

# Efficient Robotics using the Lego NXT Platform and .Net

Rui Filipe Guedes <sup>(1)</sup>, Luis Paulo Reis <sup>(1,2)</sup> and Armando Sousa <sup>(1,3)</sup>

<sup>(1)</sup> FEUP - Faculty of Engineering of the University of Porto, Portugal

<sup>(2)</sup> LIACC - Artificial Intelligence and Computer Sci. Lab., Univ. of Porto, Portugal

<sup>(3)</sup> ISR-P - Institute of Systems and Robotics, Porto, Portugal

**Abstract.** This paper presents a project that intends to enable efficient implementation of Multi-Agent systems on the Lego NXT robot commercial platform, programmable by using the well known Microsoft .NET C# language. The work presented shows mainly the tools developed and the experiences performed. Some examples of the available APIs are mentioned and discussed. The developed tools will enable efficient programming and easiness of communication using Bluetooth technologies that, in turn, allow very easy development of remote user interfaces on PCs and on portable (PDA) devices. Some experiments on a single robotic agent are described proving the efficient deployment of several low-level skills in the context of the framework developed.

**Acknowledgements:** The research was supported by FCT under grant POSI/EIA/63240/2004. The authors wish to thank the Portuguese FCT - National Science and Technology Foundation for the fellowship granted.

## 1 Introduction

The field of robotics is composed by several branches, as whole it is very diversified and complex. This paper focuses mainly on software development and problem analysis as a commercial hardware platform is used.

The robotic platform of choice for this project is the Lego Mindstorms NXT. The standard Lego NXT package ships as a kit containing all the required mechanical parts and hardware (sensors, processor, and motors). The mechanical configuration has very high flexibility as is a trait for all the Lego products and presents itself as a worthy candidate in all limited complexity robotics projects. Lego has also made a great effort in developing an open platform to software and hardware developers[4] [5] [6] without compromising performance, which makes the NXT a good choice for this project.

The combination of Lego NXT and Microsoft C# is hoped to be the enabler of efficient intelligent robotics applications, even for multi platform problems.

The goal of this project is to enable efficient implementation of Multi-Agent robotic systems using NXT robots and C#. Although the set of solutions and approaches in study will be applicable to any context, we've decided to focus on

a single validation task: a physical simulation of a robotic rescue mission. Our agents will have to undertake several tasks in order to solve a physical version of the RoboRescue simulator [1] [18] [19].

At the current projects stage, we are focusing on the individual skills, that is, the development of agents that can navigate and interact with the world allows a solid base for all the multi-agent /cooperative components that are to be developed. Our work so far has allowed us to analyze and document several features, strong points and weak spots of the Lego NXT platform. It also has allowed us to develop and explore several ways to interact programmatically with the NXT.

Since this paper reflects the current state of our research project, we will start by introducing our motivation in the next section. Section 3 describes the development platform and the next section the Lego NXT programming methodology. Section 5 introduces our architecture for the software, the prototypes developed, the single-agent software analysis and the experiments performed. Section 6 describes the current work under development, related with this project and finally section 7 presents the conclusions and future work on this project.

## 2 Motivation

As for the motivational force that drives this project, it can be divided into three main goals:

1. To develop a fully functional multi-agent system, composed by agents that can interact with the physical world, deploying them on to a scenario that is usually used only on simulation
2. To perfect and study the single agents skills, the Lego NXT platform has its hardware limitations, so, the development of an agent that can navigate, follow waypoints, reach goals and take some mild decisions on its own poses an very interesting challenge
3. To document all the results obtained during this process, we hope that the knowledge that will be gained is of use to similar projects, so, we hope that we can give our contribution to efficient deployment of complex projects.

## 3 Development Platform

The Lego Mindstorms is a robotics kit for researchers, students and hobbyists alike, that allows the development of models that represent real-life robots. This makes studying and deploying basic robotics concepts and techniques easy and cost-effective. The mechanical flexibility is very interesting as a large number of configurations are possible and several types of sensors are available.

The first Lego Mindstorms kit, the RCX, was somewhat limited on connectivity, processing power, programming tools and computational power. In 2006 Lego launched the second Lego Mindstorms kit, the new NXT, a new processor that supports most of the object oriented programming languages and Assembly

with an open development philosophy. Also the new NXT comes with Bluetooth and a firmware developed to support multi-robot interaction. The ARM7 processor gives it a firm computational power, allowing it to execute its fair share of logic [7]. As for software deployment, the NXT provides a series of interesting solutions. From developing client applications in NXT-G (Lego visual language)[4] that will read from remote or local code files, to full remote applications that can run either on mobile devices like pocket pcs and smartphones or on desktop pcs.

One of the NXTs greatest advantages is its flexibility when it comes to programming languages, it supports from assembly to python, passing through C, C++, java or c#, it all depends on how the software is going to be deployed. There are many IDEs available, from Microsofts Visual Studio (c#, C++) to Assembly-based NxtByteCodes (NBC), and some less complex java-based interfaces.

The remote code deployment is possible due to the way that Lego developed the firmware on the processor: all the core functions can be set or read from an outside source. This allows the development of software in virtually any object oriented language possible, as long that we create a way to pass information to and from the Assembly functions on the processor. The practical result is that the processor doesnt know where he is getting the information from or to where he returns it. This allows two distinct layers, one for high level operations like coordination, decision, etc and another for low level operations like giving the run order at an engine or reading a sensor output.

## 4 Programming

On our work so far, weve done a series of testing and analysis in order to determine the programming language and IDE to use for efficient deploying. An interesting challenge was to take a commercial language like C# and use it to program a robot. Our previous experience with C# showed that it is a very versatile language, the .NET framework gives it the resources to tackle complex distributed computation with the same efficiency that it tackles web development or win32 applications, so it seemed that it would be interesting to find out how far we could take C# and how would it handle the requirements for our software.

The adopted software solution is based upon the usage of an open source library, developed by Niles K. Handest [16] [17] that handles the conversion of data between the C# code and the assembly functions on the processor. We have modified this library in order to use it on the .Net Compact Framework 2.0, which is the framework for mobile devices such as pocket pc's and smartphones, since the original one was developed for the .Net Framework 2.0. Weve also made some minor changes to some of its methods in order to further tailor it to our needs.

The code developed so far is mainly event-driven and we can set several handlers per sensor, depending on the available functions and call-backs defined

by Lego. It is also possible to implement state machine workings where events cause state transitions. The states call C# call-back functions that define the high-level skills developed so far. Our current work is focused on refining the existing methods to allow a better analysis of the context in which the event was raised, thus improving the responses that our agent will give.

## 5 Prototypes and Results

In our preliminary work, several software prototypes were developed, with several local code execution tests and a client PC controlling a remote application on the NXT robot. The version used on our tests was the third prototype, which consists in software deployed from a pocket PC with windows mobile that uses Bluetooth communications for remote command of the robot. The software running in the PDA had full autonomous control of the hardware, acting like a "remote brain". The robot used the "Tribot" (see figure 1) which is the Lego default configuration for a vehicle-like robot but with pincers. The mentioned mechanical setup features the following sensors:

- Ultra-sound Sensors;
- Touch Sensors;
- Light Sensors;
- Sound Sensors.

The available actuators are:

- Two motors (synchronizable by software command);
- Two Pincer Devices.

Additional non-standard sensors and actuators are available.

The main goal of this trial run was to experiment with all the Lego sensors, in order to determine how to use them on C#, how to process their outputs and how to calibrate them programmatically.

As mentioned earlier, we have built a test application for validation purposes. For this end, we've focused on developing a roaming algorithm using a simple reactive architecture that would raise different behaviors from the robot, depending on the input we provided. It was determined that the Ultrasound sensor would be the robot's "eyes" (remote sensing), the light sensor would be used for color recognition, the touch sensor for grabbing objects and the sound sensor to activate a victory dance or halt.

The software passed through three different versions, the first two focused on basic operations and platform differences (desktop, pocket pc), the third was solely for pocket pc and had the complete algorithm.

As metrics of the work done, it can be mentioned that the mechanical setup of the robot is as short as 1 hour. The main core of the work done was porting the mind squalls library to the PDA architecture. This enabled the development to be made on the PC or PDA with client application executing on the PDA. The used PDA was a Fujitsu Siemens with Bluetooth communications. The fact that

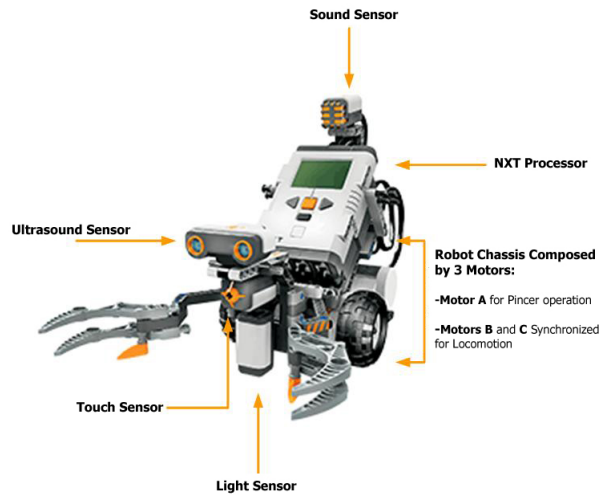
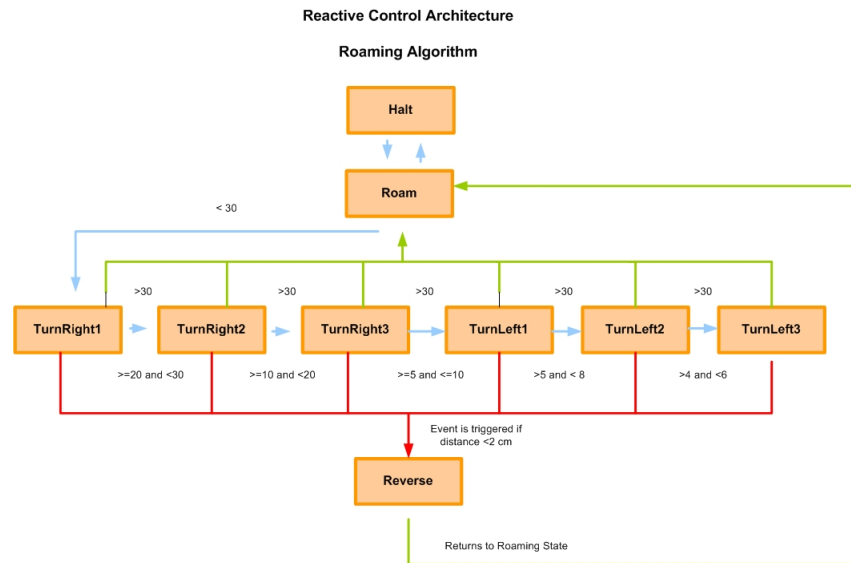


Fig. 1. Overview of Lego's Tribot

a very portable mobile control center is used may be very interesting for easily validating performance and future debugging of the application as it is perfectly possible to move robots and control center to new mission sites effectively.

The mentioned limited complexity application was efficiently developed (less than 2 days) and the client part running on the PDA has only about 400 C# lines of code. The performance was adequate for the applications, and remote code execution via Bluetooth provided responses that could be considered "real-time" for the mission at hand - roaming, obstacle avoidance and grabbing objects. As for the algorithm, the basic logic consisted in a close relationship between ultrasound sensor output and synchronized motors, the "object detected event" would call a C# function that would compare the measured distance with hardcoded values (in centimeters), for each set of distance intervals, to produce transitions for the state machines. The software's architecture is represented schematically on figure 2 for the roaming behavior. The test application also features a very simple behaviour of colored ball holding - if color is interesting, then catch it, otherwise it is an obstacle and will be avoided.

However, there were more sensors involved, so, when the touch sensors "On-Pressed" event was triggered, all others were suppressed, the light sensor would then project a beam and tried to evaluate the intensity of the light, which would raise another event that called an function to compare the obtained intensity with a hardcoded value, if the value was inside the interval, a "Grab" function would activate the pincers, if not, the robot would resume normal navigation. As for the sound sensor, it can raise an event every time some sound is received, if the received sound match clapping hands, it would call a function to perform



**Fig. 2.** Roaming algorithm reactive architecture

a victory dance and resume navigation, if it matched a sharp sound, it would call the halt function.

In spite of the relative success obtained on deploying a robot with a simple roaming algorithm that processes information from four different sensor types in an efficient way, there were some problems and difficulties that are worth mentioning.

The first one was sensor calibration, the NXT light and sound sensors are quite sensitive, and so, using thresholds to define intervals can be problem-prone solution. The light sensor readings suffer considerable variations, depending on the environment lighting, even if it is reading from its own light beam. The sound sensor lacks resolution when processing the intensity of a given sound, the readings can vary by changing the direction of the sound source.

The Ultrasound sensor however had great results, its measures were accurate and it is capable of stacking distances for eight different objects per reading, this opens up a series of possibilities if some more navigational aides are developed. The touch sensor also performed well, providing instant response to contact. The engines also had an interesting performance, they carry a sensor that keeps track of the current angle and direction of a given rotation (movement), this feature was used at full extent, providing some interesting solutions to handle the overall engine logic.

In a general way, the prototype had satisfactory results, the essential components of the roaming algorithm performed well (ultrasounds, engines, events). This confirmed that this platform has all the necessary requirements to our long

term goals project, the efficient deployment of Multi Agent systems. All the detected sensor problems could probably be overcome by adding some helper functions to our code with logic to assist on the output processing, so the conclusion is that there aren't any "core" handicaps or severe conception flaws on the platform that would interfere with the development of the single-agent software and the Multi-agent system.

## 6 Current Work

As mentioned earlier, the long term goal of our project is to make the foundations for the development of efficient Multi-Agent robotic platforms. The current goal is, however, to validate the usefulness and efficiency of a single-agent robotic platform ready for cooperation. On the current time we are undergoing several researches and testing tasks related to the development of individual skills, using the third prototype mentioned on the previous section as a base for all the forthcoming improvements and breakthroughs.

The roaming algorithm we've referred previously is one of those skills, others are Spatial Navigation, Guidance and Object Interaction. At this time, we've granted Navigation and Guidance top priority, since they are essential for the scenario in mind and they represent a basic skill for any robotic agent.

For the development of an Navigational and Guidance system, we've acquired a Magnetic Compass manufactured by one of Lego's "certified" hardware developers for NXT [20], this sensor has full integration with any of Lego's IDE's and it follows the same digital sensor template used by the Ultrasound sensor, allowing it to be integrated on the library discussed on section 4 of this paper. In addition we are pursuing several logic paths to determine the most effective solution in terms of integrating the compass sensor with existing algorithms. The existing roaming algorithm is also suffering a revision in terms of structure, since we've decided to improve it, making it more versatile in terms of programming logic. Some traditional decision algorithms are also under study for these improvements, although the usage of most of them in the referred context would be overkill, we are trying to determine if any of them would fit our needs or if it can be simplified to fit. A classic programming toolkit is the state machine approach that can easily be deployed in C# language.

On the navigation chapter, it's also worthy to mention that we are also undergoing a series of experiments using the NXT's Bluetooth device and other devices (laptop internal Bluetooth device, usb dongles) in order to study and evaluate the possibility of developing the Bluetooth positioning system by triangulation [12] [13]. In this stage we are trying to develop methods that determine distance between two Bluetooth devices, our control value was measured physically, the goal is to get a match or near-miss with an acceptable error margin.

## 7 Conclusions

This project has a series of interesting goals from the academic, technical and scientific points of view. The work produced shows that indeed the NXT platform and the C# language provide for efficient deployment of limited complexity intelligent robotics applications, for it has allowed us to develop, implement and test a working algorithm with full sensor and actuator support with relatively few development hours. Command software can be on a PC platform and, even more interestingly, on a PDA platform.

The Lego NXT platform also gave us the privilege of choice both on the type of processing and on the deployment platform. For processing we could use remote, distributed or local, meaning that we were free to develop software that would control the robot from another device, from a group of devices or deployed on the robot itself, just a matter of choice, since we could also choose the programming languages to use. For remote or distributed processing, we could either use a Desktop Pc (any OS, as long it supports Bluetooth), Pocket Pc or Smartphones.

The second proof of efficiency was obtained by the choice of programming language. We've decided to use a commercial language such as C#, which carries native support for various programming contexts via the .Net Framework 2.0 and the .Net Compact Framework 2.0, at almost no effort we were able to develop an simple pocket pc application in C# that used our custom library to communicate with the NXT. Our only real concern during the development was the structure of the roaming algorithm and sensor testing/calibration, the NXT-C# solution took care of deployment, configuration, compatibly and communication issues by itself.

As for the deployment method that was chosen, using a mobile device allowed us more flexibility, especially during tests. It also demonstrated that it is possible for future robotics software to keep up with the current technological trends of integrated mobile devices, we can now carry our robot's software on our Smartphone or other mobile device. Furthermore a mobile device is smaller and more suitable to be integrated on a robot's chassis or testing environment.

The current status of the project is to further document, experiment and develop the concepts and solutions presented on this paper. From the technological point of view, so far the platform is a solid base for our project. The programming language is also very suitable, at first it might not seem the perfect candidate, but so far not only it came through, as it opened very interesting possibilities in terms of distributed systems integration with our robots.

The prototypes and tests have been very valuable for our study, thanks to good integration between hardware platforms and programming language, we can focus on the high-level aspects of the project and other related issues, since we don't have to worry all the time about software/hardware conflicts, configurations, etc.

Looking at this stage as a preparation for the Multi-Agent System Development, it's possible to infer that we will have a good starting point for future needs. This first stage will make us explore and test the platform to its full

extension, thus providing valuable knowledge about every aspect. This will be important information for later stages of the project, where inner workings of our robots should be abstracted by consolidated and well documented software calls. Future developments will focus on communication and coordination among several agents.

## References

1. Joao Certo, Nuno Cordeiro, Francisco Reinaldo, Luis Paulo Reis e Nuno Lau, FCPx: A Tool for Evaluating Teams' Performance in RoboCup Rescue Simulation League, in Gelbukh, A. and Reyes-Garcia, C. eds, Special Issue: Advances in Artificial Intelligence, Research in Computing Science, Vol. 26, pp.137-148, November 2006, ISSN: 1870-4069
2. Luis Paulo Reis, Nuno Lau, Francisco Reinaldo, Nuno Cordeiro and Joo Certo. FC Portugal: Development and Evaluation of a New RoboCup Rescue Team. 1st IFAC Workshop on Multivehicle Systems (MVS'06), Bahia Convention Center, Salvador, Brazil, October 2-3, 2006
3. Francisco Reinaldo, Joao Certo, Nuno Cordeiro, Luis Paulo Reis, Rui Camacho, Nuno Lau. Applying Biological Paradigms to Emerge Behaviour in RoboCup Rescue Team. In Carlos Bento, Amilcar Cardoso and Gael Dias (Eds.): Progress in Artificial Intelligence, 12th Portuguese Conference on Artificial Intelligence, EPIA 2005, Covilha, Portugal, December 5-8, 2005, Vol. 3808, pp. 422-43487. Springer LCNS, 2005, ISBN 3-540-30737-0
4. Lego *Writing Efficient NXT-G Programs* 2006: Lego
5. Lego *Lego Mindstorms NXT Bluetooth Developer Kit* 2006: Lego
6. Lego *Lego Mindstorms NXT Hardware Developer Kit* 2006: Lego
7. Lego, Lego Mindstorms Website <http://mindstorms.lego.com/> 2006: Lego
8. Steffen Nissen, Steffen Larsen *Real-time image processing on iPAQ based robot (iBOT)* 2002: University of Copenhagen
9. A.Grosso, A.Grozzi, M.Coccoli e A.Boccalatte *An Agent Programming Framework Based on the C# Language and the CLI* 2002: University of Genova
10. Anders Lemke, Johan Laidlaw e Lars Zilmer-Pedersen *Developing Multi-Agent Lego Robotics* 2007: Danmarks Tekniske Universitet
11. Stuart Russel, Peter Norvig *"Artificial Intelligence: A modern Approach" (2nd Edition)* 2004
12. Josef Hallberg, Marcus Nilsson, Kare Synnes *Positioning with Bluetooth* 2003 :Lulea University of Technology/Center for Distance-spanning Technology
13. Kenneth C. Cheung, Stephen S. Intille, Kent Larson *An Inexpensive Bluetooth-Based Indoor Positioning Hack* 2006 :Massachusetts Institute of Technology
14. Microsoft, Microsoft Robotics Studio Website <http://msdn.microsoft.com/robotics> 2006: Microsoft
15. Microsoft, Microsoft Robotics Studio On-Line Tutorials <http://msdn.microsoft.com/robotics/learn/default.aspx> 2006: Microsoft
16. Niles K. Handest, Mindsqualls Library Homepage <http://www.mindsqualls.net> 2006
17. Niles K. Handest, Mindsqualls Library On-Line Tutorials <http://www.mindsqualls.net/samples/NxtRemote1.aspx> 2006
18. RoboRescue Website <http://www.robocuprescue.org/> 2007

19. RoboRescue Simulator Download <http://sourceforge.net/projects/roborescue/> 2007
20. HiTecnica <http://www.hitechnic.com/> 2006