Chapter 5

Design Styles

This chapter discusses the design activity engaged in by the students of the Robot Design project. These students have a diverse set of backgrounds; each year, there is a distribution of students from the four undergraduate years as well as a few graduate students, with a corresponding difference in academic experience at each level. For some, the Robot Design project is their first experience in creating a complete technological artifact with their own hands and minds, while others can call on past experience gained through previous academic work, industry work, or personal pursuits.

We operated the project based on an "affirmative action" policy that was meant to encourage an alternative, bottom-up style of design work. Our motive was to allow students to learn about unfamiliar ideas, materials, and methods by a playful, exploratory process rather than the traditional methods of teaching through lectures, texts, and problem sets. Some aspects of our method was discussed in Chapter 3, which described how we created the technological materials to support exploratory work. This included contest designs that encouraged creative thinking and a variety of solution strategies, and robot-building hardware that was highly interactive to encourage experimentation as way of learning.

This chapter presents the results of this agenda. In the first section, I discuss students' actual working styles as they created their robots. Most of the students used an exploratory process to learn how to work with the LEGO materials, but many found this method unfamiliar and uncomfortable. Most students reverted to the traditional top-down style when they reached the programming phase of their project work; some students referred to

this method as being the "computer science mind." In general, even though the top-down style did not produce effective results, students had trouble abandoning it.

The remaining sections of the chapter focus on the adaptability of the learning environment to students' own interests and desires. The second section discusses a variety of design excursions taken by students, reflecting the openness of the technology we provided to them as well as our own encouragement for them to make such explorations. The third section discusses ways that students adapted the ostensible goal of the project—that of building a successful robot—to suit their personal interests.

5.1 Affirmative Action for Bottom-Up Design

As creators of the course and its materials, we were participants in a bottom-up design process ourselves. We therefore had a deep sense that this sort of design was both effective and valuable. First-hand, we had learned about the properties and capabilities of the robotic systems as we created them, and as users of our own technology, we saw natural ways to extend it—making it more powerful, versatile, and easy to use, and more suitable as a substrate for the design course itself.

Rather than having students design their robots by "thinking hard," we wanted them to (for example) make a series of prototypes as part of the process. To learn about sensors, we wanted them to experiment with the devices we gave them. To learn about robot control, we wanted them to write programs for their robot and see the results. As was discussed in Chapter 3, we designed our robot-building kit to encourage such a playful exploration of ideas.

Thus, we wished to not only *allow* bottom-up design, but to *encourage* it. I would liken the effort to a sort of affirmative action for intellectual styles: top-down design is so heavily favored in academic circles that one must consider anything other than that style to be a minority deserving of special consideration and opportunity.

5.1.1 Experimenting with the LEGO Technic System

When learning to use the LEGO *Technic* system, students typically used a bottom-up style of exploring the materials and prototyping their ideas, as described by this student:

> In the "strategy" stage, I have played around with the LEGO (I had never seen Technic before) in the hope of getting a "feel" for LEGO, which I believe I will need before insight into a good structure for the robot is possible.

The most defining characteristic of this phase of the project—the LEGO design—was the aspect of re-design: a repetitive, iterative, trial-and-error process involving conceptualization, implementation, testing, analysis and iteration. Often this process happened so rapidly that the steps flowed together, making it difficult to separate them from one another. Rather than attempting to analyze all of the possibilities and implications of a design idea, students would typically build something to see if the idea was feasible, and then either improve the artifact (through another design iteration) or discard it. As one student explained the development of his robot's chassis:

> My teammates helped a lot by pointing out weaknesses in my design and by forcing me to rebuild, but I had to do the building myself. It was discouraging to dismantle the old robots, but trial and error is certainly necessary. Here's a quotable quote: *It's astounding how much going backward is entailed in going forward.* I just hope I'm through going backwards with the chassis! (Emphasis added.)

The LEGO materials are ideally suited for this style of work. Since they are disassembled as easily as they are put together, students know that they aren't committed to a particular mechanism or concept simply because they have build it. To make sure that this aspect of the LEGO system was preserved, we didn't allow students to cut or glue the LEGO parts for particular applications; they had to use the LEGO materials as they were designed. (There were a few minor exceptions to this rule, but the exceptions were designed to encourage the creation of modular, reusable parts, like easily mounted sensor assemblies.) This stands in contrast to "raw" building materials, like those used in MIT's *Introduction to Design* class.

We encouraged students to try things out as early as possible as a way to learn about the capabilities of the materials they had at their disposal, and as a way to try out their

own ideas. This was perhaps the dominant way of learning and way of designing that the students used. In the following quote, a student discusses how both his vehicle and his strategy changed as a result of explorations with the LEGO materials:

> Since the first report the vehicle has changed extensively in both structure and purpose. These changes are both the result of hands on LEGO bricks interaction and conceptual changes in strategy itself. As we proceeded with the construction of the robot we realized the difficulties [inherent] in transferring ideas into working LEGO kit contraptions.

Many students developed their mechanical ideas in this fashion. A common experience was that of discovering that a favorite idea was easier to conceive than to implement. The following two quotations reflect a typical process by which ideas were discarded:

> Playing and tinkering have had an interesting interplay. I have been the team member who has come up with complex schemes for our robot and has consistently failed to get very far in prototyping such systems with LEGO even after hours of work.

> We spent a lot of time mainly revising our strategy and playing with LEGOs. Our initial ideas of arms and shooters that span the entire board were not "LEGOizable."

It is typical for students to learn about their own capabilities and the properties of the materials by developing a series of prototypes. Here a student explains his team's process with regard to developing their robot's chassis, which is characterized by an iterative learning process:

> Towards the beginning of the week, we built a small vehicle out of LEGO to get a feel of how gear trains work and how LEGO fits together in a strong configuration. We paid too little attention to perfect vertical spacing, and ended up having diagonal braces holding together pieces awkwardly. We used too much LEGO for just a frame, had no place to put the battery or board, and had an inefficient gearbox. We dismantled this machine and built a second one that was a little more efficient in LEGO use, and turned a little better as well. Finally, we took this apart and thought out another one. When we built this one, we were careful to reinforce all structures with vertical LEGO beams, use perfect spacing, and make it the size we wanted. We also discovered that it is extremely important to use many structural vertical beams in the gearboxes so that the axles don't bend and create friction.

Here is another student's comments which describe a similar learning process:

> So far things have been built rather than discussed and virtues debated. We will agree that we want a shooter and so someone will start it. Then someone else will pick it up, play with it, make a few enhancements, and pass it on to the next experimenter. I built the first release mechanism but it was not very sturdy. John came in and quickly made it into a working system.
>
> The original pull-back mechanism came from an idea that I saw one of the alumni working on, but was largely changed. This is how we have been working. No one was really totally responsible for the shooting system since it came from many seed ideas that were worked out after much experimentation by other members of the team.

While these students felt comfortable with the iterative design process he and his teammates used, others who worked in essentially the same manner were less pleased with the way the time was spent:

> We basically decided to build components since we really did not know how to decide between strategies. We knew we needed a drive mechanism, a steering system, some way to score, and a unit for pushing the ball release button. Of course, different strategies would require changing these systems but once built it is easy to modify. *The result is probably not as efficient as a system properly designed in the first place.* (Emphasis added.)

This student's comment points directly at the tension that many students experienced. They adopted bottom-up design strategies since the ones they were used to didn't seem to apply. Even though the method they used was effective, they were left with a nagging sense that they weren't doing design the right way.

Some resolved this inner conflict by shrugging off the exploratory work as insignificant play, a prelude to the "serious" work to follow:

> Saturday, 11th January: Our team played with LEGOs all day. I designed free rotating wheel, the chassis and [an] automatic transmission box. Tom was working on a catapult mechanism and Aram was making some strange high friction apparati. This was not meant to be serious work, just trying to get some feeling and experience working with LEGOs, and see how much material is actually available, and how complex we can get.

What is most interesting about these comments is that while the students are indeed using bottom-up design practices, they themselves do not validate this process as being

instrumental to their own learning or to the successful completion of their design. Most often, students almost apologize for their excursions away from the task at hand.

Still, the Robot Design class did not *impose* a particular design methodology. Students were free to organize their own work as they like. As the following passage from a student's journal illustrates, there were differences of opinion about what methods are best, even within a design team:

> Tuesday, 7 January 1992: We attended lecture, received our kits, and went home to start building the 'bot. It was clear from the beginning that there was a distinctive clash in approach between myself and my teammates. I favored a top-down approach, immediately wanting to devise a strategy. My teammates Dave and Tom, who favored a bottom-up approach, were more interested in building pieces of the robot, nicknamed "Robbie" by Dave.
>
> . . . I also came up with two rudimentary ideas for robot strategies, believing that we couldn't begin the structural design of the robot until a strategy had been decided . . .
>
> Thursday, 9 January 1992: After the lab closed, we went to my room and played with LEGOs. I immediately wanted to set a strategy in order to create a proper design, but Dave and Tom just wanted to play with the pieces.
>
> Saturday, 11 January 1992: Dave had tried to design an automatic transmission, and on this day he tried to design a "chassis" to connect to it . . . Tom worked on a ball projectile device. I also continued clamoring that we should decide on a strategy.
>
> Sunday, 12 January 1992: Today, we finally agreed to discuss basic strategy. Things probably worked out for the best this way because we have learned much about using LEGOs in the past few days. We decided that it would be best to implement a rather simple "get the ball, put it in the goal, and go to the dispenser where the opponent isn't" strategy.

This student concluded that the "play first, decide later" strategy was effective, but not all students drew this conclusion from the same experience.

Another student explains how he had to succumb to his "juvenile instincts" in order to suppress his "computer science mind":

> My teammate and I were clueless when it came to LEGO building; we spent several afternoons and nights just staring and letting our juvenile instincts take over. . . we tried creating gear trains and looking for ways to attach LEGO bricks securely. If nothing else, one important quality we will gain is the ability to think mechanically instead of tackling problems with a computer-science oriented mind.

To rationalize the process he used to learn to build with LEGO parts, this student calls it an "ability to think mechanically"; he does not want to confuse that kind of thinking with the orderly, divide-and-conquer strategies he is used to—the type of thinking he associates with computer science.

Other students were more accepting of the bottom-up approach they used, though it was still unfamiliar to them:

> Lots of time was spent on playing with LEGO building blocks. I found out that LEGO's building blocks were more than a toy. I built gear blocks with serious gear reduction and found lots of interesting LEGO techniques. As I spent more time on playing with LEGO's, interesting ideas popped into my head and I was amazed to find that I could put those ideas into reality.

While this student expressed surprise in her hands-on design process, others took it stride. In the following excerpt, a student describes how a key design feature was happened upon by accident:

> I left off in the last report where Willy and I had decided to mount the servo motor on the front wheel of our tricycle to provide a steering mechanism. Using a chain to connect our servo on the main chassis to the front wheel, I managed to get it working while Willy was building gear reductions for the two rear wheels. We weren't crazy about the way it worked but it was a success, at least until the [wire to the servo motor] broke. I didn't feel like resoldering it that late at night, so I just set the front wheel straight forward and helped Willy finish the drive part of our chassis.
>
> We tried out the two motors and we liked the way they worked. It appears that we have decent torque as well as considerable speed. More importantly, when I went to turn off the motors, I accidentally threw one into reverse. The result was more efficient turning than we ever saw with the servo. Therefore, the servo motor is now history and we're steering just by spinning the rear wheels in opposite directions, like a wheelchair.

This student summarized his team's process succinctly:

> Some of our mechanisms were the result of careful planning, but most have just come about from playing with the pieces and seeing what we can do.

This synopsis could be applied to many of the students' LEGO-building work.

5.1.2 Programming Strategies

When the Interactive C programming environment (discussed in Section 3.2 became available, students had two different sorts of reactions to the user-friendly, rapid-prototyping development environment. Some, as we anticipated, used the system to create playful robot demonstrations that had little to do with the contest problem per se. For these students, making a simple demonstration was the best way to learn about the potential that a robot might have—knowledge they would use to guide their design process when they went about tackling the contest task.

Many students, however, explored the programming environment just enough to know how it worked, and then assumed that they already knew enough about programming that they didn't need to explore *robotic* programming. These students expressed simple surprise that the top-down methods didn't work when they actually tried to get their contest robot design to perform its task. It seemed that with regard to the mechanical aspect of design, students realized they needed to play with the materials (i.e., LEGO parts) to express their ideas, but that with regard to the programming and control side, they expected that they could construct solutions just by thinking about what should work:

> I spent most of the latter part of the week working on my wall-following procedures with limited success. *I only recently realized the need for using the robot in conjunction with programming, for often what you think will happen does not.* My first wall-following attempt looked good on the screen, but when I ran it, it looked [terrible]. (Emphasis added.)

This student's experience was typical. Many students spent time constructing elaborate control programs before they had spent time trying to get simple behaviors to run on their robots. They then attempted to "fill in" the sub-procedures of their master control program, as this student describes:

> The coding is also progressing, though at a slower rate. Our approach to programming has been top down. We have a shell of our strategy in place. We need to implement many of the subroutines that the main control program will call. We are debating various turning strategies ranging from shaft encoders, optical sensors to determine our orientation [with respect to] the playing field, and using the back bumpers to align the machine perpendicular to the side wall.

This design style is the traditional top-down method favored by most computer scientists. In the task of robot programming, it failed badly, because, as was discussed in Chapter 4, most students' robot strategies were based on assumptions of robot control that were faulty. Students were particularly susceptible to this type of design problem because of their confidence in their understanding of the problems they were facing, their confidence in their programming ability, and their confidence in the problem-solving strategies they had used in the past.

5.1.3 Time Constraints

While the existence of a particular mechanism or piece of code did not per se imply a commitment to its use, many students were confronted with the situation of time running out with less-than-ideal system components in place. As described by the student in the following quote, often a "prototype" became the final design:

> I don't know how many prototypes (for our LEGO design) we've made, but it's been a bundle for sure. We can't seem to make a robot that corresponds well with our strategy (which primarily deals with blocking the goal with our robot). We're looking for a VERY strong robot that won't fall apart if run into by the opponent's robot, hence eliminating chance of default. We also need speed (hence, a low gear ratio) to get to the ball dispenser and shoot quickly. Thus far, we've developed a few pretty decent models, however, we have been ambitious (desiring the near perfect model). This could hurt us now, as we are running out of time. We have a working model right now, and we'd like to improve it, but might have to stick with it due to time constraint.

The same phenomenon happened with code, often resulting in major simplifications of strategy and programming.

Each year a few students decided to perform major reconstructions of their design with just one or two days before the contest. Usually this forced students to realize how interdependent the different levels of their design had become. For example, one team redesigned the primary drivetrain for their robot the night before the contest. While the new geartrain was significantly better than the old one, their program no longer worked because it was peppered with hard-coded timing constants that turned out to have dependencies on the original geartrain. The team never recovered from their mechanical "upgrade."

This real-world engineering lesson of systems and deadlines was a surprise to many of the students.

5.2 Design Excursions

This section illustrates various ways in which students formulated their own investigations and little research projects based on ideas that came from working with our materials. These projects were facilitated by the technical openness of our system, as was described Chapter 3, and our own policy of encouraging such endeavors.

As part of tendency and freedom to "play around" with various ideas during the course of their projects, a number of students spent a fair bit of time on sub-projects that only sometimes had practical relevance to their main project. Some of these design excursions were a natural result of the bottom-up process that the students were using. A mechanism, chunk of code, or other artifact would get discarded when it no longer remained a necessary part of a revised strategy or functional plan. Sometimes a sub-project would be started to play with an idea or on a whim, without the expectation that it would necessarily become part of the final design. Through the normal course of this working style, artifacts that were once thought valuable would no longer be considered so, and vice-versa.

Other design excursions came from a more deliberate attempt to create something unusual. Some of these efforts were driven by functional considerations (e.g., a new type of sensor device that would play a key role in an overall strategic plan); others were simply aesthetic (e.g., additional hardware or software for creating musical sound effects).

During the *Robo-Pong* year, several teams embarked on a sub-project to develop motor driver electronics that would be more effective (i.e., would deliver more power) than the stock motor driver circuit. This phenomenon is interesting on a number of levels, because it illustrates not only the students' desire and willingness to invest time in what might be considered a frivolous sub-project, but how their efforts fed back into the design of technology and policy for the class.

The remainder of this section discusses these design excursions to illustrate the variety of learning styles that were supported by these projects. We will begin with the motor

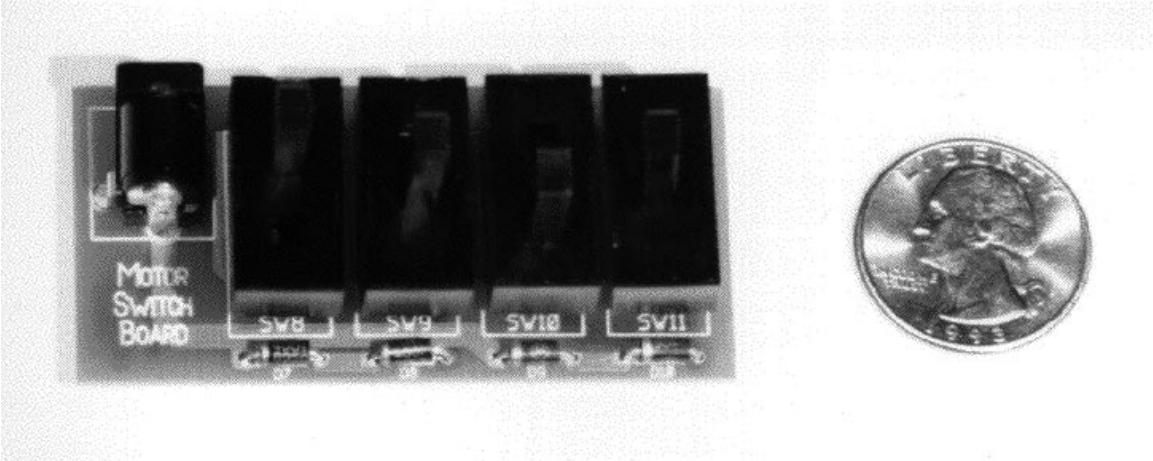

Figure 5-1: Motor Switch Board used for manual control of kit motors

driver circuits just mentioned.

5.2.1 Improving the Motor Drivers

In the first two years of the project, there was no straightforward way for students to test motors other than by driving them through the controller hardware. It became apparent that this significantly hampered students' ability to play in early stages of LEGO designs and thus develop a sense for the capability of their LEGO constructions. In order to test their vehicles, students would push their motor contacts up against the battery terminals to see how their devices would run, a clumsy and inelegant solution (though it worked and was helpful nonetheless).

We developed the *Motor Switch Board* as a remedy to this situation. Introduced in the *Robo-Pong* year, the switch board was a panel into which student would plug their motors; the panel was then plugged into a battery. Students could easily switch on and off up to four motors (see Figure 5-1).

There was one complication with the use of the switch board. The amount of power that the controller electronics delivered to the motors was significantly less than the amount of power delivered from the switch board. Motors operated by control electronics ran at about half the power than those operated by the switch board. Students were informed of this effect, and cautioned that they should not expect the same level of power when they

installed their controller boards. Nevertheless, a number of teams built mechanisms that performed adequately when powered from the switch board, but not from the electronics.

Upset when they discovered just how much less power was available from electronic control, approximately five student teams then decided to take advantage of the "$10 Electronics Rule," a rule we had established to encourage creative sub-projects. According to the $10 rule, students could spend up to $10 of their own money to purchase electronic parts for their own custom modifications.[1] In addition to restricting the monetary expenditures to $10, we set a limit of ten additional components.

Rather than redesign their mechanics, these teams set out to build their own motor control electronics to replace the "poorly performing" stock electronics. Perhaps motivated by the motto, "Stronger is faster, and faster is better," some of these students soon discovered that the latter half of this premise turned out to be an erroneous assumption.

In building motor driver circuits, two types of components are typically used: electromechanical switches (relays) and electronic switches (transistors). Some circuits use one device family, the other, or a combination of both. We allowed students to use either approach, so long as they adhered to the terms of the rule. We did not give them particular advice as to which approach was better.

As it happens, designing motor control circuitry is a non-obvious problem. There are a number of tricks that must be employed to ensure that the control circuitry is not damaged by current spikes that occur during normal motor operation. Most of the students who went about building their own circuits did not do the research required to uncover these tricks, and constructed hardware that caused significant trouble—which often did not show up until after they thought their circuit additions to be finished.

One student, who received help from an expert friend, constructed an exemplary circuit. He used more than ten components, however, having overlooked this constraint. It was upsetting to have to tell him that he could not use his circuit, particularly since it was an excellent solution to the perceived problem, but we had established the rule and to stand by it in the interest of fairness.

[1] We had agreed upon the $10 figure as an amount large enough to allow the purchase of interesting components but small enough to not favor teams with strong financial resources.

Other students encountered a variety of other difficulties. Most common was increased unreliability in the basic computer hardware as a result of poorly designed motor circuits. Unfortunately these could not be easily fixed without breaking the ten-component limitation.

One team who ended up using their motor circuit to the dire end encountered a different sort of complication. Their circuit delivered a great deal more power to the motors, as they desired, but it also made the actual power output of the motors much more dependent upon the battery voltage level (see discussion in Section 4.1.3). It seemed that the standard circuit tended to reduce differences in actual power as a function of battery level.

In order to gain reliable results, these students developed their control software with heavy dependencies on the assumption that their battery was fully charged. After just a few minutes of usage, the battery voltage would drop and their robot would no longer function properly. They didn't have time to properly generalize the program parameters as a function of battery level, so they were forced to constantly have batteries on charge and to charge batteries between rounds during the contest itself. In the end, it made for quite an unreliable design.

The battery discharge problem, and the change in motor performance that resulted from it, was a problem that was endemic to the hardware system we had created; all students had to deal with it. Depending on the design of the basic strategies for negotiating around the playing field, the problem would manifest itself with more or less conviction. The change in motor driver design, however, made the problem nearly intractable, as this group of students discovered.

After considering the experiences of the students who had attempted to construct their own motor circuitry, we were faced with a difficult decision. The attempts had caused a lot of frustration both on the part of the students who were trying to get buggy circuits to work, and on our part, those who were helping them. There were already enough hardware problems in getting all of the students up and running using the standard hardware, with too few people available to help. While these students had taken the admirable initiative to do something clever, and had a valuable learning experience, they were the cause of an untoward drain on the class resources.

The modifications were clearly unsuccessful from an overall performance standpoint.

A couple of teams had succeeded in fielding robots that were obviously more powerful than their competition, but the students were not capable of controlling that power and turning it into an advantage. We knew that we didn't want to be designing contests based on the premise that bigger is better, and we didn't see the need to encourage this attitude in students either.

In subsequent years, we decided to disallow modifications to the motor driver circuitry. To address the students' concerns about the shortcomings of our standard circuit—both perceived and real—we took a dual approach. First, we implemented a somewhat better standard circuit in subsequent years to alleviate the actual problem. Second, we installed current limiters in the hand-controlled motor switch board—deliberately "crippling" its performance so that students would be not be deceived by the amount of power they would see in their early manual testing!

While these actions mitigated the electronic power problem as well as students' perception of it, I have mixed feelings about the decision. Had the human resources for running the class not been so constrained, it would have been better to not cut off that avenue of exploration for the set of interested students.

5.2.2 Developing New Sensors

Other design excursion projects were less problematic from a practical or policy level, but still indicative of students' style of participation in the project. In the *Robo-Pong* year we distributed a number of Hall effect (magnetic) sensors in the students' kits. These sensors were unusual in that we did not have a specific purpose in mind for their use, nor did we provide a debugged circuit for employing them. One student, "Frank," became quite intrigued with the Hall effect sensor technology, and set out to design an ultrasensitive Hall effect sensor—one that would be good enough to detect the Earth's magnetic field. With the use of this device, he hypothesized, his team's robot would be able to know its orientation with respect to the playing field at all times, which surely he could turn into an advantage in game play.

Frank experimented extensively with the Hall effect devices provided in the kit, determined that the provided ones were inadequate for his purpose, and then went on to research

better devices. Aware of the $10 limit on additional electronic purchases, he requested that he be allowed to have prospective sensor devices donated to his project as engineering samples. We agreed to this suggestion; there was nothing preventing other teams from doing the same if they so chose.

Frank's work did not result in a deployable sensor device; when the engineering samples did arrive it was too late to develop circuits around them and use them in the robot design. Nevertheless, the sub-project was a valuable engineering experience for him.

5.2.3 Musical Robotics

In both the *Robo-Pong* and *Robo-Cup* years, musical hardware and software additions were a popular diversion. Students initially became excited about the possibilities of musical robots via experimentation with stock hardware and software provided—a small electronic beeper and software routines for playing simple tones of varying pitch and duration.

While some students focused their musical efforts on transcribing their favorite songs to play on the beeper (e.g., The Beatles' *Hey Jude*, Wagner's *Ride of the Valkries*, and MIT's *The Engineer's Drinking Song*), others modified the sound playing hardware to achieve greater volume or variety. The most ambitious project involved grafting an electronic keychain "zapper" (a pocket device that creates sound effects of laser blasts and other explosions) to the robot's microprocessor controller. The project involved electronically mixing the sound output from the zapper device and the microprocessor beeper circuit into an amplified speaker driver. Under software control, the circuit was able to simultaneously play tones while triggering zapper sounds. The *pièce de résistance* of the project was the musical selection that the students chose to play on their circuit: William Tell's *Overture of 1812*, complete with cannon sounds, courtesy of the keychain zapper.

For several students, the work on the musical projects, which were obviously of no utility in getting their robots to perform the contest task, was one of the few relaxing aspects of the whole project. Here is how one student describes working on the musical program:

> I wrote a short program to make the beeper play a little tune (*Flight of the Valkyries*), and Eric wrote two programs to play *Let it Be* and *Hey Jude*. I found writing this program to be the most relaxing and not nerve-wracking part

of the contest.

These excursions were precipitated by the simple provision for sound output we had built into the stock kit hardware. If for no other reason that for providing this kind of emotional outlet, I consider the audio projects to have been extremely valuable.

5.2.4 Alternate Control Systems

In the first year of the hardware contest (*King of the Mountain*), one student, "Peter," undertook a development effort that foreshadowed our work in subsequent years: that of creating on-board computational control (in the *King* contest, students' robots were controlled by off-board, desktop computer computation).

Peter was motivated by an aesthetic sense that objected to the standardized hardware we had provided that year, which required a tethered cable to join the robot to its "brain." Providing local control on the robot itself was something we would have liked to have given the students, and would provide in all following years. But in the *King* year, it was not available, and with good reason: we did not know of an inexpensive and sufficiently simple way to implement on-board computing at the time.

Peter knew of a way to implement a compromise solution. Rather than use a fully programmable microprocessor to control his robot, he proposed the use of a dedicated circuit with limited programmability, known to computer scientists as a *finite state machine*, or FSM. The hardware of an FSM consists of just a few simple integrated circuits—in fact, single-chip programmable devices to implement FSMs exist, and this is just what Peter proposed to use.

Peter succeeded in implementing a hill-climber using the FSM method, though his robot did not win the contest overall. His robot still needed the cable to provide power to operate the motors, so the aesthetics of the solution were not completely ideal, but he had the satisfaction of knowing that his was the only robot that year with true on-board control. He also did avoid any number of annoying interfacing problems that other students had to deal with.

In the contests since, a few students poured their efforts into developing alternative

approaches for controlling their robots; the case of the student determined to work in assembly language, described in Section 4.3, is one example. This student was motivated by a desire to be "close to the hardware"; others wanted to move in the opposite direction, toward more abstract models of control. In the *Robo-Cup* year, several students expressed interest in experimenting with robot control ideas developed by MIT's Professor Rodney Brooks, who developed the *subsumption architecture*, a novel philosophy and practical approach for developing robust, adaptive control systems for mobile robots (Brooks, 1986; Connell, 1988). One student in particular spend most of his month's time (to the chagrin of his teammates) in an attempt to get Brooks' software to run on the course hardware. He did not get the system functional in time to use it for the contest, but ultimately finished the project during the following academic term.

5.3 Redefining the Goal of Participation

Another way that our project encouraged involvement was by allowing students to adapt the goal of participation to their own desires. For most students, the purpose or goal of their participation in the project is to create a workable solution to compete in the final contest. Certainly, many students are interested in the project for the learning experience they expect to have in the process, but the goal of creating a working robot at the end is a big part of the motivation. Some students, however, choose goals other than this norm as what was most important to them. The protracted design excursions discussed in the previous section are examples of this phenomenon—students deciding that what was most important was spending time on issues that may or may not be most effective in producing a functional robot.

From the earliest contest, there were examples of students who redefined the contest challenge in a way to make it more interesting for their own participation. During the *King of the Mountain* contest, one team chose to take a particular rule item, which was intended to eliminate the possibility of a purely mechanical solution to the contest, as a direct challenge.

5.3.1 Mechanical Intelligence

The *King of the Mountain* contest had a clause that decreed that robots would be placed at a randomized orientation during actual contest play. This rule was established with the intent of making it unlikely that a purely mechanical solution could be successfully built. Our supposition was that robots would have to do at least a little sensing and computation to be able to climb the hill, given that they could not count on the direction they would face at the start of the round.

One team took this as a direct challenge. This team set out to build a robot, which they later named *Stupid*, that would solve the contest without any sensing and computer control at all. The design solved the problem of climbing the hill through a sort of "mechanical intelligence" based on gravitational feedback. The robot they built was a simple walking device. It had a platform base and a pair of supporting legs. In operation, the legs would swing forward, lift the base off of the ground, advance the base forward in the air, and then deposit the base back down again.

But there was a trick. Every time the legs were raised in the air, they were free to pivot about their center. Due to the way the legs were balanced, they would rotate to seek a subsequent step that was uphill from the last position! So the robot solved the problem of climbing uphill without any need for "sensing and control" in the traditional sense. As soon as the robot was given power to walk, it would start climbing uphill, regardless of initial orientation.

The design needed no additional mechanism or control to stop at the top of the hill. When it reached the top, it would indeed take a step downhill (it had nowhere else to go) and was "too stupid" to stop, but then on the following step, it would swing around 180 degrees and climb back uphill again. The design was a simple but elegant solution to the contest problem.[2] (It should be noted that none of the hill-climbers *needed* a separate stopping mechanism: after reaching the peak of the hill, they would go beyond the top, but would then just turn around and head back for the top.)

[2]The mechanics required to implement *Stupid* were not trivial. In fact, a last-minute modification to the pivoting mechanism resulted in a jam during the contest performance, causing the robot to lose a round and eventually cede the first place contest position.

Figure 5-2: *Blind* unfurls its arm toward the puck

The same team of two students returned to collaborate on a design in the *Robo-Puck*, the subsequent contest.[3] Again the students attempted to foil the contest designers and develop a machine that did not rely on computer control.

The challenge of *Robo-Puck* was to locate and gain possession of an infrared light-emitting hockey puck. Again we employed the randomized orientation rule to discourage mechanical solutions. This time, the team's solution was based on the fact that while the *orientation* between one's robot and the puck was thus randomized, the *distance* between them was always the same. (This can be seen in the diagram of the *Robo-Puck* playing table, shown on page 79.)

The students created a massive arm that swooped out and scooped the puck from its initial position. In the spirit of *Stupid*, they named this robot *Blind*, since it did not use light sensors to locate the puck, but rather attempted to grab the puck from its starting position in the center of the rink, before either opposing robot would have a chance to displace it from this location.

Figures 5-2, 5-3, and 5-4 show the arm in successive stages of deployment, sweeping the

[3]Beginning in the 1993 contest year, it was decided that students would be allowed to participate as contestants in the project only one time. This was due to the large oversubscription of the class and the need for admission lotteries in the past several years.

Figure 5-3: *Blind* brings puck to the rim

Figure 5-4: *Blind* loses control of the puck to *Bertha*

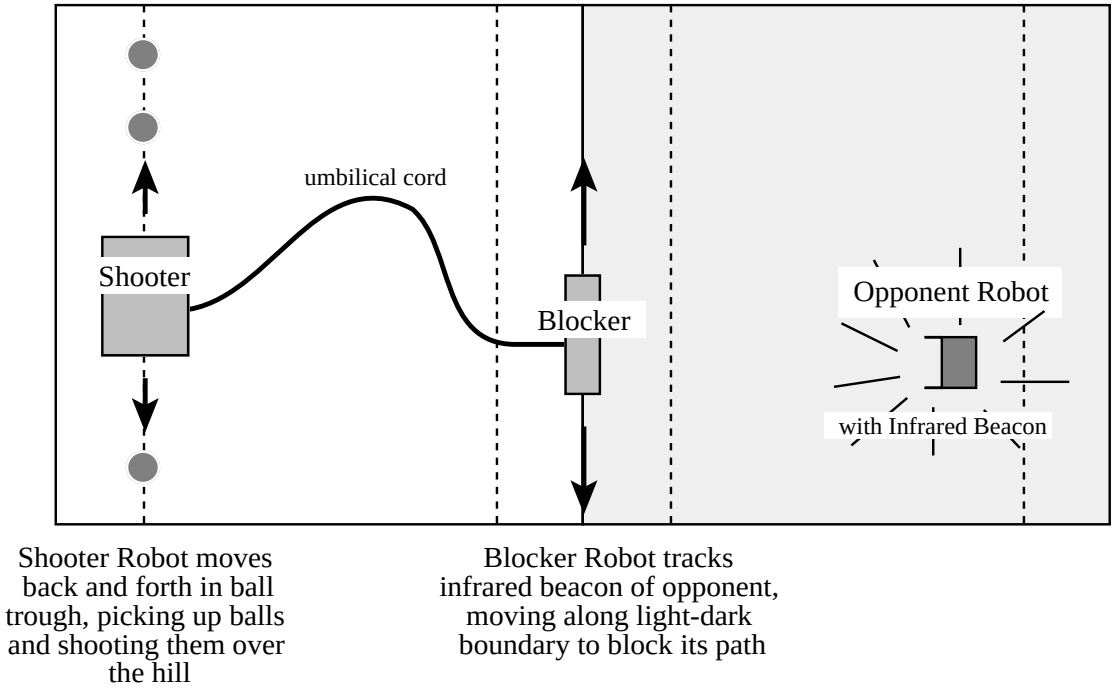

Figure 5-5: Strategy diagram for *Mutton Jeff* in *Robo-Pong* contest

puck from the center of the rink and toward the rim. In these figures, *Bertha*, a competent puck-fetching design, is hot in pursuit of the puck, and ultimately won the particular contest round. *Blind* did require a little bit of programming, to make sure that the arm triggered at the appropriate time, but the robot remained true to its designers' goals of being essentially a mechanical solution.

5.3.2 Deliberate Complexity

Other students took a more accommodating approach toward the objective of the contest, but added their own constraints to define what sort of robot they'd like to build. For example, a number of students perceived the flaw in the *Robo-Pong* contest design that would have allowed a simple but dedicated "assassin" robot to win, but rejected it for their own robot, deciding it would be too trivial a project. Students that did design attack robots ended up burdening them with enough unnecessary complexity that they failed to have the reliability needed to win the contest overall.

Some students deliberately chose designs that would be complex. One of the most

striking of these was a dual robot design created by a brother-and-brother team during the *Robo-Pong* contest. Each brother was responsible for one of the two essentially independent robots that made up the design. One robot drove into the ball trough and traveled back and forth, picking up balls and shooting them over to the other side. The other robot drove to the center plateau dividing edge and moved back and forth along the edge, tracking the opponent's infrared beacon (see Figure 5-5).

When the system worked, it made for an astounding performance. The blocker robot was effective in keeping the opponent at bay while the shooter robot delivered balls to the opponent's side. The robots sometimes had trouble disengaging from their starting configuration, but if they deployed themselves successfully they were effective and exciting to watch. The two students who designed the robotic duo were less concerned with it winning the contest, which it did not, than with it having at least one successful contest round, which it did.

Other students focused more on the *process* of their work rather than the final outcome. Here a student describes how he and his partner's proclivity for complexity got in the way of producing results:

> Our design process is pretty haphazard. Darrin is a true hacker at heart, and we both just play around with ideas and try to get them to work. Since our big idea didn't materialize (probably more for lack of time and frustration rather than poor design), this random playing around may not have been as good as a consistent plan. Also, I think our team succumbed to a little to what Fred Brooks calls the "Second System Effect" in his book on software engineering, *The Mythical Man-Month.* Brooks' thesis is that when a designer builds his second system, he is tempted to include all of the widgets and neat ideas he thought of during the construction of his first system, but wasn't able to include because of time constraints, etc. This is dangerous because it can put the second system design seriously on the wrong track as the implementors try to get all of these cute hacks to work, and the system gets bogged down, "overdesigned," and becomes unwieldy because of its complexity. In the same way, our robot is a victim of this effect as Darrin and I tried to work in the neat ideas we had last year (rotating wheel assemblies, a differential, centrally located drives, etc.) As we found out, the design got out of hand and we didn't have the time to get it right. Perhaps we should have followed the KISS rule: "Keep It Simple, Stupid!"

5.3.3 The Talent Show

Another example of students' redefining the goal of their participation was a compromise to rescue satisfaction from a failed situation. Beginning with *Robo-Puck* in 1990, we had established two nights of contest game play. In the first night, one round of matches would be run (each robot would play one time), and results would count. The second night was the public event, drawing a large crowd to see the remainder of the competition, as many rounds as were necessary. We ran a double-elimination contest, meaning that a robot was allowed to lose once and continue playing until lost again. Hence, a loss in the preliminary night did not preclude a robot from playing in the public contest, though it would be eliminated after sustaining a second loss.

Beginning in the 1991 *Robo-Pong* year, we put a clause in the rules that stated that robots had to demonstrate "minimal proficiency" in the preliminary contest round in order to qualify for participation in the main round. We created this rule to help ensure a reasonably entertaining contest night—one without too many disappointing robots.

Minimal proficiency was defined as being able to win against a non-moving, "placebo" opponent. If a robot lost in the preliminary round, it would have to be tested against such an opponent before qualifying for the main round. Students were given until midnight of the eve before the main contest to make their robots capable of defeating "the placebo."

In the *Robo-Pong* year, this qualification rule was essentially moot, as any robot that could move around the table at all was likely to "accidentally" hit one of the center plateau balls over the top of the hill and onto the opponent's side, allowing the robot to claim victory. Indeed, the *Stupid Scorpion* robot (described in Section 3.1.2) created in a last-ditch effort to get a working machine, used a rudimentary back-and-forth movement strategy and placed third overall in the *Robo-Pong* contest, much to the surprise of its creators. In 1992's *Robo-Cup*, however, a robot had to demonstrate a series of competences in order to score a single point: it had to navigate to the ball dispenser, trigger the panel to feed a ball, and then direct the ball to reach its goal. The qualifying rule clause thus became a serious affair, determining participation in the main contest.

As it happened, one team in *Robo-Cup* either decided or realized, before the night of the preliminary round, that they would be unsuccessful in qualifying for the performance in the

main contest. They then arranged to have their "performance" take place in the preliminary round: they invited a number of their friends to attend (though the preliminary round was not publicly advertised, it was open to an audience, provided they didn't complain about the pace of the contest rounds!). They decorated their robot with various accessories, and then programmed it to play a song and dance ("La Cucaracha") *rather than try to solve the contest.* Their friends gave a big cheer when their turn came, and the robot functioned as it had been programmed:

> José Luis had proposed adding music to the robot, so I got a small speaker; we think it will play "La Cucaracha," which prompted the name "BAYGON"—a household insecticide in Latin America.

I was not aware that the students had planned this until witnessing this event as it was occurring. Afterward, I attempted to persuade them to not give up yet (they had until midnight of the following evening to get their robot to qualify), but they had had enough of the project. While I was glad that they had the creativity and conviction to make the most of their preliminary round performance, it upset me that by the rules we had established, they would be kept from competing in the main contest round. Surely our rules weren't meant to keep dedicated students from having their chance to display an interesting robot!

It was too late to change the rules for these students' case. In subsequent years, rules were revised to allow students whose robots had been disqualified to create show robots for the main contest performance, or to compete against other disqualified robots in an exhibition round.