

## **An Introduction to the Lego Mindstorms**

**Daniel Cliburn**  
**Assistant Professor**  
**Mathematics and Computer Science**  
**Hanover College**  
**Box 890**  
**Hanover, IN 47243**  
**(812) 866 – 7286**  
**[cliburn@hanover.edu](mailto:cliburn@hanover.edu)**  
**<http://www2.hanover.edu/cliburn>**

### **Abstract**

This tutorial provides an introduction to the LEGO Mindstorms Robotics Invention Systems (RIS), contemporary learning tools that have been used to teach a number of concepts in technology related courses. First, a general overview of the Mindstorms and their history as educational tools is presented. Next, the process of building and programming robots using the visual programming language provided with the RIS kits is described, focusing on how the general algorithmic constructs of sequence, selection, and repetition can be implemented with the Mindstorm programming language. This tutorial concludes with a general discussion of how to successfully incorporate the Mindstorms into a course, and provides an overview of resources available to make teaching with the Mindstorms a success.

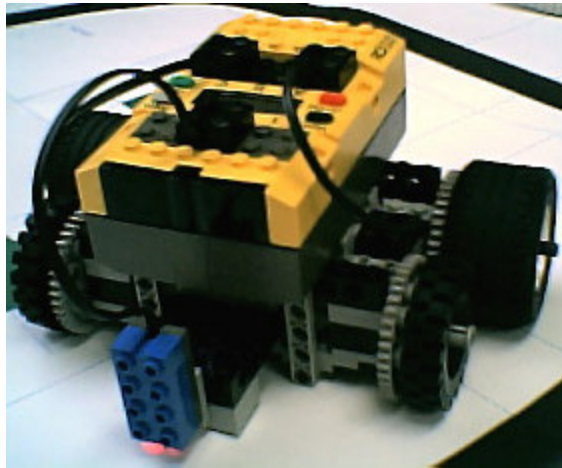
### **Introduction**

LEGO first introduced the Mindstorms in 1998, with version 1.0 of the Robotics Invention System (RIS) kits [1]. Since that time, the Mindstorms have caught the interest of numerous hackers, programmers, and even teachers. Wolz [2] was one of the first to incorporate the Mindstorms into a computing course, using them to teach students in an introduction to programming class about good design and project management concepts. Fagin, Merkle, and Eggers [3] were also some of the early users of Mindstorms in the classroom. They developed a number of instructional examples using the language Ada and the Mindstorms, which teach students about programming topics such as sequential control, variables, procedures, selection structures, and arrays. The Mindstorms have since been used in computer science sources to teach concepts from object-oriented programming [4] to artificial intelligence [5]. They have even been used to introduce children to the exciting world of computing [6]. The LEGO robots are often said to increase the interest of students in computing courses, and they make class work more fun and exciting.

The purpose of this paper is to serve as an introduction to the LEGO Mindstorms for faculty members who are interested in incorporating these robots into their courses. The Mindstorms can be a great way to introduce programming and computing concepts to students in a fun and nonthreatening manner. The next section of this paper will discuss the building of LEGO robots, followed by a tutorial on programming the Mindstorms. The final section will introduce a number of resources, as well as provide practical advice for making Mindstorms use a success in the classroom.

## Building Robots

The central component of any LEGO Mindstorm robot is the RCX brick, which contains a small computer based on the Hitachi H8 series microprocessor and 32K of RAM [7]. The RCX unit (the yellow block that can be seen in Figure 1) has three input ports (labeled 1, 2, and 3) to which sensors can be attached, and three output ports (labeled A, B, and C) through which motors can be connected. The RIS kits come standard with two touch sensors, one light sensor, and two motors. Angle and temperature sensors can be purchased separately.



**FIGURE 1**

Building robots is straight forward, and the RIS kits come with a “constructopedia” that contains directions for the assembly of a number of different designs. In my courses, I typically have students build some version of the “roverbot”, a robot designed to drive in relatively straight lines and make sharp turns. These robots are good for navigation tasks. A fully constructed robot with a light sensor attached to port 2 is shown in figure 1.

## Programming the RCX

The RIS kits contain an easy-to-use visual programming language for communicating instructions to the RCX. A number of alternative programming interfaces have been developed [7, 8, 9, 10] by programmers outside the LEGO Corporation. These will be discussed later. For now, we will use the visual programming language provided by LEGO to instruct our robots.

When the LEGO software is started, users will be presented with the screen shown in figure 2A. To begin programming, click on the options “Program” and then “Freestyle” (from the screen that comes up next). This will bring up the visual programming interface shown in figure 2B.

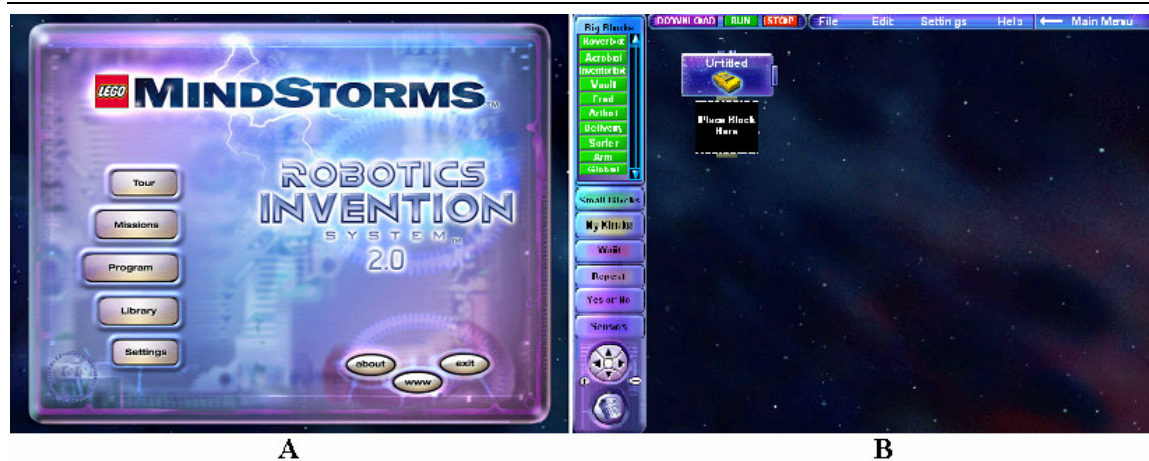


FIGURE 2

I use the Mindstorms in an introductory computer science course to teach non-majors about the general algorithmic constructs of *sequence*, *selection*, and *repetition*. The remainder of this programming tutorial will focus on designing a robot's program that introduces these three concepts. As a concrete example, let us say that we wanted a robot to continually drive around a track – defined by a black ring on a white piece of paper (such a piece of paper is actually included with the RIS kits). Our robot would need to be able to drive forward until it *sensed* with the light sensor (connected to port 2) that it was no longer over the black line. The robot would then need to turn towards the inside of the circle a small amount (for the sake of this example, we will assume that the inside of the circle is always to the robot's left). The robot would then repeat the described process.

To begin our program, we need a repetition structure that could continually instruct the robot to drive forward, looking for white paper. Clicking on the "Repeat" button on the left hand side of the programming interface exposes a number of options. We want the "Repeat Forever" block. This block can be clicked and dragged so that it is attached to our program in the black programming area on the right. Next, we need an instruction that tells the robot to drive forward. This command can be found in the "Big Blocks – Roverbot" section of the interface on the left. Click and drag the "Forward" block so that it is inside of our repetition structure. By right clicking on this block, we can adjust the amount of time that the robot drives forward before it sequentially does its next task. We really want this to be a small amount of time, so set it to 0.1 seconds. Now we need to tell the robot to check for white paper (which will notify the robot that it is driving off the black line, and needs to turn left). To do this, click on the "Yes or No" button on the left, and drag the "Yes or No" block so that it is connected to our program after the "Forward" block, but inside our "Repeat Forever" loop. Right click on the "Yes or No" block and tell the robot that we want it to: 1) use a light sensor; 2) attached to Port 2; 3) look for a brightness event; and 4) use automatic settings. You will notice that the "Yes or No" block has two paths, one for if the condition is true and the other for if the condition is false. In the yes path, we want to tell the robot to turn left, so select the "Left" block from the Big Blocks – Roverbot section, and set the time to 0.5 seconds. On the "No" path, place a "Forward" block that executes for 0.1 seconds. Figure 3 shows the complete program.



FIGURE 3

The wonderful thing about this programming interface is that students can visually identify the algorithmic constructs they are using in their programs. Green blocks represent instructions that are to be executed *sequentially*; the purple blocks are *selection* structures, and the orange blocks are *repetition* structures. This is a programming language that anyone can learn. The interface is very nonthreatening for students who are computer phobic, and can be a lot of fun.

To get our robot to execute the program, it must first be downloaded to the robot using the IR tower (which needs to be connected to the computer through the USB port). Make sure that the RCX is on and facing the tower, then click the download button at the top of the interface to send the program to the RCX. When the program is downloaded, you will hear a series of beeps. To run the program, press the green “run” button on the RCX.

If you try out the program exactly as I have it above, you will quickly notice that the RCX will turn left much too long if told to go 0.5 seconds. A smaller number, like 0.2, is probably better, but will not make the robot work every time. It is likely that the robot will eventually “overturn” and head into the middle of the track. This is actually a good learning experience for the students. To fix the problem, we can use a nested repetition structure inside our selection structure. Try and add a “Repeat While” structure inside the “Yes or No” block that makes the robot turn left, in 0.1 second increments, as long as the light sensor detects brightness. Thus, when the robot detects the dark of the black line, it will stop turning and just go straight. This should make the robot work every time. The complete program can be found in figure 4.

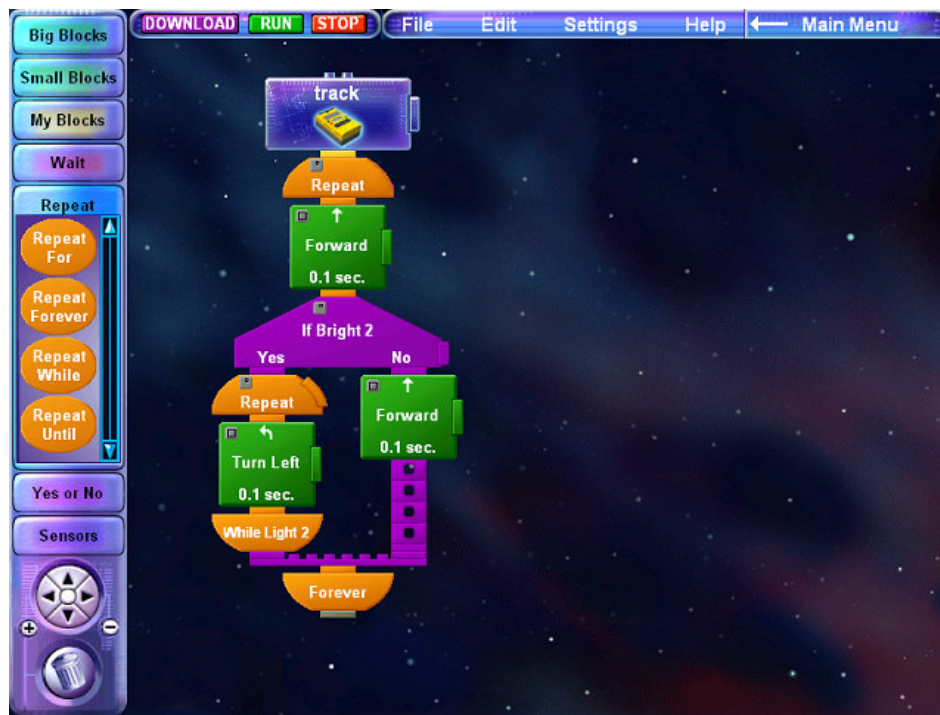


FIGURE 4

The LEGO software included with the kits contains interactive tutorials that teach students the basics of programming the RCX unit using this visual programming language. To access these tutorials, simply choose “Missions” from the main screen (shown in figure 2A), then “Training Missions” from the screen that comes up next. These training missions make great laboratory activities to get students started with the Mindstorms.

### LEGO Mindstorms Resources

The LEGO Mindstorm Robotics Invention Systems 2.0 kits can be purchased through target.com and eBay for about \$200. A criticism of the RIS kits is that they do not contain a large number of sensors (and do not even include the angle and temperature sensors). Extra Mindstorms parts can be purchased through LEGO Education [11]. The extra sensors can make it possible for robots to perform more interesting tasks.

As mentioned earlier in the paper, a number of other options are available for programming the Mindstorms, in addition to the visual language provided with the RIS kits. If you are interested in teaching a programming class with a significant Mindstorms component, you may prefer a language that more closely resembles what is already taught in your course. Dave Baum has developed a C-like language (called NQC) that can be used for writing Mindstorms programs. He has authored a text [7] which both describes NQC and several robot designs not found in the constructopedia. A Java like programming environment is also available [9] for the Mindstorms, as well as a development environment for C and C++ using the gcc and g++ compilers [8]. There is even a LISP programming environment for the RCX [10]. Patterson-McNeill and Binkerd have published a detailed listing of LEGO Mindstorm programming interfaces [12]. Erwin’s text [13] contains a number of additional interesting robot designs and projects.

## Using the Mindstorms in a Computing Course

The Mindstorms often have great appeal to both students and faculty at the beginning of a course. Playing with LEGOs can be quite fun. A few years ago, many faculty members felt that the Mindstorms were the next great tool for teaching computer science. However, this excitement has begun to wane somewhat, as many instructors have observed that the Mindstorms do not have the positive impact on student performance in computing courses that many had hoped [14]. Incorporating the Mindstorms into a course can also be more difficult than it initially seems. The robots do not always perform in exactly the same manner, even when executing the same program. Environmental factors such as friction or battery power can cause the motors (and wheels attached to them) to behave differently each time a program runs. For instance, the time it takes a robot to make a 90 degree left turn on carpet, is likely different than the time required on a table top. The light sensors are also difficult to calibrate. Shadows and glare from light sources can make a robot's light sensors provide very different readings across the same surface. These issues can frustrate students, and should be taken into account when planning assignments for computing courses. I usually give students a number of trial runs and just count the best performance, when their robots are competing in graded challenges. This seems to ease the students' anxiety somewhat, but the inconsistencies of the robots can still be very aggravating.

As mentioned earlier in this tutorial, I use the Mindstorms in a computer science course for non-majors, to teach introductory programming concepts. The Mindstorms are very effective teaching tools for a course such as this. Most students come into the class with no background in computer programming, and the Mindstorms can make learning this skill fun. I give the students lab time over a three week period to build and program their robots, so the Mindstorms do not add an out-of-class burden for the students. I have also tried to incorporate the Mindstorms into introductory programming courses for computer science majors. In this course, the Mindstorms were not as effective. In order to continue covering the same amount of material as had been covered in previous semesters, I was not able to give the students as much lab time for robot building, as in the non-major course. Thus, the students in the introductory programming courses had to work on their robots outside of class. This became difficult for a number of reasons. First, students had to work in groups since it was not economically feasible for each student to have his or her own kit. The students often had difficulties finding times when they could all meet. Typically, one student ended up doing all of the robot building. A number of parts also became lost when the students took the RIS kits to their rooms. After several semesters, enough parts can go missing from kits that they need to be replaced.

An approach that has been tried in other introductory programming courses [12], which has been successful, is to introduce programming early with the Mindstorms using the visual language provided with the kits. Later, students move on to a more powerful language like NQC. This approach has the advantage that students can reuse robots from earlier portions of a course while they learn a new programming language, allowing them to focus on the language itself and not robot building.

The LEGO Corporation is planning to replace the Robotics Invention Systems 2.0 kits with a new and improved model, the Mindstorms NXT, in the fall of 2006. The NXT kits will contain 3 servo motors, 4 sensors (light, touch, sound, and ultrasound), and support wireless Bluetooth technology, for a price of around \$250 [1]. This could have several ramifications for the RIS

kits. In some sense, the technology is becoming obsolete. However, the RCX powered robots will still be capable of teaching the topics discussed in this paper; and the number of third party programming options available for the NXT will not be as plentiful as for the RCX (for at least a few years). The cost of RIS kits will likely go down once the NXT kits become commercially available too, so if price is a consideration, the RIS kits will still be an excellent option.

Robots add a lot to the classroom. They can provide motivation for students who do not have much interest in technology. They can serve as concrete examples of programming concepts that may otherwise seem very abstract. Robots can also offer a change of pace in a course that may sometimes get bogged down in technical details. However, developing good Mindstorms projects takes time and careful consideration. For students to enjoy working with LEGO robots, instructors need to plan appropriate challenges and provide ample opportunities for students to be successful.

## References

- [1] LEGO.com Mindstorms, <http://mindstorms.lego.com>.
- [2] Wolz, U., "Teaching Design and Project Management with Lego RCX Robots." *Proceedings of the 32<sup>nd</sup> SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2001)*, Charlotte, North Carolina, 2001, pages 95-99.
- [3] 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, pages 73-78.
- [4] Barnes, D., "Teaching Introductory Java through LEGO MINSTORMS Models." *Proceedings of the 33<sup>rd</sup> SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2002)*, Covington, Kentucky, 2002, pages 147-151.
- [5] Klassner, F., "A Case Study of LEGO Mindstorms<sup>TM</sup> Suitability for Artificial Intelligence and Robotics Courses at the College Level." *Proceedings of the 33<sup>rd</sup> SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2002)*, Covington, Kentucky, 2002, pages 8-12.
- [6] Weisheit, T., "Using practical toys, modified for technical learning: A class aimed at increasing children's interest level in computer science." *ACM Crossroads*, Volume 10, Issue 6, 2004.
- [7] Baum, D., "Definitive Guide to LEGO Mindstorms, 2<sup>nd</sup> Edition", New York: Apress, 2003.
- [8] Noga, M., "brickOS", <http://brickos.sourceforge.net/>.
- [9] Solorzano, J., "leJOS", <http://lejos.sourceforge.net/index.html>.

- [10] Klassner, F., "Enhancing Lisp Instruction with RCXLisp and Robotics." *Proceedings of the 35<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2004)*, Norfolk, Virginia, 2004, pages 214-218.
- [11] LEGO Education, <http://www.legoeducation.com>.
- [12] Patterson-McNeill, H., Binkerd, C., "Resources for Using the LEGO Mindstorms." *The Journal of Computing Science in Colleges*, Volume 16, Number 3, 2001, pages 48-55.
- [13] Erwin, B., "Creative Projects with LEGO Mindstorms." Boston: Addison-Wesley, 2001.
- [14] Fagin, B., Merkle, L., "Measuring the Effectiveness of Robots for Teaching Computer Science." *Proceedings of the 34<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003)*, Reno, Nevada, 2003, pages 307-311.