Session #2793

Mobile Robots as Instructional Technology Tools for CELTS

Richard W. Freeman, Katherine C.S. Whitaker
Iowa State University

Computer Engineering 181X and 182X, Computer Engineering and Problem Solving I and II, were designed as pilot courses for Computer Engineering students involved in the new Learning Community within Iowa State University's Electrical and Computer Engineering Department. As a pilot program, the learning community was designed to be flexible, but include subject matter elements of Computer Engineering, at the same time focus on the retention of first year students. As part of the retention effort, mobile robots were incorporated as an instructional technology tool.

The Mobile Robots served to teach problem solving, as well as reinforce C Programming, teamwork, and other social skills. The robots also served as a teaching platform for second and third year Computer Engineering concepts. First year Computer Engineering students were introduced to the concepts of Digital Design, Computer Organization, and Embedded Systems.

In the flexible environment established for the Learning Community, the students formed teams and assembled the robots. This was not the original intent of the staff, but allowing the students to assemble the robots satisfied their need to build their own mobile computing platform.

This paper focuses on the objectives, selection criteria, and exercises associated with using mobile robots as an instructional technology tool for a first year Computer Engineering Learning Community.

Introduction

"I want to be a Computer Engineer because I like building/programming computers." This is not an uncommon statement made by first year Computer Engineering Students. While students are given catalogs, bulletins, and flowcharts that explain their learning program, they expect to start "doing" Computer Engineering from Day One. Instead they face a year of Physics, Chemistry, Calculus, and other foundation courses and some students become discouraged. Some ultimately choose to change majors or transfer to another university.

Research universities should foster a community of learners. Large universities must find ways to create a sense of place and to help students develop small communities within the larger whole.[1] Learning Communities, at Iowa State, have been established to allow students to network with each other, build relationships with Computer Engineering faculty, increase student retention, and teach valuable social and team skills. Nonetheless meeting student expectations to begin their Computer Engineering studies was still an issue that needed to be addressed.

It is incumbent upon the faculties of research universities to think carefully and systematically not only about how to make the most effective use of existing technologies but also how to create new ones that will enhance their own teaching and that of their colleagues.[1]

Mobile robots were included as instructional technology tools as platforms for teaching the skills necessary for student success using cooperative learning, coping with different learning styles, and understanding engineering problem solving techniques. The original vision was of robots being built by us, with the students then using the robots to solve engineering problems. In practice, the students asked to build the robots themselves, with us supervising. The decision to allow the students to build changed the objectives of the Learning Community to include being a cornerstone course. This allowed Computer Engineering 181X and 182X to meet students' needs for hands-on experiences.

Computer Engineering Learning Teams

During the 1998-99 academic year, our Learning Community developed a distinctive personality. We started with 25 students, a budget, and a vision. Given the opportunity to name this Learning Community, the students decided on CELTS- Computer Engineering Learning Teams. Our staff included an Associate Professor, Adjunct Instructor, two graduate students, and three members of Iowa State's Project LEA/RN.

Project LEA/RN is Iowa State's effort to promote the use of cooperative learning techniques among faculty and students. The Project LEA/RN members of CELTS introduced and coached the students on using cooperative learning skills, as well as helping develop the course curriculum. During work in the lab, these members provided very valuable in observing student behavior and reinforcing the use of cooperative learning techniques.

Cooperative Learning

Students were asked to work in teams. The students operated as cooperative learning groups. By definition the students worked in groups to meet shared goals. "Students discuss material with each other, help one another understand it, and encourage each other to work hard."[2] The work in the lab caused informal cooperative learning. Informal cooperative learning means having students work together to achieve a joint learning goal in temporary, ad-hoc groups that last from a few minutes to one class

period.[2] Most groups chose to work together until the robot was complete, while others tended to form and reform groups. Formal groups were not assigned.

Engineering Problem Solving

Engineering 161, Engineering Problems with Computational Laboratory in C, is a required course for Computer Engineering students. This course is an introduction to problem solving tools and concepts such as C, Statistics, SI Units, and significant figures. The goal of CELTS, during the 1998-99 academic year, regarding Engineering Problem Solving was to enhance the problem solving and C programming taught in Engineering 161. CELTS The mobile robots proved an excellent tool for integrating the concepts of problem solving and C programming.

Selection Criteria

Once the decision to incorporate mobile robots in the courses was made, the biggest decision then was to select a robot for use. The Computer Engineering Learning Teams Staff compiled a list of three criteria:

1. Mobile Platform
2. Flexible at performing tasks
3. Use of C as a programming language

A fourth criterion was included after the selection process began:

4. Capable of being used with other courses

A mobile platform allows the students to experience the effects of integrating software and hardware components into a system. Many students were used to writing code. Some students modified code to take advantage of the mobile platform. One group completed Lab 3, and then modified their code so the robot spun and played a tune when it completed its task. Additionally, using a mobile platform allowed some students to debug their programs by watching the behavior of the robot.

A flexible platform meant we could create a variety of problems and exercises. Additionally, this type of platform allows the students to design multiple solutions to solve the same problem. An example of this is Lab 2 (described below). This platform could also be used to model several important computer concepts such as Interrupts.

All Computer Engineering students at Iowa State University have at least four courses that teach, or require knowledge of, C (Engineering 161- Engineering Problems with Computational Laboratory in C, Computer Engineering 211- Introduction to Microcontrollers, CPR E 301- Microprocessor- Based Design, and CPR E 308- Software Systems Integration). Many employers recruit Iowa State students with C Programming skills. An instructional technology tool that uses C as its programming language allows

us to reinforce C programming skills, and introduce the concept of non-standard forms of computer languages.

The last criterion also served as a powerful benefit to CELTS. In justifying the cost of Rug Warrior Pro™ robots and RugBat™ Sonar Navigation Kits, we were able to demonstrate use of this technology to explain and illustrate concepts from introductory and advanced Computer Engineering courses. Some of the students from last year's team continue to refer to experiences with the robots to understand concepts covered in their current Computer Engineering courses.

Proposed versus Actual Role of Robots

The Rug Warrior Pro was intended to be a tool for teaching problem solving, C programming and cooperative learning. Since we were in the first year of the Learning Community, designing lab exercises and design challenges was only a small part of our task.

We needed to determine how to use the robots in such a way that the technology tool did not become the focus of student learning, but remained a tool, just as pencils and computers are tools. We determined that slowly introducing the robots was the best approach.

Bloom's Taxonomy categorizes the levels of abstraction of questions that commonly occur in educational settings. Following this taxonomy, we planned to evaluate student learning. The levels of the taxonomy are:

- Knowledge
- Comprehension
- Application
- Analysis
- Synthesis
- Evaluation

Applying this taxonomy to the use of mobile robots, we planned the following steps for activities:

1. Gradual immersion with the robot by creating exercises that introduce each sensor
2. Introduction to the use of multiple sensors
3. Increases in the demands on students to write code
4. Introduction to the concept of problem specifications
5. Creation of open-ended design challenges
6. Inclusion of student-designed challenges

Using the robots solely as an instructional technology tool was the major part of the plan to keep the tool from being the major focus of learning. The plan called for us to have the robots built and ready for student use before the end of the Fall Semester. The more

we talked about using the robots, the more excited the students became. Finally, a student suggested we allow them to build the robots. By agreeing to this request, the course took on the feel of a first-year design course.

We allowed the students to form teams, claim a Rug Warrior Pro™ Kit, and begin building. There was some concern about the students building the robots. "Would they read the directions? What if someone soldered something wrong? How much would it cost us in time and money to repair their mistakes?" We pondered those questions as the students used their two-hour Friday lab to build robots. As supervisors, we got little work done on our own robots, as we helped the students build theirs. After four months of work, all the student groups had finished building, and were ready for RugBat™ Kits and lab exercises. After students completed the robots, they were given the option of naming the robot. "Squash, The Bug", "Norman the Blind Man", "Godbot", "Maynard G", "Scrooge", "Spanky 2", and "Toby" are examples of the names students used for the first 16 robots built. Later robots are named after the 1998-99 Learning Community students.

Our fears were not realized. Most of the students worked very well. Many had either no, or very limited experience soldering and assembling electronic devices. Some of the students were easily distracted and had trouble staying on task. As a result, we had to rebuild portions of some robots, but this work was completed in less than a week. Building the robots had gotten the students past the "Wow" phase, and they were ready to solve problems.

The company we purchased the robots from offered to ship them assembled for an additional $3,000.00. We decided to build them ourselves. We expected that having the students build the robots would cost several hundreds of dollars for replacement parts. Our errors cost more than those of the students. The students built ten of the sixteen robots, with us building the rest. We managed to destroy six Shaft Encoders while building the robots. The students collectively destroyed one, and one Brain Kit (Processor Board). Our total cost for replacement parts was less than $500.00. The confidence and learning gained by the students was well worth the cost of the replacement parts.

"*Experimentation is the easiest way to learn.*" comment on student evaluation form

After building was completed, it was time to design labs. We wrote skeleton code and original code as protection mechanisms for the robots. We had to repair a robot we were using because of a coding error that sent it flying off a lab bench. Our new fear was that students would also not be careful and accidentally damage a robot.

By having students review the code and understand that the use of this code, in early programs, was intended only as a protection for the robots, allowed them to build their confidence in using the robots. Some unintended results were students exploring most, if not the entire Interactive C library files, students writing files for the library, and students writing code that used multiple protections.

In using the robots as a tool, students developed an appreciation for debugging skills. Students compared the expected and observed behaviors and determined what corrective actions were needed. We helped the students debug their programs by engaging them in conversations about expected versus observed behaviors. We asked students what program variables controlled behavior, and questioned their logic structures when the behaviors were not the same.

Robot Exercises

The self-formed teams of students completed four exercises. The first was used as a general introduction to the workings of the robot as a problem solver. The other labs were problem solving type labs. At the beginning the labs were very restricted. The students did very little coding and, instead, used guesswork to succeed. As the labs progressed and we all learned more about Interactive C and the robots, more problem solving and programming was required in the lab. The final lab left many options open to the students in both approach and programming.

Lab 1: Diagnostic Lab

This lab was used as an introduction to the software and the actual functionality of the robots. Students were asked to configure various sensors and setting of the robots. The stated objectives were:

Introduce various sensor functions
Calibrate sensors
Introduce to Interactive C
Record sensor bias for robot

The three sensors that need are the photocells, near infrared (IR) emitters and detector, and the collision detectors (bumpers).

To calibrate these sensors, the students used a program that came as the test program with the robots. Then they followed the instructions for each sensors' test as given in *Rug Warrior Pro Assembly Guide*[4]. The results of these tests were recorded on a separate sheet to be kept with the particular robot for future reference.

Students were also asked to find out the correct motor bias for a particular robot. The motor bias is used in several of the library functions to correct the rotation of the motors allowing the robot to move in a straight line. Because any two motors will not have the same rotation cycle, the motor bias is calculated in an equation along with the percentage of power that the motor should be given, causing the robot to move straight, assuming the bias was correctly obtained. The bias is figured by loading a program written by us that moved the robot in a forward direction for 10 seconds (see figure 1). The students then observed in which direction the robot curved. The bias was then changed and the process

repeated until the path was observed to be as straight as desired for the particular activity in which the robot is to be used. This number was recorded with the other sensor data.

Generally the students were able to set the bias correctly with little trouble. One of the robots, "Norman the Blind Man," did cause some problems. He was unable to effectively use his motor bias from one trial to another. Changing the bias sometimes changed the path but not always as expected. With a large change in bias the path sometimes did not change at all or only slightly. With a small change of 1 point, he sometimes went all the way in the other direction. On some of the other robots as bias of 36 was required while others only needed a bias of 5 or 10. This bias sometimes needed a regular check as the batteries die. The response time of the motors on low batteries is slower and this caused the robot to stagger a bit.

At the end of the lab, students were asked to fill out a debrief questionnaire. This asked them to think about the preparation for the lab, the purpose of the lab, the cooperative learning processes in the lab, and the practical applications of the lab. Most of the students found that the lab met their expectations. A few would have liked to have had a little more information about the programs and the functions called by those programs. Some of these students were able to piece together the information from the library files provided with the robot software and could then help the others. The most common response, as expected, when asked about why this lab was so important for future work, was that by recording the setting for the motor bias, that step could be eliminated in future labs. We asked them to respond to what would improve the group work. Most of them commented on the importance of communication skills, but they could not always see the connection between that and the success of the lab. Most of the students were excited to actually be able to start working with the robots and were reluctant to find ways in which the lab might be improved.

Lab 2: Distance and Wall Following Lab

This lab covered the basics of moving the robot a set distance forward. It also incorporated the IR sensors and bumpers to create a wall following behavior. The lab was divided into three separate exercises. The first two dealt with distance measurements and the third with wall following. This lab worked to change the focus of the class from the technology of the robots, to using them as problem solving tools. Even though the exercises for this lab were centered on particular parts of the robots, the overall objective was to solve the problem of traveling or measuring out a set distance.

Exercise 1: Distance Trial 1

Objective: Using a set number of seconds to run the robot a measured distance.
Process: The students were to have the robot run for 10 feet. This meant that they needed to determine the number of seconds to test. We gave the students code to use that required as a parameter the number of seconds the robot was to run (see figure 2). The

students were asked to study the code and then run the program. They were to select the number of seconds to run and then observe how far the robot went altering the number of seconds until the robot stopped at the finish line. Some of the students took the challenge to the extreme and worked until the robot stopped exactly at the line.

Exercise 2: Distance Trial 2

Objective: Introduce using the shaft encoders for measuring distance.
Process: The shaft encoders work by counting the transitions between black and white from the patterns on the wheels. The patterns that came with the robots have 32 transitions in one revolution of the wheel. These transitions are sensed by a photoreceptor on mounted next to the wheel and counters are then incremented. By reading these counters the number of clicks can be determined. The code we wrote for the students to use for this exercise required the students to input the number of clicks to travel (see figure 3). Some of the students first measured the wheel and then divided the distance to travel by the distance of one revolution and then mathematically determined the number of clicks to use as a first attempt. Depending on the motor bias this number would vary slightly. Others just randomly picked a value for the clicks and then observed the robot's behavior. Once again some of the students were not satisfied with the robot stopping somewhere near the mark, but instead worked until the robot stopped exactly at 10 feet. Some went beyond the actual problem and added imaginative flourishes, such as a little song and dance routine when the finish line had been reached.

Discussion of Exercises 1 and 2:

After the entire lab was completed, the students were asked to discuss the accuracy of the two different methods. The general observation was that by using the encoder clicks, the robot was guaranteed to travel the same distance every time. But by using a given time, as the batteries wore down the output of the motors would decrease causing the robot to travel a shorter distance. There are some issues with always using encoder clicks, because of how those clicks are stored the registers of the robot. In order to use them as a standard measuring device, the students needed to read and study the code for the encoders.

Exercise 3: Wall following behavior

Objective: Introduce wall following behavior using IR sensors and bumpers.
Process: This part of the lab was to introduce the students to the idea of following a wall by using the IR sensors and the bumpers. The robot was to keep along side of the wall using the IRs. When the wall was out of "sight" the robot was to stop and turn back toward the wall to correct the path. If the robot ran into the wall as sensed by the bumpers, after stopping the robot backed up a short distance and then turned to correct the path. If the motor bias was not set correctly the robot appeared to bob down the hall. The students were given a skeleton program that took care of the programming for the bumper sensing and allowed time for positioning the robot in the hall against a wall.

This lab flowed better than the first one. The students felt as if they were actually getting to do something fun and challenging. This lab did not have a debriefing because the students were very involved in the exercises and we did not have time left at the end of the lab session as planned. The students left the lab feeling that they had accomplished something. This lab also caused some excitement in the building because the wall following exercise was carried out in the hall outside of the lab. It drew a substantial audience of student and professors and the students certainly enjoyed the attention they were given.

Lab 3: Sonar Lab

This lab covered the basics of installation, testing, and using the sonar to judge the distance between the robot and an object. The main objective of the lab was to introduce the idea of using the sonar to create an avoidance behavior. This lab was broken into two different exercises. Each part was more open ended than the previous labs.

Exercise 1: Installation and testing of the sonar device

Objective: To install the sonar device and test that the robot is receiving the signals from the device.
Process: The students were asked to read the sonar program we wrote and then look at the sonar files that were supplied with the sonar unit (see figure 4). The program we wrote did the function calls for the students so that they could observe the behavior and use of the functions before they had to write a program that used the sonar.

Exercise 2: Using the sonar to avoid an object

Objective: Introduce the idea of creating a sonar behavior which will stop the robot within two feet of a wall. Introduce the idea of writing code to modify an existing program.
Process: The students were asked to look at a skeleton code we had written and understand what protective measures we included. This code contained a short section at the beginning of the program to allow for positioning of the robot and a section to make sure that when the robot sensed something with the bumpers, it would stop and start in a different direction. The students then added their own code inside the bumper code to have the robot go until it was 2 feet away from an object. This was the first lab in which the students were expected to write some of their own code for the robots. The part the students had to write focused on the logic for stopping the robot when it had reached its destination.

This lab also had a debriefing questionnaire for the students to fill out. Generally the students enjoyed having more of the responsibility of writing the code. They liked the open-ended part of the lab. Mostly they learned more about Interactive C and about how to use the robots as tools.

This debriefing questionnaire also served as the closing comment on the lab portion of the course. The students asked if they were to create a lab, what would they do. The responses were varied. Some said run a maze, others play tag and others just "do something fun." In general, the students would have liked a few lab sessions in which they could just "mess around" with the robots. Those who had finished the prepared labs early, frequently took the extra time to alter the given code to allow the robot to print extra information to the LCD.

Proposed labs for the future

As the year ended, we developed ideas for additional labs, but did not have the time available to perform them. One of these was suggested by another faculty member. The specification for this challenge was to put the robot in an empty, rectangular room and it would figure out the dimensions of the room. This challenge has multiple solutions. The room could be measured by the wall following behavior using the encoders to calculate the distance, or using the wall following behavior and the sonar to calculate the distance. There are of course a number of other solutions that do not involve wall following at all. One would be to place the robot in the center of the room and use just the sonar and spin the robot to get the dimensions. The students would be asked to think through the problem and then, using whatever method they chose, write a program that would perform the task.

Another idea was to develop a game of tag. We would program a few robots to play and then give the specifications to the students and have them program other robots to join in. The specifications would include how the robots "talk" to each other, how does a robot know it has been tagged, and what does it do once it has been tagged.

We have also considered doing a maze. The students would be allowed to choose how the robot is to make its way though the maze. They could use the sonar, IRs for wall following, or bumpers. Another very different way would be to coach the robot through the maze by using a flashlight in a dimmed room and using the photocells to find the light.

In addition to these proposed labs, we contemplated giving the students the opportunity to create their own labs and submit proposals for the activity. This would allow the students to find their own uses for the technology tool offered in the robot. The students would have to include the objectives, specifications and pseudocode for the lab.

Appendix: Programs for Robot exercises.

```
/* This program cause the robot to move forward for 10 seconds at 60% power. */

void go()
{       int i;
        tone(880.0, 0.25);              /* warning tone */
/* allow for time to place the robot on the floor */
        for (i = 0; i < 4; i++)
                {sleep(5.0);
                tone(440.0, 0.25);
                }
        tone(220.0, 0.25);

        driveb(60, 0);          /* go forward */
        sleep(10.0);            /* keep going for 10 seconds */
        driveb(0,0);            /* stop */
        tone(880.0, 0.1);       /* I'm finished tone */
}
```

Figure 1: Program for Lab 1.

```
/* This program causes the robot to move forward for sec number of seconds.  The first
part of the code allows for time to position the robot on the floor. */
/* Requires common.c from the RWP library files */

void go(float sec)
{       int i;
        tone(880.0, .25);               /* warning tone */
/* allow for time to place the robot on the floor */
        for (i = 0; i < 4; i++)
                {sleep(5.0);
                tone(440.0, .25);
                }
        tone(220.0, .25);

        driveb(60, 0);          /* go forward */
        sleep(sec);             /* keep going for sec seconds */
        driveb(0,0);            /* stop */
        tone(880.0, .1);        /* I'm finished tone */
}
```

Figure 2: Program for Lab 2 Exercise 1.

```
/* This program causes the robot to move forward for clicks number of clicks. */
/* Requires common.c from RWP library files */

void go(int clicks)
{       int gone;               /*the total number of clicks the robot has traveled so far*/
        gone = 0;
        tone(880.0, 0.25);      /* initial warning tone */

/* allow 20 seconds of time to place the robot on the floor */
        for (i = 0; i < 4; i++)
                { sleep(5.0);
                tone(440.0, 0.25);
                }
        tone(220.0, 0.25);

/* as long as clicks number of clicks has gone by go forward at 60% power for 0.5
seconds and then recheck the clicks traveled */
        while(gone < clicks)
                { driveb(60, 0);
                sleep(0.5);
                gone = gone + get_left_clicks();
                }

        driveb(0,0);            /* stop the robot */
        tone(880.0, 0.25);      /* I'm finished tone */
}
```

Figure 3: Program for Lab 2 Exercise 1.

```
/* Requires Rugbat.c from the RWP library files */
void sonar()
{       float dist;       /* distance to the nearest object */
        dist = -1.0;      /* set as though the ping was unsuccessful */
        init_sonar();     /* initialize the sonar registers */

/* The range function returns a –1 if the ping was unsuccessful or the distance to the
object if successful. Only print out the distance if the ping worked.  */
        while(dist < 0.0)
                { ping();
                dist = range();
                }
        printf("%d\n", dist);   /* This prints to the robot's LCD */
        tone(880.0, 0.25);      /* I'm finished tone */
}
```

Figure 4: Program for Lab 3 Exercise 1.

Bibliography

1. The Boyer Commissions of Education Undergraduates in the Research University. *Reinventing Undergraduate Education: A Blueprint for America's Research Universities.* (1998).
2. Johnson, D. W. & Johnson, R. T. Making Cooperative Learning Work. *Theory into Practice*, 38(2), 1999.
3. URL: http://www.coun.uvic.ca/learn/program/hndouts/bloom.html; Learn Skills Program.
4. Jones, J. L. *Rug Warrior Pro Assembly Guide.* Natick, MA: A K Peters, Ltd.

RICHARD W. FREEMAN
Richard W. Freeman is an Adjunct Instructor in the College of Engineering and Department of Electrical and Computer Engineering at Iowa State University. He instructs for both the College of Engineering's LEAD Program Learning Community and the Computer Engineering Learning Teams. He was awarded the Warren B. Boast Award for Teaching Excellence from Iowa State University's Department of Electrical and Computer Engineering in 1999. He received his B.S in Computer Engineering form Iowa State University, and his M.B.A. from Southern Methodist University.

KATHERINE C.S. WHITAKER
Katherine C. S. Whitaker is a graduate student in Computer Engineering at Iowa State University. She received her undergraduate degree in German at Oberlin College.