Program your NXT robot with Imagine

Pavel Petrovič, *ppetrovic@acm.org*
Department of Computer and Information Science, NTNU Trondheim

Abstract

Our aim is to develop a versatile toolset for constructive learning. We achieve it through the marriage of powerful learning tools: a universal learning environment Imagine Logo (Kalaš and Hrušecká, 2004) and the recent and most popular educational robotics kit LEGO® Mindstorms® NXT.

LEGO Mindstorms NXT is the second generation robotics toolkit. Performance and robustness have been improved, infrared communication medium was replaced by BlueTooth® radio, new sensor types and motor actuators are more powerful with built-in encoders. Educational software RoboLab has been completely rewritten based on the collected experiences. And perhaps the most important is that the platform is open and well documented down to the low-level hardware layer. Schools and educational centres already own the sets. However, the chosen approach used in the RoboLab educational software does not necessarily suit all needs. In particular:

- Despite the communication capabilities of the robots, the software does not provide any means for writing applications for the PC, which could interact with the robots;
- The software requires learning yet another language, which may be a too high obstacle for educators;
- The coding style is based on drawing flow-chart diagrams, which easily become complex and difficult to understand, modify, debug, and analyze; the language lacks structure, the possibility to view programs as plain text; and it implies serious limitations;

Other environments for NXT programming are already available, but they usually suffer from some of these shortcomings. Our approach is different. Let us use the existing power of Imagine Logo, and give its users the possibility to control and even program their NXT robots. We demonstrate how this on three different levels, all utilizing the BlueTooth communication:

- Direct interactive control of robot sensors and actuators from Imagine environment;
- Loadable Imagine project with procedures for interfacing robots from Imagine projects;
- Possibility to download and run a piece of Logo program directly on NXT robots, while this program may communicate with an Imagine project running on the PC or other NXT robots.

The programs running on NXT robots may feature multithreading and get access to all features of the NXT brick. User works only with the procedures that were implemented in Imagine Logo. The logo interpreter – a program written with Next Byte Codes is automatically downloaded to NXT from Imagine environment. The solution does not require the firmware replacement. Our implementation is open-source and available for further development and improvement.

In this paper, we describe all three interaction modes on demonstrative examples. In the second part, we develop two educational experiments for physics and mathematics. These include the Imagine project and robot setup.

In the mathematics experiment, the children are presented with a black-box with gears, where their task is to write a program in Imagine Logo, which will utilize the sensors in order to measure the angular velocity of the input and output of the gear box, compute the conversion ratio, and thus determine the contents of the box given the known set of gear wheels. The physics experiment demonstrates the use of pulleys and a division of force. Students measure and observe the difference in force applied by the motor in various pulley setups.

Keywords

Imagine, LEGO Mindstorms NXT, Educational Robotics.


Figure 1. Robogjeng – after-school robotics club at secondary school Katedralskole in Trondheim at public presentations: MiniSumo playing robots (left), and Labyrinth-navigating and Line-following robots, right.

Robotics in Education

Robot is a machine which behaves, works, and often looks like a human being. Robots – as other machines – are usually used for the benefit of the human: to liberate us from heavy work or provide functionality that would otherwise be inaccessible. Thanks to the advance of the technology, robots are part of everyday life today. From this perspective, it is natural to expect their appearance in the educational process. This for the sake of: assisting the teachers and students in various tasks (for instance for disabled) on one hand, and becoming part of the learning experience on the other hand. Robots in education may:

- ✓ demonstrate phenomena in novel and more ample ways,
- ✓ provide creative platforms for hands-on exploration for individual or group student work,
- ✓ increase entertainment experience during the learning process,
- ✓ increase motivation for learning,
- ✓ spawn interest in technology among students.

Using robots in education has severe drawbacks and challenges:

- ✓ high cost of robots,
- ✓ extra time, space, work and competence required from teachers and schools,
- ✓ shortage of curriculum materials and guidelines.

These can be overcome generally only through added-value enthusiasm of teachers and students who see the potential and value in such work. Once enough experience and potential is generated, institutions may eventually integrate robotic platforms into more or less standardized curriculum. In this sense, LEGO is doing a pioneering work, which naturally is not always a profitable business.

Important role play after-school robotics clubs and centres, which bring together young people with enthusiasts and generate deal of material that can possibly be digested into curriculum by educators. Figure 1 shows the students from secondary school in Trondheim who meet regularly to program robots for fun and robot competitions (Balogh, 2005) – such as RoboCup Junior (Sklar et.al. 2000), MiniSumo, or Micromouse.

Teaching Robotics Materials

The work towards the use of robots in schools is slowly starting to materialize into teaching materials. Most of the pilot programs performed up to day focus on teaching basics of control and programming – learning about sensors, feedback, and mechanics. Examples of such are

the project of London Grid for Learning or woks of (Rosenblatt M. and Choset, 2000, Johansson, 2001). LEGO provides a set of inspiration booklets on mechanics, buildings, energy, and robots, and a set of 16 activities with worksheets and supporting CD: Science & Technology Activity Pack. Extensive work of LDAPS team (LEGO Design and Programming System), which was at the start of RoboLab system produced multiple creative project ideas (LDAPS url). These are described also in the book (Erwin, 2001) and in the recent book of Barbara Bratzel, a teacher at Shady Hill School, a preK through 8 independent school in Cambridge, Massachusetts, who has developed a project-based course that teaches classical mechanics through engineering (Bratzel, 2005). Comprehensive set of materials is provided by CMU Robotics Academy (CMU Robotics Academy url). Yet, there is a large potential for more curriculum material that would be easy to use for teachers without extensive previous experience and technical competence.

LEGO Robotics

Today, LEGO has a long tradition of producing educational toys and robotics programmable sets. The LEGO Educational department founded in 1980 was renamed to LEGO Dacta in 1989 and produced programmable sets that could be controlled from computer running a dialect of Logo. Already since 1984, Dr. Papert had been working with LEGO's development staff on linking the LOGO computer programming language to LEGO products. LEGO Logo that was developed in that cooperation was a successful product that allowed controlling non-autonomous LEGO robotics sets. However, today, this product is almost completely forgotten somewhere in the history of the early 90s. With the autonomous robotics sets, new concepts arose and ought to be treated anew.

A precursor of the autonomous programmable brick was CodePilot released in 1997 as part of the 8479 Technic set. It could be programmed by swiping over barcodes. However, the highlight, LEGO Mindstorms with wonderful programmable brick RCX arrived in 1998, and it is still enjoying attention (RoboCup Jr. Slovakia url). RCX can be programmed using graphical iconic parallel language of Robotics Invention System, using RoboLab – an educational iconic language with greater functionality, using Lejos – a limited version of embedded Java, using Not Quite C (NQC) – a C-like interpreted language with limitations, or even using BrickOS – a standard C/C++ based on the GNU C/C++ with libraries for RCX control. Many other systems were developed, for instance PbForth – minimalistic stack-oriented procedural language. Several more or less compatible flavours of the LEGO Mindstorms were produced, targeting mainly entertainment market: CyberMaster, Scout, Micro-Scout, Spybotics. An important step forward was achieved by releasing the LEGO camera, which however, requires wired connection to the computer, and provides very limited functionality. The recently released LEGO Mindstorms NXT sets support compatibility with previous LEGO electric parts (motors, sensors), and align the LEGO robotics product with the current standard for embedded devices: radio BlueTooth communication, I2C bus, fast ARM7 microprocessor, direct USB connection, graphical display, servo motors, large flash memory, EIA-485 fast serial interface, compact Li-ion rechargeable battery. However, the main break-trough is that LEGO released the source-code of the firmware, detailed schematics of all parts and documentation. This allows the non-LEGO developers to maximize the educational benefit of the new robotics system. In addition to new version of RoboLab – NXT-G, which is now the main programming iconic language for NXT, less than a year after its release, NXT can be programmed in limited Java – remotely using iCommand, and autonomously using Lejos, in a new version of NQC (now called Not eXactly C), in C-like language RobotC, and using the low-level byte-code language Next Byte Codes (NBC). Third-party developers play a very important role in making the LEGO robotics products useful by providing various sensors. The main players are HiTechnic, Mindsensors, and Techno-stuff.

The LEGO company has a strong focus on the user community, and provides ideas, hints and inspirations of professional quality at their websites (mindstorms.lego.com, legoeducation.info/nxt/). In addition, thousands of users and third-party developers exchange their experience at various forums, which include LEGO Users Group (lugnet.com), and blogs (thenxtstep.blogspot.com, nxtbot.com).

Logo for NXT

Logo programming language has a successful history of being used for teaching programming. Recently, LEGO introduced its second generation robotics educational sets on the market. Their combining is made possible and easy by using the radio Bluetooth link between a PC running Imagine Logo and a NXT brick. Both learning tools complement each other. In concert, they create a learning studio of endless series of possibilities for educational projects. That is our main motivation for the Logo for NXT project. It consists of the following parts:

1. Imagine project with simple controls for immediate remote control and status querying of NXT brick.
2. Imagine loadable text project nxt.imt containing set of elementary procedures that directly interact with NXT brick.
3. Logo interpreter running on the NXT brick that allows running simple logo programs on NXT brick autonomously.
4. Set of example programs for both the second and the third level.

Imagine NXT Controller

For immediate control of NXT robot, testing the functionality, calibrating sensors, or even simple navigation including speed regulation and direction steering, one can use the provided controller project, see figure 2. The user can setup types of sensors, configure and operate the motors. In the upper-right part of the screen, the nine controls allow steering a two-wheeled robot in all main directions. Since NXT motors have built-in rotation sensors, NXT brick firmware supports automatic synchronization of the motor speed to a specified value. Furthermore, the firmware can automatically control the ratio between the rotational speed of two selected motors. The controller project has controls to test this functionality. The NXT brick only needs to be on and have standard or compatible firmware installed. Logo interpreter program is not required.

Control NXT from users' projects

Users can load the nxt.imt project and call many different procedures that interact with NXT brick. These are of two main types:

1. Direct commands and commands according to LEGO NXT Communication protocol (LEGO, 2006), 34 different commands at the time of writing.
2. Higher-level commands that are based on direct commands and provide some higher-level functionality, or easier to understand interface, about 10 different commands at the time of writing.

Examples of the commands from the first set are **nxt_getoutputstate, nxt_resetmotorposition,** or **nxt_openappenddata,** which determine the state of the motors, reset the built-in rotation sensors, and open a file in NXT's flash memory for appending, respectively. All commands are described in the documentation of the Logo for NXT project (Logo for NXT url).

Higher-level commands provide the following interface:

(motor m cmd [arg]) – operates on motor output m, supported commands are: on, off, brake, regulate [speed], sync [ratio], onrot [nrot], onms [ms], power [pwr], rst;

(readmotor m cmd) – reads the rotation sensor of specified motor in one of three ways (rotations, tacho, blocktacho);

(sensor N) – reads the value of specified sensor;

(setsensor N type_or_cmd) – configures the specified sensor as: raw, switch, tmpC, tmpF, refl, angle, lightA, lightI, db, dba, lowspd, spd9V. Also allows resetting sensor: rst.

(nxt_connect port) – establishes connection to NXT brick on specified virtual serial BT port;

(nxt_disconnect) – closes connection to NXT brick;

(download filename) and **(upload filename)** – transmits a file to or from NXT;

User projects can use commands from both groups interchangeably. Attention needs to be paid to the fact that BlueTooth radio communication despite its relatively high bandwidth in one-directional communication has an inherent delay when switching the direction of communication. Therefore the commands that require responses from NXT will usually take about 60ms to complete.

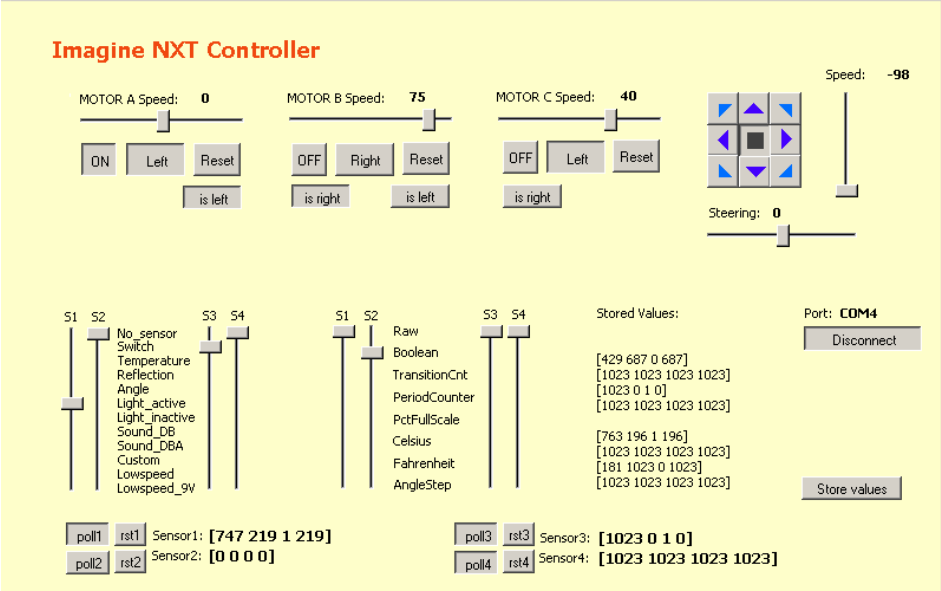

Figure 2. Direct control of NXT brick using Imagine project.

Logo interpreter for NXT

Finally, as a simple and straightforward extension to the interface described above, the command **(remote expression)** allows evaluating Logo expressions directly on the NXT brick. This command assumes the logo interpreter program is already running on the brick, but if it is not, the running program can be controlled from Imagine with the help of **nxt_startprogram, nxt_stopprogram**, and **nxt_getcurrentprogramname** commands. The expression is transmitted to NXT over BlueTooth radio, evaluated on the NXT, and the return value is sent back to Imagine Logo running on the PC. The expression can contain all supported commands. For example, the command **(remote [load "myprogram.lgo])** will load the specified program from the file stored in the NXT flash. NXT Logo programs can define procedures and operations, and they can themselves, in an analogical way, request expressions to be evaluated on the PC by Imagine Logo (provided that the expression-polling is turned on) using the same remote command.

At the time of writing, only very limited functionality of the interpreter is available, but we expect the following features to be available shortly:

NXT Logo supports the following primitive procedures and operations:

1. Arithmetic: *add, sub, mul, div, mod, neg, abs*
2. Logic: *and, or, xor, not*
3. Bitwise: *bitand, bitor, bitxor, bitnot*
4. Lists and words: *first, bf, last, bl, item, se, list, word*
5. Maps & co.: *map, apply, foreach, run, remote*
6. Loops and conditions: *repeat, while, foreach, if, ifelse*

7. Predicates: *empty? number? list? word? equal? gt? lt? ge? le? member?*
8. Variables and procedures: *make, let, to, op, stop, erase, launch*
9. NXT file handling: *load, printto, fopen, fwrite, fread, fclose, fremove*
10. Sounds: *tone, sound*
11. Motors, sensors: *motor, readmotor, sensor, setsensor, battery*
12. Other: *wait, random, print, draw, ascii, char, boot, gc, fm*

The expressions must be in the prefix notation. Currently all procedures have a fixed number of arguments, and therefore the parenthesis are not needed (and are not recognized by the parser). Variables are used in the same way as in Logo: **"name** denotes variable name, and **:name** refers to its value. There are global variables (**make**) as well as local variables (**let**, and arguments). Lexical scoping applies as usual in Logo. There is currently no support for objects, or images, or other primitive types. However, the **draw** command allows drawing images stored in the files in the NXT flash, and the **sound** command allows playing NXT sound files. The NXT software architecture has a limited support for multithreading, and the logo contains the **launch** command for starting a separate thread. See the documentation (Logo for NXT, url) for more details on the syntax and purpose of all supported commands.

Example programs

A set of documented example programs demonstrates the basic use of the Logo primitive commands. In this way, users who are not familiar with Logo programming language can learn to use Logo for NXT directly. In fact, we do not require the user to own Imagine Logo. Logo programs that are uploaded to NXT can be started directly from the interpreter. The communication with the PC will be disabled, and the remote command will have no effect (return “false), but the logo programs will run.

In addition, example programs for line-following, and mini-sumo playing robots, as well as some other are provided.

Implementation issues

Logo for NXT interpreter is written in NBC, a very low-level programming language that is compiled into byte-code, which is then interpreted by the standard NXT firmware. NBC programs have at their disposition 32 Kbytes RAM for “dynamic” memory. Logo interpreter allocates the whole available memory as one large array, and features its own C-style memory management (malloc, free) with support for garbage collection on demand. That means that all data that can be de-allocated immediately are de-allocated immediately. However, lists, and words are being freed only after verifying that there are no pending references. We aimed at very compact and memory-efficient representations utilizing almost every data bit.

System works with 16-bit unsigned words, where typically the upper 14 bits form a value or a pointer (given max. 32 KB memory, 14 bits are sufficient to address 16-bit words). The remaining two bits are used for the type information: direct integer value (00), pointer to word (01), car-pointer to list (10), cdr-pointer to list (11). List elements are normally stored in a continuous block, and the cdr-pointer is used only if the list grew out of its original memory block.

Lists of all words, all global variable names, and all procedure names are maintained. These lists are not simple linked-lists, rather sorted linked-lists of tables containing 8 records each in order to make the search faster (**symbol tables**). When any of the tables becomes full, it is automatically split in two. Logo words are stored in a single memory block each. There is a separate support for the columned and quoted words (“hello, :var) so that there is no duplicity (i.e. columned and quoted words are only manifesting its type and point to the memory block with the unquoted word). Unquoted system reserved words are especially marked so that they are never de-allocated. All unquoted words have one 16-bit slot reserved for the variable pointer to the current context, which contains the current value, and one 16-bit slot reserved for the

procedure pointer to procedure symbol table. The former is updated each time a variable is created/erased/shadowed. Words are never stored in duplicates in the memory, and as a consequence, comparison of pointers is sufficient. List items are 16-bit values as described above. The pointers to list nodes can point to middle of a continuous sequence of list elements. When lists are allocated, there is automatically a small buffer of free elements provided before the start of the list to allow for growing forward without the need of re-allocation. This is to avoid memory blocks containing only a single list element, which would mean too large overhead. Lists are stored in reversed direction so that garbage collection process can safely traverse through a pointer to a random element in the middle of the list to the start of the block and mark it as used. Each memory block requires one word (16-bytes) of overhead containing a 14-bit value – the block size, one bit indicating if the block is free, and one bit used by the garbage collection. The procedures are represented as lists of commands, preceded by a value containing the number of arguments, and a pointer to the list of argument names. The names of primitive procedures are loaded to memory from a separate file (logo.def) during the start-up of the interpreter.

The **core engine** consists of parser, evaluator, and printer. The **parser** is started when an expression is sent using the remote command, or when a logo program file is loaded from the NXT flash. Its role is to simply convert the string expression into internal expression/value representation. The opposite transformation is the responsibility of the **printer**. The **evaluator** has two flavours: one that evaluates list-expressions – such as procedure or loop-command bodies. These contain commands that do not return values, and the whole expression can optionally return a value using the **op** (output) command. The second flavour evaluates single immediate values, such as procedure arguments, and always returns a value. The evaluator supports recursion, and variable shading. It works with a current context, which is a structure containing a reference to local variables symbol table, parent context, and expression being evaluated as well as the position in the evaluated expression. Since the parser translates all variable references to pointers, the values can be immediately retrieved through the symbol-table pointers contained in the block of the word. When returning from a call, current context is released, and the previous symbol-table pointers are retrieved from the local symbol-table that stores the previous pointer.

The communication with PC is maintained by a communication thread perpetually running in the background. It calls the parser and evaluator when an expression arrives. If communication is not established, user can enjoy a limited interaction with the logo interpreter using the buttons and LCD display.

Garbage collection works only on demand. The logo program must call the **gc** command in order to free unused memory. At that point of time, all global variables, all local variables in all parent contexts, all referenced lists, words, and procedure bodies are marked as used. Finally, the whole memory is scanned, and all blocks which are not marked are claimed to be free. The fast ARM7 processor running on 48MHz frequency is able to scan the whole 30kB memory thousand times a second, therefore even if the logo interpreter itself is interpreted, the performance can be acceptable, provided that the garbage collection is not started in critical periods. We believe this is a feasible approach for programming embedded systems in general, where sudden unplanned interruption by garbage collector could cause undesirable results. On the other hand, the programs for embedded devices typically contain safe periods when garbage collecting break is possible.

Two Projects

While LEGO and robotics provide multiple advanced challenges, such as robotics competitions, which are strongly motivating students with over-average performance, LEGO in the schools can be the motivational force for the students who are struggling with the usual curriculum material due to their difficulties with maintaining attention and finding interest when links between theoretical knowledge and practical applications are missing. This section discusses two simple LEGO Logo-style projects, to be used directly at the standard math or physics lessons.

Gearboxes – learning about fractions

The first project can be used at mathematics courses in junior high-school for teaching computing with fractions. Students are presented a closed box with the same kind of gear wheel in the middle of both sides of the box. The wheels are connected using gears in some way, i.e. turning the wheel on one side by 1 rotation turns the wheel on the other side by a/b rotations (see top of the figure 3, the inside of the gearbox is shown on the right). Teacher can prepare several gearboxes with different ratios. It is possible to simply attach a motor with a rotation sensor on one of the wheel, by connecting it on top of the gearbox, and another rotation sensor to the second wheel by connecting it to the bottom of the gearbox (both modules are shown at the bottom of the figure). The task for the student is to first measure the ratio between the numbers of rotations on both sides, then describe this ratio by a fraction. Finally, the student needs to make estimation (a drawing) of the contents of the gearbox, knowing that only standard gearwheels with 40, 24, 16, and 8 teeth are used inside of the gearbox. This requires mastering fraction algebra. Students can use Logo for NXT to write program that measures the number of rotations, see figure 4.

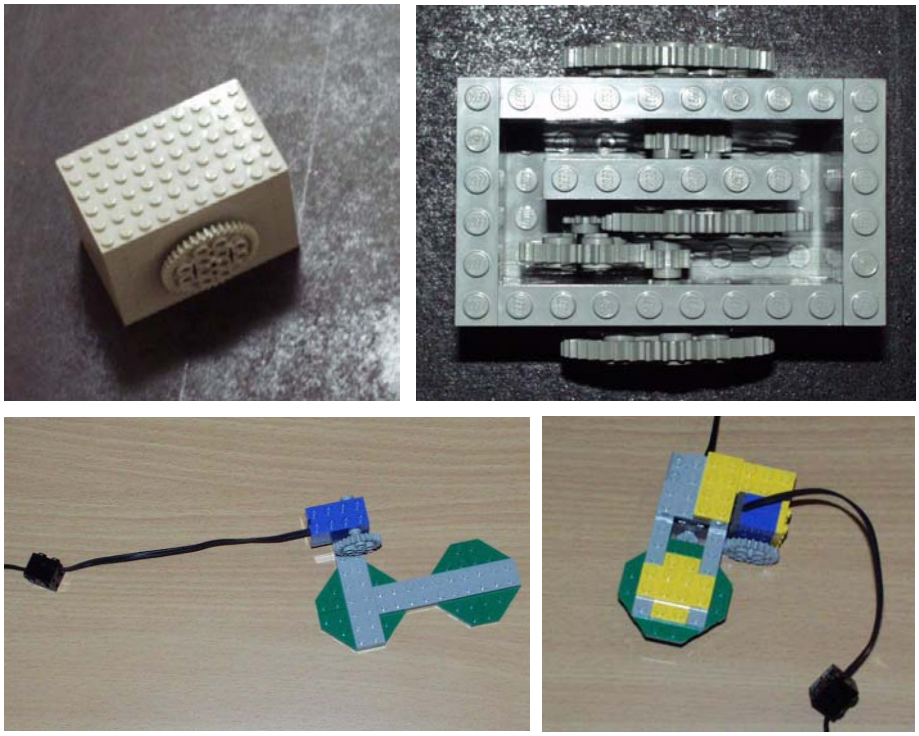

Figure 3. Gearbox project: motivated learning about fraction algebra.

```
to measureRatio
  setsensor 1 “angle
  setsensor 2 “angle
  setsensor 1 “rst
  setsensor 2 “rst
  motor “A “power 50
  ; rotate the first wheel 10-times
  while [gt? 160 sensor 1]
  ; sensor2 measures the opposite side, output the ratio
  op list sensor 1 sensor 2
end
```

Figure 4. Gearbox project: Logo for NXT program to measure the ratio between numbers of rotations.

Analysis: Including a practical experiment in a series of mathematics lessons not only improves the manual skills of the pupils, but mainly creates important connections between the theory material that is part of the curriculum and the real life. Such practical demonstrations of the learned theory are **essential**, if the school aims at preparing the pupils for the real life, particularly in the lower-grades and in the schools for the widest general public.

Pulleys and division of force

The machine shown on figure 5 is a demonstration of practical application of pulleys. It is inspired by a model in the science museum in Bremen. A mobile platform can slide in the track from right to left and opposite. It is connected using a LEGO plate brick to a string, which can be rolled on a coil using a motor controlled by NXT brick. A light sensor can signal when the platform reaches its left-most position. The platform can either be connected directly, or using the two pulleys. There is a load (on the figure, it is a LEGO 9V battery box) placed on top of the platform. Various loads can be used, especially such that allow gradual increase of weight – for example a set of weights from the physics laboratory). The task for the student is to measure the force the motor applies on the platform in both machine configurations. For reasonable results, the motor is powered with only 10-15% of power. The force can be measured using Newton-meter from the physics laboratory, or by loading the platform with different weights (and taking into account friction coefficient), or by attaching a string with weights hanging down from the other side of the platform over another pulley. Figure 6 shows a simple program in Logo for NXT that can be used to run an experiment.

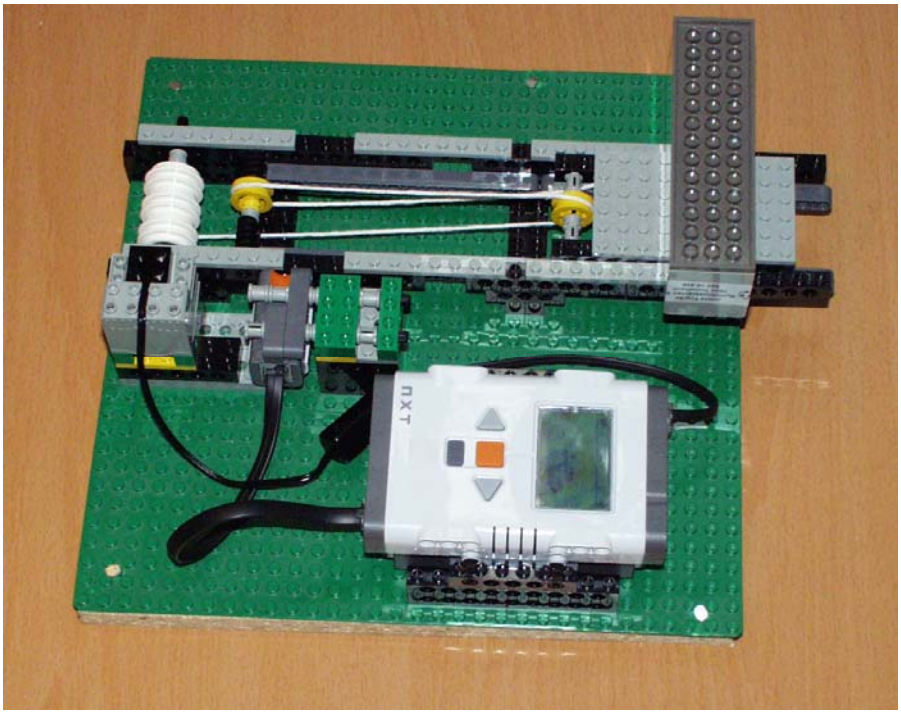

Figure 5. Pulleys project: learning about pulleys, force, friction, work.

```
to movePlatform
  setsensor 1 “lightA
  let “threshold 30
  motor “C “power 15
  motor “C “on
  while [lt? sensor 1 :threshold] []
  motor “C “off
end
```

Figure 6. Pulleys project: Logo for NXT program to run the platform and stop it at the end of the ramp.

Conclusions and Future Work

We presented an alternative way of programming NXT robots, which we believe is the way the LEGO programming system could have been designed in the first place. In fact, the original LEGO Dacta systems that were connected using the wired interface were using a powerful dialect of parallel Logo. The transition from the wired remote control to autonomous RCX required abandoning that path due to the limited memory, CPU, and communication capabilities of the RCX. NXT resolves these bottlenecks and it is time to come back with powerful Logo language as the programming platform for the educational use of LEGO robotics systems. Our work does exactly that. It still has major limitations due to the fact that it is implemented using interpreted NBC, that is Logo programs are currently interpreted by an interpreter that is interpreted by the firmware. We chose this solution for three reasons: 1) to determine the limitations of the NBC and to see if it can provide a reasonable performance and 2) to liberate us from the arduous challenge of understanding how to write our own firmware, and 3) to provide as compatible solution as possible, one that does not require the user to change the firmware on the NXT brick. Despite the limitations (these are: 1) 30 kB memory limitation for all data, code, and internal stack compared to normally available 64 KB RAM, and 2) limited performance compared to interpreter that would execute directly on the CPU), our system can already be used successfully to program NXT robots in Logo. In addition, it smoothly plugs into Imagine Logo educational suite. This paper presents the system as we wish it will perform. At the time of writing, portions of the functionality are still under intense development. The Logo for NXT project is open-source, and available for the user customization. Source-code is extensively commented, and documentation is provided. We demonstrate two simple projects, where Logo for NXT is used in practical student tasks within standard curriculum of junior high school.

Acknowledgments

Tomas Gunnarsson from Nardo Robot Club, and Gustav Henrich Bernhardt from Katedralskole in Trondheim provided extensive support and photographs from public presentations.

References

Balogh, R. (2005) *I am a Robot - Competitor, A Survey of Robotic Competitions*, International Journal of Advanced Robotic Systems, vol.2, number 2, p. 144—160.

Bratzel B (2005) *Physics by design*, College House Enterprises, LLC.

Erwin B. (2001) *Creative Projects with LEGO MINDSTORMS,* Addison-Wesley.

Johansson H. (2001) Robotic courses based on LEGO, Master Thesis, Department of Computer Science, Lund University.

Kalaš I. and Hrušecká A. (2004) *The Great Big Imagine Logo Project book*, Logotron.

LEGO (2006) *LEGO Mindstorms NXT BlueTooth Development Kit*, LEGO NXT'reme document, version 1.00.

Rosenblatt M. and Choset H. (2000) Designing and Implementing Hands-On Robotics Labs, IEEE Intelligent Systems, vol.15, nr. 6, p. 32-39.

Sklar, E.I. and Johnson J.H. and Lund H.H. (2000) *Children Learning From Team Robotics: RoboCup Junior 2000*, Educational Research Report, Department of Design and Innovation, Faculty of Technology, The Open University, Milton Keynes, UK.

CMU Robotics Academy url, http://www-education.rec.ri.cmu.edu/

LDAPS url, http://www.ceeo.tufts.edu/ldaps/htdocs/

Logo for NXT url, http://virtuallab.kar.elf.stuba.sk/robowiki/index.php/Logo_for_NXT

RoboCup Jr. Slovakia url, http://www.skse.sk/