Experiences with the LEGO Mindstorms^{TM} throughout the Undergraduate Computer Science Curriculum

Daniel C. Cliburn
Assistant Professor of Computer Science
Hanover College
Hanover, Indiana 47243 cliburn@hanover.edu

Abstract - **The LEGO Mindstorms Robotics Invention Systems are contemporary learning tools that have been used in a number of courses throughout the undergraduate Computer Science curriculum. However, with a few exceptions, very little information is available describing the degree to which Mindstorms projects improved (or hindered) the learning process in these courses. This paper describes personal experiences and provides practical advice for successfully incorporating Mindstorms projects into an undergraduate computer science course, as well as highlighting the types of assignments that have not worked well. Specifically, the author has incorporated Mindstorms projects into five courses: Fundamental Concepts of Computer Science (a non-major Computer Science course), Programming I, Programming II, a general programming language practicum, and the survey of Programming Languages course. These courses will be summarized, and the paper will conclude with a general discussion of lessons learned about Mindstorms use in the classroom.**

Index Terms – Curriculum, Education, LEGO Mindstorms, Pedagogy

INTRODUCTION

Since their introduction in 1998 the LEGO Mindstorms Robotics Invention Systems (RIS) [1] have captured the imagination of large numbers of computer science faculty and students. The Mindstorms have been used in a number of undergraduate computer science courses across the entire curriculum. From teaching basic programming concepts such as variables and procedures [2], to artificial intelligence [3], the Mindstorms have helped to add a new and exciting element to the structure of existing courses.

Early opinions about use of the Mindstorms in the classroom were generally very positive. However, in one of the first formal assessments of the Mindstorms as teaching tools, Fagin and Merkle [4] found that Mindstorms use in introductory programming laboratories had a negative impact on student performance, as students who attended the Mindstorms labs scored lower on exams than the students who had traditional programming labs. Fagin and Merkle also did not find that Mindstorms use in the classroom helped recruit students to the computer science major. In fact, the students in the Mindstorms sections were actually less likely to declare a major in computer science than those in the traditional sections. The author of this paper has used the Mindstorms in several undergraduate computer science courses with a mix of success and failure. The purpose of this paper is to provide what the author feels is missing from the literature: an informal discussion of the types of projects and courses for which the Mindstorms can be effective teaching tools, and those courses in which the Mindstorms should not be used.

The next section of this paper describes the efforts of other instructors who have incorporated the Mindstorms into their courses. Following this discussion, the paper will describe the author's own experiences with the Mindstorms in a number of classroom settings, and comment on their usefulness as teaching tools in these courses. The paper will conclude by sharing some general observations about how to make Mindstorms use a success in the undergraduate computer science curriculum.

RELATED WORKS

Fagin, Merke, and Eggers [2] were some of the first to incorporate the Mindstorms into introductory programming courses, developing a series of exercises for the Mindstorms with the language Ada that taught students about concepts such as sequential control, variables, procedures, selection structures, and arrays. Barnes [5] also used the Mindstorms in an introductory programming course, but instead used Java and leJOS [6]. Barnes found that the Mindstorms could serve as excellent physical models for teaching object-oriented programming principles, although he comments that the physical limitations of the RCX environment could make it difficult at times to observe good object-oriented programming style. Wong [7] also incorporated the Mindstorms into an introductory programming course, as well as in intermediate and advanced programming courses. The Mindstorms were reported to have positive impacts on student performance and enthusiasm levels in all of the courses.

Perhaps the most natural place to incorporate the Mindstorms is into an artificial intelligence course that uses an agent based approach, as implemented by Klassner [3]. The Mindstorms were found to be effective tools for such a course. LEGO robots have also been used successfully to teach other topics as well, such as concurrency [8] and peer learning [9].

The Mindstorms have even been used in other disciplines, such as chemical engineering, to teach process control [10].

The following sections discuss the author's efforts to incorporate the Mindstorms into five courses: Fundamental Concepts of Computer Science (a non-major Computer Science course), Programming I, Programming II, a general programming language practicum (entitled the LEGO Mindstorm Challenge), and a general survey of Programming Languages course.

FUNDAMENTAL CONCEPTS OF COMPUTER SCIENCE

Perhaps the author's most successful implementation of the Mindstorms in the classroom has been with Fundamental Concepts of Computer Science, a computer science course for non-majors [11]. Students typically take this course to satisfy a college wide general degree requirement in Abstraction and Formal Reasoning. The syllabus for the course follows a bottom-up model of computing, covering topics such as: digital logic design, computer organization, operating systems, networks, and then computer software. The course concludes with an introduction to programming.

The author has found that teaching programming to beginners can be very difficult. Instructors of such courses typically require students to learn simultaneously about algorithm design along with the syntax of a high level programming language. For many college freshmen, absorbing both of these concepts at the same time can be very difficult. The author wanted to teach students about algorithm design and high level language programming in separate course units. The hope was that students would find it easier to learn programming if they had previous experience with algorithm design. The LEGO Mindstorms have provided an excellent way to do this. The visual programming language of the Mindstorms is straight forward, and students (even those with little desire to learn programming) can pick it up very quickly. The visual programming interface makes it possible to program the Mindstorms without writing any code; however, students still learn about control structures. Sequential, selection, and repetition constructs form the basis of almost every Mindstorm program and can be clearly identified visually. Figure 1 shows a program written for the Mindstorms using this visual programming language. The green blocks are commands to be executed sequentially (such as Forward or Turn Right), and the purple blocks are selection constructs that allow the robots to make decisions based on sensor input. The orange blocks allow programmers to specify a series of instructions that should be repeated.

Students work in groups of two or three to build and program robots that compete in two challenges during the algorithm design section of the course. All robot building and programming are done during lab times that are devoted to this unit of the course. The first challenge requires the robots to push empty soda cans out of a square with black borders. The idea for this challenge came from a summer computing camp at which one of the author's students worked [12]. This challenge requires the students to implement sequence, selection, and repetition structures in their programs.

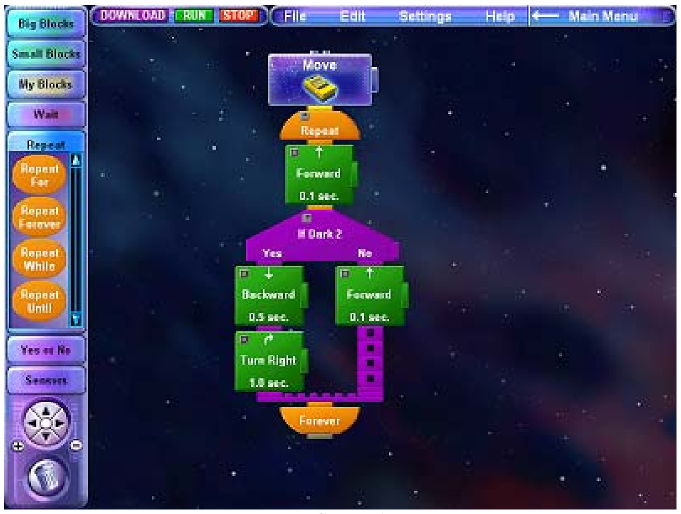

FIGURE 1
A SAMPLE LEGO MINDSTORMS PROGRAM WRITTEN USING THE VISUAL PROGRAMMING LANGUAGE INCLUDED WITH THE ROBOTICS INVENTION SYSTEMS KITS

The second challenge requires students to build a robot that can find its way out of a maze. A potential design for the maze can be found in figure 2. This challenge is more difficult than the first, as it requires students to implement nested selection structures. When a robot makes contact with a wall (as detected by its touch sensor), then it is to move right if it "sees" a piece of black tape with its light sensor, and left otherwise.

In the author's opinion, the Mindstorms challenges have been very effective teaching tools in Fundamental Concepts of Computer Science. Students come into the course looking forward to the time when they get to "play" with the LEGOs, and the fact that class time is allocated to robot building makes the burden (and sometimes frustration) of working with the LEGOs something very tolerable. The students certainly appear to prefer building and programming robots to the classroom lectures. Students have made the following comments on course evaluations, "Legos were exciting" and, "The robots were a lot of fun and very useful in learning." The Mindstorms also appear to be effectively teaching students about algorithms and control structures (as they were intended to do). Of the eleven students who have taken Fundamental Concepts of Computer Science prior to Programming I, eight received a "C" grade or better (72.7%) in Programming I; of these, five received a "B" or better (45.5%).

PROGRAMMING I

Inspired largely by the success of the Mindstorms in Fundamental Concepts of Computer Science, the author wanted to incorporate use of the robots into an introductory programming course. Specifically, it was hoped that the Mindstorms could help to teach object-oriented programming concepts. The author felt that the Mindstorms would be excellent concrete examples of "real world" objects that could serve as illustrations in lectures, and then provide the basis for the final programming assignment in the course.

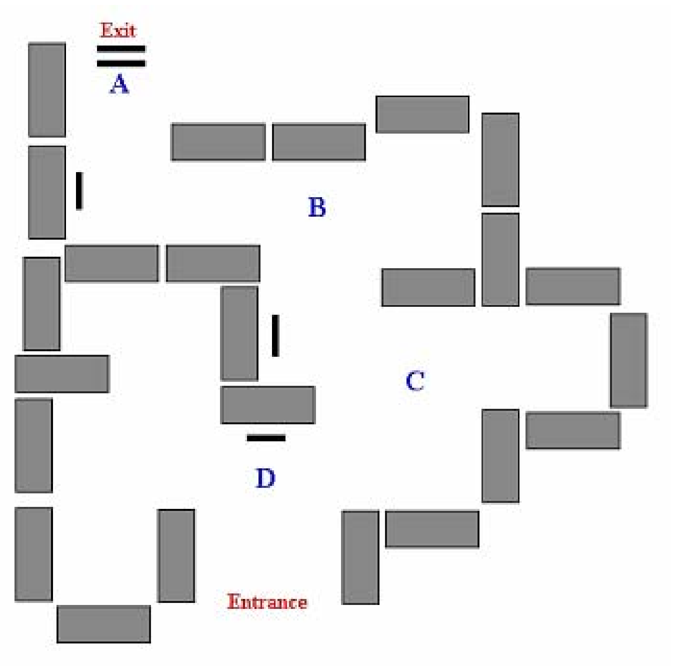

FIGURE 2
A POTENTIAL MAZE CONFIGURATION FOR THE SECOND MINDSTORM CHALLENGE IN FUNDAMENTAL CONCEPTS OF COMPUTER SCIENCE

Programming I at Hanover College has traditionally used Visual C++ as its integrated development environment (IDE). The author felt that it might be too much of a burden on the students to learn a different programming environment at the same time that they were learning object-oriented programming fundamentals and the LEGO Mindstorms. Thus, he chose not to use one of the well-known programming environments such as brickOS [13]. Instead, he chose to use a Win32 interface developed by Daniel Berger [14]. With this interface, the program executes on a PC and communicates instructions to the RCX brick in real time through the IR tower. The advantage of such an interface is that applications are not subject to the physical constraints of the RCX (like the 32K of RAM). The disadvantage is that the IR communication is slow and the robot needs to be oriented facing the tower so that communication can occur. These limitations can potentially cause problems. For instance, if the robot makes a turn causing its IR receiver to face opposite the IR tower, the program can loose contact with the RCX. The other drawback is that on occasion, communication between the RCX and IR tower is not fast enough for the robot to take action in an appropriate amount of time. For example, if the robot needs to stop when it reaches a piece of black tape, sometimes the robot will drive over the tape before the appropriate information from the light sensor is sent to the host computer. The robot will just continue to drive past the tape. The author noticed that these sorts of difficulties could be minimized by having robots drive slowly (with low power settings on the motors), but these problems could never be eliminated altogether.

For the final assignment in Programming I, students were required to work in groups of three to build and program a robot that could compete in the maze challenge described earlier (see Figure 2). This particular challenge minimized some of the potential difficulties that could be encountered with the programming interface. For instance, the maze was designed in such a way that the robot's IR receiver was never more than a 90 degree angle from the IR tower. Students were also allowed to hold the IR tower facing their robot as it traveled about the maze. The communication lag between the tower and robot was a non-issue with this challenge as well, since the robot would run into a wall and continue to just try to drive forward until the RCX was able to communicate the touch sensor information to the tower and receive appropriate instruction.

In the author's opinion, the Mindstorms were reasonably successful in this course. However, there were also many students who grew frustrated with the inconsistencies of the Mindstorms. For example, the light sensors are subject to environmental conditions. Depending on glare and ambient light in a room, the light sensors can give two different readings for the same color surface. This is a problem difficult to correct for (and impossible to fix in code). While typical computer programming is deterministic (a program should always behave the same), the robots are not. Different behaviors can be observed in trial runs through the maze, even though the robot is executing the same program. Battery power also has a tremendous impact on robot performance. The amount of time that a robot needs to rotate in one direction to make a 90 degree turn depends on the amount of battery life left. Thus, students who feel that their program is working fine one day, may return the next to find that the robot no longer works they way they thought. These issues are very irritating for students. The instructor gave the students a number of attempts at the maze to help correct for these problems, but ultimately had to grade assignments based on code and not actual robot performance.

Despite these difficulties, many students seemed to enjoy the assignment. The change of pace was welcomed at the end of the term. However, what concerned the author most about the Mindstorms was that two students who had gotten "A"s on the final assignment scored a "C" or worse on the final exam. This was problematic since the Mindstorms were supposed to teach the students about object-oriented programming, and the bulk of the final was over object-oriented programming. It became apparent to the author that in several groups, one student had done the programming, and the others built the robot. Thus the group work required on the final assignment made it possible for some students to get by without ever writing any object-oriented code before taking the final exam.

PROGRAMMING II

The author planned to carry on the Mindstorms theme the following term with the students who continued on from

Programming I to Programming II. A series of assignments was developed that allowed students to implement object-oriented programming concepts such as encapsulation, inheritance, and polymorphism. Students also gained experience with graphical user interface (GUI) development using the Microsoft Foundation Classes (MFC). These projects are described briefly here.

Encapsulation

Students were required to write a robot class and a short driver program that created a robot object, opened a data file, read some instructions from the file, and executed those instructions through the robot object. The robot class had a number of attributes that were "hidden" inside each robot object. Interaction with the robot had to be performed through its methods.

Inheritance and Polymorphism

Students wrote a new class called "Launcher" which inherits from the robot class of the first project. The launcher class redefined a number of virtual movement functions that were defined in the robot class, for movement specific to the launcher. The challenge for this assignment involved having the robot drive to a black line and launch a ping-pong ball as far as it could. The idea for this challenge was also borrowed from the summer camp at which one of the author's students had been employed [12]. Grades were assigned based on the distance that the ball was launched.

Graphical User Interface

Students designed a remote control GUI interface using MFC which could control the robot. The interface communicated with the robot in real time executing commands as the user selected them with the interface. A sample GUI remote control is shown in figure 3.

For the most part, the Mindstorms were not successful in Programming II. Given that several students in Programming I had gotten by on the last assignment relying on their partners to do the coding, the author wanted all students to write individual code for Programming II. However, the college did not own enough RIS kits so that all students could have their own. The solution the author came up with involved breaking the class into teams for robots building, but required students to write their own program for each assignment. This arrangement did not work well. Each team was given an RIS kit to share, but almost always one team member would end up building the robot himself or herself. Team members had a great deal of difficulty finding time to share the robots so that they could test their programs. Multiple students told the author that they submitted assignments without ever having tested their program on an actual robot. In the author's opinion, no student seemed to enjoy working with the Mindstorms by the end of the term. Building and testing programs on the robots just seemed to add for students an additional burden on top of learning the "real" material for the course.

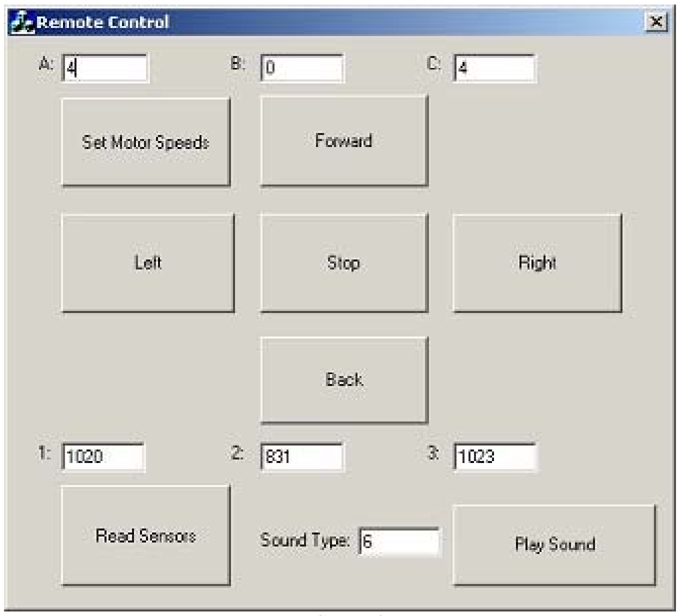

FIGURE 3
A SAMPLE GRAPHICAL USER INTERFACE IMPLEMENTING A ROBOT REMOTE CONTROL

LEGO MINDSTORM PROGRAMMING CHALLENGE

During the winter term of 2003, the author offered a one-hour credit course with the LEGO Mindstorms. The author's objectives for the course were for the students to get experience with group work, planning and implementing solutions to problems, and also to learn new programming interfaces. There were no lectures; students were required to work in small teams to create robots that could compete in various challenges. The author was teaching the course as an overload and made it clear on the first day of class that the course was going to be taught in the form of a directed study. Students groups were on their own to seek help from the instructor if it was needed. Three groups were formed (each consisting of 3 or 4 students), and each group was given an RIS kit. The instructor and students would only meet again formally for the three challenges during the term. These challenges are discussed next.

Challenge 1

For the first challenge, student groups were required to build a ping-pong ball launcher (the same challenge discussed for the inheritance and polymorphism assignment for Programming II). Students were allowed to use the visual programming language of the Mindstorms that comes with the kit. Of the three groups, only one did an excellent job of collaborating and constructing a robot that was very successful in the challenge. One group failed to do much of anything before coming to the first challenge, and one of its members dropped the course shortly thereafter.

Challenge 2

The second challenge involved designing and programming a robot that could trace a pattern and then redraw it using a marker. Groups were given a third motor (the RIS kits only come with two) that could be used to lower the marker at the appropriate time. Students were also required to use a more sophisticated programming interface, and were presented with the following options: NQC [15], legOS (now brickOS) [13], leJOS [6], and Lego/Scheme [16]. Once again, only one group actually constructed a robot that was completely successful with the challenge (the same group who excelled at the first challenge). At this point, several students began to complain that their group members were having a difficult time meeting to build robots, and had little time to learn these programming APIs on their own.

Challenge 3

The final challenge was a "Capture the Balls" contest similar to that described in [3]. The students were to build a robot that could travel around a room with black tape lines on the floor, collect ping-pong balls, and return them to their "nest". All groups were able to build a robot that was able to find at least one ball (which earned them a passing grade on the assignment); however no group really excelled at this challenge.

This was the first course the author ever taught using the LEGO Mindstorms. On the first day of class, students were very excited about the course and looked forward to getting started. However, the excitement quickly waned. One student commented on the course evaluations, "*This course took a lot more time – and added a lot more stress – than I expected*." Other students wrote, "*Should be full credit*," and "*I think the work load is that of a regular class, so either adjust the workload or give full credit.*" Again, the author's objectives for the course were for the students to get experience with group work, planning and implementing solutions to problems, and also to learn new programming interfaces. These objectives were met; however, the students seemed very frustrated throughout the course and did not particularly enjoy the experience. They clearly felt that the course was too time consuming given the amount of credit received. Whether or not the Mindstorms were to blame for the course problems is unknown; however, their use in the course was certainly not enough to make the experience positive for all of the students.

PROGRAMMING LANGUAGES

The author wanted an exciting way to teach the functional programming paradigm to his students in a Programming Languages course, and believed that the RCXLisp interface developed by Frank Klassner [18] might be an excellent solution. The author teaches Programming Languages as a general survey of the major programming paradigms. Most of the assignments in the course were game-like, and therefore somewhat interesting to the students. The Lisp/Scheme assignments, however, had consisted of writing a number of Scheme functions (that the students found very dull). With RCXLisp, the students could write functional programs that were downloadable to the RCX brick. The author felt that this assignment would be considerably more enjoyable for students than the functional programming projects of years past.

The assignment for this unit of the class was the "push the cans out of the square" challenge described earlier in the paper for the Fundamental Concepts of Computer Science course. Students were given in class time to learn RCXLisp and to build their robots. Unfortunately, not all student groups came to class ready to participate in the challenge. One group in particular was making final modifications to their robot up until the final few seconds before they were required to compete in the challenge. While some students enjoyed working with the LEGOs, the majority found building robots to be more of a burden than fun. The students probably did need to spend more time on this particular assignment than on the functional programming assignments in the past, and two students in particular obviously resented this. The author learned a valuable lesson from the experience: the Mindstorms should not add extra work for students. If they do, then the students will probably not enjoy using them in a course.

LESSONS LEARNED

Until recently [19], opinions about LEGO Mindstorms use in the classroom that have been reported in the literature are overwhelmingly positive. Generally, the Mindstorms are said to increase student enjoyment of computer science courses. While the Mindstorms can be inherently more interesting than many typical programming assignments, they do have the potential to create a greater workload for students. Thus, the advantages of using Mindstorms must be leveraged against the additional burden which they can place on students.

The following are three principles for success with the Mindstorms that summarize the author's experiences. The first principle is that an appropriate programming interface and project should be chosen carefully. If the interface and assignment are too difficult for the students to master easily, then the Mindstorms just become an additional layer of abstraction over the concept that the project is intended to teach. Even if students enjoy "playing" with LEGOs, they will not enjoy a project if it requires them to do more work. The programming interface should support the objectives of the course and be relatively straight forward to learn.

The second principle is that group sizes should be no more than two or three (ideally one). Assignments must be designed with this is mind so that all students fulfill important roles in the project's completion, and are subsequently involved in the learning process. When teaching courses such as introductory programming, it is absolutely imperative that all students learn to write code. Group work in such courses can make it possible for students to earn high project grades without having to learn the appropriate programming skills. If enough funding is available so that all students can have their own RIS kit, then this problem can be avoided. However, for most institutions, this is not feasible. Ensuring that all students develop the requisite programming skills can be difficult when you put them into groups.

The third principle is that students should be given time in class for robot building. A number of problems can occur when students take kits out of class, such as lost parts, difficulty finding meeting times, and insufficient motivation for assembling robots. These difficulties can be minimized by providing students with lab times to build their robots. When students are allowed to construct robots in class, they take a much more positive attitude towards the projects, and the out-of-class workload is reduced significantly. Another option, if class time cannot be allocated to assembling robots, is for instructors to build the robots ahead of time for the students. This is an excellent solution to a difficult problem when all students are expected to use the same design. However, this is not an option when the robot's design is part of the grading criteria.

After several years of experience using the Mindstorms across the computer science curriculum, the author has decided that they are not appropriate for every course (and in fact, are only appropriate for a few courses). They can be fun for many students, but not for everyone. They can be great examples, but only in the right contexts. The only course in which the author is still currently using the Mindstorms is Fundamental Concepts of Computer Science. Most students in this course are non-majors, so they have no (or very few) preconceived notions about computer programming and enjoy spending two weeks of class time building robots. The visual programming language provided by LEGO does an excellent job of allowing students to focus on algorithm design without the need to write code.

The primary message of this paper is that instructors should carefully consider whether or not the Mindstorms will add to their course's content before choosing to adopt them. The general conception that the Mindstorms are fun for students is usually true, but the fun quickly wears off if the projects become long and complex. This is particularly true for projects that do not really need a tool like the Mindstorms to teach the intended concept. When considering the Mindstorms for use in a course, proceed with caution.

REFERENCES

[1] "Mindstorms", http://mindstorms.lego.com/

[2] Fagin, B., Merkle, L., Eggers, T., "Teaching Computer Science with Robotics using Ada/Mindstorms 2.0", *Proceedings of the 2001 Annual ACM SIGAda International Conference on Ada*, Bloomington, Minnesota, 2001, pp. 73-78.

[3] Klassner, F., "A Case Study of LEGO Mindstorms'™ Suitability for Artificial Intelligence and Robotics Courses at the College Level", *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2002)*, Covington, Kentucky, 2002, pp 8-12.

[4] Fagin, B., Merkle, L., "Measuring the Effectiveness of Robots for Teaching Computer Science", *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003)*, Reno, Nevada, 2003, pp 307-311.

[5] Barnes, D., "Teaching Introductory Java through LEGO MINSTORMS Models", *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2002)*, Covington, Kentucky, 2002, pp 147-151.

[6] Solorzano, J., "leJOS", http://lejos.sourceforge.net/index.html

[7] Wong, K., "Teaching Programming with LEGO RCX Robots", *Proceedings of the 18th Information Systems Education Conference (ISECON 2001)*, Cincinnati, Ohio, 2001.

[8] Jacobsen, C., Jadud, M., "Towards Concrete Concurrency: occam-pi on the LEGO Mindstorms", *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2005)*, St. Louis, Missouri, 2005, pp 431-435.

[9] McGoldrick, C., Huggard, M., "Peer Learning with Lego Mindstorms", Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference, Savannah, Georgia, 2004.

[10] Moor, S., Piergiovanni, P., Metzger, M., "Process Control Kits: A Hardware and Software Resource", *Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference*, Indianapolis, Indiana, 2005.

[11] Cliburn, D., "A CS0 Course for the Liberal Arts", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2006)*, Houston, Texas, 2006, pp 77-81.

[12] Sci-Tech Ventures, http://www.scitechcamps.com/

[13] Noga, M., "brickOS", http://brickos.sourceforge.net/

[14] Berger, D., "Advanced Lego Mindstorms Programming in Visual C++", http://www.kyb.tuebingen.mpg.de/bu/people/berger/mindstorms.html

[15] Baum, D., "Definitive Guide to LEGO Mindstorms, 2nd Edition", New York: Apress, 2003.

[16] Wick, A., Klipsch, K., Wagner, M., "Lego/Scheme Compiler V0.5.2", http://www.cs.indiana.edu/~mtwagner/legoscheme/

[17] McNally, M., Goldweber, M., Fagin, B., Klassner, F., "Panel Session: Do Lego Mindstorms have a Future in CS Education?", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2006)*, Houston, Texas, 2006, pp 61-62.

[18] Klassner, F., "Enhancing Lisp Instruction with RCXLisp and Robotics", *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2004)*, Norfolk, Virginia, 2004, pp 214-218.

[19] Baum, D., Gasperi, M., Hempel, R., Villa, L., "Extreme Mindstorms: An Advanced Guide to LEGO Mindstorms", New York: Apress, 2000.