Teaching Mobile Robotics with a Modular Construction System

C. Ronald Kube
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
kube@cs.ualberta.ca

March 17, 1995

In Canada north of the 53rd parallel, where seasonal temperatures often range between -30 and +30 C, is a group of autonomous robots playing pinball for University credit. Although not your average pinball game, these robots were created by a group of Computing Science students at the University of Alberta as part of a unique course, under the direction of Dr. Hong Zhang, that is testing an experimental teaching methodology called Interactive Learning (IL). The undergraduate course in mobile robotics takes a non-traditional approach to both teaching and learning in which students design and build autonomous robots that play a derivative of the arcade game pinball called pinBot, devised by Rod Johnson and myself. In pinBot, robots score points by accomplishing tasks on a large 8' by 8' game table, with the winner having accumulated the most points in a ten minute round.

This article discusses our approach to teaching mobile robotics in the Department of Computing Science over the past four years, and how the evolution of our hardware, software and curriculum has affected the way in which we build our robots.

Captivating Imagination with Robotics

There is a fascinating curiosity intrinsic to robots. Something about these machines that captivates the imagination of both young and old alike. To see this first hand, just watch the reaction children have when they encounter a mobile robot in a shopping mall or the attention generated by a robot handing out business cards at a trade show.

Previously, in the Department of Computing Science, the first exposure to robotics by a student was a graduate level "Introduction to Robotics" course heavy on theory with little hands-on application. We wanted to introduce undergraduates to the multidisciplinary field of robotics in a manner that was both fun and educational. What better way, we thought, than to create a course in which students design and construct a small autonomous robot capable of displaying some purposeful behavior. To be successful, students would have to combine and integrate their knowledge from other courses to become a mechanical/electrical/software guru all rolled into one; a roboticist.

We were also interested in delivering this course in a non-traditional way. Traditionally, courses taught in Computing Science involve, for the most part, a one-way communication from professor to student. Students typically attend three hours of lectures, taking notes that will hopefully be useful later on in laboratory assignments and exams. Cooperation and collaboration on assignments is not normally encouraged as students all strive to fall on the right side of the dreaded bell curve. But learning is really a two-way street, so we wanted to encourage information exchange between students and instructors, between the students themselves, between past and present students by way of the previous year's reports, and between the students and anyone else that could help them learn.

We call our approach Interactive Learning and it is based on a few simple ideas: No lectures, no exams or traditional teaching methods in the course. Instead the course is project based and centers around the creation of a small autonomous robot capable of predefined tasks. In place of lectures we hold meetings in which we brainstorm on ideas from both students and instructors, no matter how far fetched the ideas appear. Laboratory work immediately follows in which students implement and test their ideas. Students are encouraged to share solutions, design tips, code and ideas in building their robots, with instructors active in helping students bring their ideas to life. The cohesive factor in the approach is the students' robot. Students are involved with their robots from day one with course milestones chosen which incrementally add knowledge about sensor and actuator technology. As students acquire skills so do their robots until both have reached some level of competence. The philosophy behind the approach is best captured in the "constructionist" paradigm (Papert 1980) [1] in which new information is reinforced

by building artifacts.

Enthusiasm climaxes towards the end of semester which features a final event highlighting the robots and their inventors to local media. In the first year the course was offered, this took place as a robot talent show in which the robots demonstrated some purposeful behavior such as "serving cookies" accomplished by a unique combination of pyroelectric and sonar sensing with the assumption that the target detected by the human seeking pyroelectric sensor was the same one detected by the sonar. This allowed the robot to deliver its cookie tray while stopping by the person and ensured the same person wasn't served twice.

Our second year targeted a more task oriented objective versus the "build a robot that does anything" approach of the first year. The end result was a game of Cops 'N Robbers in which the autonomous "cop" robots had to seek and capture our human controlled "robber" robot. Finding the robber was simplified by installing a bright light on the robber which the cops could track using photocells.

A desire to increase the variety of tasks which the robots could perform has led to our current concept of robot pinball we call pinBot. In pinball the motion of a steel marble is directed by the player towards targets that accumulate points. In pinBot we replace both the player and steel marble with an autonomous mobile robot which earns points by identifying objects and accomplishing tasks on the gameboard. We will spend the remainder of this article describing pinBot in detail and how our curriculum is organized around building a robot which plays this game.

Captivating the imagination of students in this course through robotics has resulted in their requesting a follow on course, in which they design and build an autonomous underwater vehicle (AUV). These robots, unconfined to the two dimensional flat world of their wheeled counterparts, must navigate in three dimensions within the confines of our swimming pool. In fact, enthusiasm for this second course—in which students built a robot called Delphis—has resulted in the students continuing to work on their underwater robot well after graduation.

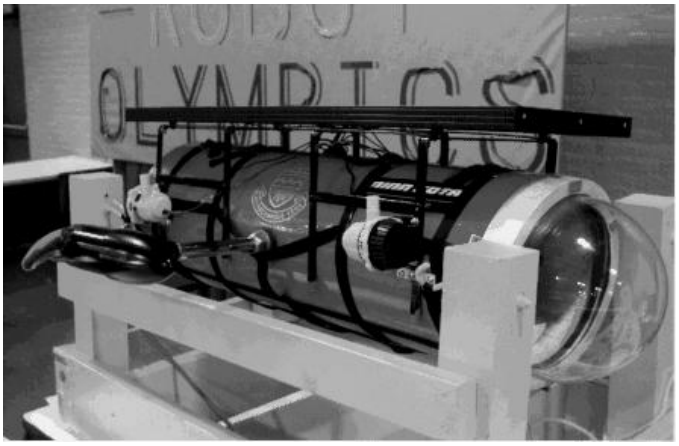

Figure 1: Delphis the Autonomous Underwater Vehicle designed by Adrian Smith, Kelly Kearney, Andrew Morris and Steve Hryniw went to the BEAM Robot Olympics in Toronto.

Figure 2: Delphis without its solar panel during a pool test.

Teaching Mobile Robotics

The course objective is to allow students the opportunity to learn about a subset of the knowledge in robotics by implementing a real physical system, a mobile robot. The current project is to build a robot that plays pinBot, a game involving the identification and completion of a number of tasks on an 8' by 8' playing area. Each robot behaves reactively to sensor stimulus like the vehicles of Braitenburg (Braitenburg 1984) [2]. The 13 week semester begins with a series of introductory seminars designed to bootstrap the construction phase of the course, which is

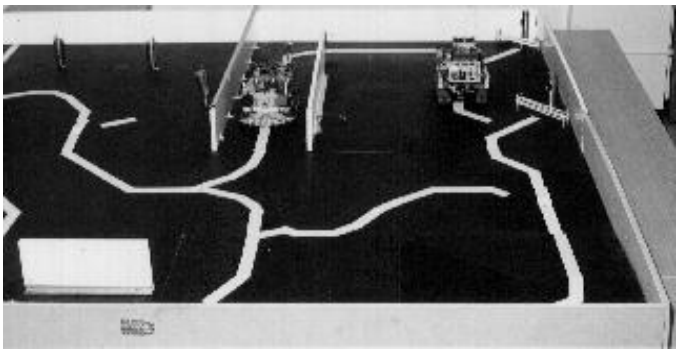

Figure 3: Two robots playing pinBot. In the bottom right corner is a wedge which the robots climb and score points by playing a unique tone on their speaker, thereby identifying the incline, top and decline of the wedge.

centered around five milestones and ends with the final pinBot game, a 10 minute winner-takes-all "run till your batteries choke" event.

The game board is a black playing surface with a white line that weaves through six playing areas (see figure 4). By following the line the robot enters each playing area and accumulates points by accomplishing the playing area's task. In figure 4 the large black dots labelled S1 to S5 represent track switches which change the order in which the playing areas are encountered. Beginning at the top right of figure 4 the robot follows the line into the first area: the Room of Doom, where the track disappears. The task is to locate the exit while avoiding obstacles placed in the room; a successful robot is awarded 200 points. The next task is to identify the area marked "Light Channel" which is equipped with a break-beam sensor that turns on an opposing pair of lights, which the robot must indicate it has seen by sounding a unique tone on its speaker. Next the robot must identify the incline, top and decline of a wedge located in the bottom left of the board. Once again, unique tones are used to signify recognition. The Pinball Corridor consists of two bumper/light obstacles located at deadends. The robot must contact the bumper, activating its light, and signal its recognition with a tone before backing up and continuing to the next bumper/light obstacle. The track then guides the robot through the Flashlight Corridor in which an arbitrary number of lights are illuminated which the robot must count and then report the number seen by clicking its speaker the appropriate number of times. A correct count is awarded 150 points. The last playing area requires the robot to leave the track and find illuminated bumpers which are turned off when hit. Once all three bumpers have been turned off the round is complete and the robot must once again find the track.

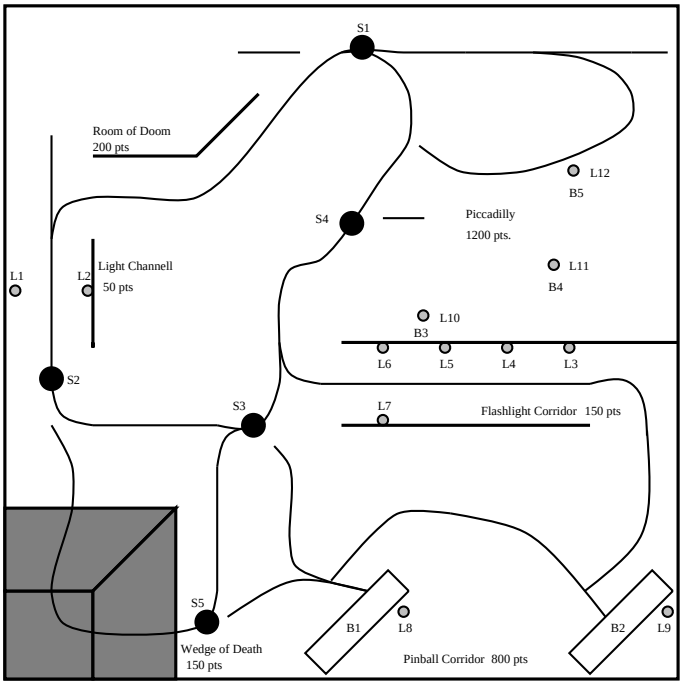

Figure 4: The layout of the pinBot game board. Robots navigate by tracking the line and score points by identifying objects and performing actions.

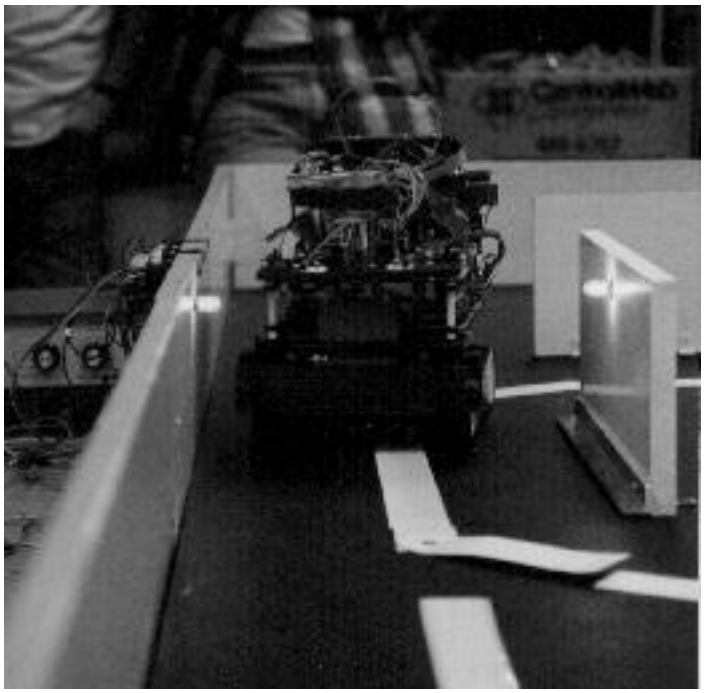

Figure 5: Burning Rubber navigating the light channel and scores points by recognizing the pair of lights. When it identifies the lights it plays a unique tune.

Bootstrapping the Learning Process

In general, Computing Science students have little background in soldering and building circuits. Although the students that are typically interested in this course have some hobby experience with electronics, we start everyone off by building a simple one sensor one actuator robot kit called the Muramator from Johuco Ltd. (Connell 1991) [3]. In an afternoon students learn how to read schematics, identify parts and solder. If all goes well they are rewarded with a little robot capable of avoiding obstacles and following walls using its one infrared sensor.

Our next objective is to prepare the students for the planning stages of their robot designs. A team of two students will spend approximately 200 - 300 man hours designing and building their robot. To avoid having to preplan the entire project, we require students to produce proposals for each of the five milestones. The milestones are:

1. Programable Motion.
2. Velocity Control and Line Following.
3. Obstacle Avoidance.
4. Recognize Lights and Inclines.
5. Counting and Searching for Lights.

The milestones are geared towards accomplishing tasks in each playing area of the board. The proposals for each milestone must include sections on Function, Electrical Schematics, Mechanical Drawings, Parts Lists and Wire Lists for each module in the design. These modules must refer to the robot's overall system block diagram. The proposal is essentially the first draft of the deliverable the team is expected to hand in on the due date of the milestone. These documents will then form the basis of the team's final report documenting their robot and experiences. For example, the first milestone requires each robot to demonstrate programable motion; something as simple as forward, backward and turning. A block diagram might look like figure 6. The only restriction is a robot must use differential steering with left and right wheel motors. Teams choose, from among sample circuits, how to implement each module. For example, the DC Power module shown in figure 6 could be a linear or switching power supply; the Motor Driver module might be a discrete or an IC H-bridge. These documents are used to transfer knowledge from one year to the next, as future students have access to the documentation from previous years.

Charting the team's progress through the semester is accomplished by awarding "stars" for completed tasks on each milestone. Ninety percent of these tasks are not time dependent allowing teams to play catchup if unforeseen circumstances halt their progress. In this manner, more

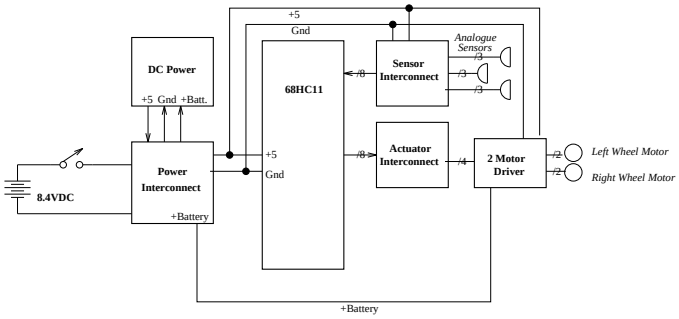

Figure 6: A system block diagram for the first programable motion milestone.

emphasis is placed on completing the tasks, allowing students more time to evaluate and test the more riskier implementation strategies.

Once a week each team must present a brief five minute status report on the current state of their robot, highlighting the problems encountered and their solutions. Unsolved problems are immediately the subject of brainstorming sessions which involve both students and instructors, with implementation left to the teams to complete. Verbal status reports are summarized in one page written reports which are added to the team's project file.

Laboratory Equipment

The course enrollment has been kept to under 10 students per semester while we develop the curriculum. As a result, lab space requirements were minimal with students having access to a variety of Unix workstations and PCs on which to write their code. The lab is equipped with an oscilloscope, multimeters and power supplies which we teach students to use during the bootstrapping weeks of lab introduction. Common tools are shared, as is access to parts and building material. Construction techniques, explained in the next section, are modular to ensure reuse of all material upon course completion.

Building Modular Robots

Ours is a quest, a kind of technological evolution towards a modular hardware and software system that facilitates the rapid prototyping of our student's ideas about building robots. When we first began developing the curriculum and building robots in 1991 our hardware resembled undocumented spaghetti code in an obscure language that nobody understood. Wires were everywhere, huge gobs of solder that made better mechanical than electrical connections were rampant. Robots were considered successful if they moved a few inches before crashing their embedded microcontrollers. Although not pretty, it was fun.

This of course was to be expected; after all, we are

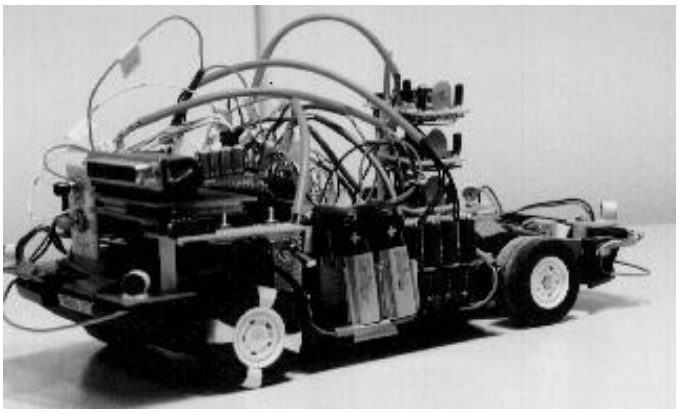

Figure 7: Behavior Bot created by Douglas Doole and Dani Beaubien in 1992 chased loud noises tracked with two microphones.

Computing Scientists and although our software might be modular our hardware certainly was not. Student projects typically took 200 - 300 man hours to complete with 60 percent devoted to hardware and the balance on software and documentation. A great deal of the hardware construction time was spent on mechanically building the robot and mounting its sensors and control electronics (see figure 7 of Behavior 'Bot). Because the approach to construction was adhoc, little of the robot's circuits or body was recoverable at course completion; this was expensive.

Student robot projects cost roughly $Cdn 500 with $150 for the microcontroller; $50-75 for the mobility base; another $50 for batteries; $100 for sensors and electronics; $100 for connectors and mechanical construction material. Initially we were only able to recover the microcontroller and battery along with some of the sensors; the mechanical construction material, which consisted of acrylic and metal was lost due to all the mounting holes that were drilled. However, after building over 50 robots in the past four years, we have started to refine our moving gobs of solder and wire.

A Minimalistic Approach to Hardware

Our effort to create modular electronics has taken two approaches. In the first, we have reduced our functional blocks to printed circuit boards (PCBs) with connectors for input and output. Our second approach is to move some of the hardware functionality to software. For example, obstacle avoidance was initially implemented in hardware using an infrared (IR) emitter/detector pair and a thresholding circuit to give a binary output; this required a thresholding circuit using a LM339 comparator. In the second year the course was offered one of our student teams (Edward Mantey and Steve Hryniw) decided to try feeding the output of the IR detector directly into

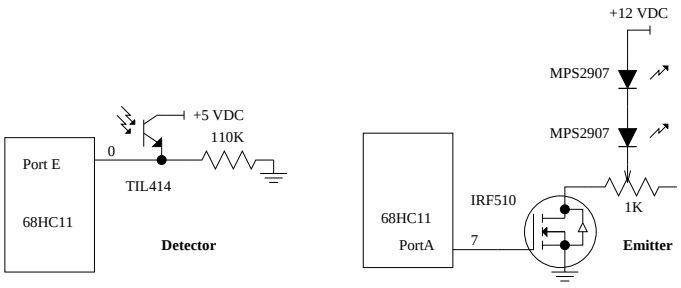

Figure 8: Obstacle avoidance using an IR pair. An ambient reading is first obtained by reading the IR detectors with the emitters turned off; then the reflected reading is taken with the emitters turned on.

the A/D port of the 68HC11 microcontroller and perform the thresholding in software. Not only did this approach use less hardware, but it had the added advantage of delivering a continuous distance-to-obstacle value based on the reflected intensity of the IR. This soon became the defacto standard among our students for using IR (see figure 8).

Another rather novel approach to obstacle avoidance is to use two photocells directed forward approximately 30 degrees off center and a "move towards the brightest light" algorithm. This was discovered purely by accident by two students (Kelly Kearney and Allan Sampson) while testing their light-seeking behavior. It seems for an indoor environment that light falling on an object from above casts minute shadows and objects appear darker than open space. It was amusing to watch the robot avoid a room full of table legs especially since no explicit obstacle avoidance system was present. Thus we favor solutions in software versus hardware whenever possible.

To solve the never ending hole drilling, needed to mount sensors and electronics, during the mechanical construction phase, students use a perforated PVC sheet cut to size and stacked using hex spacers. This material obtained from local plastics distributors is 1/8" thick with evenly spaced holes throughout. Originally used to make "basket strainers" for the chemical process industry, the material is ideal for mounting both sensors and PCBs using a standoff/grommet combination (see figure 9 of Labotomy). Our intention for the next offering of the course is to start each team with a basic kit of parts which they will augment with task specific sensor systems.

Sensing the Key to Interesting Behavior

Students choose from a wide range of sensing technology including: infrared sensors for obstacle detection; micro switches for tactile sensing; photocells for measuring light intensity; pyroelectric sensors for warm body detection; sonar modules for distance measurements; microphones

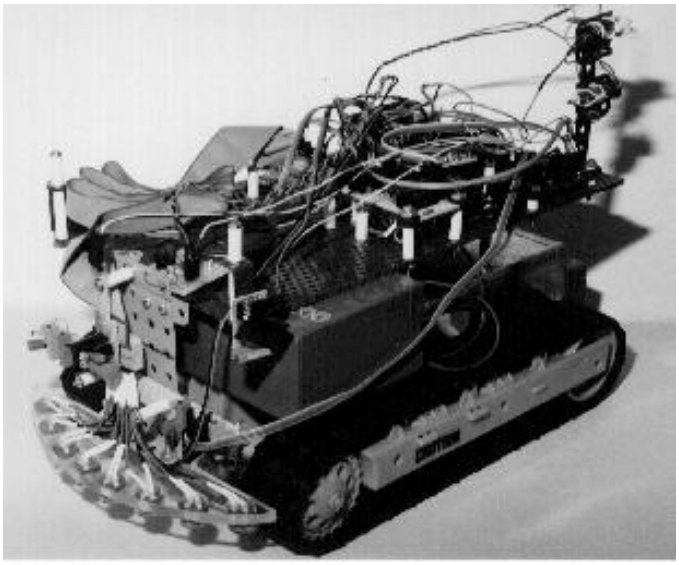

Figure 9: Labotomy, a pinBot designed by Andrew Hess and Jerry Jeremiah, makes use of a curved photocell array to detect and follow a white line.

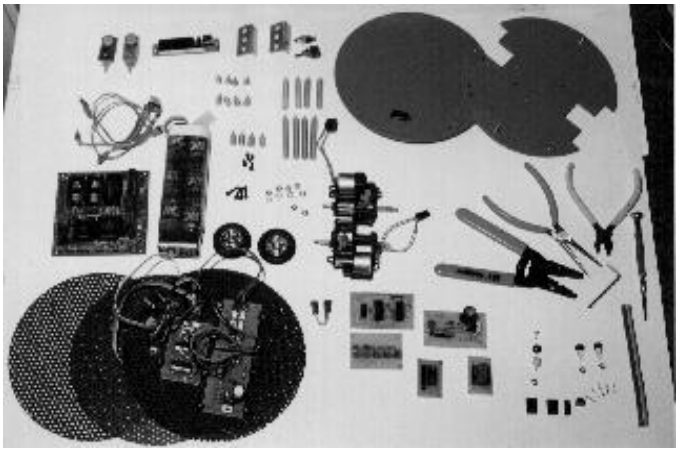

Figure 10: A modular robot construction kit includes aluminum discs for base construction and a supply of perforated plastic discs for mounting sensors and electronics.

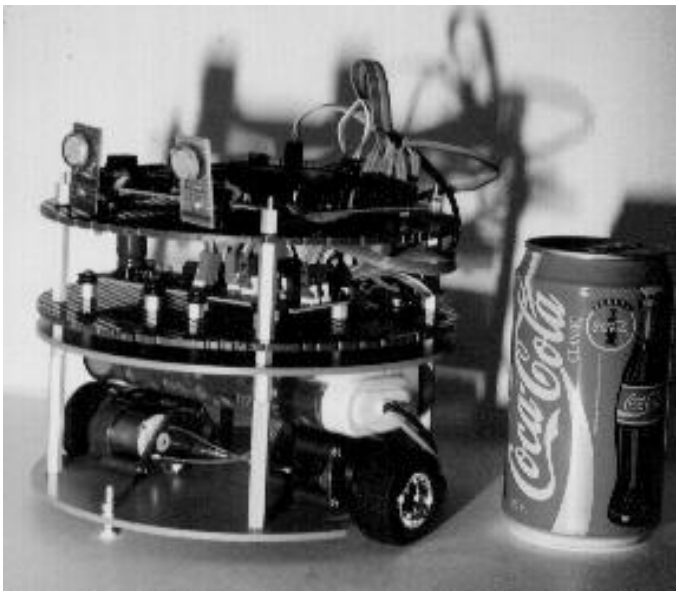

Figure 11: A robot base made from the modular construction kit.

for noise detection and mercury switches for tilt sensing. The task then is to combine these sensing elements into a task specific sensor system. For example, the photo of Labotomy's line sensor in figure 12 shows one approach to following lines based on an arc of cadmium sulphide photocells (Radio Shack 276-116), whose resistance vary as a function incident light intensity.

In order to recognize an inclined surface, teams had to design a slope detector. The task is to identify when the robot is on the incline, has reached the top flat surface, and moves onto the decline. Labotomy, designed by Andrew Hess and Jerry Jeremiah, used two mercury switches, commonly found in heating thermostat systems, shown in figure 13. Another solution, designed by Jerry Yee and John Bartoszewski and shown in the photo of their robot Burning Rubber (figure 14), makes use of a curved plastic tube with a steel ball bearing inside and two contact switches at either end of the tube. When the robot is on an incline the back contacts are closed by the ball bearing. To debounce this type of switch and eliminate false readings, while the robot navigates the board, the software polls the switch and requires a minimum number of closed readings.

In the pinBot game each playing area has a mutually exclusive sensing requirement so that behaviors can be designed for the unique perceptual task; this avoids incorrectly identifying different playing areas since the sensor readings are unique. Given a handful of sensors, wheel motors and a place to put them on, how do you stick it all together?

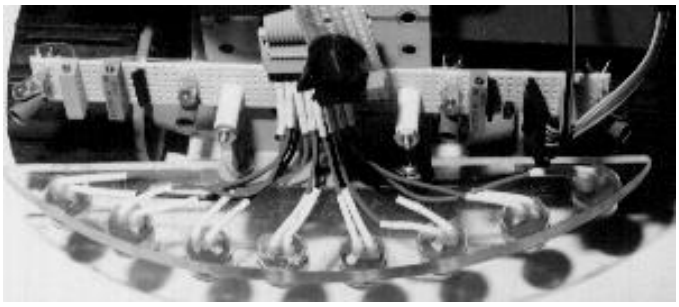

Figure 12: Labotomy's line sensor made from cadmium sulphide photocells whose resistance varys as a function of incident light intensity.

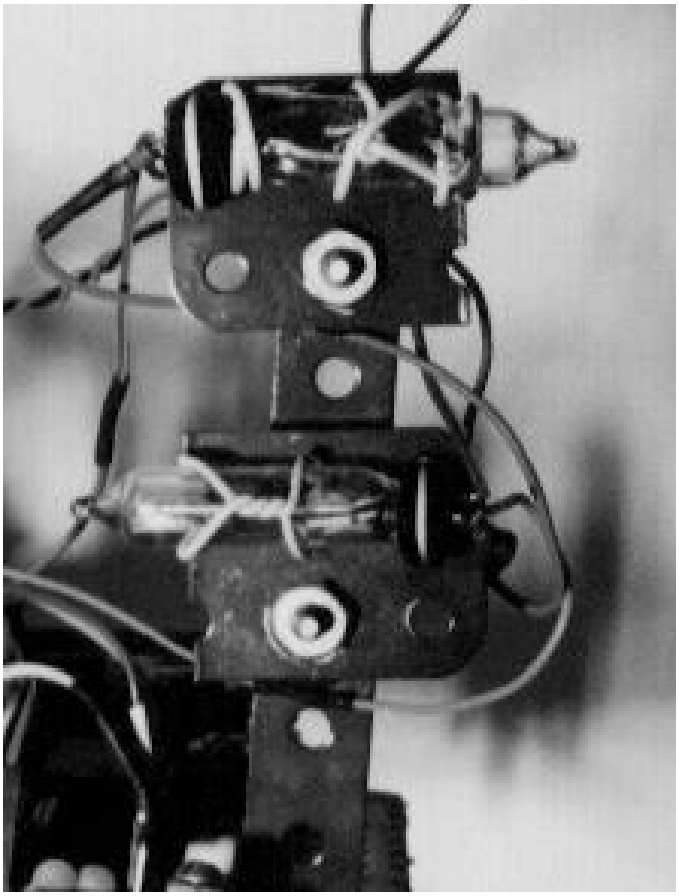

Figure 13: Labotomy's incline sensors made from two mercury switches commonly used in a heating thermostat.

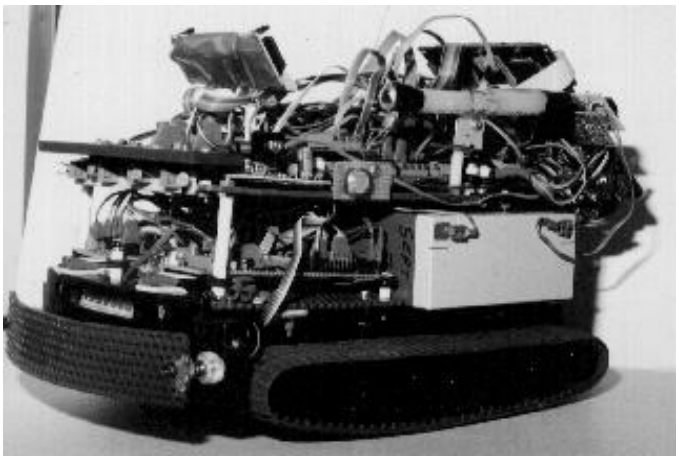

Figure 14: Burning Rubber, designed by John Bartoszewski and Jerry Yee, uses a curved plastic tube with a steel ball bearing inside (shown on the right above the battery) to sense incline surfaces.

Forth: The Glue that Binds Hardware and Software

When faced with having to choose a microcontroller development environment two possibilities came to mind. We could develop our robot code with a cross compiler and download it to the robot's controller. This would mean having to wait and test any sensor or interrupt routines after the code was running on the robot. This approach would also require a common hardware platform for the cross compiler; an oxymoron in a university environment consisting of Unix workstations from a number of vendors, both DOS and Mac based PCs, not to mention the plethora of machines students have at home. We wanted the students to have the flexibility of pulling their robot from their knapsack and programming it with any type of computer they had available to them.

Our solution is a 68HC11 microcontroller from New Micros Inc. in Dallas, Texas [4], complete with an interactive language/compiler/OS based on the Forth programming language invented by Charles Moore to control radio telescopes. To program the controller, all that is needed is some terminal emulation software and a serial cable. Programs, which map sensor input to motor output in a behavior based fashion, are created by extending the set of subroutines, resident in the New Micros 68HC11's ROM, into an application specific language. Because of its interactive nature, the best way to learn Forth was to let the students jackin to their robots and play. The first milestone was motion control and each wheel motor was controlled by the 68HC11 using two bits: direction and power enable. Forth is considered an extendable language since the base set of procedures (called words in Forth lingo) that comprise its dictionary and define the tokens

in its language, can be expanded by creating new words composed of existing words from the dictionary and data. An example might help.

Imagine you just plugged your PC into the robot and reset its microcontroller, a little >ok prompt tells you everything is working and you may now talk to the robot's hardware. Your robot has a left and right wheel motor connected to the controller's port A as in figure 15. You want to turn the left motor on by enabling bit 5 and setting its direction to forward with bit 6 equal to 1, and the same for the right wheel motor, so you type: 50 PORTA C! (C! writes a byte just like a POKE in Basic). The motors turn on, and you decide to call this action FORWARD and compile a new word in the dictionary by typing, : FORWARD 50 PORTA C! ; (The ":" defines a new word and the ";" ends the definition.) Before the robot runs right out of the room you create a few more new words, like: STOP, REVERSE, LEFT, and RIGHT. Now you can move your robot around using the newly created commands and decide to create a small demo so you can unplug the tether to your PC; so you type, : DEMO FORWARD WAIT STOP LEFT WAIT STOP FORWARD WAIT STOP ;

One of the nice features of the Forth language you discover is its ability to serve as both a low and high level language; great for the low level sensor hacking as well as the high level behavior programming. If any of your routines need more speed you can directly embed assembler into newly created Forth words. How long does a new word take to run? Just add up the time of each primitive dictionary word used in your newly created word and add a small constant. This can also be done for assembly language programs since the number of cycles used for each instruction is listed in the microcontroller's technical data, but is difficult if the program is written in other languages like C or Basic.

Like any new language, Forth with its stack based parameter passing takes a little getting use to, but the payoff is in the quick code/test cycle and the small space its programs take. In fact, students have developed their code including a multitasking kernel to easily fit within the 8K RAM supplied with the board. When is comes to gluing hardware and software together, once bitten by the interactive Forth bug it's hard to turn back.

Conclusion

I think the ultimate success in any course can be judged by the enthusiasm it generates in the students for the subject, and by what the student takes away after the experience is over. Building robots is a hands-on learning experience that is educational for both students and instructors alike (not to mention a whole lot of fun!). Students become so engrossed with their robot that many have continued

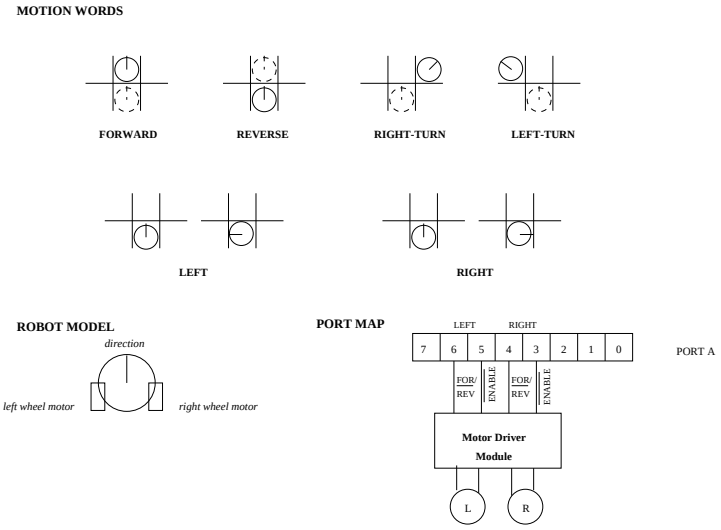

Figure 15: A simple motion scheme for two wheels connected to the controller's port A.

their development long after course completion.

For most of us, interaction with a computer means typing on a keyboard and staring at the screen for a response. But to the students in the Department of Computing Science, here at the University of Alberta, interaction with computers also includes creating autonomous robots capable of exploring the world for themselves. As the evolution of computing moves from the internalized world of crunching numbers on a spreadsheet, to allowing computers to see their world through sensors and to explore it through mobility and manipulation, we stand witness to an enabling technology that might affect the 90's with a greater impact than the microprocessor of the 80's. With embedded processors in our cars, cameras, appliances and even toasters, the enabling technology will soon become ubiquitous without us giving it a second thought. By introducing students to this technology first hand, in a manner they enjoy and remember, we are helping usher in the age of intelligent machines. [1]

References

[1] Papert (1980). *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York.

[2] Braitenburg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, MA.

[3] Connell, J. (1991). Design your own robot. In *Popular Electronics*, August, 25-34.

[4] New Micros Inc., 1601 Chalk Hill Rd., Dallas, Texas 75212.

[1] Want more information? Come visit our WWW home page at http://ugweb.cs.ualberta.ca/~c498/ we'd love to hear from you.