Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Embedded Implementation of Mobile Robots Control

A. Valera, M. Vallés, P. Albertos, R. Simarro, I. Benítez, C. Llácer

Instituto Automática e Informática Industrial
Universidad Politécnica de Valencia, Valencia, Spain.
P.O.Box 22012 E-46071
e-mail: {giuprog, mvalles, pedro, rausifer, igbesan}@isa.upv.es,

Abstract: In some embedded applications, the global control objective can be split in different control objectives from simpler to more complex ones. In this sense, simple controllers can achieve the basic control problems and controllers with more computational resources can be in charge of more complicated decisions. In these situations, a control architecture for coordinating all the components is needed.

This work will present a distributed network-based control of mobile robots, each of them having embedded control system based on a microcontroller. In the control architecture proposed, in addition to the mobile robots, there is a PC working as a server and a set of PCs to control the mobile robots.

A wide range of different control tasks can be done because all the computers are networked connected. The paper will also show some control examples like mobile robot control, automatic path generation free of collisions, control algorithms based on agents, etc.

1. INTRODUCTION

The strong increasing presence of Embedded Systems (ES) in products and services creates huge opportunities for the future in different areas such as industrial control systems, avionics, health care, environment, security, mechanics, etc. (Chinook, 2007).

ES normally impose a strong constraint in size and weight, assuming a high level of autonomy and distribution. For different reasons (distance, location, etc.), ES must be adapted to changing conditions or update, and because it can operate on critical systems that may cause damages to people or to itself, ES require very extensive testing to satisfy a set of strict rules. In addition, a process is needed, in order to detect and perform actions to reduce the effect of failures to continue to work in degraded or faulty mode (ARTIST, 2007).

Nowadays, there is a growing scientific interest on conceptual and practical tools for ES development. In particular, their use in control applications (and especially, robot control applications) is becoming very popular (Baliga, Kumar, 2005)

From the robot control algorithm point of view in ES, specific design methodologies should be use like fault-tolerant control, multimode control, decision and supervisory control, reduced order models, event-triggered control, sampling rate changes and non-conventional sampling and updating patterns, battery monitoring and control, degraded and back-up control strategies, etc. (Albertos *et al.*, 2005).

This paper presents an embedded implementation of mobile robots control. The robots are based on LEGO Mindstorms and K-Team Khepera II, and a robot control architecture is proposed.

In order to demonstrate the possibilities of the robot control system provided, different embedded robot control examples are presented at the last part of the paper.

2. MOBILE ROBOTS WITH LIMITED PROCESSING CAPACITY

2.1 Introduction

Nowadays it is possible to acquire several very interesting platforms based on robotics (ActivMedia's Pionner robot (ActivMedia Robotics Web site), MIT's HandyBoard and Cricket controller cards (The Handy Board home page), The LEGO Group's LEGO Mindstorms (LEGO Mindstorms home page), Khepera robot (K-Team corporation home page), etc.). These platforms usually consist of controllers, electronic sensors, mechanical parts and/or small robots. Some of them perhaps do not provide the same precision than industrial robots, but they are sufficient for research and educational purposes.

In order to program and to control these mobile robots, they are equipped with microcontrollers or processors. Despite depending on the processor the mobile robots can do complex tasks, although in general all these kind of robots have limited capacity of processing, therefore serious difficulties can be found with the implementation of some functionalities.

In order to analyze and compare the performances and characteristics of the system, two different robotics platforms have been used: the LEGO Mindstorms and the Khepera II K-TEAM solutions.

2.2 LEGO robots solution

The 2006 LEGO Mindstorms set is an educational toy for children aged 8 years and older. Apart from those familiar beams, bricks and gears, the kit contains dc motor actuators, a range of sensors, and, the most interesting, the NXT Intelligent Brick component. The NXT is the LEGO's programmable brick that allow models not just to move, but to sense and respond to their environment. It is based on an ARM7 microcontroller. This 32 bits CPU provides most of the control logic for the NXT, including serial I/O, Analog to Digital Converter and built–in timers. It even contains 256Kbytes of FLASH memory and 64Kbytes of static RAM. In addition, there is an interface to three output ports and four input ports (one includes an IEC 61158 Type 4/EN 50 170 compliant expansion port for future use), one Bluetooth wireless communication (Bluetooth Class II V2.0 compliant) and one USB full speed port (12 Mbit/s) to communicate with a desktop computer or another RXC) as well as a 100 x 64 pixel LCD graphical display. The NXT also has a loudspeaker - 8 kHz sound quality with a sound channel with 8-bit resolution and 2-16 KHz sample rate.

The LEGO Mindstorms system was initially developed as an educational tool through the collaboration of LEGO and MIT (Resnick *et al.*, 1996), (Papert, 2000). The initial intended use of the LEGO system was for research and educational activities. The combination of the versatile construction blocks with the easy-to-use programming and I/O interfacing of the NXT provided a fast prototyping system to support these activities. Although the commercialization of this product has focused on the recreational and K-12 educational markets, the flexible and expanding world of LEGO Mindstorms is widely accepted as a tool for research and higher education.

Enthusiasts have extended the hardware and software in various ways (Gasperi *et al.*, 2007), (Hansen, 2007), (Valera *et al.*, 2005). Recent issues of IEEE Robotics and Automation Magazine (Geenwald, Kopena, 2003), (Weinberg, Yu, 2003), (Klassner, Anderson, 2003) or IEEE Control Systems Magazine (Gawthrop, McGookin, 2004) argue that these extensions show that the LEGO Mindstorms kit can be used to good effect in an education context. In particular, the kit is relatively cheap, robust, reconfigurable, reprogrammable, and induces enthusiasm and innovation in students.

In addition to the construction blocks, the components of the LEGO Mindstorms can be categorized under the headings of sensors and actuators. The NXT has four sensor ports, each of which can accommodate one of the different LEGO sensors: touch, light, sound, and ultrasonic visual sensors. In addition, there are rotation sensors (rotational encoder) built into the motors that provide the position of the shaft with 1º resolution. Information about these and new sensors can be found in (Gasperi's LEGO Minsdstorms web page) or (Tehno-stuff web page).

The standard LEGO actuator component is a good quality permanent-magnet dc motor with high inertia and low friction. NXT actuators deliver a high torque thanks to their internal speed reduction gear train. Powered by a 9V, they can provide torques about 16.7N.cm working at 117rpm with 0.55A current consumption.

A more detailed description of the LEGO Mindstorms actuator can be found in (Lego Technic Motors web page).

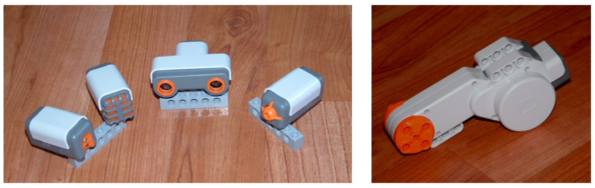


Figure 1. LEGO NXT sensors and actuator

2.3 Khepera robots solution

Khepera II is a miniature mobile robot with functionality similar to that of larger robots used in research and education. Khepera was originally designed as a research and teaching tool for a Swiss Research Priority Program at EPFL in Lausanne. It allows real world testing of algorithms developed in simulation for trajectory planning, obstacle avoidance, pre-processing of sensory information, and hypotheses on behaviour processing, among others.

Very modular at both the software and hardware level, Khepera has a very efficient library of on-board applications for controlling the robot, monitoring experiments, and downloading new software. A large number of extension modules make it adaptable to a wide range of experimentation.

The Khepera robot is a miniature, circular, compact and robust robot. It has a diameter of 70 mm, it is 30 mm high and its weight is 80g. The robot is supported by two wheels and two small Teflon balls placed under its platform. The wheels are controlled by two DC brushed servo motors with incremental encoders (12 pulses per mm of robot advancement) and they can rotate in both directions. This geometrical shape and the motor layout of Khepera provide an easy negotiation of corners and obstacles.

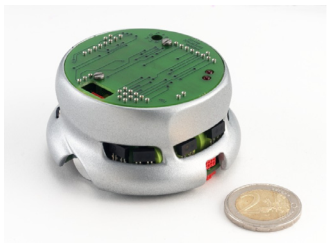


Figure 2. Khepera mobile robot

In its basic version it is provided with eight infrared proximity and ambient light sensors with up to 100mm range. These sensors are placed around its body (six on one side and two on the opposite one), and they are based on emission and reception of infrared light.

The robot is equipped with a Motorola 68331, 25MHz processor, with 512Kbytes of RAM memory and 512Kbytes of Flash memory programmable via serial port.

3. MOBILE ROBOT CONTROL ARCHITECTURE

3.1 Control architecture

In order to establish the mobile robot control implementation, this work proposes the control architecture shown in figure 3. There is a PC working as server that it is equipped with a webcam. This server processes the vision images and provides (via Ethernet) the interest information to the control architecture computers.

Because the robots used in this platform are small robots with a limited capacity of processing, it has been chosen to use remote computers to execute the control tasks that require a great processing To do this, they obtain the server information, calculate the control actions and/or the required instructions and send them to the mobile robots using wireless communications.

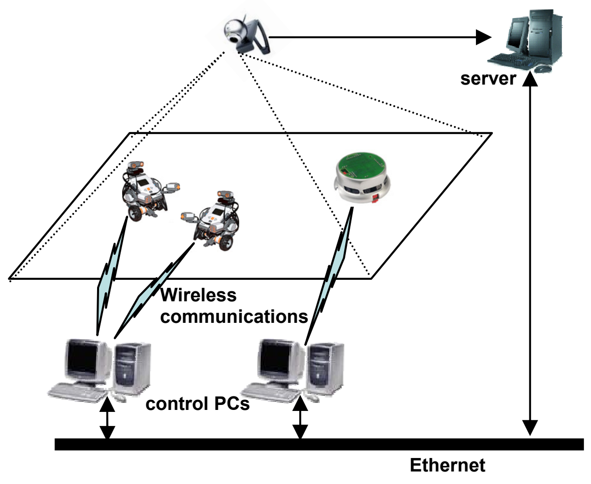


Figure 3. Mobile robot control architecture

The information that the control PCs send to the mobile robots depends on the kind of control application. For example, by means of the wheels encoders information and the kinematic model of the robot, the position of the robot within the environment is considered. From the desired trajectory and with the kinematic controller, the remote computer calculates the reference velocities for the two wheels of each mobile robot. In this kind of control applications, normally a simple dynamic velocity controller (like a PID controller) is found in these mobile robots. This controller must verify that the robot wheels follow the velocity references as close as possible.

Because the PCs have more calculus performance, they can compute any vectorial or trigonometrical calculations without any problem, solving the typical limitations that usually appear when trying to implement them in the embedded systems, so more complex controllers or/and applications can be developed. Figure 4 depicts the control scheme implemented in this architecture.

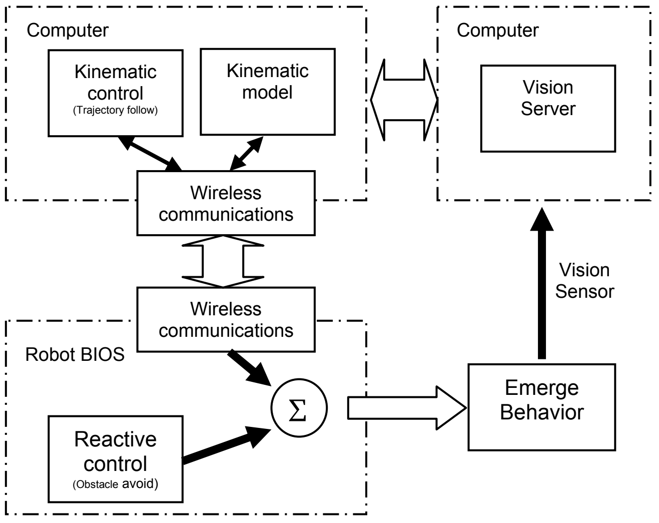


Figure 4. Mobile robot control scheme proposed

On the other hand, this control architecture also allows the PC to start and stop different control programs previously stored in the robot microcontroller. The robot local control programs can use the mobile robot sensors (distance, orientation, inclination, etc. sensors), so a very wide range of activities can be done, e.g. automatic collision avoidance, pre-programmed movements, etc. The process of stopping one program and starting a new one is very fast, so these changes cannot involve a great deal of drawbacks for the robot control.

In addition, the PC can also compile, send and store a new control program to the robot. However all these actions require a little more of time, so it is recommended to do this only when the mobile robot is in several (secure) conditions.

3.2 Programming environments

For the control architecture implementation, different programming environments can be used for each kind of mobile robots. For the LEGO robots, the NXC and the C# of Visual Studio 2005 languages have been utilized.

The first one LEGO programming environment is NXC (Not eXactly C). It is a high level language to program the LEGO microcontroller that provides arrays, functions, tasks, control flow sentences, sensors and actuators access, communication system, etc. (Next Byte Codes web page).

For the NXC routines programming there are different solutions, but the more used is Bricx Command Center (Bricx web page). This is a free software environment that allows not only the edition of the programs, but also different tools, for instance, to compile and transfer the programs from the remote PC to the LEGO microcontroller.

The second language used for the LEGO programming is C# language of Visual Studio 2005 environment. This language is an object oriented language developed and standardized by

Microsoft as a part of its .NET platform. .NET is intended as a new software development platform with emphasis in network transparence, platform independence and that allows developing applications quickly.

Using the C# language, a supervisor has been implemented to analyze the information of the robots positions through the webcam images. It sends the corresponding trajectories to the control PCs using a connection based on TCP/IP sockets. With the server and the local sensors information, the PCs control the movement of the corresponding LEGO NXT. To do that, they send the adequate information (velocities references, commands to change the NXT control algorithm, etc.) through the Bluetooth communication channel.

For the Khepera robot, different remote control software, cross-compilers and simulators can be used. The remote control software allows the remote control, interaction and visualisation of the robot behaviour from the remote computer. These programs are generally standard software tools which communicate with the robot to send commands to the Khepera and control motors, sensors and all other capabilities of the robot. Of this type of programs the Khepera support, for example SysQuake, LabVIEW® and MATLAB®.

The Cross-compilers allow to generate code that is executed on the robot itself. Khepera currently provides libraries and documentation for the GNU C public domain cross-compiler.

In order to implement the Khepera control algorithms, in this work the complete development environment called KTProject has been used. It is a graphical C development environment for Windows that includes the GNU C compiler, the Cygwin Environment (copyright RedHat Inc.), Source Navigator (copyright RedHat Inc.) and KTDebug, a serial port terminal written in Java (copyright J-M Koller).

Low level functions are provided for interactions with motors and sensors, for communications with extensions or a host computer, and for basic multitasking management. (Bureau 2002).

Figure 5 shows the basic subdivision of the BIOS. Different managers were designed to control only a specific part of the system (K-Team, 1999).

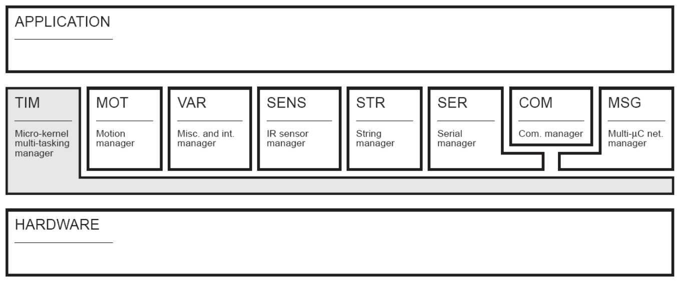


Figure 5. General topology of the BIOS

The code is completely relocatable and is designed to allow an easy interface with a high level language such as C.

As already mentioned, each physical part of the system is under the control of a specific manager. Here is the list:

BIOS: Global core of the BIOS
COM: I/Os communication manager
MOT: Khepera movement manager
SENS: infra-red sensor manager
MSG: multi-microcontroller communication manager
VAR: misc device manager (jumpers, LEDs, etc.)
TIM: multitasking manager
SER: serial link manager
STR: string manager

4. EMBEDDED ROBOT CONTROL EXAMPLES

4.1 Path generation and trajectory robot control

In order to test the control architecture proposed, some path generation and trajectory robot control problems have been developed (Valera *et al.*, 2007). The first one is shown in Figure 6, where two different trajectories have been used for the LEGO. In this case the robot references are periodic and non-periodic curves, and the figures contain the references, the real robot system response, and the simulated result of a robot model. The real robot system obtains better response than the simulated system.

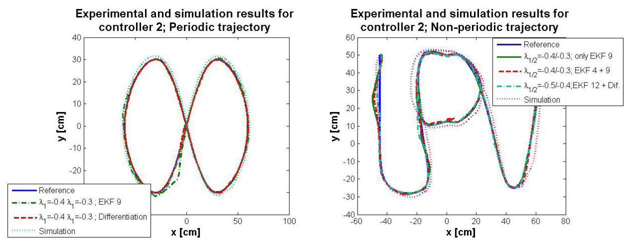


Figure 6. Reference and LEGO mobile robot response

Using artificial vision, the webcam not only is used to obtain the robot position, but also to detect the obstacles, so it is possible to obtain a path by cell decomposition methods (Nourbakhsh, Siegwart, 2004).

Figure 7 shows the image obtained by the webcam, the path generated and the error between the reference and the LEGO robot movement.

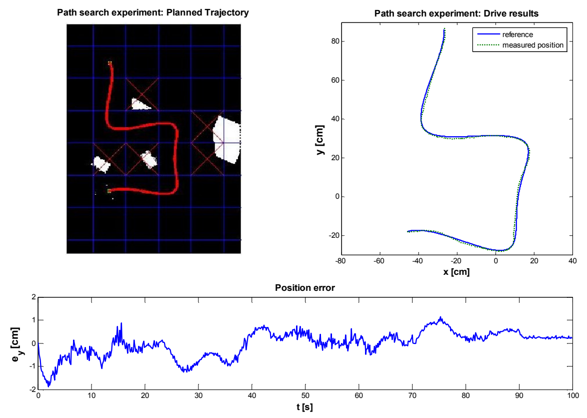


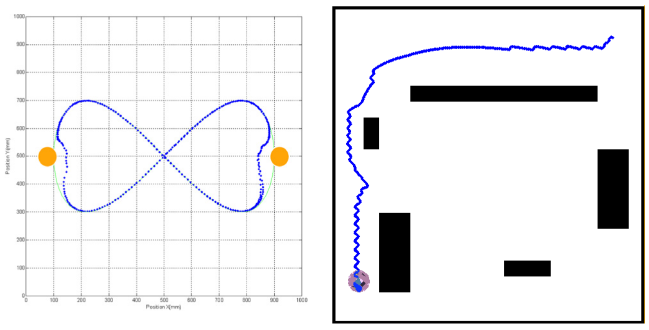
Figure 7. Reference and mobile LEGO robot response

4.2 Distributed structure behaviors based

In the next example, a distributed structure behaviour based is used. The mobile robot (a Khepera) must follow a trajectory and avoid the moving obstacles of the environment.

In order to avoid the obstacles not contemplated in the trajectory planning, a reactive control of obstacles avoidance is used. It is based in the reactive control model of the Braitenberg vehicles (Braitenberg 1984), and that is implemented within the robot BIOS. This task is high-priority with respect to the trajectory pursuit, and therefore, when approaching an obstacle the robot reacts avoiding it and trying to follow, as far as possible, the trajectory planned initially.

Another test that can be made within the same scheme is switching two different controls, one implemented in the remote computer and other simpler within the robot BIOS. According to the available processing time at every moment and the tasks that are being executed, a decision can be made on the execution of one control or the other. The objective always will be to send a control action to the motors of the wheels to follow the pre-selected target or in last instance to take the robot to a safe situation. An example of this implementation is displayed in Figure 8 (a) and (b).

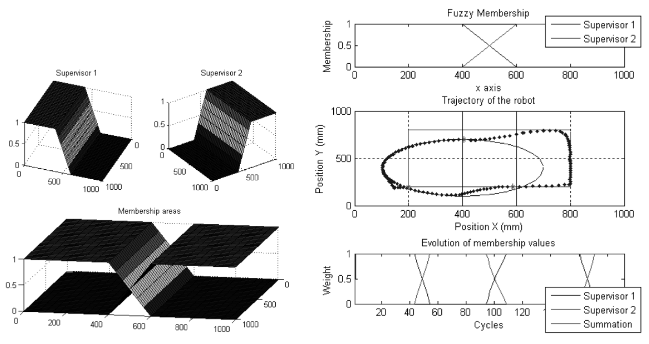


(a) (b)

Figure 8. (a) Trajectory pursuit with two not planned circular obstacles (b) To reach an objective in a nonstructured environment.

4.3 Multiagent systems

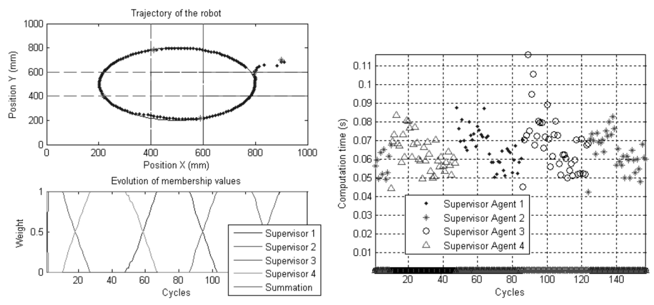
The last example applies the principles of multiagent systems (Ferber, 1999) to build a distributed control architecture based on multiple supervisor agents. The control and supervision of a group of mobile robots is carried on by the supervisor agents according to the position of each robot. Each agent takes the control of a mobile robot whenever the robot moves in a region of its influence. The available environment is distributed into areas of influence by the use of fuzzy membership functions (see Fig. 9(a)). This method allows an intermediate transition state between two regions of influence, therefore allowing a soft change from the influence and control of one agent to the other.



(a) (b)

Fig. 9. Experiment with two supervisor agents. Fuzzy membership functions (a), and results of experiment (b)

Figure 9 depicts an example of this implementation, designed with the KIKS (Nilsson, 2001) tool. Two supervisor agents are implemented, each one with its own area of influence, delimited by a fuzzy membership function (Fig.9(a)). The trajectories defined in both agents are different, being one circular and the other a square trajectory. Figure 9(b) displays the result of the control experiment on a Khepera robot, with the two agents interchanging the control duties, and a transitional state between the two trajectories.



(a) (b)

Fig. 10. Results with four supervisor agents: robot trajectory (a), and computation time of each agent (b)

Figure 10 is another example of this implementation, in this case with four supervisor agents and their corresponding areas of influence, placed at each of the four corners of the simulation environment. The agents interchange the control on the mobile robot as it moves through their influence regions. Figure 10(b) displays the distribution of computation time or cost dedicated by each agent on the control of the robot. The multiagent design allows a parallel allocation task of secondary computations (such as future control actions) on idle agents or processors.

5. CONCLUSIONS

A new architecture for mobile robots applications has been presented. This architecture allows implementing control activities from simple to more complex ones.

As example of the capabilities of this architecture several mobile robots control applications have been developed. These deal with aspects such as distribution of control tasks, switching between controllers and an application with multiple supervisor agents.

For these examples two kinds of mobile robots have been used: a LEGO based one and a Khepera one; the development environments used for their programming have also been described.

ACKNOWLEDGEMENTS

This work has been partially funded by FEDER-CICYT projects with references DPI2005-08732-C02-02 and DPI2005-09327-C02-01 financed by Ministerio de Educación y Ciencia (Spain).

REFERENCES

ActivMedia Robotic Web site: http://www.activrobots.com/

Albertos, P., A. Crespo, M. Vallés and I. Ripoll (2005). "Embedded control systems: some issues and solutions". *16th IFAC World Congress*, Praga (República Checa)

ARTIST. Advanced Real-Time Systems. Selected topics in Enbedded Systems Design. European Project IST-2001-34820 (http://www.artist-embedded.org/), 2007

Baliga, G., P.R. Kumar (2005). "A Middelware for Control over Networks", *44th IEEE Conference on Decision and Control and European Control Conference ECC*

Braitenberg, V. (1984). "*Vehicles*". MIT Press.

Bricx Command Center, http://bricxcc.sourceforge.net/

Bureau, P. (2002). *"Khepera 2 programming manual"*. K-Team S.A.

Chinook Project webpage: Embedded System Links, http://www.cs.washington.edu/research/chinook, 2007

Ferber, J. (1999). *"Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence"*. Addison Wesley.

Gasperi's LEGO Mindstorms NXT/RCX Sensor Input Page: http://www.extremenxt.com/lego.htm, 2007

Gasperi, M., I.L. Hurbain, P.E. Hurbain (2007). "Extreme NXT: Extending the LEGO Mindstorms NXT to the next level". *Technology in action press.*

Gawthrop, P., E. McGookin (2004). "A LEGO-Based Control Experiment", *IEEE Contr. Syst. Mag.*, vol. 24, no. 5, pp. 43-56.

Greenwald, L., J. Kopena (2003). "Mobile robot labs", *IEEE Robot. Automat. Mag.*, vol. 10, no. 2, pp. 25-32, 2003.

Hansen. J.C. (2007). "LEGO Mindstorms NXT power programming". *Varian Press.*

Klassner, F., S.D. Anderson (2003). "LEGO mindstorms; Not just K-12 anymore", *IEEE Robot. Automat. Mag.*, vol. 10, no. 2, pp. 12-18.

K-Team corporation home page, http://www.k-team.com

K-Team (1999). "Khepera BIOS manual". K-Team S.A.

LEGO Mindstorms home page, http://mindstorms.lego.com

Lego Technic Motors compared characteristics, http://www.philohome.com/motors/motorcomp.htm

Next Byte Codes & Not eXactly C. http://bricxcc.sourceforge.net/nbc/

Nilsson, T. (2001). *"KiKS is a Khepera Simulator"*. PhD Thesis, Umea University.

Nourbakhsh, I. R., R. Siegwart (2004). *"Introduction to Autonomous Mobile Robots"*. Campridge MIT Press, MA.

Resnick, M., F. Martin, R. Sargent, and B. Silverman (1996). "Programmable bricks: Toys to think with", *IBM Syst. J.*, vol. 35, no, 3&4, pp. 443-452.

Papert S. (200). "What's the big idea? Towards a pedagogy of idea power", *IBM Syst. J.*, vol. 39, no, 3&4, pp. 720-729.

Valera, A., M. Vallés, J.L. Díez, C. García (2005). "Development of Bluetooth Communications for LEGO-Based Mobile Robot Laboratories", *44th IEEE Conference on Decision and Control and European Control Conference ECC*

Valera, A., M. Weiss, M. Vallés, J.L. Díez (2007). "Control of Mobile Robots using Mobile Technologies", *Int. J. Engng. Ed.,*vol. 23, no. 3, pp. 491-498.

The Handy Board home page: http://www.handyboard.com/

Tehno-stuff Robotics, http://www.techno-stuff.com, 2007

J.B. Weinberg and X. Yu (2003). "Robotics in education: Low-cost platforms for teaching integrated systems", *IEEE Robot. Automat. Mag.*, vol. 10, no. 2, pp. 4-6.