Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Robotic Competitions: Teaching Robotics and Real-Time Programming with LEGO Mindstorms

Raffaele Grandi and Riccardo Falconi and Claudio Melchiorri *

* *DEI, Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Italy; e-mail: {raffaele.grandi, riccardo.falconi, claudio.melchiorri}@unibo.it*

Abstract: A laboratory activity started a few years ago within a course of Foundation of Industrial Robotics for the Master Degree in Automation Engineering at the Engineering School of the University of Bologna. The goals were on one side to drive students to acquire practical knowledge on mobile robotics, on the other to teach them how to write efficient code for real time control of automatic machines and robots. The tasks assigned to the students required them to mechanically design and program autonomous robots to be used in a competition among teams. Because of its low-cost, the modularity and the possibility of building several different devices, the Lego Mindstorms Kit has been adopted. With respect to the software, the Java-based firmware LeJOS has been selected as a tool to teach how to develop structured software. The experience has being very positive both for the students and the teaching staff. Besides acquiring many technical skills, the students felt involved in practical activities, thus increasing their interest for the subject.

1. INTRODUCTION

The design of modern automatic machines and robots is based on different and complex competencies deriving from mechanics, automatic control and computer science. In high-level education it is therefore advisable to organize teaching activities to let students acquire not only the basic knowledge, but also some more advanced competencies for the integrated design of automatic machines and real-time, object-oriented control software. From this point of view, mobile robots are a good example of automatic machines in which the mechanical design (e.g. number and positioning of actuators or sensors) must be accompanied by a clear design of the control software, being aware that choices made in one of the two areas have strong implications also on the other one.

The pervasive growth of micro-controllers and microchips has allowed to use quite powerful controllers in advanced toys such as Lego Mindstorms, making available to a wide public the possibility of designing and controlling relatively complicated devices. Being a relatively low-cost but at the same time computationally powerful platform, Lego Mindstorm have been widely used in recent years for education purposes [Galvan et al., 2006, Cliburn, 2006, Lew et al., 2010, Butterworth, 2012, Kolyubin, 2012].

As a consequence of the Lego Mindstorm spread out, many companies have been interested in producing Lego compatible sensors that are not usually provided as part of the Mindstorm kit, thus leading the whole Lego experience to a new level [Klassner and Anderson, 2003, Wallich, 2001]. These sensors have been used for the control of complex systems, such as the inverted pendulum (such as the one shown in Fig. 3) presented in [Ferdinando et al., 2011], where PD controller exploits the data provided by on board accelerometer to maintain the system vertical. Other examples that can be found in the literature implement even more complex control algorithms, such as Extended Kalman Filter [Pinto et al., 2012] and SLAM (Simultaneous Localization And Mapping) techniques [Oliveira et al., 2009]. Other interesting Lego Mindstorm applications involve remote control experiments [Casini et al., 2011, Grandi et al., 2011], Multi-Agent Systems [Benedettelli et al., 2009] and Swarm Intelligence based algorithms [Brigandi et al., 2010].

During the last years, we have integrated the course of Foundations of Industrial Robotics within the MSc Course in Automation Engineering at the University of Bologna with laboratory activities that have been designed in order to improve the problem solving skills of our students. In particular, we found that the laboratory activities where students have to compete between them have developed their creativity and have pushed them to exploit all their knowledge in order to solve the related problems. The aim of this work is to present the different challenges proposed to the students in the last years (2010-2013).

The paper is structured as follows. In Sec. 2 a brief introduction of the Lego kit and some of its applications in many research fields, especially in robotics, are presented. In Sec. 3, the laboratory activities proposed to the students are described, while in Sec. 4 the programming paradigms taught to the students are briefly presented. Pedagogical considerations regarding the obtained results are discussed in Sec. 5 while concluding remarks are reported in Sec. 6.

2. LEGO MINDSTORMS KIT

The series of Lego Mindstorms Kits, with the programmable *brick* computer, a set of modular sensors and

Copyright © 2014 IFAC

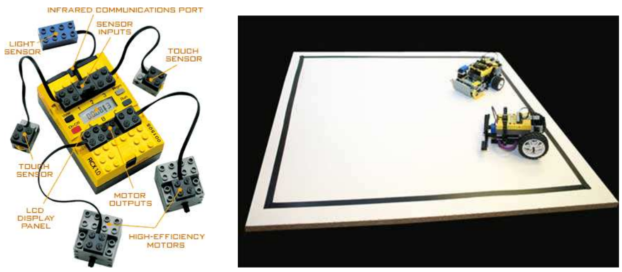


Fig. 1. RCX brick and its application in a Sumo robots competition.

motors and standard Lego parts coming from the *Lego Technics line*, allow to design and implement small, customizable and programmable robots both from hardware and software point of view.
The basic hardware/software element of *Mindstorms Robotics Invention System* kit is a programmable brick that derives from the one created at the *MIT Media Lab*. The first brick ever released was programmed using Brick Logo, an improved version of the more famous programming language Logo. The software innovation was the possibility to program the brick in a visual fashion. The first visual programming environment was called *LEGOsheets*; it was defined by the *University of Colorado* in 1994 and based on *AgentSheets*, educational software for the cyberlearning [Gindling et al., 1995, Repenning, 2013].
Normally the brick may be programmed by uploading a program written using the bundled graphical software or one of the several available languages (see Sec. 2.2).

2.1 The Lego Mindstorms kits

Different versions of Lego Mindstorms Kit have been released over the time, each of which is characterized by an increasing level of complexity, thus allowing the development of more sophisticated and complex robots both from the mechanical and the control software point of view. In the following, a short overview of the Lego Mindstorms Kits is presented. For further details please refer to related websites.

Lego Mindstorms RCX

The first generation of Lego Mindstorms was based on a brick known as RCX (Robotic Command eXplorers). Even nowadays, after many developments in the field of single on-board computers (SOCs), the performances of RCX are still quite interesting, especially for educational purposes. An example of application with a RCX brick is shown in Fig. 1.

Lego Mindstorms NXT

The second version of Lego Mindstorms (called NXT from *NEXT*) was released by Lego in July 2006. Compared to the RCX version, the NXT resulted to be a significant improvement not only from the hardware, but also from the software point of view. In fact, the developers abandoned the standard programming interface an adopted the philosophy of visual programming, where the program can be designed by simply connecting basic blocks representing the basic functionalities of the Lego NXT brick. This kit has been intensively tested in our laboratory experiments as described in Sec. 3. An example of application with NXT brick is shown in Fig. 2. The main NXT features are summarized in Table 1.

Fig. 2. A line-follower application developed with a Lego NXT differential wheeled robot.

Fig. 3. An inverse pendulum designed as a biped robot and a scorpion built with NXT-EV_3.

CPU	32 bit Atmel AT91SAM7S256 at 48MHz (ARM7), coprocessor 8 bit Atmel ATmega48 at 8MHz (RISC)
RAM	256k flash and 64k RAM (CPU), 4k flash and 512 byte RAM (coprocessor)
Ports	3 motor ports and 4 sensor ports
Motors	3 servomotors equipped with an integrated position sensor. More powerful in terms of torque and speed
Sensors	Colour sensor, 2 touch sensors, audio sensor and an ultrasonic sensor with a range of approximatively 60 cm
Communications	USB 2.0 or Bluetooth. BT also for a communication up to 4 team-mates
Display	LCD, 60x100 pixel of resolution
Pieces	619 pieces of Lego Technic
Prog. interface	NXT-G graphical environment
Power supply	Cable or batteries, also rechargeable

Table 1. Main features of the Lego NXT kit.

Lego Mindstorms NXT Kit EV_3

Because of the worldwide success of Lego Mindstorms Kits, Lego Company decided to produce the Lego Mindstorms NXT EV_3, which is going to be the most popular robotics platform ever developed by the company. Presented in January 2013 at Consumer Electronics Show (CES), its release is planned in Fall 2013 and it is going to be the most powerful of the platforms. Some application examples of the new version of NXT is shown in Fig. 3.

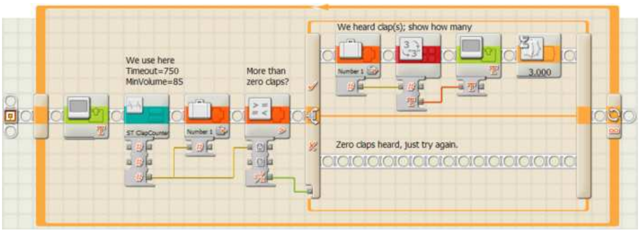


Fig. 4. A screenshot of the Lego NXT-G programming environment.

2.2 Lego NXT Programming Environment

NXT-G is the standard interface for programming the Lego Mindstorms Kits (see Fig. 4) that comes in bundle with each NXT kit. Developed by National Instruments, it is based on *Visual Programming* philosophy, i.e. it exploits the same well-known interface used in scientific software like Matlab/Simulink and LabVIEW. Despite the NXT-G user-friendly interface, it is not suitable for teaching high-level programming techniques. In fact, many technical details are hidden by graphical blocks and the options available for the programmers are quite limited, thus reducing the programming possibilities. Many other software tools can be used as valid alternatives to NXT-G. They are based on C/C++ or Java, Visual Basic, Logo and so on, although for some of these it is necessary to replace the original firmware of the brick. As one of the main aims of the laboratory experiences was to teach the basic skills necessary to design efficient software for automatic machines and multi-thread concurrent programming, among all the possibilities LeJOS [Lew et al., 2010] environment was selected. The chosen programming environment allows a powerful integration with the most common IDE tools and it can be used not only for controlling a single robot, but also to deal with multi-robot systems and to remotely control them.

More in details, LeJOS is a firmware replacement for Lego Mindstorms programmable bricks that currently supports both Lego RCX and NXT bricks. It also includes a Java virtual machine (inherited by the most famous TinyVM), which allows Lego Mindstorms robots to be easily programmed with Java language and to interface the software with most Java resources in term of API. Some of LeJOS main features are:

- object oriented language (Java);
- pre-emptive threads (determined context switching);
- (multi-dimensional) arrays;
- recursion;
- synchronization;
- exceptions;
- types of variable including float, long, and String;
- most of the standard Java classes are available (as java.lang, java.io, java.util);
- well-documented robotics APIs;
- a wide community of support.

The standard Java packages providing the language core, which can be downloaded from the official website [1] (Fig. 6(b)), have been extended with specific packages borrowed from other Java libraries or developed on purpose in order to facilitate the interaction between the user and wireless communication devices. Moreover, these packages provide basic solutions to some of the most classic problems related to mobile robotics, such as navigation, odometry and, more in general, the interaction with the environment.

[1] http://lejos.sourceforge.net/nxt/nxj/api/

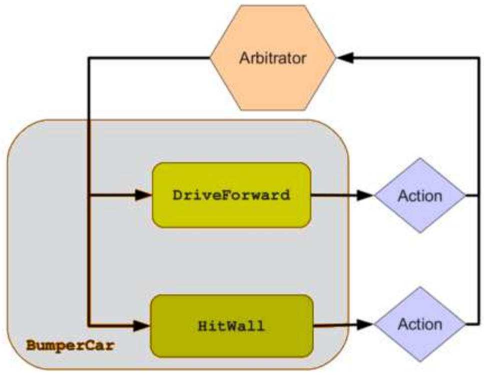


Fig. 5. Example of the subsumption architecture used to control a bumper-car.

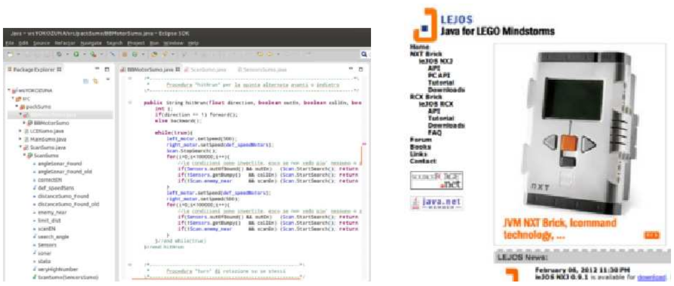
While a part of the extended packages included in LeJOS are derived from Java ME, a version of Java normally used in mobile devices, another consistent part of them, named *robotics packages*, have been developed ah-hoc for the LeJOS platform. Indeed, the *Subsumption Architecture* (SA), introduced in [Brooks, 1991], has been implemented within the set of robotics packages as an add-on specifically designed to development robots. This reactive software architecture can be exploited to design behavior-based control programs for any kind of robots (e.g. in Fig. 5).

The main feature introduced by the SA approach is that it allows to decompose complicated behaviors into simpler behavior modules, each of which is in charge of calculating the control action corresponding to a predefined sub-goal. The layers are then organized in bottom-up fashion and the overall system is monitored by a thread (namely Arbitrator) acting as a supervisor.

This powerful architecture has some drawbacks, for example it is not able to manage more than one behavior at the same time or, for complex global behaviors, it is difficult to create disjoint sub-behaviors. In order to overcome these problems, hybrid techniques, like the one based on neural network presented in [Ghinassi, 2013], can be implemented.

3. LABORATORY ACTIVITIES

Based on the positive feedback of the students in previous laboratory activities, where they used an *ad hoc* robotic simulation package [Falconi and Melchiorri, 2008] in order to improve their knowledge of the course subject, it clearly follows the importance of developing an innovative teaching method to improve the involvement of the students. As consequence, the laboratory activities based on Lego NXT were included in the curriculum activities of the students. The activities have been attended every year by an average of 30 to 40, usually divided in teams in order to design, build and control a mobile robot able to compete in the execution of a predefined task.



(a) (b)

Fig. 6. Eclipse IDE (Fig. 6(a)) can easily be interfaced with LeJOS (Fig. 6(b)).

Once the task to be accomplished is assigned, each team has to work autonomously. As typically few constrains were specified regarding the robot design (e.g. maximum size and number of motors and sensors), given the same resources each team usually developed solutions that differ a lot one from the others. From the software point of view, in our laboratory activities we strongly encourage our students to utilize Eclipse IDE (see Fig. 6(a)). In fact, by installing the proper plug-in, this developing environment allows not only to fully exploit all the potentials of the LeJOS libraries, but also to directly download the compiled code on the brick by using USB or Bluetooth connection.

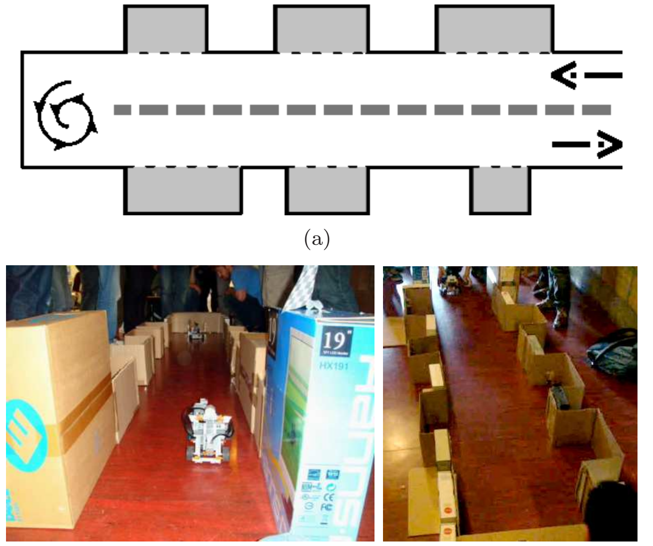
In the following, some of the applications developed with Lego NXT during the laboratory activities of the last Academic Years (2010-2013).

Car Parking

The main task was to build a self parking vehicle under the constraint of implementing a car-like kinematic. The scenario used for this task was a dummy street, built using carton boxes. The street had parking areas with different dimensions on both sides, some of of them big enough for the vehicles, some other too short to contain a car. Moreover, the street was designed with only one entrance (as a dead-end street) and it was large enough to do not allow a vehicle to perform a U-turn, thus forcing the car that reached the end of the street to perform complex maneuvers in order to invert the direction and then check the other side of the street for a suitable parking area (see Fig. 7(a)).

Given this scenario, the competition, that involved five groups of students, consisted of five trials with a different starting sequence. The test was considered successful both in case the car had found a proper parking area or in case it went out of the testing scenario, after exploring the whole street on both sides, because any parking lot was not available. The time of each trial was taken, and the winner team was the one that succeeded with the minimum average time (a time penalty was assigned to the teams unable to perform the task successfully).

Given the same Lego kit to all the groups, it is worth to notice that they develop quite different and interesting solutions. In particular, as two motors were used for the vehicle (steering and forward/backward motion), only one group utilized the third motor to move the sonar provided in the kit, thus creating *de facto* a radar able to scan at the same time both the side of the street.



(a)

Fig. 7. The dummy street used by the students in their competition. Parking lots are on both sides, while the ending area can be used by the robots to invert the motion direction.

Some images of this task are shown in Fig. 7.

Sumo Robot

This task, inspired by Japanese Sumo combat, took place in circular arena delimited by a black border. The only constraint was that each robot had to occupy an area smaller than 25×25 cm and that it could not be equipped with devices able to injury the opponent. Many mechanical solutions were adopted, although all the teams developed their robots on the basis of the differential wheeled robot kinematics by using two of the three motors provided by the Lego kit, while the third motor was used by teams in very different fashions. For example, some groups developed a shovel lifter placed in front of the robot to overturn the opponents, while other groups used the third motor to develop a radar turret to find the opponent and anticipate its move by exploiting very sophisticated predator-prey algorithms.

The whole competition was then organized as a tournament where each team disputed against all the others. Each match ended after two minutes or if a robot was pushed out of the arena. To evaluate the performances of each team, two points were assigned to the robot still on the arena, while one point was assigned to each competing team if, after the two minutes, both their robots were still in the arena.

A remarkable result that arose from this laboratory activity is that each group developed a really unique solution both from the mechanical and the control point of view. In particular, each team developed complex strategies that required a very intense work of research on subject that are not usually considered part of the course of Industrial Robotics. Two of moments of the final competition are reported in Fig. 8.

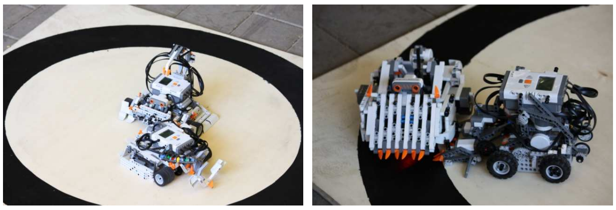


Fig. 8. Two moments of two different one-to-one matches in the Sumo robot competition.

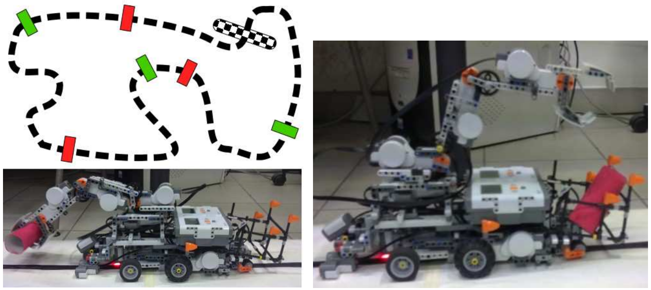


Fig. 9. Mobile Gripper. The sharp curves on the track are difficult to be followed because of the size of the robot. Notice that the robot is developed by using two kits.

Mobile Gripper

The goal of this task was to build a vehicle able to navigate by following a line on the floor and to recognize obstacles eventually placed on the path and, depending on the color, remove them from its way or collect them in a basket placed on the back of the robot. This task, carried out by a single team of three students, required to build a four degrees-of-freedom robot arm to be mounted on the vehicle (Fig. 9). The task was considered accomplished when all the objects were correctly processed and the right colored objects were collected in the basket.

To solve this task, two main difficulties had to be faced. First of all, as each Lego brick can control only three motors, the students had to use two control bricks, requiring them to implement also the communication protocol to interface the two control units via Bluetooth. Moreover, the path to be followed was designed with some sharp curves that were really hard to follow due to the size of the vehicle. To overcome this problem, a dynamic PID controller was implemented in order to improve the ability of the robot to follow the given line.

Pallet Transportation

The challenge in this task was to develop a robot able to pick up an object with known shape and transport it to a final destination. More specifically, see Fig. 10, the robot designed by the each team were placed in the area R0 and, by exploiting the guidelines on the floor, it had to reach and pick up the object placed in P0. Then, the robot had to turn around and travel through an area scattered with obstacles by exploiting only on board sensors and odometry. Once the next guideline was found, the robot had to deploy the object in the position P1 and then it had to move to the final area R1.

More than the other activities, this particular tasks required the students to develop many different skills not

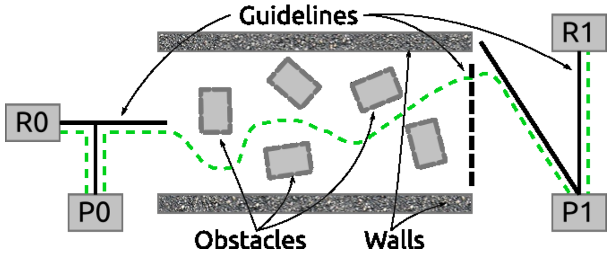


Fig. 10. Starting in R0, the robots have to grasp the box in P0 by exploiting the guideline on the floor. Subsequently, they have to safely navigate through the obstacle area and, after storing the box in P1, they have to conclude the task by parking in R1.

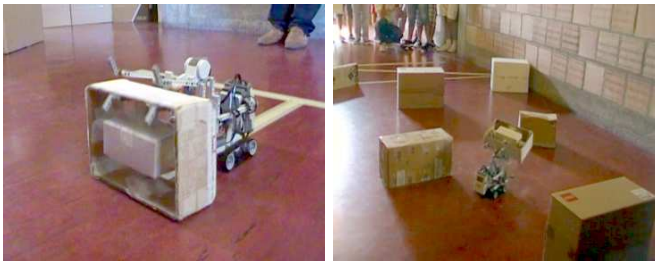


Fig. 11. The pallet transportation is a very challenging task, where odometry, obstacle avoidance, mobile robot navigation and task scheduling techniques must be exploited.

only from the mechanical point of view (as usual, many different solutions were implemented), but also from the control point of view. The final competition shown that better results were achieved by those groups which exploited more theoretical concepts in the development of the robot control software. Some pictures taken during the final competition are shown in Fig. 11.

4. CONTROL ALGORITHM DESIGN

The laboratory activities were carefully design in order to push the students to acquire new skills. In particular, one of the main targets was to teach them how to properly write programs in a modular way, thus improving not only the understandability of the control code, but also the possibility to easily write the code for each sub-behavior of the robot. Independently from the programming approach chosen by each group, i.e. exploiting the SA or developing their own architecture, it emerges from the code's analysis that they all developed over the years a similar software structure. In particular, they all developed programs with a main core with no more than three strongly interconnected *primary* threads to compute the behavior of the robot and a series of *secondary* threads to manage sensors and motors. This approach allowed to create compact programs quickly executable by the micro-controller of the robot, and to tune the interaction with the environment in order not to saturate sensors and motors.

5. COMMENTS ON PEDAGOGICAL RESULTS

As already mentioned, during the laboratory activities the students were usually organized in groups of approximately five members. Each group worked autonomously to achieve the goals of the assigned task, self-organizing the hardware/software design. Minimal suggestions and corrections are given by the supervisors only if strictly necessary. Each group received the specifications of the project and only some general constraints must be respected such as the kind of kinematics used for the mobile robot and its dimensions. The date of the final test was defined at the beginning of the work course and the groups had to plan and organize the development of the work in order to respect the deadline. Some hours of the course were devoted to teach Java and the LeJOS software environment. Typically, students spend a couple of weeks to learn Java and LeJOS/Lego basis and three-four weeks to complete the laboratory activity. As a part of the final exam of the course, each group had to present a final technical report on the experience, describing software and mechanical design problems encountered during the development and the solutions developed to solve them. All the comments given along the years by the students about this activity are absolutely positive and enthusiastic. Besides the acquired technical skills, the fact to acquire experience in team-work, also interacting with other groups and other solutions for the same problem, seems to be quite formative and appreciated by students. It often happens that groups created in these experiences have been maintained also in laboratory experiences attended in other courses. Finally, it must be noticed that the skills and knowledge acquired in this experience are exploited by students in many other courses.

6. CONCLUSIONS

In this paper, some of the laboratory activities related to mobile robotics carried out by the students during the course of Foundation of Industrial Robotics at the Engineering School of the University of Bologna are described. The proposed activities were developed to push the students not only to develop skills related to the subject, but also to learn how to cooperate with each other as part of a working team. The key to push them to develop new solutions for the given tasks was the idea of creating competitions, thus giving them a goal to accomplish and keeping their attention focused. Even if the proposed tasks were always challenging, after years of teaching we can finally say that the idea of involving the students in this activities was a complete success.

ADDITIONAL MATERIAL

Some videos of competitions carried on by the students in our lab are available on our Youtube channel [2].

REFERENCES

D. Benedettelli, M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino. A lego mindstorms experimental setup for multi-agent systems. In *Proc. 3rd IEEE Multi-Conference on Systems and Control, Saint Petersburg, Russia*, pages 1230–1235, July 2009.

S. Brigandi, J. Field, and Yunfeng Wang. A lego mindstorms nxt based multirobot system. In *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME Int.Conf. on*, pages 135–139, 2010.

R. A. Brooks. Intelligence without reason. In *In Proc. of the 12th Int. Joint Conf. on Artificial Intelligence, Sydney, Australia.*, pages 569–595, 1991.

D.T. Butterworth. Teaching c/c++ programming with lego mindstorms. In *Proc. 3rd Int. Conf. on Robotics In Education (RiE2012)*, 2012.

M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino. A lego mindstorms multi-robot setup in the automatic control telelab. In *Proc. 18th IFAC WC, Milano, Italy.*, pages 9812–9817, August 2011.

D.C. Cliburn. Experiences with the lego mindstorms throughout the undergraduate computer science curriculum. In *Frontiers in Education Conference, 36th Annual*, pages 1–6, 2006.

R. Falconi and C. Melchiorri. Roboticad: An educational tool for robotics. In *Proc. of 17th IFAC World Congress, Seoul, Korea*, volume 17.1, July 2008.

H. Ferdinando, H. Khoswanto, D. Purwanto, and S. Tjokro. Design and evaluation of two-wheeled balancing robot chassis: Case study for lego bricks. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 Int. Symp. on*, June 2011.

S. Galvan, D. Botturi, A. Castellani, and P. Fiorini. Innovative robotics teaching using lego sets. In *Robotics and Automation, 2006. ICRA 2006. Proc. 2006 IEEE Int. Conf. on*, pages 721–726, 2006.

G. Ghinassi. A learning method based on neural network for autonomous mobile robot navigation in partially structured or unknown environment. Master's thesis, School of Engineering and Architecture, University of Bologna, March 2013.

J. Gindling, A. Ioannidou, J. Loh, O. Lokkebo, and A. Repenning. Legosheets: a rule-based programming, simulation and manipulation environment for the lego programmable brick. In *Visual Languages, Proceedings., 11th IEEE Int. Symp. on*, 1995.

R. Grandi, R. Falconi, and C. Melchiorri. Unibot remote laboratory: A scalable web-based set-up for education and experimental activities in robotics. *Proc. 18th IFAC WC, Milan, Italy*, pages 8521–8526, August 2011.

F. Klassner and S.D. Anderson. Lego mindstorms: not just for k-12 anymore. *Robotics Automation Magazine, IEEE*, 10(2):12–18, 2003. ISSN 1070-9932.

S. Bobtsov A. Pyrkin A. Kolyubin. Lego mindstorms nxt for students' research projects in control field. In *9th IFAC Symposium "Advances in Control Education" (ACE 2012)*, pages 102–106, June 2012.

M.W. Lew, T.B. Horton, and M.S. Sherriff. Using lego mindstorms nxt and lejos in an advanced software engineering course. In *Software Engineering Education and Training (CSEE T), 2010 23rd IEEE Conf. on*, pages 121–128, 2010.

G Oliveira, R Silva, T Lira, and LP Reis. Environment mapping using the lego mindstorms nxt and lejos nxj. In *14th Portuguese Conf. on AI, EPIA*, 2009.

M. Pinto, A.P. Moreira, and A. Matos. Localization of mobile robots using an extended kalman filter in a lego nxt. *Education, IEEE Trans.on*, 55(1):135–144, February 2012. ISSN 0018-9359.

A. Repenning. Making programming accessible and exciting. *Computer*, 46(6):78–81, 2013. ISSN 0018-9162.

Paul Wallich. Mindstorms not just a kid's toy. *IEEE Spectrum*, pages 52–57, September 2001.

[2] http://www.youtube.com/feed/UCOudJUXA74dNXHXyM2TUw9g