Robots in the Undergraduate Curriculum

Deepak Kumar
Department of Math. & Computer Science
Bryn Mawr College
Bryn Mawr, PA
dkumar@brynmawr.edu

Lisa Meeden
Computer Science Program
Swarthmore College
Swarthmore, PA
meeden@cs.swarthmore.edu

1. Introduction

In this paper, we describe the augmentation of a traditional computer laboratory with materials and equipment required for conducting robot building exercises. We present our experiences in incorporating robot exercises in Artificial Intelligence (AI) courses. We also discuss the possibility of using robots across the computer science curriculum. At the undergraduate level, we feel that such a model is viable for small colleges. It may pose substantial administrative challenges for programs with large enrollments.

2. Motivation

The advent of computer technology, both in terms of hardware and software, has had a significant impact on the teaching of computer science, especially at the undergraduate level. Most schools are constantly struggling with the issue of the choice of programming languages (Pascal, Scheme, C++, Java), platforms (PC, MAC, UNIX), and operating systems that will be used for laboratory exercises in various computer science courses. As operating systems, and software environments become increasingly user-friendly and sophisticated, a laboratory environment is well insulated from the lower-level features of the underlying machine and its software components.

Typical undergraduate computer science curricula focus mostly on the development and analysis of algorithms. Thus, indirectly, the emphasis is on software development. This creates a void in that the students may never experience the underlying hardware of the computers they utilize. Students are required to take a core course on computer organization and perhaps an elective on computer architecture. However, even in these courses, there is minimal exposure to actual physical hardware components. Construction of physical agents intrinsically involves the handling of controller boards (microcomputers in themselves), interfacing of sensors to these boards, and, more importantly, dealing with the physical connection between a computer and the controller board (via a serial port). Developing software to control the behavior of physical robots also involves working under constrained resources (especially with respect to speed and memory) imposed by the robot control board. This provides a direct exposure to the time as well as the space complexity of algorithms. Yet another link that can be emphasized via the laboratory exercises is that between the algorithms embodied in the agents and their equivalence as a whole to an automaton.

3. The Robot Laboratory

The robot building laboratory is essentially a low cost extension of an existing computing laboratory. Each computer workstation is augmented with additional external hardware and materials and can serve teams of 2-3 students in a robot exercise. The following materials are needed for each workstation:

- A computer (PS/MAC/UNIX) with an available serial port.
- A robot controller board.
- Control software
- Robot building materials: LEGO Technic resource sets, sensors, motors, etc.

3.1 Robot Controller Board

The key hardware component, besides the computer, is the robot controller board. We are using a controller board called the Handy Board [Cite: Fred Martin/HB]. It is a third generation controller board, preceded earlier by the Miniboard [cite: Miniboard], and the 6.270 board [cite: 6.270].

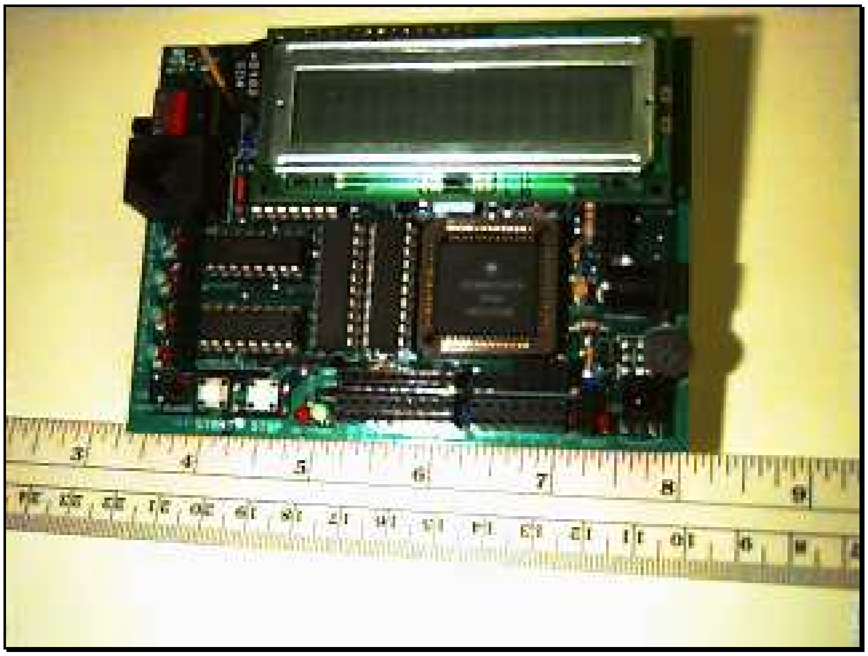

Figure 3.1 The Handy Board

The Handy Board is based on a Motorola 6811 microcontroller that has a built in 8-bit A/D converter. It is powered by rechargeable batteries. The complete unit is a single, solid-state package measuring approximately 4x3x1 inches and weighs under 1 lb. The board provides a serial interface for connecting to the computer, two programmable input switches, 7 analog input ports, and 8 digital output ports. Additionally, the board has an infra-red receiver capable of accepting signals from a standard television remote, a piezoelectric beeper, and a 2 line LCD display. The board also includes 4 ports for controlling analog output devices that are most commonly used for controlling DC motors. A variety of off-the-shelf sensors can be easily interfaced via the input ports. A completely assembled, ready-to-run Handy Board costs approximately $300, including all cables. It can also be purchased, unassembled, for less than half the price. Unless one is an expert electronics technician, we would highly recommend purchasing the completely assembled version. The latter, comes with really good assembly and testing instructions, and is more for hobbyists.

3.2 Control Software

Control programs for the Handy Board are written in Interactive C (IC). IC is a subset of the C programming language augmented with library routines for access to the interface components of the board (switches, input and output ports) as well as for multi-tasking. Interactive C software runs on several computer platforms (MAC/PC/UNIX). IC programs are written by users on the workstation and then downloaded for execution on the Handyboard via the serial interface. Currently, there are at least three vendors that sell Handy Boards and IC. The cost of pruchasing an IC license is approximately $35. There is also a freeware version available.

3.3 Robot Building Materials

Several materials can be used for building the bodies of robots. In our laboratory, we mostly use LEGO Technic materials. It is also possible to build mobile robot bodies by retrofitting toy cars. Several hobby stores also sell ready-made bodies for small mobile robots. Other sources for building materials are children's motorized building kits from Meccano, K'Nex, and Fischer Technik. In our experience, we have found LEGO materials to be the most effective in price and performance. Also, the power characteristics of the Handy Board are well matched with those of LEGO components. A comprehensive LEGO Technic building kit costs around $200. About 4-5 such kits will suffice for ten workstations. One would need additional LEGO motors which can be bought separately for approximately $20 each.

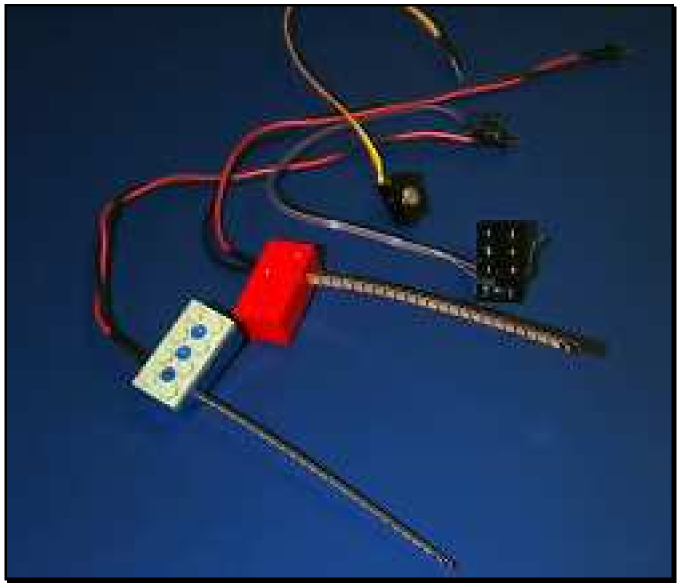

Figure 3.2 Sensors (Touch, bump, and photoresistors)

In addition to the materials above, several sensors will also be required for a typical robot building station. The ones we have used include ON/OFF switches, photoresistors, bend resistors, and a standard television remote (for infra red signals). These sensors are relatively cheap. A set of 5 photoresistors costs under $2 at Radio Shack, cheaper at electronic catalog stores. Typically, some rudimentary soldering is required for wiring an adaptor to the sensors for plugging into the Handy Board ports. Soldering equipment is typically available in a physics/electronics laboratory in most colleges.

We estimate that the total cost of augmenting a traditional computer laboratory with the described robot building equipment would be well under $1000 per workstation. Typically 2-3 students would work together on each workstation.

4. Incorporating the Robot Laboratory into the AI Course

There is a growing consensus among computer science faculty that it is quite difficult to teach the introductory course on Artificial Intelligence well \cite{hearst,AI-Ed95}. In part this is because AI lacks a unified methodology, overlaps with many other disciplines, and involves a wide range of skills from very applied to quite formal. In the funded project\footnote{This work was sponsored in part by grant XXX-XXXXXXX %DUE-9651472 from the National Science Foundation's ILI/IP program, and grants from XXXXXXXXX College and XXXXXXXXX College.} described here we have addressed these problems by

- Offering a unifying theme that draws together the disparate topics of AI;
- Focusing the course syllabus on the role AI plays in the core computer science curriculum; and
- Motivating the students to learn by using concrete, hands-on laboratory exercises.

Our approach is to conceive of topics in AI as robotics tasks. In the laboratory, students build their own robots and program them to accomplish the tasks. By constructing a physical entity in conjunction with the code to control it, students have a unique opportunity to directly tackle many central issues of computer science including the interaction between hardware and software, space complexity in terms of the memory limitations of the robot's controller, and time complexity in terms of the speed of the robot's action decisions. More importantly, the robot theme provides a strong incentive towards learning because students want to see their inventions succeed.

This robot-centered approach is an extension of the agent-centered approach adopted by Russell and Norvig in their recent text book~\cite{russell95}. Taking the agent perspective, the problem of AI is seen as describing and building agents that receive perceptions as input and then output appropriate actions based on them. As a result the study of AI centers around how best to implement this mapping from perceptions to actions. The robot perspective takes this approach one step further; rather than studying software agents in a simulated environment, we embed physical agents in the real world.

This adds a dimension of complexity as well as excitement to the AI course. The complexity has to do with additional demands of learning robot building techniques but can be overcome by the introduction of kits that are easy to assemble. Additionally, they are lightweight, inexpensive to maintain, programmable through the standard interfaces provided on most personal computers, and yet, offer sufficient extensibility to create and experiment with a wide range of agent behaviors. At the same time, using robots also leads the students to an important conclusion about scalability: the real world is very different from a simulated world, which has been a long standing criticism of many well-known AI techniques.

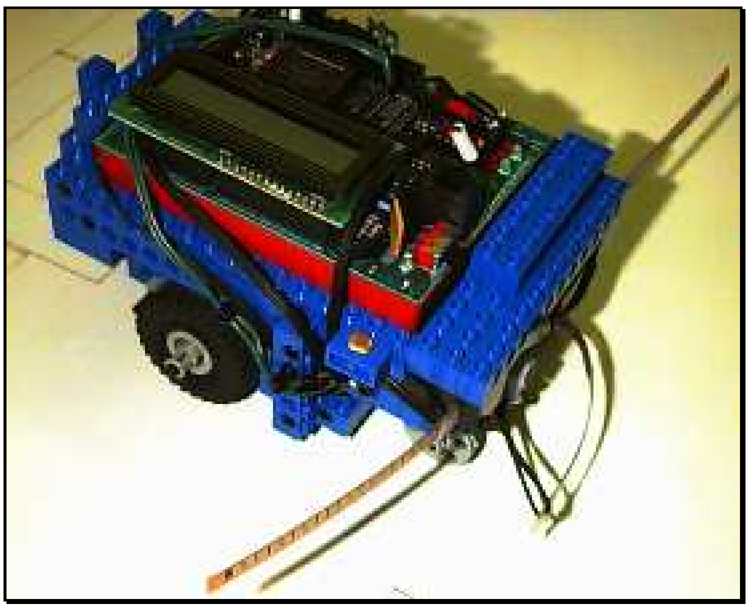

Figure 4.1 A small robot.

The course has weekly, 2-hour, laboratory sessions. Students form groups of 2-3 to work jointly on their robot exercises. Each group is issued a robot kit that can be carried out of the laboratory in a small activity box. Laboratory sessions in the first half of the semester are conducted as closed laboratories. The objective of these sessions is to introduce the robot building materials and to familiarize the students with standard techniques of interfacing sensors and writing simple robot control programs. Each laboratory session has structured exercises. Students start the exercises in the laboratory and spend the rest of the week completing it. Robots resulting from previous week's exercises are demonstrated at the beginning of the next week's laboratory session. A laboratory manual containing these exercises has been prepared and is available via the world-wide web~\cite{manual}.

The laboratory exercises address the problem of building an intelligent robot from scratch. There are two main steps involved: developing the hardware and then writing the software to control that hardware. Typically, the tasks only provide an abstract description of the desired mapping from perception to action. Much of the detailed specifications about how to behave in a particular situation is not given and may not be known. For example, a typical robotics task is to navigate from a starting point to a goal point. How will the robot recognize that it has arrived at the goal location? What obstacles can the robot expect to encounter along the way? If a map is available, how do positions on the map correspond to the robot's sensor readings? Due to the open ended nature of robotics problems, a lengthy process of trial and error is often necessary to answer these types of questions clearly enough to develop a working algorithm. Students must repeatedly run their programs on the robot and watch how well it performs. They must analyze the failures and determine how they emanate from the program. They must then modify the programs in order to correct the failures or improve the performance. During this debugging period, students learn a great deal about the interaction between the robot's sensors and its physical environment, and in turn must translate this knowledge back into their programs.

Another key area that is incorporated into the laboratory is the idea of working in teams. By working on the same project together students are able to accomplish more than what they could alone. Though not everyone in a team has the same level of expertise in programming, engineering, and structural design, team organization provides an opportunity to learn from each other.

5. Across the Curriculum

We have also begun to realize the potential of using robot-based exercises in other computer science courses. For example, programming of simple behaviors is essentially no different than the programming of simple problems as studied by students in introductory computer science courses. We have used the robot building laboratory as a basis for teaching introductory computer science \cite{meeden96}. A new course titled, {\em Building Intelligent Robots} was offered at XXXXXXXXX College to serve as a broad introduction to computer science as well as artificial intelligence. Since this course had no prerequisites, students came from a wide variety of disciplines including economics, engineering, literature, mathematics, psychology, and religion. This course stressed both application and theory by dividing class time equally between laboratory sessions and discussion sessions. Robots fascinate the typical undergraduate student, and we should use this interest to draw students into the computer science curriculum. Building a robot from start to finish is an intensely engrossing experience that motivates the students to learn about many of the less glamorous theoretical aspects of computer science.

The Handy Board can also serve as a laboratory platform for other upper-level computer science courses. It is conceivable that students in a class on constructing compilers could write interpreters/compilers for the Handy Board. As such, the Handy Board comes with a pcode interpreter. Students could design compilers for languages and generate pcode for running on the Handy Board. It is also feasible to implement a different virtual machine (JAVA VM, for example) for the Handy Board. Since the Handy Board is a small computer in itself, it can become an ideal tool for experimenting with implementations of small operating systems. Further, IC also supports multitasking. In our AI course, we have designed exercises that utilize multitasking to implement quasi-parallel programs. Students can also engage in designing and implementing drivers (which may require the use of assembly language) for interfaces to the Handy Board. Because, the Handy Board is so inexpensive, and physically disembodied from the laboratory's main computers, it can be used for exercises where students may inadvertently hang the computers in carrying out their exercises.

6. Summary