Student Feedback on Robotics in CS1

Susan P. Imberman, Roberta Klibaner, Sarah Zelikovitz

College of Staten Island, City University of New York
imberman@mail.csi.cuny.edu
klibaner@mail.csi.cuny.edu
zelikovitz@mail.csi.cuny.edu

Abstract

We describe a robotics assignment for CS1. This assignment has been used at our college since 2002. Recently we have been surveying our students as to whether the lab reinforced the programming concepts taught in the course and if students wanted to see more robotics in future courses. Student responses were positive with respect to both issues.

INTRODUCTION

Traditional introductory programming courses ask students to demonstrate their knowledge by writing programs that incorporate the concepts covered in each lecture. These assignments typically involve sitting at a keyboard, inputting computer code, executing this code, and observing the output. Students are required to desk check their output to determine if the expected results were obtained These types of assignments tend to mirror the "chalk talk" lectures delivered in the classroom; dry, dull and predictable.

Robots provide a robust learning environment which allows students to absorb these same concepts. Students are becoming more aware of the new uses for robots, such as robot surgeries, robot Mars rovers, and robot rescuers. It is not difficult to get students interested in this new and emerging discipline.

Working with robots gives students a view of programming environments other than the typical compiler, such as the Microsoft Visual C++ normally used. Dealing with "real world" hardware problems is not usually an experience that CS1 students have. In robotics, issues such as, *it didn't do what it was expected (instructed) to do,* are quite prevalent. Many students studying robotics are familiar with sensors not being accurate, wheels that don't turn at the expected or same speeds, parts that break, and power issues. Experiencing these issues early in a student's programming career makes it easier for them to understand the need to incorporate error handling in future projects.

In light of this, we at the Department of Computer Science, at the College of Staten Island (CSI), decided to implement a required robotics programming assignment in our computer science 1 classes [Imberman and Klibaner]. The assignment was designed to be given mid semester, after students have been introduced to functions, taught simple looping constructs, such as `for` and `while`, and decision constructs, such as `if-else`. Our intention with this assignment was to show students that even with limited programming skills, they could still write interesting programs that would illicit complicated behaviors from hardware objects such as a robot.

We had several motivational goals for this assignment as well. Because of limited programming skill, creating relevant and interesting assignments is difficult. After several "cute" problems, students tend to become bored with programming, thus questioning their initial interest in the major. We felt that different and interesting assignments that have an intrinsic appeal to students would help to prevent such ennui and encourage students to continue their computer related studies. We have also implemented an assignment with GUI constructs to prevent the same kind of ennui.

Our robot assignment has been offered since spring semester 2002. The BS in computer science at CSI is an accredited degree. To maintain our ABET accreditation; we are required to assess and evaluate whether or not the course of study satisfies the goals we've laid out to accomplish. The robotics lab requires a significant commitment with respect to the purchase and maintenance of the robot equipment, as well as technical support in the laboratory. To see if the assignment fulfilled its objectives, we included questions pertaining to this assignment on the assessment survey given to each student at the end of the semester.

The organization of this paper is as follows: even though our computer science 1 project is detailed in [Imberman and Klibaner], in the interest of completeness, we will describe the project in the first section. The next section will display and describe the survey results. Last, we will discuss the results, give our conclusions, and indicate our future direction.

THE ROBOTICS ASSIGMENT

Although robot construction adds to the fun (and frustration) of a robotics project, dealing with the issues inherent in building a robot within the time allocated for this assignment caused us to decide to build a suitable fleet of robots for class use. We used the design outlined in Fred Martin's Robotic Inventions [Martin] for the Handybug 9645. We chose this because we wanted to use a standard robot architecture for all the robots built for this lab assignment. Instructions for robot construction were clear and well documented with picture illustrations. The Lego Dacta 9645 kit contained most of the parts needed. In short, the HandBbug 9645 offered a simple, effective way of producing 15 working robots. The cost of each robot, including parts and HandyBoard was estimated to be about $435.

There were some slight modifications made to the basic HandyBug architecture. As described in [Martin], Handy Boards are supposed to be attached to the robot via LEGOS that are hot glued to the board. Since we use these boards in other courses [Imberman], in lieu of glue additional LEGOS were added to securely hold the handy board. Figure 1 shows a modified HandyBug robot.

.

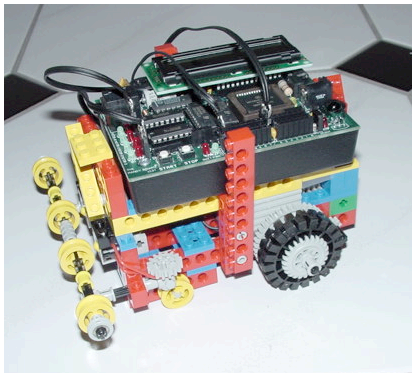

Figure 1 - Modified HandyBug

The entire Computer Science department became involved in robot construction. Faculty, students, technical staff and secretaries contributed to our fleet. Figure 2 shows a picture of our finished products.

The Assignment

The robotic assignment is broken into several tasks. [www.cs.csi.cuny.edu/~imberman/csc126/ROBOLAB.htm] Each task gets progressively more complicated. The first task, Robo-Rap, requires students to write an IC program that will beep, wait 1 second, beep again, wait one second, and then beep yet again. Robo-Rap illustrates the interactive C (IC) environment and how to "talk" to the robot. Students name their robot in the command mode of IC and then write their first IC program. This gives us an opportunity to discuss and demonstrate the difference between command mode, interpreted and. compiled languages. Students learn that not all C programs require `#include` statements, the handy board that we use has the necessary C libraries already preloaded. Since the handy board is programmed in C rather than C++ we must discuss the standard I/O and how it differs between the two languages.

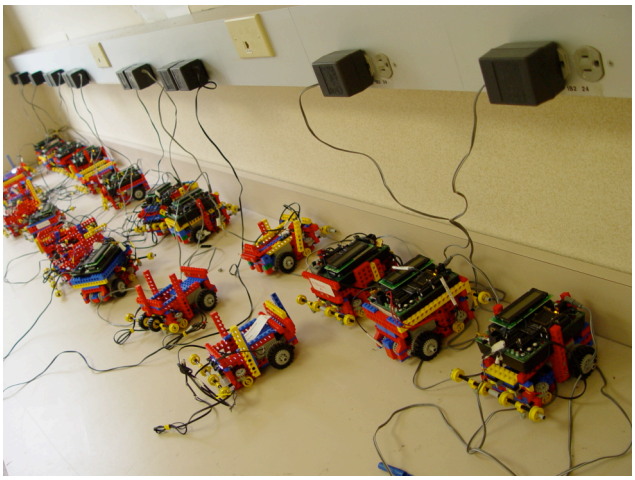

Figure 2 - Our Robot Fleet

The second task, Tickle Me Robot uses the touch bumpers and an endless loop. Students write a program that continuously interrogates the digital ports, if the right sensor is depressed a message is printed on the LCD that says: *Hee! Hee! You tickled my right side!!!* Depressing the left sensor displays the message: *Hee! Hee! You tickled my left side!!!* This assignment is designed to illustrate the concept that not all endless loops are bad. An endless loop is required to keep the robot active and *laughing*. During the execution of the assignment students discover that hardware failure can and does happen. It is a good time to discuss *mean time to failure* and the necessity of always having spare parts available. Also, the sensors are not always accurate and do not always respond as expected.

Robo-Rumba! It's time to make the robots move. In this assignment the robot is programmed to continuously turn right or left while moving forward, then move backwards while turning right or left. The *dance* ends when you catch your robot and turn the power off. Motors control the movement of the wheels and the same command is given to each motor but the robot usually favors one direction over the other. Students learn from this task that the expected power of each motor may not be consistent. If they want the movements left and right to mirror each other, adjustments may have to be made to the amount of time spent turning or moving forward.

Finally, we put it all together in Robo-Ruckus!! The robot is instructed to move forward until it hits an object, if the right touch sensor is activated the robot moves back and

turns left and then continues forward once again. If the left touch sensor is activated instead, the robot will back up and turn right before moving forward. During all the forward motion the robot continuously beeps. All of these commands have been incorporated in previous tasks and do not pose a problem for the students; however, it is the hardware that causes the most difficulty. Since this is the last exercise of the assignment, LEGO® pieces fly as robots hit obstructions. The front bumper may fall off or a wheel may become undone. The robots are running on rechargeable battery power and may require recharging before this exercise can be completed.

Working in a *live* environment re-enforces the concept that programs can and will affect others. It shows the importance of checking and double checking our output to determine the correctness of our project. It also illustrates why code for embedded processors should include fail safe code to avoid possible shut down should a processor not respond as required.

SURVEY RESULTS

As part of an ongoing assessment project, associated with our ABET accreditation, the Computer Science Department at CSI does an evaluation of courses each semester. Questionnaires are distributed to all students in the classes that fulfill CS major requirements. Questions for all courses typically ask about the professor teaching the course, the level and workload of the course, usefulness of textbooks and assignments, as well as other generic questions that are appropriate for all CSC major courses.

Our CS1 students had four specific course-related questions on their questionnaire, two dealing with the robotics lab that is presented in this paper, and two dealing with a GUI lab that we recently introduced into the course.

The students were asked to rate the following two statements, as *strongly agree, agree, disagree, strongly disagree* or *not applicable:*

1. The robot lab reinforced C++ programming constructs

2. I would like to see more robotic programming in later courses

Over the academic year 2005-2006, incorporating the Fall 2005 and Spring 2006 semesters, a total of 121 students submitted this survey. Out of these 121 students, 39 of the students were Computer Science majors, 32 were Engineering majors (all types), 8 were Information Systems majors, 2 Biology, 5 Business, 1 Psychology, 6 Mathematics, 1 Education, 2 Economics, 5 Accounting, and 20 did not declare a major yet, or neglected to fill in this question. Not all students answered every question on the survey, but almost all answered the ones about robotics. The results on question 1 were as follows: Out of all students, 40 students strongly agreed, 57 students agreed, 16 disagreed, 4 strongly disagreed, and 4 wrote not applicable. This corroborated our informal discussions with both the students taking the course and the professors teaching the course regarding the utility of the robotics lab as a pedagogical tool for teaching important basic programming concepts.

The answers to question #2 were as follows: 33 students strongly agreed, 45 students agreed, 23 disagreed, and 8 strongly disagreed. 12 students answered not applicable to this question, which probably means that they do not plan on taking any further courses in Computer Science. This result is also overwhelmingly positive, showing us that students would like to see further robotic programming, although it is not as positive as question #1. A summary of percentages of positive replies, organized by student's majors can be seen in Table 1.

What is most interesting about the results in the table is that our computer science students' replies were not very different than students from other majors (especially if the not applicable results are not counted). For both questions, our computer science major response was slightly more positive than the others. However, other majors more often answered not applicable. Out of the 39 Computer Science majors, 5 disagreed to question 1 and 1 strongly disagreed to question 1, while 6 disagreed to question 2 and 3 strongly disagreed to question 2.

	Computer Science Majors	**Engineering Majors**	**Other Majors**
question #1	82.1%	71.9%	80.0%
question #2	69.2%	62.5%	66.0%

Table 1: Percentage of replies that strongly agreed or agreed

DISCUSSION

Although the results indicate a favorable response by students to the incorporation of robots into the course curriculum, the drop in percentage with respect to the positive attitude of students toward continued robotic experiences raises some questions as to why there wasn't a more positive response. Clearly a majority of the students were in favor of robotics, but reasons for the disfavor need to be addressed. In [Fagin], students cited the unavailability of robots outside of class time as an issue

with robot projects in CS1. Our CS1 robot assignment takes place over a period of two weeks. Approximately 90% of the students are able to complete the assignment within the given time frame. Should students need more time, robots are made available via a sign-out procedure. Software and the other necessary connections needed for robot use are available in one of our open laboratories.

We have tried to make hardware failure as much a non issue as possible, but as our robots age, wear and tear on sensors, and motors, along with a decline in battery charge life present problems that can often lead to student frustration. Unavoidable hardware issues, such as the unleveled floors, lead to erratic robot behavior, again contributing to student frustration. One of our solutions has been to have one of our technical staff in the lab during this assignment. Another possibility that we are considering is to maintain enough "extra" robots so that should one fail, a student need only swap the defective robot for a new one. Cost constraints have been a factor in determining how many extra robots we can provide.

Students are surveyed at the end of the semester. The fact that the assignment occurred weeks before the survey was taken can account for some of the disfavor. By the time the survey was taken, students might have forgotten their initial excitement.

For the future, we intend to do a more detailed survey. Our survey should be timed at the end of the robotics assignment so that the experience is fresh in the minds of the students. Questions directed towards what needs to be improved with the assignment most definitely should be included.

CONCLUSIONS

In this paper we discuss a robot assignment suitable for a CS1 class. The assignment illustrates several programming constructs. Students were surveyed at the end of the semester in accordance with our requirements for ABET certification. Students felt that the assignment was effective in illustrating the programming constructs taught. The surveys also indicated that most students enjoyed their robot experience. These responses were essentially the same irregardless of the student's major area of study. The survey results, however, did raise some questions as to why some students did not want to see more robotics in their future courses.

References

Fagin, B.S., Merkle, L.D., and Eggers, T., 2001 Teaching Computer Science With Robotics Using Ada/Mindstorms 2.0, *Proceedings of the 2001 annual ACM SIGAda International Conference on Ada*, September 30-October 04, Bloomington, MN

Imberman, S. 2004 A Laboratory Exercise Using LEGO Handy Board Robots to Demonstrate Neural Networks in an Artificial Intelligence Class, *AAAI 2004 Spring Symposium Series Report*, Stanford, CA. March 22-24,.

Imberman, S. Klibaner, R. A Robotics Lab for CS1, 21st Annual Consortium for Computing Sciences in Colleges Eastern Conference, October 2005

Martin, F., Robotic Explorations, Prentice Hall, 2001