LEGO MindStorms: Not Just for K-12 Anymore

Frank Klassner, Scott D. Anderson

Department of Computing Sciences
Villanova University
Villanova, PA 19085
Frank.Klassner@villanova.edu

Department of Computer Science
Wellesley College
Wellesley, MA 02481
Scott.Anderson@acm.org

Abstract: We describe the possibility of using the Lego Mindstorms robots to support the ACM Computing Curriculum 2001, using them in lab exercises and projects for classes from beginning courses in programming to advanced courses in operating systems, compilers, networks and artificial intelligence. We first describe the limitations of the robots, both hardware and software, and some third-party programming environments that overcome some of these limitations. Finally, we describe our own work on a package of tools called MTM that eliminates most of the remaining limitations. MTM includes enhanced firmware that allows point-to-point communication and the reading of the machine state, a C++ API for programming the robots, and packages, in both Common Lisp and Java, for programming the robots and for remotely controlling them.

Keywords: ACM Computing Curriculum 2001, curriculum development, robotics, Lego Mindstorms

Introduction

The fields of Robotics and Computer Science have been fruitfully interacting at the graduate research level for many years. For example, Artificial Intelligence (AI) researchers expanded their understanding of planning in order to accommodate different robot chassis designs. Similarly, robot designers modified hardware to support scheduling for real-time operating systems. With this kind of potential for intellectual synergy, it might seem natural for educators to ask whether robotics-inspired projects could help students at the undergraduate level understand core computing concepts better by exploring them in a setting other than desktop computing. Our own (and others') experience with hands-on robotics projects in *specialized* courses has shown us that students' motivation to learn computing principles increases significantly when they have the opportunity to apply those principles in constructing robots and designing problem-solving code [4,8,10,11,13,18]. Unfortunately, until recently two factors combined to seriously limit the use of this promising tool in undergraduate computer science departments nationwide. The first was the high cost of robotics kits (often $5000 apiece in the early 1990s), and the second was the lack of a framework for comprehensively integrating robotics across several courses in a computing curriculum. These factors led to the situation where most departments either avoided the use of robotics altogether or isolated the use of a few expensive robots to one small upper-level course.

Since the middle 1990's, several manufacturers have released standardized, low-cost robot platforms. Among the more recent models are ActivMedia's Pioneer robot [1], MIT's HandyBoard and Cricket controller cards [12], and LEGO's MindStorms kit. With kit prices now under $800, and with the ACM/IEEE Computing Curriculum 2001 (CC2001) [20] committee preparing new curriculum guidelines, we believe both impediments have been removed and that the time is now ripe to address the question of how to integrate robotics effectively *throughout* the CC2001, from introductory courses through appropriate senior electives.

In this article we demonstrate that LEGO MindStorms has become a suitable platform for college students to investigate a broad range of topics within the CC2001. We selected LEGO MindStorms for four reasons:

1. **Cost:** A single MindStorms kit, with 750 construction pieces, sensors, and programmable hardware, costs

approximately $200 and thus is one quarter the cost of a HandyBoard-based robot kit and one tenth the cost of a ActivMedia-based robot kit – two of the more commonly used platforms in colleges.

2. **Flexibility:** The MindStorms platform supports a suite of reusable snap-together sensors (touch, rotation, light, temperature), effectors (motors, lights, infrared emitters), building blocks, and a programmable control unit that can serve as the basis for a wide variety of programming projects.
3. **Student Interest:** Many students have played with LEGO building blocks as children, and therefore they are intrigued with working on LEGO-based classroom exercises.
4. **Professional Curiosity:** When talking with other liberal-arts college instructors, we were struck by how often they cited the MindStorms kit as a great source for pieces to build a robot chassis to hold a HandyBoard controller card. Yet they admitted ignoring the programmable control unit that came with the kit and that contributed to over half the cost of the MindStorms package. We wondered whether the unit was as limited as conventional wisdom seemed to indicate.

To make our case in this article, we first describe the range of CC2001 areas over which we believe a candidate robotics system should support laboratory projects. After describing the basic MindStorms kit and its strengths and weaknesses, we then discuss the third-party MindStorms software tools that address some of the platform's weaknesses. Based on this evaluation we describe a set of programming environments and tools in C++, Java, and Common Lisp that we have developed to fill the holes in MindStorms' support for college-level computer science projects. Our tool set, which we call "More Than MindStorms" (MTM), has two features that distinguish it from other third-party packages designed by hobbyists for programming the MindStorms platform. The first is that they were developed to support both remote control and on-board programming of MindStorms robots in Java, Lisp, and C++. The second is that they work with an extended firmware we developed to support targeted communication between multiple LEGO robots and command-center desktops rather than the broadcast protocol supported by standard LEGO MindStorms firmware.

Robotics and CC2001 Curricula

At least seven out of the fourteen knowledge areas in the CC2001 Draft [20] contain topics that could be motivated or enhanced through robotics-oriented projects: Programming Fundamentals, Algorithms and Complexity, Programming Languages, Architecture, Operating Systems, Intelligent Systems, and Net-Centric Computing. All of these areas contain material that the curriculum committee has designated as "core" (rather than "elective") topics. In developing MTM we worked to insure that the finished product would easily support projects in these curriculum areas.

Possible pedagogic uses of robotic projects in the CC2001 curriculum include:

1. **Programming Fundamentals**: learning the differences between writing a program that controls a robot (or other external device) remotely from a desktop and one that runs autonomously on board a robot; learning how conditionals in programming languages work by using them to make a robot do different things based on the environment; learning how loops, procedures, and object-oriented methods can encapsulate common or repetitive robot actions and make programming more manageable; learning how multitasking constructs make it possible for a robot (or computer) to interleave "simultaneous" actions.
2. **Algorithms and Complexity**: appreciating the need to design time-efficient algorithms to minimize demands on real-world constraints on battery voltage and motor speed; visualizing the design of distributed algorithms in networks of cooperating robots.
3. **Programming Languages:** understanding of the concept of a virtual machine interpreting instructions within the embedded environment of a robot; comparing concurrency constructs across programming languages; exploring compiler optimizations' impact on code reduction for use with limited RAM.
4. **Architecture:** understanding the design of machine instruction sets and their support for various high-level language features; exploring the hardware issues in transporting data from sensors to a robot's CPU.

5. **Operating Systems:** experimenting with the use of concurrency in monitoring sensors shared by different tasks; appreciating the role of device drivers (or interrupt handlers) in configuring sensors.
6. **Intelligent Systems:** understanding the role of various AI techniques in improving an agent's autonomy; exploring the use of machine learning to develop control algorithms for robots; learning how to design groups of cooperating robotic agents.
7. **Net-centric Computing:** comparing the efficacy of distributed algorithms with centralized algorithms for coordinating robots; exploring the reliability of various message-passing protocols in networks of robots.

While these sample project descriptions are meant to be platform-independent, there are some assumptions in the hardware and software behind them. Any single robotic platform for this range of projects would need the following features:

1. support for multiple sensors, with multiple modalities
2. adaptable chassis for a wide variety of projects
3. some means of communicating among robots and/or desktop PCs
4. programmability in several languages commonly used in computer science education
5. a fast enough CPU, enough RAM, and adequate programming infrastructure to support moderately complex programming
6. a price that is low enough so that a small college can afford enough robots for teams of students in two or more courses at a time to conduct experiments.

We believe the MindStorms kit can, with some additional software, fulfill these needs.

The Basic MindStorms Kit

The basic LEGO MindStorms kit costs approximately $200 and contains 750 building block pieces. The kit's centerpiece is the programmable control unit, called an RCX. The RCX has a 16MHz CPU (Hitachi H8/3292 microcontroller), 32KB RAM, and houses an infrared (IR) transmitter/receiver for sending and receiving data and commands from a desktop PC or from other RCXs. The IR transceiver has a range of 15-25 feet, depending on lighting conditions and reflectivity of walls. The RCX unit has 3 input ports, 3 output ports, and a 5-"digit" LED display. Power is supplied by 6 AA batteries; rechargeable batteries usually provide a continuous-operation time of 1-2 hours, while nonrechargeable batteries usually provide a continuous-operation time of 2-3 hours. The MindStorms platform supports four kinds of sensor: touch, light, angle (rotation), and temperature. The platform currently supports two kinds of effector: motor and light.

LEGO provides two tools for programming the RCX. The first is a development environment for programming the RCX with an interface that models programming as a process of dragging puzzle pieces (representing program steps) together to produce a chain (complete program). This GUI environment supports basic programming constructs such as loops, subroutines (though not true procedure calls), and concurrency. LEGO's second programming tool is a library for generating Visual Basic programs to control the RCX.

The RCX's "operating system" is a replaceable firmware that models a primitive virtual machine. The RCX's firmware can be used in autonomous mode (the robot's behavior depends only on the program loaded into its memory) or in direct-control mode (a control program on a desktop computer broadcasts instructions to the robot). When used in autonomous mode, the firmware supports 32 16-bit global integer variables that can be shared by up to 10 threads ("tasks" in MindStorms parlance). Each thread can allocate up to 16 private variables. There is no firmware support for dynamic memory allocation. Only integer arithmetic is supported by the standard firmware. Since there is no firmware support for call stacks, nested function calls are not possible. With regard to direct-control mode, it should be noted that the RCX firmware supports only a broadcast protocol through its IR transceiver. Thus, if more than one RCX is within range of a controlling PC, the PC cannot individually address the units.

Evaluating Basic MindStorms' Applicability to CC2001

Based on our list of features a robotics system should support if it is to be applicable to any CC2001-based curricula, the basic MindStorms kit does not have good applicability. To be sure, there are strong points in the platform's favor. The kit's building block supply does support a very wide variety of chassis designs. The kit's price makes it feasible for most Computer Science departments to consider buying enough kits to support laboratory team projects in more than one course per semester (i.e. 6-12 kits). While it is true that the MindStorms' platform only supports three input ports, this does not limit the platform's sensory capability as much as it would seem. In our experience, it is possible to "stack" more than one touch or light sensor on the same port and to interpret the port's measurements appropriately. That is, by configuring a port to be in "raw mode," one will obtain an integer value between 0 and 1023 that can be analyzed through trial and error to determine when one, two, or no sensors on the same port have been activated. In some situations, one can even determine which particular sensor in a stack has been activated. With one minor misgiving, we therefore believe that the MindStorms sensor suite does cover a reasonable range of modalities for its price. The minor quibble we have is that the rotation sensor (aka "angle sensor") is crucial to any robust solutions to the robotic navigation problem, yet it is not included in the basic MindStorms kit. We highly recommend that anyone contemplating use of MindStorms for college robotics projects add this sensor to their development budget!

The basic MindStorms kit, however, does have serious deficiencies with regard to CC2001 curricula. The inability to program with commonly used languages such as Java, C/C++, and even Lisp means that educators seeking to adopt the platform face the overhead of familiarizing students with new languages that do not support some of the features in the more traditional languages. The platform's 32KB of onboard RAM, while not small compared to more expensive platforms, is still a significant barrier to those who would want to run large "intelligent agent" software packages, regardless of source language. The basic platform's lack of support for floating-point arithmetic, while not fatal, represents a serious challenge to educators wanting to introduce students to the mathematics behind robot navigation. The basic MindStorms' broadcast IR protocol is a major obstacle for exploring networks, distributed algorithms, and message-passing protocols within CC2001 curricula. Despite these problems, we believe that with some supplementary software, the Lego Mindstorms robots can be helpful in teaching the CC2001 curricula.

Related Work for Improving RCX Applicability

The standard LEGO Mindstorms RIS (Robotics Invention System) comes with a CD of Microsoft Windows software that provides a graphical programming environment based on a metaphor that a program is a set of linear sequences of "jigsaw puzzle" pieces representing commands (turning a motor on), sensor readings (is the light-sensor reading less than 50?), and control constructs (loops). This language is compiled into bytecodes that run on firmware that is installed on the RCX by the CD. The RIS language is simple enough for children, but very limited in expressive power. Therefore, a number of hobbyists have replaced this graphical programming language with standard languages such as C, Ada, Scheme and Java.

A subset of the C programming language is available thanks to Dave Baum's implementation of NQC: Not Quite C [3]. It provides a conventional, textual language that compiles to the same bytecodes that the standard RIS language compiles to. Thus, NQC programs can be downloaded along with RIS programs, all accessible via the buttons on the RCX. Because it targets the standard MindStorms firmware, NQC suffers from some of the same limitations: there is only integer arithmetic (the firmware lacks floating point), the run-time stack is very limited. All variables in NQC are 16-bit integer, and the number is severely limited: the RCX firmware allows up to 32 global variables. The NQC language includes inline functions, including arguments, but those functions cannot have return values. NQC also allows subroutines--which are not inline, thus saving scarce RAM--but subroutines cannot take parameters and cannot call other subroutines. The RCX is limited to 8 subroutines. NQC extends C in adding "tasks," which run as concurrent threads on the RCX hardware; the RCX hardware supports up to 10 concurrent tasks. NQC is free software available under the Mozilla public license, and runs under Mac OS and Windows.

A subset of the Ada language is available for the RCX, thanks to an implementation of a cross-compiler by Barry Fagin [8]. The Ada language is converted to NQC by a program called "ada2nqc." The program is then compiled and downloaded to the RCX just like any normal NQC program. The Ada2NQC translator also checks

that the program doesn't use any features that NQC or the RCX hardware cannot support. An Ada compiler is also required to validate the Ada program. The software comes with a library of Ada code describing the RCX API so that the programs use of the sensors, effectors and other features can be validated by the Ada compiler, prior to translation to NQC. Like NQC, there are a number of restrictions: the only datatype is integer, procedures may only appear within the main procedure, and there are no separate compilation, standard packages, or user-defined types.

Wick, Klipsch and Wagner [19] implemented a Scheme compiler targeting the RCX firmware. As with the NQC and Ada projects, they implemented primitives to create threads, read the sensors and control the effectors. Their compiler is implemented using Chez Scheme or Petite Chez Scheme, running under Windows. As with the previous languages, the programs are limited by the scarcity of variables and the lack of subroutine calls and floating point arithmetic.

LabVIEW is a industry-standard graphical programming language, and ROBOLAB is an adaptation of LabView for the RCX, developed by Chris Rogers [16] for use by beginning robotics students on either the Mac OS or Windows operating systems. ROBOLAB is superficially similar to the basic Mindstorms GUI language, but it is more sophisticated in that it does allow RCXs to send asynchronous messages back to the desktop computer.

Software that replaces the RIS firmware is "legOS," [14] a multi-tasking operating system running on the RCX hardware and providing, of course, an API to all the RCX sensors and effectors. To program in legOS, one uses a C cross-compiler running on a host compiler, so it is similar to NQC in that the language is C, but the target is quite different. Essentially, legOS provides a library of system calls for use in a C program for the RCX. The legOS software provides memory management, call stacks, threads and other high level language features, though of course it is still subject to the memory constraints of the RCX. Like NQC, legOS is free software available under the Mozilla public license, and runs under Linux, Unix, and Windows.

The "leJOS" software package [17] is an implementation of part of the Java Virtual Machine (JVM), running on the RCX hardware. LeJOS comes with firmware that replaces the original RIS firmware; the new firmware implements many of the core Java classes, including integer and floating point arithmetic, memory allocation, strings, threads, and exceptions. It also, of course, provides classes and methods to interact with the sensors and effectors. Even more than with NQC, programming in leJOS is easy, since it is plain Java plus an extra "import" statement. Debugging is easier because exceptions are reported on the RCX's LCD, specifying the method signature and the exception class. The primary disadvantages of leJOS compared to NQC or LegOS are memory usage and performance. It was originally developed by Jose Solorzano as an Open Source project hosted by SourceForge.net. It is free software that runs under Linux, Unix, and Windows.

This wide variety of programming languages and environments mitigates the basic MindStorms platform's language deficiency with respect to the CC2001. In addition to developing support for programming the RCX in common languages, hobbyists have also worked on special-purpose libraries and tools. The NQCIPC package by Brian Connors [6] adds semaphores to the NQC language, making it more feasible to use the RCX to experiment with concurrency issues in an operating systems course.

The usefulness of the RCX as a platform for exploring distributed programming has been improved with the release of Denis Cousineau's "RCX_command" tool [7] and Ross Quillan's Perl module "LEGO::RCX." [15] RCX_command is a graphical environment for writing programs that not only allow a desktop PC to remotely control an RCX but also to synchronize execution of other programs on the desktop with data transmitted by the RCX. Programs for RCX_command are written in the PRO-BOT language, which is similar to NQC in terms of the number and kind of supported RCX commands and control structures. The LEGO::RCX module provides a collection of Perl subroutines that send individual commands to RCX standard firmware over an IR transceiver. The commands are similar to those supported in NQC and include the ability to poll for individual RCX state variable values. Since these subroutines can be embedded within a larger Perl program, the module makes it possible to experiment with simple distributed systems involving the interleaving of a desktop PC's analysis with remote control of a data-gathering RCX. The module's implementation in a relatively common programming language makes it easier to integrate into an academic curriculum than RCX_command.

The MTM Tool Set

The work described in the previous section significantly improves the applicability of the Lego MindStorms robots to CC2001-based curricula. However, a few shortcomings remain. This section describes those difficulties and the solutions we have developed to address them. We have named the collection of solutions "MTM" for "More Than Mindstorms."

The RCX has a hard limit of 32KB of memory, which presents a serious restriction for instructors who want to give students experience in working with the large real-time schedulers and planners that are often used in robotics. One possible solution to this problem is to offload most of the computation to a desktop computer and to have the desktop application do the planning and scheduling and then send commands as necessary to the RCX to control its actions. To do this, the desktop application needs an API for sending action commands (as opposed to programs) to the RCX, and the RCX needs software that will listen for action commands and execute them. The MTM tool set includes several ways to use this approach, which we'll discuss later.

A second shortcoming is that the MindStorms platform does not support point-to-point wireless protocols over its built-in IR port—all IR communication is broadcast. In other words, if you have three RCXs, two of them cannot exchange messages without the third receiving them and possibly altering its behavior. Furthermore, if you have a desktop application controlling the several robots of a team, it needs a way to address them individually. The standard Lego firmware, however, is limited to using a broadcast protocol that does not support such targeted message-passing, and therefore, so is software based on the Lego firmware, such as NQC. Since firmware is responsible for parsing IR messages for the MindStorms robots, any solution to this problem requires development of a modified firmware that checks for "source" and "target" fields in a message before passing the message along to application software on the RCX. MTM includes such modified firmware, which we call the Mnet firmware.

While the RCX makes a good platform for projects in the areas of Operating Systems, Computer Architecture, and Programming Languages because of third-party support for C/C++ and Java, the lack of support for Common Lisp limits its usefulness for projects in the area of Intelligent Systems (IS). IS is a knowledge area of the CC2001 that covers Artificial Intelligence and related fields, including topics such as search, knowledge representation, agents, machine learning, planning and robotics. While research and development in these areas can and is being done using languages other than Lisp, Lisp has long been important in IS and the lack of ability to control the RCX using Lisp is an unnecessary impediment to IS courses. MTM includes two ways to use Common Lisp with the RCX: the desktop application can run a Common Lisp application and remotely control the robot, or a control program can be written in a subset of Common Lisp and downloaded to the RCX.

A related issue occurs with respect to machine learning (ML). Machine learning algorithms often require a way to get a vector of state information, and the different elements of the state vector must be measured more or less simultaneously. If we assume that the ML algorithm is running on the desktop, it can construct a state vector by querying the RCX for the motor settings, sensor settings and values, button state, clock values and so forth, thereby building up the state vector by a sequence of IR messages to the RCX. We have identified as many as 60 queries that need to be made to construct a state vector. Each IR message and response takes about 7ms, so it can take quite a while to build up the state vector, possibly violating the assumption of simultaneity. Furthermore, if one such state vector is constructed soon after another, the last state value of the previous vector may be closer in time to the first one of the second state vector than it is to the first one of its own vector. One possible solution is to enhance the firmware so that a single instruction will cause the firmware to transmit all 60 state values in one long burst of IR. This solution, which in included in the Mnet firmware, makes it faster and easier to construct data important for many ML algorithms

The "remote control" paradigm that we described earlier as a possible solution to the limitations of memory and computing power onboard the RCX itself has a disadvantage common to centralized systems: the central computer (the desktop application) may not be able to respond quickly enough to environmental stimuli. An alternative solution is to have the desktop application create, compile and download a control program for the RCX that is appropriate to the current situation and goal. MTM includes tools that allow on-the-fly compilation of dynamically generated source code targeted to the RCX.

We have developed the MTM toolkit to address these shortcomings and to complement the capabilities of the third-party programming environments discussed in the previous section. The MTM toolkit contains the following:

1. An enhanced version of the basic LEGO MindStorms firmware (Mnet) that supports the addition of "source" and "target" byte fields in message passing, thereby allowing point-to-point communication when teams of RCX robots are in use. Furthermore, Mnet supports an instruction that assigns a network identifier to an RCX and another instruction that restricts what other RCXs a given RCX will process messages from. These instructions make point-to-point communication easier to use. The Mnet firmware also supports a bytecode for requesting all of the state information of an RCX [5], which is useful for monitoring and statistical analysis as well as the machine learning applications mentioned above. The Mnet firmware is backwards compatible with the standard Lego firmware, which means that software based on the Lego firmware (such as NQC), could easily be adapted to use Mnet.
2. A new API for LegOS that uses C++ and supports targeted communication with the RCX [9], thereby making the enhanced firmware immediately useful to those developing in C or C++. This API is implemented by C++ classes and methods to control the robot, such as setting motor power and querying sensor values. Of course, the software also supports standard OS services such as creating threads and managing memory (though with much less capability than most operating systems). The low-level software that manages the IR communication allows the same point-to-point communication as the enhanced firmware. Finally, the API allows for a single method that causes a vector of state information to be transmitted via the IR port.
3. A Common Lisp package for remotely controlling RCX robots. The package is a library of functions and macros to open connections to the RCX, setting the type and mode of sensors, querying the values of sensors and the state of buttons, and controlling motors and the tone generator. Whenever possible, functions are named like those in NQC. The package is compatible with both the original Lego firmware and with the enhanced MTM firmware, and so there are functions to assign numbers to an RCX, allowing messages to be individually targeted. Thus, a Common Lisp application running on the desktop can. control RCX units using all the power of Common Lisp, with no practical memory limitation.
4. A package of Common Lisp functions that can be used to define programs that can be compiled and downloaded to the RCX. Thus, you can program the RCX in Common Lisp, just as you would in NQC or leJOS. Furthermore, because of the nature of Common Lisp, programs can be created on the fly by the desktop application and then compiled and downloaded to the RCX. An artificial intelligence or machine learning application could determine particular code to be executed, based on prior experience, and download the code to the RCX when needed. The programming language is just a subset of Common Lisp that can be compiled and downloaded in this way, so this could be viewed as "not quite Common Lisp." The subset of Common Lisp includes the standard control structures, functions to control the RCX, and the ability to define macros and threads. Of course, the object language for this cross-compiled Lisp is the Mnet firmware.
5. A Java package (class library) for remotely controlling RCX robots via methods similar to NQC functions. The package is compatible with the MTM firmware and is backward compatible with standard RCX firmware. As with the Common Lisp library, this software allows a desktop application to control the RCX using all the power of the Java language and without any practical limitations on memory.
6. Native-language compilers for a subset of Java that is targeted to the enhanced firmware and that allows on-the-fly compiling and downloading of code to the RCX. This compiler is similar in spirit to the Common Lisp compiler in item 4. Both of these compilers allow a desktop application to do more than just remotely control the RCX; it can create and download controllers that are customized to current conditions and which have the responsiveness that a desktop application lacks due to communication delays.

Conclusion

While the LEGO MindStorms kit is still somewhat limited in RAM and number of sensors and effectors, we believe that the third party programming environments, supplemented by the MTM toolkit, address the software limitations, thereby allowing the use of these inexpensive, durable robots in college-level courses. Indeed, the robots have been used for classes at Villanova in Artificial Intelligence and in Operating Systems. Pamela Lawhead has told us that they are used in her introductory programming classes the University of Missisippi. Another aspect of our project is the development of lab exercises using the MindStorms robots to illustrate and explore computer science concepts from across seven of the core areas of the ACM/IEEE's Computing Curricula 2001. These lab projects are currently under development and would not have been possible without the support provided by the third party programming environments and the MTM tools.

Acknowledgments

The authors gratefully acknowledge the programming work done by their students Joseph Chojnowski, Andrew Chang, Nicholas DiPasquale, Drew Jacobs, David Immordino, Mary Chang, and Erika Symmonds.

LEGO MindStorms is a trademark of the LEGO Group, which does not sponsor, authorize, or endorse any of the third-party work cited in this article. None of the authors of this article have any financial relationship with the LEGO Group.

This material is based upon work supported by the National Science Foundation under Grant No. 0088884. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] ActivMedia Robotics, Available at http://www.activrobots.com/

[2] Bagnall, Brian, *Core LEGO MINDSTORMS Programming: Unleash the Power of the Java Platform*, Prentice Hall PTR; ISBN 0130093645.

[3] D. Baum, "Not Quite C (NQC)," Available at http://www.enteract.com/~dbaum/nqc/index.html

[4] R. Beer, H. Chiel, and R. Drushel, "Using autonomous robotics to teach science and engineering," *Communications of the ACM*, Vol. 42 (6) June 1999, pp. 85 - 92.

[5] A. Chang, "A new serial communications protocol for LEGO MindStorms RCX 1.0," MS Independent Study, Department of Computing Sciences, Villanova University. Villanova PA. 2000.

[6] B. Connors, "Not Quite C IPC (NQCIPC)" Available at http://www.geocities.com/ResearchTriangle/Station/2266nqcicp/nqcicpdoc.html

[7] D. Cousineau, "PRO-BOT 2000." Available at

http://prelude.PSY.UMontreal.CA/~cousined/lego/4-RCX/PRO-BOT/index.html

[8] B. Fagin, L. Merkle, and T. Eggers, "Teaching basic computer science concepts with robotics." In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (2001).*

[9] C. Hook, "Using and extending LegOS, an open source alternative to the LEGO MindStorms firmware." MS Independent Study, Department of Computing Sciences, Villanova University. Villanova PA. 2001.

[10] F. Klassner, "A case study of LEGO MindStorms' suitability for artificial intelligence and robotics courses at the college level," in *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (2002).*

[11] D. Kumar, and L. Meeden, “A robot laboratory for teaching artificial intelligence,” in *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education (1998).*

[12] F. Martin, “The MIT HandyBoard Project, 2000,” Available at http://lcs.www.media.mit.edu/groups/el/Projects/handy-board/

[13] N. Mazur and M. Kuhrt, “Teaching programming concepts using a robot simulation,” Journal of Computing in Small Colleges, Vol. 12, (5) 1997, pp. 4-11.

[14] M. Noga, “LegOS,” Available at http://www.noga.de/legOS/

[15] R. Quillan, “RCX.pm,” Available at http://members.home.com/quillan/lego/rcx.pm.html

[16] C. Rogers, “RoboLab,” Available at http://ldaps.ivv.nasa.gov/

[17] J. Solorzano, “LeJOS,” Available at http://lejos.sourceforge.net/

[18] L. Stein, Interactive programming: revolutionizing introductory computer science,” ACM Computing Surveys 28A(4), December 1996.

[19] A. Wick, K. Klipsch, and M. Wagner, “LEGO/Scheme compiler, v. 0.52,” Available at http://www.cs.indiana.edu/~mtwagner/legoscheme/