From the Logo Turtle to the Tiny Robot Turtle: practical and pedagogical issues

Moro Michele
University of Padova, Department of Information Engineering
mike@dei.unipd.it

Alimisis Dimitris
School of Pedagogical and Technological Education (ASPETE)
alimisis@otenet.gr

ABSTRACT

This work proposes a teaching method, which attempts pedagogical utilization of educational robotics, in the form of an interdisciplinary assembly project, in the secondary/high school education. The work in question is part of a program training pupils on how to program a robot to behave as a Logo Turtle. The pupils involved construct and program that robot with the use of a LEGO MINDSTORMS NXT set. The knowledge areas involved are those of Technology, Informatics and Mathematics.

KEY WORDS: *Educational robotics, Logo Turtle, Robot turtle*

INTRODUCTION

The educational activity described in this work includes a proposal referring to the construction and programming of a robot turtle controlled by Logo-like commands. As a matter of fact, the proposal in question concerns the transfer of the known virtual Logo turtle from the computer monitor to the ground. The pupils are intended to learn how to design and construct a robot-car with the use of the right Lego Mindstroms materials (Technology), how to use the Lego Education NXT-G software in order to program the motion of the said robot (Informatics) and how to achieve the right measurements and calculations with a view to determining the right parameters (Mathematics) aimed at a successful programming of their models. The conduct of the assembly task requires 6 to 8 teaching hours.

The current didactic approach in informatics for the introduction of students in programming uses languages of general purpose and emphasizes on work with numbers and symbols in an abstract way ignoring students' needs for learning through concrete action. During last years programmable several kits for mechanical constructions have been introduced in school education that can communicate with a computer and can be programmed to move and to interact with their environment through sensors (Lego-mindstorms, Pico-crickets kits from the Media Lab at MIT , Resnick et al, 1996). Using these new technologies the construction and programming of robots is not a black box for students (as it happens in the robotics industry which aims at humans using pre-programmed pre-fabricated robots) but following the constructionist paradigm students can construct and program themselves their own robots in a transparent way (Kynigos 2008).

The activity described below is meant for learners without previous experience in educational robotics. Pupils are required to construct a car model with the use of material taken from the Lego kit and to program same with the use of the educational LEGO Mindstorms Education NXT software. The pupils are to carry out their entire work in the computer workshop, forming groups (of 3-4 individuals), while discussing and making decisions within the groups and the plenary assembly of the class. At the end of each

teaching unit, the groups will record their impressions in a task diary, either of printed or electronic form. In the framework of the TERECoP project (www.terecop.eu) we have suggested that robotics activities like those presented here should be integrated in a constructionist pedagogical setting (Alimisis et al. 2007) and we have proposed an analytical project-based methodology that teachers can follow to implement robotics in classroom settings (Papanikolaou et al 2008)

PROPOSED TEACHING COURSE

First experiences with robots in an educational environment should start with very simple architecture and actions: first, robots could use only one motor on linear trajectories to analyse very basic relations like those among the power applied, the spanned space, time, the motion type, and more specifically, the ratio between the angle of rotation of the motor and the linear space travelled by the robot, which is a function of the wheel radius. In these first experiences sensors can be integrated just for understanding their functions and operative range, influencing the motion in not complex ways (like stopping and/or switching the direction).

When the teacher wants to introduce robots as learning tools, he/she knows what kind of didactic goal the proposed experience should promote. Therefore the robotic architecture, the task to be performed and the robot programming must be carefully designed to make the students more and more aware of the new knowledge or principle or abstraction or feeling the teacher wants to present.

Referring to the LEGO Mindstorms NXT kit and its programming environments, you could first prepare a very simple robot but with two wheels and two motors, one motor per wheel, and something to allow the robot to follow not linear trajectories, like a free rotating wheel or a support with a little friction. Because you can control separately the power applied to each motor, the first observation is that only when the two wheels move with the same angular speed and direction, the robot moves linearly, forward or backward. But when the two rotations are different, with a different speed and/or direction, the trajectory is not longer linear. All the time the two speed are constant, the trajectory is circular and it is interesting, as a first deduction, that the centre of the trajectory is on the axis connecting the centres of the two wheels, external to the robot if the two directions of rotation are concordant, on one wheel if only the other is moved, within the wheels if the directions are opposite one another. In the latter case, if the two speeds are the same in modulus, the robot more or less pivots: even the pivoting angle is in a constant ratio with the angle performed (in opposite directions, as mentioned) by the two motors. Moreover, at this point the linear motion can be understood as the limit case of a circular motion when radius tends to infinite.

Figure 1. The tiny Robot Turtle

All these things can be put together giving this kind of robot the appearance and behaviour of the Lego Turtle, whose main motion commands are linear forward and backward (*forward*, *backward*) for a certain space, measured in a suitable little unit, and pivoting for a given angle (*right*, *left*). With the observations above the implementation of such commands as robot sub-procedures is quite straightforward. For instance, NXT-G allows the user to define 'My Blocks' as reusable building blocks similar to the basic ones provided by the environment but user-defined. Fig.1 shows the robot that we called *tiny RobotTurtle*.

In fig. 2 you see a step in defining the *forward* command (the *backward* command is similar). For simplicity the command parameter is represented by the so called 'duration' of the composing *move* block (usually this kind of basic command pilots contextually a couple of motors), measured in degrees of the external motor axis. In fact an NXT servomotor is internally de-multiplied in order to permit one degree of resolution on the external axis. Thus the minimum space travelled by the turtle in a forward 1 or backward 1 command is that one corresponding to the rotation of 1 degree of both the wheels.

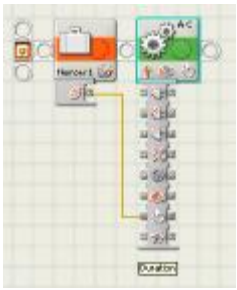

Figure 2. *Defining the forward command*

With a specific procedure provided in the NXT-G environment, you can transform this skeleton into a parameterised sub-command with its own representative icon (see Fig. 3)

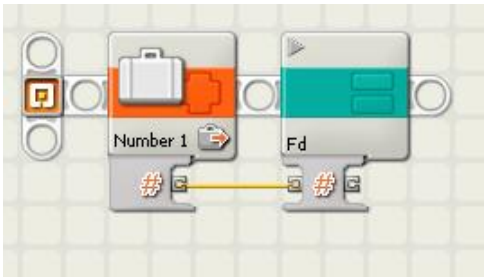

Figure 3*.The parameterised subcommand (left) and its representative icon (right)*

The procedure to define the other two commands, *right* and *left*, should be also similar but, due to more precise definition of the command parameter, with represents the rotation angle of the whole robot, you must calculate the 'duration' of the motor movement transforming linearly such an angle into the two angles of rotation of the two motors in opposite directions. For this reason in this case you must use one *motor* block for each motor and some pre-calculations. Once empirically established the ratio to be calculated, namely about 2.22 for the realised robot, you must also cope with the limitation of the robot firmware which supports only integer calculations. In order to reduce the loss of precision, you can multiply/divide the command parameter for a couple of rather large integer whose ratio is more or less the required one: we chose 12445/5600 (fig. 4a). The complete *left* sub-program is showed in fig. 4b.

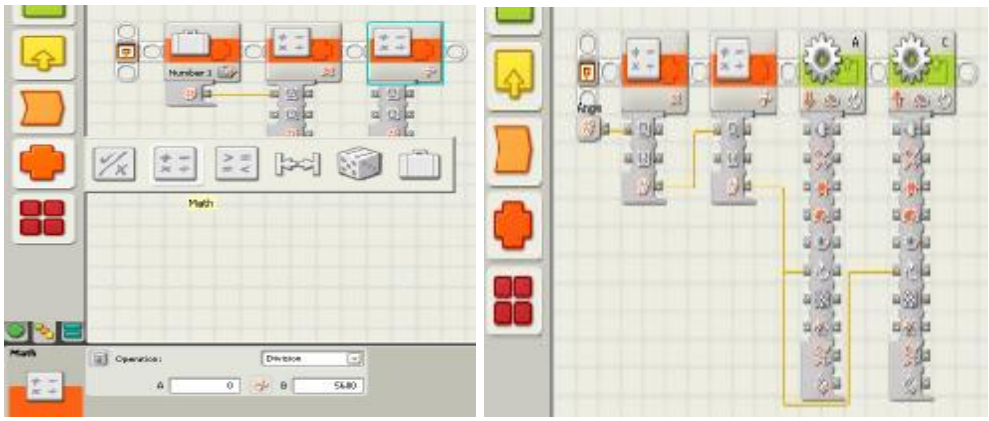

Figure 4. *The ratio calculation (right) and the left command (left)*

Supposing to have programmed the four basic turtle command as 'My Blocks' called Fd, Bk, Lt, Rt, you can now 'simulate' the drawing of a square of edge L, corresponding to the Logo code: repeat 4 [Fd L Rt 90]

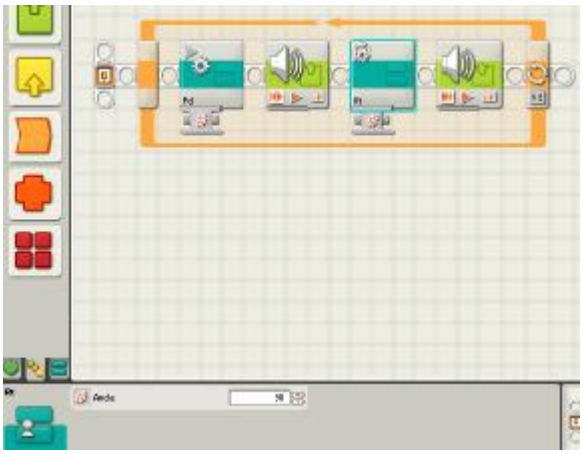

Figure 5. *The main program*

With a simple program like that one of fig.5, which includes a couple of blocks emitting sounds in the main loop.

POSSIBLE EXPANSIONS

The tiny Robot Turtle is an example of a sensorless robot able to execute predefined relative motion commands: linear steps and rotations are given as values relative to the current position and orientation. But even most of the computer implementations of the Lego turtle provide an absolute Cartesian reference and the possibility to move the turtle to absolute positions and orientations.

One of the relevant problems in robotics is to teach the robot about its position/orientation and to maintain this information in the robot with sufficient precision. An alternative solution is to equip the robot with an external system, e.g. one or more videocameras, able to calculate the coordinates of the robot itself and send them to it, or to equip the robot with an on-board sensor system suitable to give it a sufficient ability of self-positioning, possibly with the help of some external known markers (Arlegui et al. 2008).

ANNOTATION: by means of the educational procedure proposed, pupils construct their programmable models and learn fundamental programming concepts and techniques (programmable artifacts?) by creating programs that determine the behavior of their models not on the computer monitor but in the physical environment, a fact that offers a high degree of interaction between pupil and actual object. As a result, the pupil can relate the reactions of the model to the program's commands and observe the effects that they have on its behavior. That possibility gradually leads to ameliorated solutions to the problem under examination. The ultimate workmanship product of the pupils is a mechanical robot-turtle, which "understands Logo-like commands" and, thus, can constitute an invaluable tool in the pupil's hands for further development of activities regarding problem solution and application of more complex programming techniques.

References

Alimisis, D., Moro, M., Arlegui, J., Pina, A., Frangou, S., Papanikolaou, K. (2007) Robotics & Constructivism in Education: the TERECoP project, In Ivan Kalas (ed.), Proceedings of the 11th European Logo Conference, Comenius University, Bratislava, ISBN 978-80-89186-20-4

Arlegui, J., Menegatti, E.., Moro, M., Pina, A. Robotics, Computer Science curricula and Interdisciplinary activities, Workshop Proceedings of SIMPAR 2008, International Conference on Simulation, Modeling and Programming for Autonomous Robots, Venice (Italy), ISBN 978-88-95872-01-8, pp. 10-21.

Kynigos, C. (2008) Black-and-white-box perspectives to distributed control and constructionism in learning with robotics. Workshop Proceedings of SIMPAR 2008, Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots, Venice (Italy), ISBN 978-88-95872-01-8, pp. 1-9.

Papanikolaou, K Frangou, S. Alimisis, D.*Teachers as designers of robotics-enhanced projects: the TERECoP course in Greece,* Workshop Proceedings of SIMPAR 2008, International Conference on Simulation, Modeling and Programming for Autonomous Robots, Venice (Italy), ISBN 978-88-95872-01-8, pp. 100-111.

Resnick, M., Martin, F., Sargent, R., and Silverman, B. 1996b. Programmable bricks: Toys to think with. *IBM Systems Journal.* Volume 35, No. 3&4.

Acknowledgement and disclaimer: The paper is based on work done in the frame of the project TERECoP co-funded by the European Commission-European Programme Socrates/ Comenius /Action 2.1, Agreement No 128959-CP-1-2006-1-GR-COMENIUS-C21 2006 – 2518 / 001 – 001 SO2. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.