Implementing Applications on Small Robots for Educational Purposes: Programming the LEGO Mindstorms

Theodore H. Kaskalis, Vasilios Dagdilelis, Georgios Evangelidis, and Konstantinos G. Margaritis

***Abstract*-- Commercially available robotic construction kits become more and more accessible to the modern classroom, as regards their financial cost. Their application in a wide educational framework gradually acquires a more realistic base. Many researchers have already considered the demanding task of the programming of such robotic kits. As a result, a great number of programming tools have already appeared. Concentrating on the LEGO Mindstorms product, we attempt to identify and present the available approaches. Our main interest, however, is their potential educational exploitation. This comparative presentation leads us to useful results, in order to identify the appropriate programming tool for a given course curriculum.**

***Index Terms*-- Robotic Construction Kits, Programming Environments**

I. Introduction

DURING the last decade a great number of "robotic construction kits" appeared in the scientific and industrial field [7]. These simple robotic constructions aim to become a learning aid for various educational topics, including mathematics, physics, computer science, engineering etc [4]. These products include small motors, sensors, wheels, rods, belts and tires, equipping the student with all he needs in order to construct a simple robot.

This approach follows the principles, introduced by Jean Piaget [10], as revised by Seymour Papert [9]. The center of the whole educational process is the active membership of the student, who is expected to broaden his knowledge through the means of construction, management and guidance of artificial objects. Practical experience plays a central role, giving the students the ability to engage in the development of a new idea, through a direct and immediate manner. Unfortunately, when this engagement is virtual, over-simplification tends to appear, due to the abstraction followed during the training phase. This phenomenon leads to a problematic understanding of the world as the robot itself is realizing it. The main cause of such cases is the fact that many topics are taught in un-realistic and abstract ways. Students tend to become acquainted in such methods of learning, and generalize this way of thinking [5].

One of the most attractive and at the same time popular approach is the one offered be the LEGO company, under the name *Mindstorms*. This robotic construction kit has been specially developed, so as to allow the construction of fully autonomous robots, since all the processing power is included in the implemented machine [6]. The "heart" of LEGO Mindstorms is the element, known as RCX, which contains a simple, programmable micro-controller (Hitachi H8). Moreover, it integrates a speaker, an LCD screen, three sensors (input ports), three actuators (output ports) and an infrared port, utilized for communication reasons and program downloading (www.legomindstorms.com).

All authors are with the University of Macedonia, Thessaloniki, Greece. E-mail: {kaskalis,dagdil,gevan,kmarg}@uom.gr

II. Educationally Oriented Lego Mindstorms Programming

Of course, the LEGO Mindstorms are basically a commercial product, oriented towards a variety of uses: everything from simple house entertainment to "best construction idea" contests. Being in the market for over three years, they have already attracted the attention of a large number of buyers. People from all ages prefer this product in order to experiment with a self-complete, flexible and inexpensive robotic kit.

The first need of the LEGO Mindstorms user is, therefore, a simple, yet powerful, method for the programming of their "smart" part, the RCX brick. As one would expect, the manufacturing company itself supports their product with various programming tools. However, a dilemma soon arises: how user-friendly this programming environment should be? It is not hard to realize that full flexibility and power control is generally opposed to user friendliness, especially if the consumers target group covers practically all ages. The LEGO company faced this problem, equipping their product with two types of programming tools: one simple and comprehensible and one advanced and demanding, as regards prior knowledge of programming principles. We will further discuss these tools in the next section.

Moreover, the growing popularity of the product led many researchers and programmers to develop a great number of programming environments and methods. Tools varying from high level, graphical user interface programs to low level, machine language interpreters already exist. Of course, flexibility and in-depth control follows an opposite direction from user friendliness and intuitive comprehension.

These programming environments constitute very interesting proposals and what is more important about them is that they can be found and downloaded from the Internet. We located no less than 12 different proposals, including the tools from LEGO itself: official SDK, ROBOLAB, Visual Basic, C++, Smalltalk, Logo, C (NQC), QBasic, TCL, Brick Programmer, Forth and Java. We, therefore, understand that a comparative presentation of these programming environments exhibits particular interest.

We will mainly turn our attention to the identification of the advantages and disadvantages in an educational framework [8]. Those characteristics which make a certain programming language or environment more accessible and comprehensible to a group of students, will be mainly stressed. At the same time, the issues of flexibility and in-depth control will also be taken under consideration.

III. Presentation Of The Programming Environments

The following presentation assumes an order of a loose decreasing level of user friendliness. We will also group some tools, according to certain common characteristics that they possess. Furthermore, we provide the respective Internet addresses, where one can find the actual programs under consideration, along with further details and support.

A. Official Software Development Kit (RCX Code): http://mindstorms.lego.com

This is the official software development environment, provided by LEGO, which also comes with the Mindstorms product itself. Being very friendly and intuitive to users of all ages, it does not require prior knowledge of any kind of programming principles or languages. The user is only assumed to have a basic familiarization with a computer, to develop his first "thinking" robots. Programs are actually constructed by putting together certain graphical elements or blocks. The idea behind this approach is directly connected with the physical construction of the robot itself: LEGO bricks are assembled in order to build it. This approach becomes even more attractive to students of small ages, because the visual feeling and the functionality of the environment remain very close to that of a child's toy.

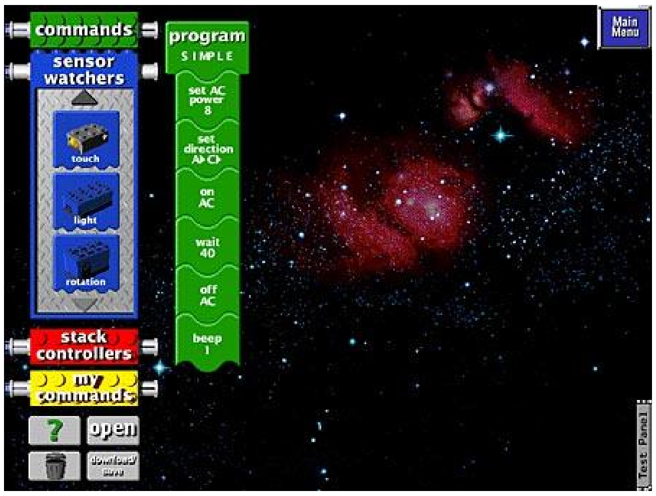


Fig. 1 Official SDK (RCX Code)

As regards users of higher ages, this tool covers all the basic needs for people who just want to do simple experiments and have not used any other programming language before. The official SDK can be indeed an excellent starting point towards certain programming notions' understanding. Fig. 1 presents the graphical user interface of the respective environment.

On the other hand, this proposal appears to be quite restrictive, when one demands sophisticated programming techniques and full control over his robot. The level of flexibility remains low and the potential to create advanced programs is often narrowed. One of the most serious problems is the lack of variable support, which accordingly prohibits the development of serious control over the construct. Moreover, an attempt to create "large" paradigms quickly leads to complex visual structures, which discourage the analysis and comprehension of a program in an educational framework.

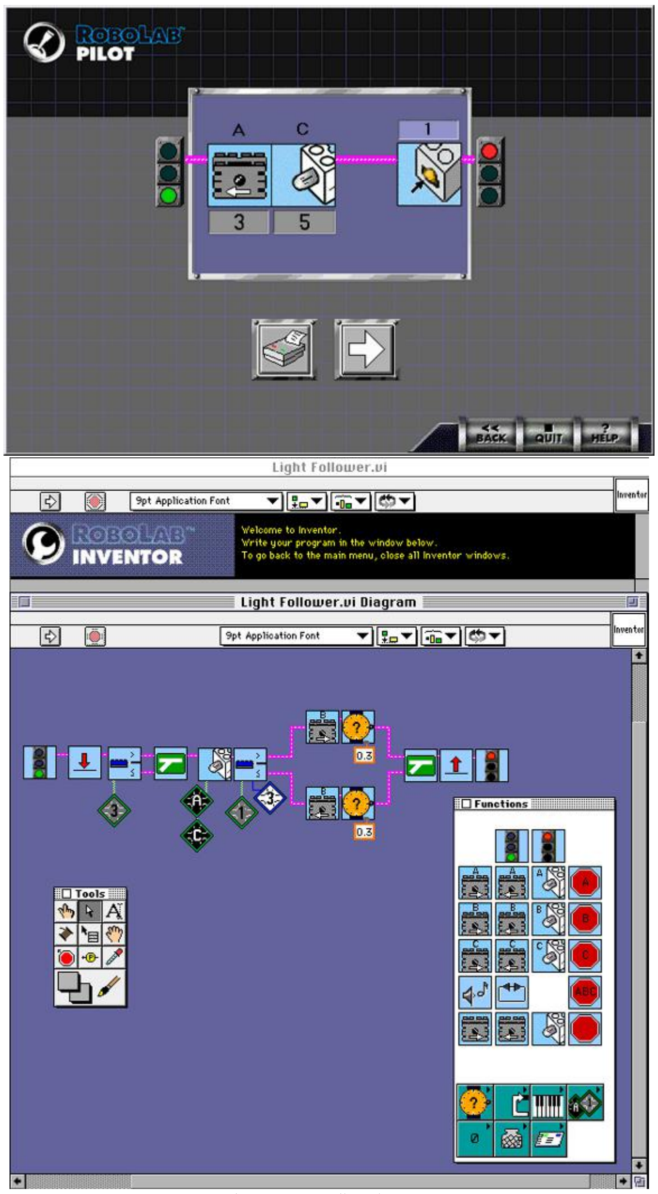
B. ROBOLAB: http://www.lego.com/dacta/robolab

The previous proposal was developed by LEGO to support the Mindstorms product in an integrated retail pack. At the same time, a more "enriched" version was developed. This version was implemented with an orientation towards school labs. It contains a greater number of building blocks (bricks) and RCX elements [11]. Just as before, the ROBOLAB tool respects the "graphical programming environment" philosophy. Through icons and objects the user mainly follows a drag-and-drop concept, in order to build his paradigms. Students learn to think logically, like a computer, as they program a series of events (or commands) for the RCX to perform.

The programming capabilities are divided up into two sections to accommodate the widespread needs of students. Pilot, the basic elementary section, and Inventor, the more advanced section, both use icons to represent commands or structures. In the Pilot section, the number and order of icon options is restricted to ensure the success of the user. This section requires very little reading, making it easy to use in the primary grades. It uses common images like traffic lights, arrows, and watches to allow first time users to construct elementary programs intuitively and quickly. To meet the needs of the middle and upper grades as well as other ambitious programmers is the Inventor section, which offers a new level of flexibility and power, coupled with a slightly different but still graphical interface. Similar to the Pilot section, there are 4 levels that gradually add complexity. Comparing the RCX Code with ROBOLAB, we can say that the former may be somewhat easier to use, especially by younger children, while the latter is probably more powerful from a programming standpoint. RCX Code has an interface that feels something like a computer game; ROBOLAB is built on top of National Instruments' LabView, a visual programming environment for process engineering controls, and is more complex visually. Fig. 2 presents the two main sections of ROBOLAB.

C. Visual Basic: http://www.legomindstorms.com/sdk – http://emhain.wit.ie/~p98ac25

As already mentioned, the LEGO company also considered the cases of programmers, already familiar with "classic" programming languages, willing to take full advantage of the RCX capabilities. The solution offered follows the concept of Active-X objects, in order to allow the development of high level programs through commercially available programming environments. Microsoft Windows' Active-X technology allows programmers to build self-contained programs, that function as plug-ins in all kinds of applications. This solution, although easy to use, it implies the prior installation of commercially available programs (e.g. MS Visual Basic). We consider this fact as a disadvantage in an educational framework, since most of the presented solutions come at no charge.



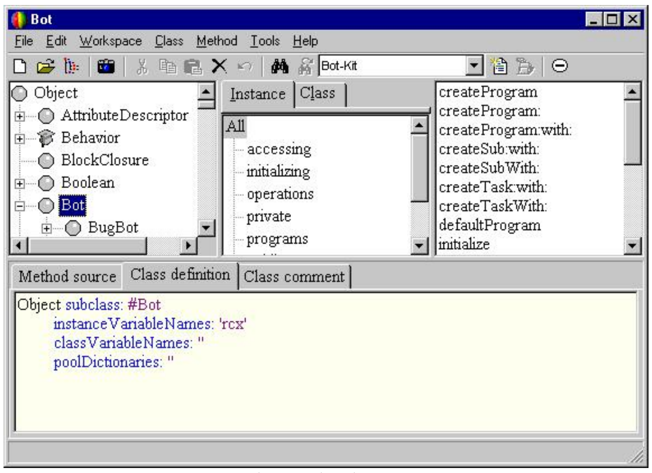
Welcome to Inventor.
Write your program in the window below.
To go back to the main menu, close all Inventor windows.

Fig. 2 ROBOLAB Pilot and Inventor

The way it works is the following. An interface is offered, which permits a program to control the RCX brick, through certain communication controls. The main file is named Spirit.ocx and after its installation, the operating system "acknowledges" its existence to all available applications. It offers a specific set of procedures (methods) you can call, certain variables (properties) you can set and so on. All these are actually embedded in a syntactically "classic" Basic (or Visual Basic) program, allowing a much better hardware control through software structures. If students feel already comfortable with certain programming principles (or, even better, Basic syntax), this tool proves to be very powerful in a potential autonomous robotic construct. Assuming that a school can anticipate the additional cost for the purchase of a commercial programming environment, further benefits arise. Sophisticated tracing and debugging tools become available to programmers. This fact permits the development of highly complex projects and paradigms.

D.
C++: http://www.geocities.com/SiliconValley/Hills/8306/Lego/mindprog.html

In the same context, as the previous approach, we identify the ability to program the RCX brick, using C++. The drawback of the additional cost, discussed earlier, is balanced by the ability to use a powerful, popular, object-oriented programming language. The Spirit.ocx object appears as a class, under the name CSpirit and it can be attached to the main project. A rich set of programming and control components becomes available, helping the user to develop his programs. We should note, however, that the learning curve of this proposal is indeed quite steep. In many cases it can go beyond the needs of a certain robotic control curriculum. This problem characterizes all the tools utilizing the LEGO Active-X object approach.



```
Object subclass: #Bot
    instanceVariableNames: 'rcx'
    classVariableNames: ''
    poolDictionaries: ''
```

Fig. 3 Bot-Kit Environment

E. Smalltalk (Bot-Kit): http://www.object-arts.com/Bower/Bot-Kit

This environment combines the Spirit.ocx object with the commercially available programming language Dolphin Smalltalk. All the comments made so far, also hold for this case, with the only difference located at the type of the used language. In a framework, where Smalltalk is supported, it can prove to be a useful solution. Bot-Kit is capable of controlling a robot in one of two modes. In Immediate mode the computer directly controls the robot by sending signals across the infrared link and receiving back information from the robot's sensor inputs. The robot can be manipulated by typing commands directly or by writing a Smalltalk program to perform the appropriate tasks. In Retained mode a Smalltalk program can be downloaded to the RCX brick so that the robot is able to operate independently of the PC. Fig. 3 presents the Bot-Kit environment.

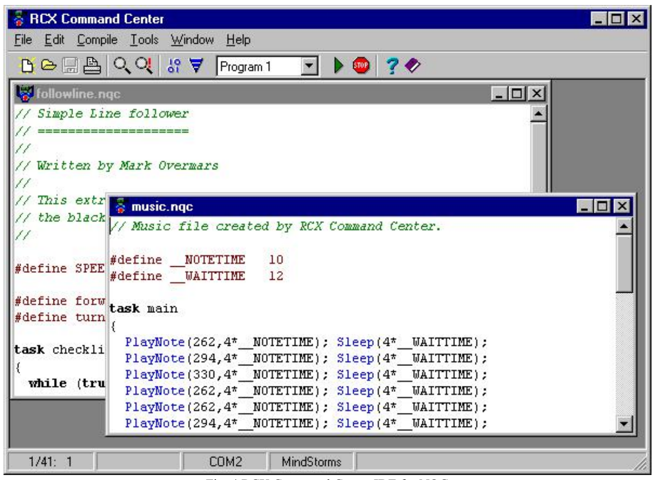
F. Logo (BrainStorm): http://www.netway.com/~rmaynard

BrainStorm constitutes a very interesting programming environment, based on the Spirit.ocx object and supported by the freely available implementation of UCBLogo. It does not require any additional commercial programming tool and it implements a classic "introductory" programming language. Logo has been successfully integrated in a number of curricula and has established a wide user support in the educational community. Through this tool, Logo can now be utilized to direct and program robotic kits. This idea is somewhat inherent in the famous Logo turtle notion, and it comes quite natural to students of younger ages.

G. C (NQC): http://www.enteract.com/~dbaum/nqc

The NQC (Not Quite C) programming environment [2] introduces one of the most interesting independent proposals, as regards the control of RCX. It has already acquainted extensive user support with new versions and updates appearing at a constant pace. In general, it is a very active project, offered for free, and has attracted the attention of many researchers and hobbyists. The core programming language is very close to the traditional C, utilizing the LEGO firmware, but bypassing the Spirit.ocx object.

The extensive support of variables, counters, arrays and subroutines allows the development of powerful control programs. Moreover, the user can have direct control over the RCX, up to a level low enough to install new versions of firmware. Of course the prior knowledge of basic programming in C, can become a problem at introductory levels. Moreover, NQC must live within the constraints of the standard LEGO firmware. For example, since the firmware does not provide floating-point support, NQC cannot provide it either.



```
// Simple Line follower
// ====================
//
// Written by Mark Overmars
//
// This extr
// the black
//

#define SPEE

#define forw
#define turn

task checkli
{
  while (tru
```

```
// Music file created by RCX Command Center.

#define __NOTETIME   10
#define __WAITTIME   12

task main
{
  PlayNote(262,4*__NOTETIME); Sleep(4*__WAITTIME);
  PlayNote(294,4*__NOTETIME); Sleep(4*__WAITTIME);
  PlayNote(330,4*__NOTETIME); Sleep(4*__WAITTIME);
  PlayNote(262,4*__NOTETIME); Sleep(4*__WAITTIME);
  PlayNote(262,4*__NOTETIME); Sleep(4*__WAITTIME);
  PlayNote(294,4*__NOTETIME); Sleep(4*__WAITTIME);
```

Fig. 4 RCX Command Center IDE for NQC

NQC was created as a command-line tool, but certain users have created Integrated Development Environments (IDEs) for it. Basically, they provide a nice user interface on top of the standard NQC compiler. They also add features such as remote control of the RCX, syntax highlighting, etc. In general they provide a friendlier way to write NQC programs. One such IDE is the RCX Command Center, presented in Fig. 4. WinNQC and Visual NQC 2001 are two other graphical user interface environments.

H. QBasic: http://www.phesk.demon.co.uk/lego

Turning our attention again to the Basic programming language, we identify an independent proposal, attempting to utilize the freely available (bundled with MS-DOS) QBasic. Although the project is still in early stages, its objective exhibits interesting aspects. The LEGO technology itself is "not so hardware demanding", as regards the platform where the program implementation takes place. Since many school computer labs do not have state-of-the-art machines, this feature can come in handy in certain situations. It proves that modern operating systems are not actually necessary and the RCX can be effectively programmed through low-end machines (MS-DOS or Windows 3.1 based). As a result, the LEGO Mindstorms technology can reach even more laboratories, raising the need for expensive hardware upgrades.

I. TCL: http://www.autobahn.org/~peterp/rcx

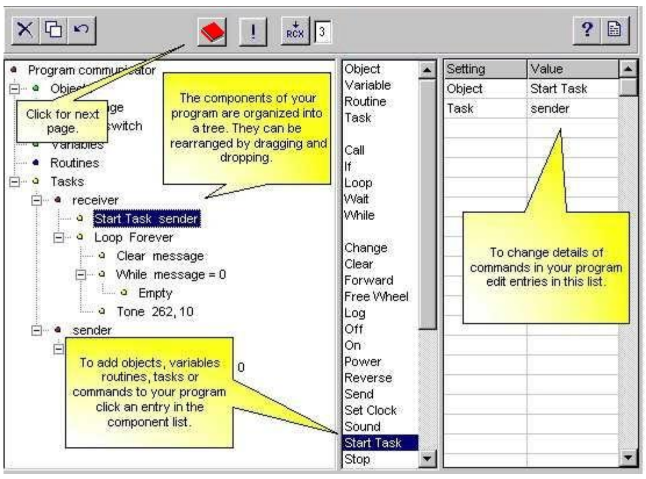
Moving on to scripting languages, we identify a solution permitting the RCX brick to be programmed through the TCL language. Aim of the researchers, developing the project, is a visual programming environment through the "natural extension" of TCL, TK. For the time being, an interpreter has been realized and it allows the control of the RCX with commands of the respective language. From an educational point of view, interest can be found only in situations where the TCL is supported in similar teaching subjects, a fact that is indeed not so common today.

J. Brick Programmer: http://www.umbra.demon.co.uk/gbp.html

Returning to visual programming environments we identify the Brick Programmer tool. Although its development is not so active, compared to other existing projects, it offers a very flexible framework. We could say that it is mainly directed to students of higher ages, providing more general access to variables stored inside the RCX than the simple counters in the Mindstorms system. The programming model supported is that provided by the RCX byte code interpreter and includes direct access to its tasks and routines, while supporting simple symbolic debugging facilities. The events (sensor watchers) found in Mindstorms can be programmed with greater generality, using tasks containing loops and conditions based on sensor values. The user can easily create advanced programs, without going into complex visual structures. The Brick Programmer visual environment appears in Fig. 5.

K. Forth (pbForth): http://www.hempeldesigngroup.com/lego/pbFORTH

Our last two presented environments share the common characteristic of the RCX firmware replacement. The standard firmware from LEGO is by default loaded into the RCX brick. This works very well for most of the applications, users are likely to build. There are certain situations, however, when one needs advanced control over the RCX. The ability to use more variables, to log thousands of sensor readings and to do sophisticated control algorithms require a more powerful environment.



```
Program commun ator
  Objec
    ge
    switch
  Variables
  Routines
  Tasks
    receiver
      Start Task  sender
      Loop  Forever
        Clear  message
        While  message = 0
          Empty
        Tone  262, 10
    sender
```

Setting	Value
Object	Start Task
Task	sender

Click for next page.

The components of your program are organized into a tree. They can be rearranged by dragging and dropping.

To add objects, variables routines, tasks or commands to your program click an entry in the component list.

To change details of commands in your program edit entries in this list.

Fig. 5 Brick Programmer

The pbForth tool can be thought of as an interactive scripting language, integrating Forth. Once the firmware is loaded into the brick, the user can either type pbForth programs or send them as ASCII files using a standard terminal emulator. This is a quite powerful concept since one can literally try his code out and debug it in real time. The brick compiles the code directly. These advanced features, in accordance with the utilization of a sophisticated and well-supported language, can prove valuable tools in courses where serious robotic construction development takes place.

L. Java (TinyVM): http://tinyvm.sourceforge.net – (leJOS): http://lejos.sourceforge.net

The cases of TinyVM and leJOS constitute parts of an ambitious program aiming to substitute the LEGO firmware, so as to have native Java support. Their main difference is the final firmware footprint, loaded in the RCX. TinyVM occupies a space of 10KB

, offering some really impressive features, including: preemptive threads, exceptions, synchronization, arrays (including multidimensional ones), recursion, access to RCX buttons, an emulation tool and well documented Application Programming Interfaces (APIs).

The leJOS proposal, on the other hand, aims to higher levels of completeness and performance, occupying 17KB of RAM. It further supports floating-point operations, mathematical functions (e.g. sin, cos, tan, atan, log), string constants, multi-program downloading and more enriched APIs.

Indeed this project appears to take advantage of the full technological features of the RCX hardware. Having Java as the core programming language, it can constitute advanced robotic curricula, aiming at students of higher ages. In a framework of object oriented programming, it can be directly utilized and it exhibits increased user support. In addition, a visual environment exists, enhancing user friendliness and permitting easier control.

IV. CONCLUDING REMARKS

The LEGO Mindstorms' programming environments that were presented allow us to reach certain conclusions, as regards the integration of the product in educational curricula. First, we can make a classification of the available tools, according to the ages of students they can be oriented to. In an ascending order of student ages we propose the following list: official SDK, ROBOLAB, Brick Programmer, C (NQC) and Java.

This classification can be modified, according to other supported subjects. The solutions of C and Java can be replaced by Logo, Visual Basic or C++. The cases of Smalltalk and Forth, although competitive, they seem to be somewhat "far-fetched" for the current educational state of computer related subjects. Finally, the QBasic proposal can prove to be valuable in low-end school labs, where computer hardware poses difficulties.

We can safely state that the research and educational community seems to actively support robotic construction kits. We can now take more and more advantage of the educational features of LEGO Mindstorms. The low cost of contemporary hardware and software can make robotic kits reach almost every related school laboratory, permitting the hands-on experimentation of students with programmable, constructed machines.

REFERENCES

[1] D. Baum, R. Zurcher, Definitive Guide to LEGO Mindstorms (Technology in Action), Apress, 1999.
[2] D. Baum (ed.), M. Gasperi, R. Hempel, L. Villa, Extreme Mindstorms: an Advanced Guide to LEGO Mindstorms, Apress, 2000.
[3] I. Harel, S. Papert, Constructionism, Ablex Publ. Corp., 1991.
[4] Y. Kafai, M. Resnick, Constructionism in Practice: Designing, Thinking, and Learning in a Digital World, Lawrence Erlbaum, 1996.
[5] H.H. Lund, "AI in Children's Play With LEGO Robots", Artificial Intelligence and Computer Games, D. Dobson & K. Forbus (eds.), TR SS-99-02, AAAI Press, pp. 60-63, 1999.
[6] F.G. Martin, Circuits to Control: Learning Engineering by Designing LEGO Robots, Ph.D. Thesis, MIT, 1994.
[7] O. Miglino, H.H. Lund, M. Cardaci, "Robotics as an Educational Tool", J. of Interactive Learning Research, vol. 10, no. 1, pp. 25-48, 1999.
[8] S. Papert, Mindstorms: Children, Computers and Powerful Ideas, NY: Basic Books, 1980.
[9] S. Papert, Constructionism: A New Opportunity for Elementary Science Education, A MIT Proposal to the National Science Foundation, 1986.
[10] J. Piaget, B. Inhelder, La Psychologie de l' Enfant, PUF, 1966.
[11] M. Portsmore, "ROBOLAB: Intuitive Robotic Programming Software to Support Life Long Learning", APPLE Learning Technology Review, pp. 26-39, Spring/Summer 1999.