we use via a host of modular devices. Supermarkets and department stores are purveyors of abstractions; for the most part, we do not want or need to know how a loaf of bread or piece of clothing are made. However, becoming conscious of the extent to which we treat much of the world as black boxes is a first step to asking important questions about what is inside these boxes: *e.g.*, where does the water come from? what ingredients are used in the bread? who manufactured the clothing? An informed citizenry needs to appreciate the ubiquity of abstraction and modularity and to understand both their benefits and their drawbacks in a wide range of domains.

EXHIBITIONS, NOT COMPETITIONS

A critical element in the organization of *Robotic Design Studio* is that it culminates in an *exhibition* rather than a *competition*. Our course was in good measure inspired by MIT's "6.270" *Autonomous Robot Design Competition* course (Martin, 1992). In 6.270, students build robots to compete in a tournament style contest in which robots play a game against one another, with winners advancing to the next round. While competitions are exciting and motivational for many students (particularly the winners), we believe that an exhibition format is more welcoming to novices, attracts a broader range of students, and allows room for a greater range of creative expression, while still maintaining the motivational benefits of a public display of the projects. The exhibition in our course is widely publicized, much like an art gallery opening might be, and is attended by about 250 people, including many children. The advantages to this approach include:

- *Personal expression:* Students experience a deep thrill in being able to start with nothing more than a vision and a "blank canvas" and end up with a tangible, almost living, expression of that vision. Our hope is to capture a feeling similar to that experienced by an artist and novelist when they create a work. Allowing students the freedom to work on a project of their own choosing increases the level of personal investment they feel in their project. This personal connection is, according to constructionist learning theory, a critical factor in creating an environment that is conducive to learning. Furthermore, at the exhibition, students receive the benefit of feedback on their work from a varied and appreciative audience. The fact that robotics is a very broad area and encompasses so many different domains makes it easy for students to find some way to express themselves through robots.

- *With an exhibition, you can't lose:* A core design principle of our course is that we are trying to attract and be welcoming to novices. A competitive event is, for many novices, not very welcoming; the prospect of having to compete against a least some local experts in a public forum is daunting. Exhibitions provide an opportunity in which all participants can be successful. Moreover, the fact that there is not a particular pre-ordained problem to be solved leads to a more forgiving and less stressful environment for the final project. If a final project idea is simply not working out, it can be changed even relatively late in the game and still result in a successful project and positive experience for the group.

- *Low floors, high ceilings:* While our course is an entry level experience for many, some students do come to it with a considerable amount of relevant experience. It is important to provide a suitable challenge for these more experienced students, and the open-ended nature of the exhibition format allows this.

- *Addressing the gender gap:* We cannot help but notice that most robot competitions (as well as most engineering professions) are overwhelmingly male in composition. As we designed our course we were guided by the intuition that an exhibition format would be more likely to attract female participants compared to a competitive format. Of course, since all of Wellesley's students are women, we are not in a position to test this hypothesis. It would be interesting to see what gender mix would be found in a course like ours if it were offered in a co–educational environment.

Competitions do have some advantages over exhibitions. Engineers rarely have the luxury of picking their own problems or having few constraints on the resources they use. In this respect, competitions better reflect the real world; robot contests typically involve solving a problem specified by someone else using a limited set of materials. Furthermore, when everyone is working on the same problem, they can gain a better appreciation for solutions developed by others. Some of the negative aspects of competitions can be minimized by steering away from head-to-head contests and inter-team comparisons and instead emphasizing "tests against nature" that focus on improvements in a team's design relative to its early prototypes (Sadler *et al.*, 2000).

We recognize these benefits and also recognize that different students have different learning styles. For some students, a competition may be more motivating than an exhibition. We address this issue in *Robotic Design Studio* by giving students a final project option of starting to build a robot that will compete in the annual "Fire Fighting" robot contest held at Trinity College.[6] During the past three years, three groups of students have built robots for this contest.

ROBOTS THAT TELL A STORY

The ground rules governing the robots that our students build are intentionally kept extremely loose. We provide abundant resources and simply ask the students to build some sort of robot that they would like to show off at the final exhibition. Not surprisingly, over the years an incredibly wide range of projects has been presented. There have been all sorts of whimsical creatures and contraptions, such as a friendly smoke breathing dragon, a gorgeous six-foot-long car wash, a robot that can play "rock, paper scissors" with you, a robotic mother that comforts her crying baby with a bottle of milk, and a road-crossing, egg-laying chicken. There have been re-enactments of great scenes in literature and cinema, such as *The Wizard of Oz* , *Romeo and Juliet*, *The Tortoise and the Hare*, and the story of the Trojan Horse.

Short of attending the exhibitions, probably the best way to get a sense of this variety is to visit our online robot project museum at `http://cs.wellesley.edu/rds/museum.html`, which contains web pages made by each team of robot builders for their projects. Figs. 4 through 10 below present a sampling of the 60 projects students have built in *Robotic Design Studio* during the six years it has been taught, giving a sense of the variety and richness of their efforts.

Looking at the robots built over the years, a few unmistakable themes emerge. There is a strong narrative element to many of the projects; students often use their robots to tell a story, and they enjoy telling stories about their robots. Students often build robots that reflect their interest in other disciplines and extracurricular activities. Finally, most projects manage to combine good engineering with artistic flair and dramatic expression.

[6] `http://www.trincoll.edu/events/robot/`

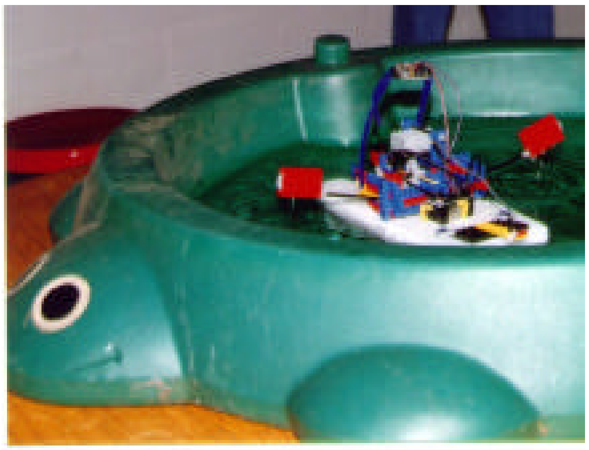

Fig. 4. Inspired by the designer's interest in competitive rowing, *Row-Bot* featured a realistic rowing motion that enables it to paddle around a turtle shaped pond.

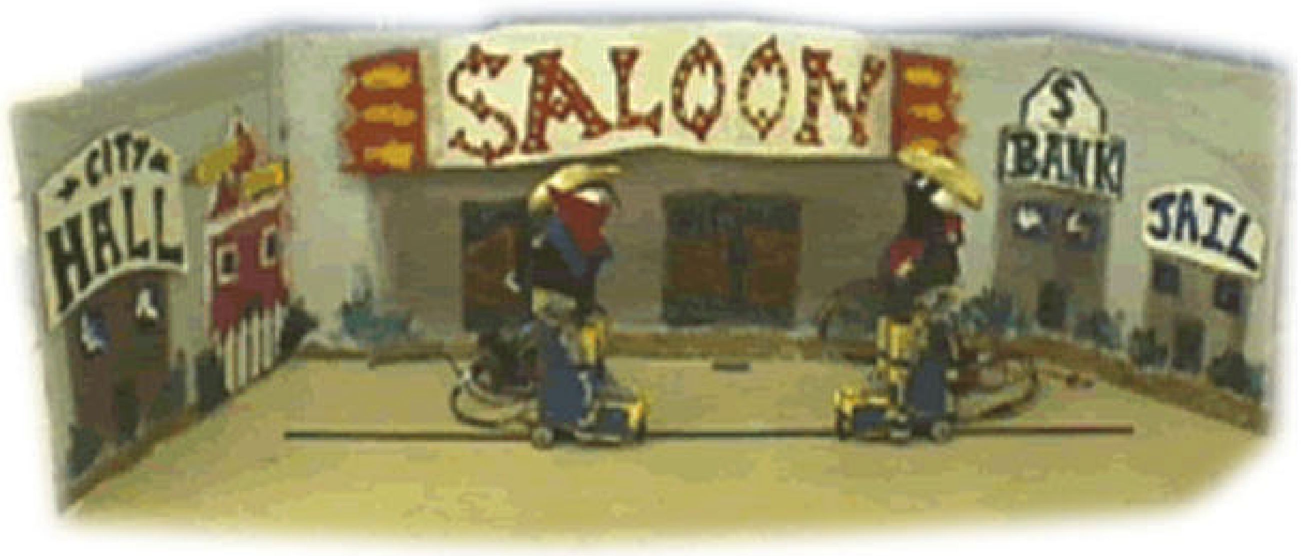

Fig 5. *Western Duel* starred a pair of robotic gunslingers performing the classic pace, turn and fire sequence found in countless cowboy movies. It featured a sense of drama.(the winner was determined randomly) and brilliant mechanical design (the loser slouched over).

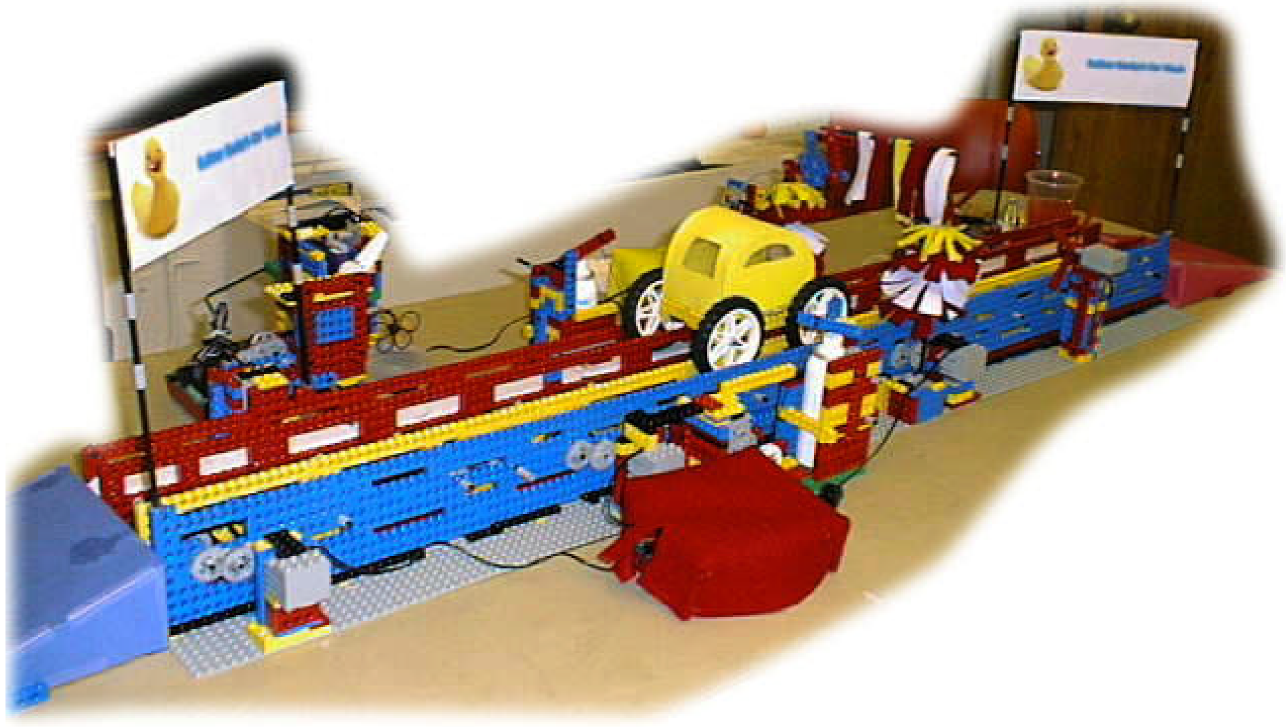

Fig. 6. Inspired by a visit to a local auto wash, the 6 foot long *Rubber Ducky's Car Wash* featured soap bubbles generated by fans blowing on a bubble wands, gentle mists of water supplied by LEGO® driven aerosol cans, and a visually spectacular buffing action provided by colorful felt brushes.

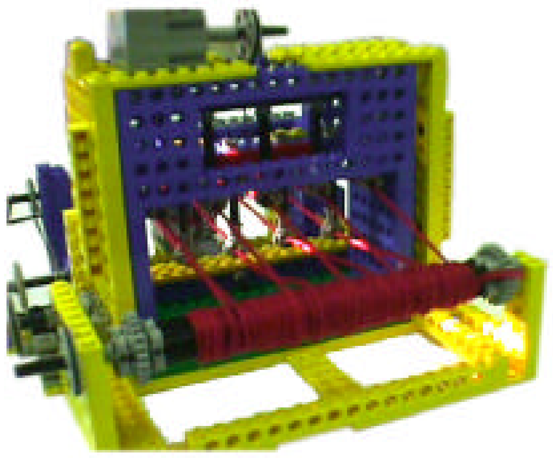

Fig. 7. *Loom* was the brain child of an avid weaver. Built entirely out of LEGO® parts, it could actually perform a rudimentary type of weaving!

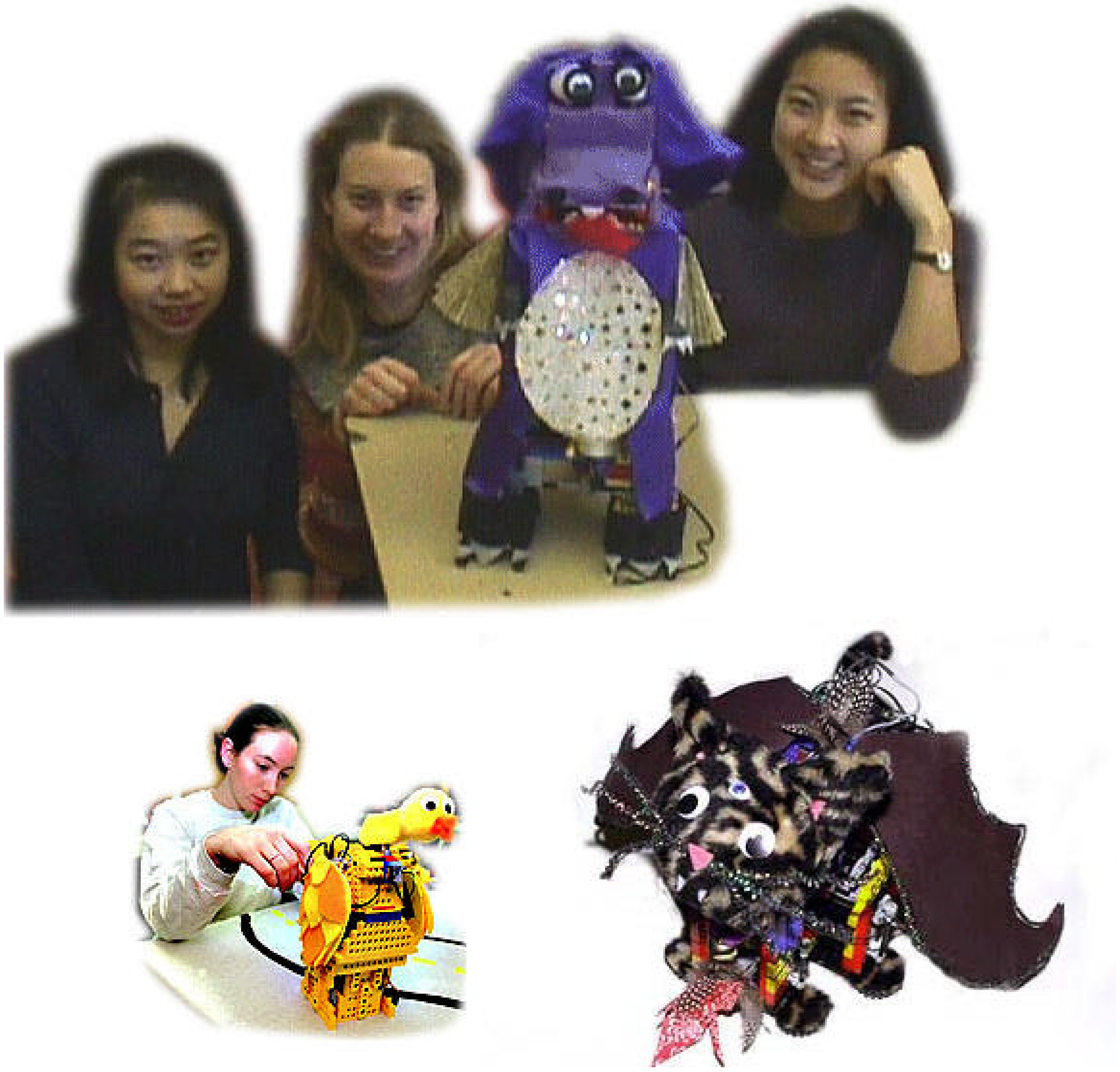

Fig. 8. A Gallery of Whimsical Creatures. Smoke-breathing *Star Dragon*, shown with his creators in the upper half of the figure, used a clever mechanism that involved pouring room temperature water into a container of liquid nitrogen in order to create a billowing vapor whenever he roars. (See the group's discussion of this mechanism in the text.) *Robotic Chicken* (lower left) always looked both ways before waddling across the proverbial road. It also laid a (candy) egg whenever anyone cupped a hand behind it. *The Chimera* (lower right) was an incredible display of artistry and mechanical ingenuity. Not only did it flap its wings, with a wonderfully life-like motion, but it also walked on eight paws and responded to patting its head (purring) or pulling its tail (meowing).

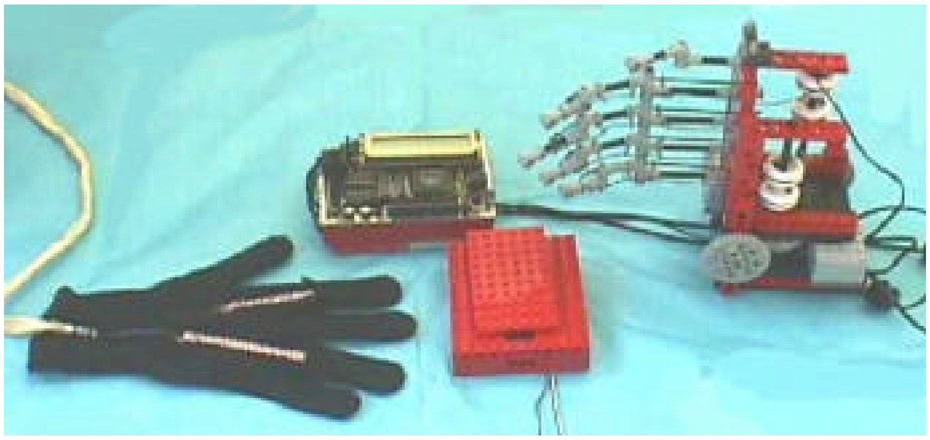

Fig. 9. *Paper, Scissors, Rock* is a robotic version of the common children's game. A robotic hand with four moveable fingers and a stationary thumb is capable of closing itself into a fist for rock, throwing out just two fingers for scissors or throwing out four fingers for paper. The robot randomly "decides" what to throw. The human opponent wears a glove equipped with bend sensors so that the robot "knows" what the human threw and can determine the winner.

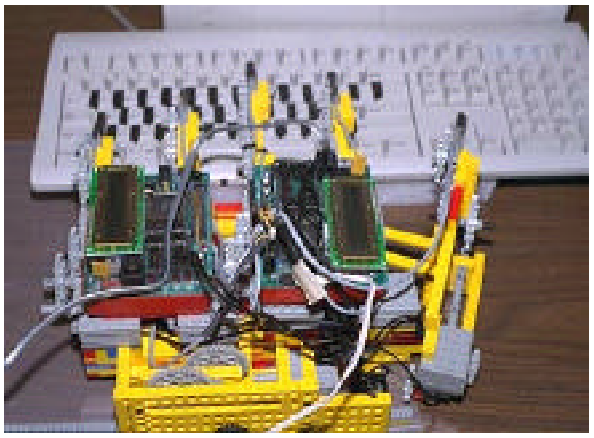

Fig. 10. In the *Handroid* project, a student used two Handy Boards to control a six-motor LEGO® hand that could (slowly) type any specified string of characters. The *Handroid* moved back and forth along a gear rack, and reflectance sensors at the tips of each of its fingers counted keys as it moved.

ENGINEERING FOR EVERYONE

The creativity and range of student projects serves to underscore what a good match robotics is to a liberal arts culture, where people are encouraged to explore and find connections across a range of disciplines. The wide variety of projects that have been built in *Robotic Design Studio* over the years reflects the diversity in enrollments that we have been successful in attracting. The 152 students who have taken the course were roughly evenly distributed between the first-year, sophomore, junior, and senior classes and, as the table below shows, represent a wide range of departmental majors. (The numbers in the table sum to a number much larger than 152 because of the large fraction of students who double major.)

Table I: Enrollments in Robotic Design Studio 1996-2002, by Major.

Africana Studies	**1**	*Computer Science*	**34**	*Middle Eastern Studies*	**1**
American Studies	**2**	*Economics*	**8**	*Neuroscience*	**5**
Anthropology	**1**	*Education*	**1**	*Philosophy*	**4**
Architecture	**3**	*English*	**12**	*Physics*	**16**
Art	**9**	*Environmental Studies*	**1**	*Political Science*	**4**
Astrophysics	**1**	*French*	**2**	*Psychology*	**7**
Astronomy	**2**	*Geology*	**2**	*Religion*	**1**
Biology	**14**	*Greek*	**1**	*Russian*	**2**
Biochemistry	**4**	*History*	**6**	*Sociology*	**2**
Chemistry	**4**	*International Relations*	**3**	*Spanish*	**2**
Chinese	**1**	*Italian*	**1**	*Theater Studies*	**2**
Chinese Studies	**2**	*Latin American Studies*	**2**	*Women's Studies*	**1**
Classical Civilizations	**2**	*Math*	**19**	*Undecided*	**4**
Cognitive Science	**10**	*Media Arts*	**4**		

There is a large and growing community of educational robot builders (Druin and Hendler, 2000; Stein, to appear). Robot contests from MIT's 6.270 to cable television's *BattleBots*[7] have gained high visibility and inspired many. *Robotic Design Studio* represents an alternative vision of how robot design can be used to teach engineering in a way that is more inclusive and provides more room for artistic expression than contest-centered formats.

The model provided by college-level courses such as ours can potentially have a very high leverage outside the university. The format of activities developed at colleges tends to filter down to earlier ages; for example unmistakable echoes of the 6.270 contest can be found in the

[7] `http://www.battlebots.com/`

FIRST LEGO League[8] or *Botball*[9] contests aimed at middle school youth. We hope that *Robotic Design Studio* can also serve as a model for robot builders of all ages.

In our view the ancient Greeks were wrong to separate out (and hence devalue) engineering activities from what they mistakenly believed were more lofty intellectual pursuits. Sadly, the traditional liberal arts curriculum of today often makes this same mistake. Doing the kind of engineering introduced in a course such as *Robotic Design Studio* is a liberating activity that *should* be a component of a liberal education.

ACKNOWLEDGMENTS

Robotic Design Studio has benefited greatly from our collaborations with Fred Martin, Mitchel Resnick, and Brian Silverman. Wellesley students Ruth Chuang '96 and Rebecca Lippmann '98 played important roles in the early development of the course.

The development of *Robotic Design Studio* was supported by NSF grant DUE-9650969, funding from the Howard Hughes Medical Institute, and several Wellesley College Educational Research and Development grants. The New England Consortium for Undergraduate Science Education supported a *Robotic Design Studio* workshop at Colby College in October 1997, organized by Batya Friedman, that helped us to disseminate information about our course to other New England area colleges. The Consortium for Computing in Small Colleges hosted us to lead a similar workshop at Sacred Heart College in April 1998.

REFERENCES

Abelson, H., and Sussman, G. J., with Sussman, J. (1996). *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, MA.

Beer, R. D., Chiel, H. J., and Drushel, R. F. (1999). Using Autonomous Robotics to Teach Science and Engineering. *Communications of the ACM* 42(6): 85-92.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*, MIT Press., Cambridge, MA.

Cheng, M, Openshaw, A, and Wang, S. (2000). Star Dragon (website for *Robotic Design Studio* final project). URL: `http://nike.wellesley.edu/rds/rds00/star_dragon/`.

Druin, A. and Hendler, J. (Eds.) (2000). *Robots for Kids: Exploring New Technologies for Learning Experiences*, Morgan Kaufman /Academic Press, San Francisco.

[8] `http://www.legomindstorms.com/fll/`

[9] `http://www.botball.org/`

Farrington, B. (1949). *Greek Science*, Penguin, Harmondsworth.

Samuel Goldberg (Ed.) (1990). *The New Liberal Arts Program: A 1990 Report*, Alfred P. Sloan Foundation, New York, NY.

Henderson, J. M., Desrochers, D. A., McDonald, K. A., and Bland, M.M. (1994). Building the Confidence of Women Engineering Students with a New Course to Increase Understanding of Physical Devices. *Journal of Engineering Education* 83(4): 337--342.

Koerner, J. D. (Ed.) (1981). *The New Liberal Arts: An Exchange of Views.* Alfred P. Sloan Foundation, New York, N.Y.

Kolodner, J. L., Crismond, D., Gray, J., Holbrook, J., and Puntambekar, S. (1998). Learning by Design from Theory to Practice. In *Proceedings of the International Conference of the Learning Sciences 1998*, The EduTech Institute (Georgia Institute of Technology), Atlanta, GA, pp. 16-22.

Jones, J., Flynn, A., Seiger, B. (1999). *Mobile Robots: Inspiration to Implementation (2nd edition),* A.K. Peters, Wellesley, MA.

Macaulay, D. (1988). *The Way Things Work*, Houghlin Mifflin, Boston.

Martin, F. (1992). The 6.270 Robot Builder's Guide for the 1992 LEGO Robot Design Competition, Epistemology and Learning Group, MIT Media Lab. URL: `ftp://cherupakha.media.mit.edu/pub/el-publications/Manuals/robot-builders-guide/`. See `http://web.mit.edu/6.270/www/` for more information on 6.270.

Martin, F. (1994). Circuits to Control: Learning Engineering by Designing LEGO Robots. Ph.D. dissertation, MIT Media Laboratory.

Martin, F. (1995). The Art of LEGO Design. *The Robotics Practitioner: The Journal for Robot Builders* 1(2). Also available at `ftp://cherupakha.media.mit.edu/pub/people/fredm/artoflego.pdf`.

Martin, F. (2000). *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice Hall, Upper Saddle River, NJ. For more information on the Handy Boards used in this book, see `http://www.handyboard.com/`.

Martin, F., Mikhak, B., Resnick, M., Silverman, B., and Berg, R. (2000). To Mindstorms and Beyond: Evolution of a Construction Kit for Magical Machines. In (Druin and Hendler, 2000). For more information on Crickets, see `http://llk.media.mit.edu/projects/cricket/`.

McCartney, R. (1996). Introduction to Robotics in Computer Science and Engineering Education. *Computer Science Education* 7(2): 135-137.

Norman, D. A. (1990). *The Design of Everyday Things*, Doubleday, New York, NY.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*, Basic Books, New York, NY.

Papert, S. (1994). *The Children’s Machine*, Basic Books, New York, NY.

Resnick, M., and Ocko, S. (1991). LEGO/Logo: Learning Through and About Design. In Harel, I., and Papert, S. (Ed.). *Constructionism,* Ablex Publishing, Norwood, NJ.

Resnick, M., Martin, F., Sargent, R., and Silverman, B. (1996). Programmable Bricks: Toys to Think With. *IBM Systems Journal* 35 (3): 443-452.

Resnick, M., Berg, R., and Eisenberg, M. (2000). Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation*.* *The Journal of the Learning Sciences* 9(1): 17-35.

Reuben, J. (1996). *The Making of the Modern University: Intellectual Transformation, and the Marginalization of Morality*, University of Chicago Press, Chicago.

Saddler, P., Coyle, H., and Schwartz, M. Engineering Competitions in the Middle School Classroom: Key Elements in Developing Effective Design Challenges. *The Journal of The Learning Sciences* 9(3): 299-327.

Sippel, J. (1999). Taking On Engineering's Gender Gap. *NewsSmith* 13(1). Smith College, Northhampton, MA. (`http://www.smith.edu/newssmith/NSSpring99/cover.html`). See `http://www.science.smith.edu/departments/Engin/` for more information on Smith's engineering program.

Stein, L. A. (1998). What We've Swept Under the Rug: Radically Rethinking CS1. *Computer Science Education* 8(2): 118-129.

Stein, L. A. (Ed.) (to appear). *Proceedings of the 2001 AAAI Spring Symposium On Robotics in Education.*