Different robotics platforms for different teaching needs

Vicente Matellán
vmo@gsyc.escet.urjc.es
tel: 916 647 472
Universidad Rey Juan Carlos

José M. Cañas
jmplaza@gsyc.escet.urjc.es
tel: 916 647 468
Universidad Rey Juan Carlos

Rafaela González-Careaga
rafaela@gsyc.escet.urjc.es
tel: 916 647 400
Universidad Rey Juan Carlos

Abstract

When facing the problem of teaching the basis of robot control programming to computer science students, apart from the syllabus of the course, some other requirements have to be taken into account. For instance, which are the most appropriate robotic platforms, and which are the best programming tools for teaching it. In this paper we describe the platforms and programming environments chosen for different senior and graduate robotic courses at Rey Juan Carlos University, focusing on the reasons under our choices. We also point out the practical assignments demanded on the courses. Finally, we will present the results along four years, the feedback from our students, and the lessons we have learned.

Keywords: Teaching, mobile robots

1 Introduction

We are currently teaching robotic courses at two different levels, undergraduated and graduated level. The main goal of this paper is the description of the teaching experiences gathered along four years, including the choice of robotic platforms and programming environments suited for the courses, and the typical assignments we demand from our students.

We teach two different robotic courses, both named *Robótica*. One to undergraduate senior students at third year of computer engineer degree, and another to graduate students, at first year of PhD. program in computer science. In both cases, the subjects are integrated into computer-engineering programs, which means that their orientation is more focused on the programming issues related to mobile robots, than in the automation or robot construction problems. Surprisingly, after four years of experience, the syllabus of both courses has become very close, however the robotic platforms used are very different.

In the design of both course syllabus we decided to follow a constructionistic approach [9], where students not only have to learn the basic concepts of robot control, but also to really build and control real robots. We claim that playing with robots is the best way to learn real robotics, and to realize robotic problems and capabilities. In addition it keeps alumni motivated, as the number of student who choose our course shows.

Next sections describe the different platforms used. First, the platforms used by undergraduate students are presented, as well as the assignments that students have to do. Then, in section 3 the robots used in research activities are described. Finally, some conclusions are noted and the further improvements we are considering are introduced.

2 Undergraduate platforms: Mindstorms and EyeBots

In fact, we have to distinguish two cases among undergraduate students, the first ones are regular students who attend the *Robótica* course; the second one are senior students who choose to works on their Final Studies Project, FSP hereafter (*Proyecto Fin de Carrera* in Spanish) in our group.

2.1 The *Robótica* course

The target for the regular course is to provide the very first introduction to robotics. This includes the presentation of sensors, actuators, and their different configurations; some useful processing algorithms of the raw sensor data, and the basic control techniques. Also the principles of the intelligent control architectures proposed are described (reactive, behaviors based, symbolic, hybrid, etc.). Finally commercially available robots and their applications, as well as the most relevant people in the robotics history are introduced.

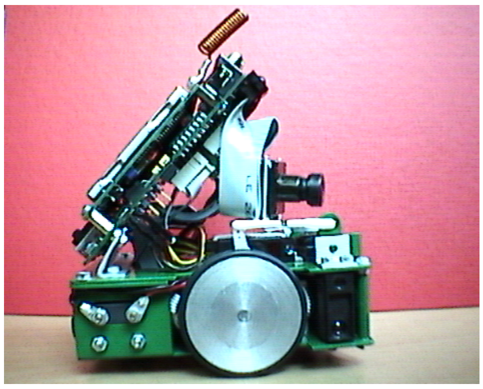
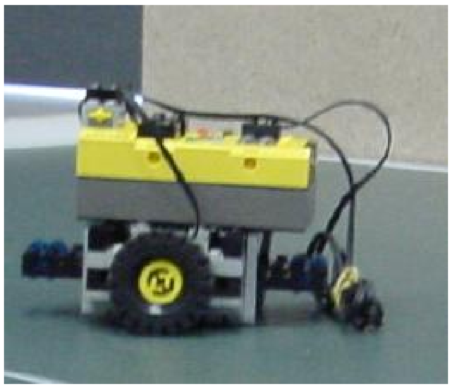

Figure 1: Eyebot and LEGO Mindstorms robots used for undergraduated students

The platform chosen for the practical assignments of the *Robótica* course is the LEGO Mindstorms[1], that is displayed in right side of figure 1. The main reasons are its flexibility and being a completely end-user product, all in a convenient price. To build a robot using the LEGO kit does not require any soldering, and the building blocks are intuitive and well known for all the students. The Mindstorms are sold as complete kits, which comprise more than 700 LEGO pieces. Each kit includes two motors, two contact sensors, one light sensor, and the processing brick, called RCX, made up of an Hitachi H8/300 microprocessor with 32Kbytes of RAM, three output ports to connect motors, and three input ports where sensors can be attached. A graphic software environment for developers, and the equipment for downloading programs into the robot are also included in the box.

In the software side, there are different tools available for programming the LEGO Mindstorms robots [7]. The first one is the child-oriented graphic language provided by the manufacturer with the kit. This is a very intuitive tool based on visual programming, but quite limited. The more extended alternatives are NQC and BrickOS. NQC [3] stands for "Not Quite C", and it is a simple language with a C-like syntax that can be used to program LEGO's RCX brick. It is the simplest option to the drag and drop icon programming tool included with the regular kit. It uses the native operating system provided by LEGO, and adds built-on-top programming capabilities.

BrickOS is the alternative chosen for our course. BrickOS is the current name of the better known LegOS, an open-source embedded operating system designed for the Mindstorms brick, mainly designed by Markus Noga [10]. This means that the original operating system has to be removed. Compared to the original LEGO operating system, BrickOS offers vastly superior performance and flexibility. It offers an API that supports dynamic loading of programs and modules, full infrared packet networking, preemptive multitasking, dynamic memory management, drivers for all RCX subsystems, 16 MHz native mode speed, access to the 32k RAM, etc. The development environment we offer to the students comprises a BrickOS running in the Lego robot and a personal computer under GNU/Linux. The robot programs are written in C language using any editor, compiled and debugged in the computer. Actually we use a cross-compiler available for GNU/Linux machines, which fully supports all C capabilities like pointers, data structures etc. Once the executable programs are generated they are downloaded to the Mindstorms, where the BrickOS makes them run.

The students of the *Robótica* undergraduate course were grouped into couples which received a complete Mindstorms kit, and each group received an additional light sensor, and a rotation sensor in order to allow them to develop more sophisticated behaviors (to implement odometry calculi, for instance). Currently, the robotic teaching lab contains 25 Mindstorms kits which allow 50 students in the course. The practical assignments include building the robots and programming them. Actually the mechanical design of the robot has to be decided by the students accordingly to the behaviors they have to program.

Two different tasks have to be solved by the students, although they can be seen as a single one, because the first one is just a subset of the second one. In the former the robot has to travel from an starting point to a destination goal inside a maze, and come back again to the initial location. The maze, shown in figure 2, is given in advance, so they can implement some kind of path-following in the robot program. In the second task the robot has to collect two different cans placed inside the previous maze. The exact location of one can is given in advance, but the location of the second one is completely unknown, forcing the robot to search for it and detect it. The idea here was to motivate students to use general architectures, instead of local solutions.

[1]http://www.legomindstorms.com

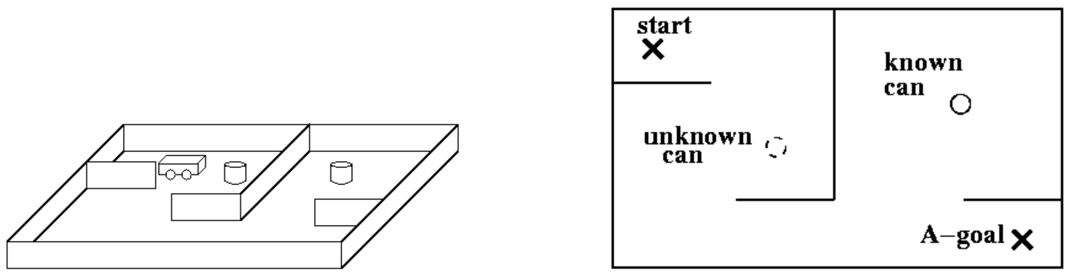

Figure 2: Practical assignments: maze and can collector

For instance, the ones who tried to pre-compile the trajectory in the maze for the known can, were in general less efficient searching the second one. Additionally, the cans can be white or black ones at random. In the case of black ones, the robot has to carry them to a destination location A. For white ones, it has to move them to another location B. So the robot has to distinguish between black and white cans, carrying each type to a different point in the maze. The guiding idea for these assingments was to make students face the sensor noise problems, and make them use the concepts presented in the theory classes like reactive local navigation, map building and use, etc.

The textbook that we currently recommend in theory part of the course is the Ronald C. Arkin one [2]. For practical assignments the construction of a LEGO-based robot requires basic notions of mechanics, but giving the students small manuals, as the official Constructopedia included in the kits, or the one by Fred Martin[2], has proved to be enough.

2.2 The Final Studies Projects

Students working on their FSP usually affront more complicate problems, which require more perceptive capabilities and more processing power than the ones LEGO brick offers. Mindstorms main drawback is its poor sensorization: available sensors, like infrared or encoders, provide few and noisy data. We decided to center most of the FSP works around a single topic: building a robotic soccer team according to the F-180 of the RoboCup [6]. Our idea is to promote collaboration among students by giving them a common problem, a higher objective where their works are integrated. So, a student working, for instance, in an adhoc networking protocol to communicate robots may feel herself part of the same group than the students working in image analysis, or the ones working in the control of a single robot, or with the ones working in the cooperative architecture.

We explored several platforms for this purpose, like Kephera[3], Tritt[4] and EyeBot[5]. Prices were very different and finally we chose the EyeBot robot [4], due to its processing power and because it was the only one providing a camera, which bursts possibilities. This robot, shown in left side of figure 1, is equipped with three infrared sensors, two shaft encoders and a 82x62 pixels color camera, giving 4 fps. It also has two DC motors, two servomotors and it is equipped with a Motorola 68332 microprocessor, and 1Mb. of RAM memory. In addition, EyeBots are supplied with a radio communication module. All those devices are managed by the *RoBios* operating system, provided by the manufacturer, which offers a programming API to the robot devices.

FSPs are carried out in the robotic research lab. Currently we have 6 EyeBot robots especially configured for the RoboCup. They have a kicker, and appear in one of two configurations: with a servo for panning the camera, or that servo for tilting the camera. All the robots are used by all the students, that is, robots are not assigned to any particular student. This way they have to ensure that their programs are not overfitted to the characteristics of a singular platform.

Some of the FSP recently finished develop a follow wall and follow ball behaviors, an adhoc communications protocol, and a teleoperator for the EyeBot robot. FSPs currently in progress are the following:

- Soccer behaviors based on vision, like go-to-point.
- Self localization from local vision.
- Reliable communications library.

3 Graduate platforms: The Pioneer and the Aibo

With graduate students the aim of the course is more research oriented, as expected for PhD candidates. The course contents lie around autonomous navigation and robot behavior generation. Besides demanding the read lot of papers on the field, we offer two

[2]http://constructopedia.media.mit.edu/

[3]http://www.k-team.com/

[4]http://www.microbotica.es

[5]http://www.ee.uwa.edu.au/ braunl/eyebot

current state-of-the-art hardware platforms for experiments: a pioneer robot from ActivMedia, and an Aibo, the Sony robotic dog.

The idea of choosing these robots was to use the most standard platforms available. This way our research works can be validated by other groups owning the same robot, because they can check the results we would claimed, and vice versa. Several topics were considered when deciding which were the most appropriate platforms. First, we had to choose the size of the robot, then we had to decide its sensor equipment. In the 90's, the successors of the famous Xavier [11] populated most research robotic centers. The most widely used platforms were the B21 robot from Real World Interface (now the research division of iRobot Inc.), and the Nomad, from Nomadic technologies. However, these robots' sizes are considered too large today. The same functionality can be got in smaller robots, with the advantage of being easier to carry and handle. This is the case of the Pioneer robot manufactured by ActivMedia that we chose.

The sensorization is another major issue when selecting a robot. For instance, the cheapest technology are the infrared sensors. However, these sensors are the least accurate, because they are very sensitive to lightening conditions. Another alternative is the use of sonar sensors, which are less sensitive to lightening conditions and provide more information. Nevertheless they are very noisy and uncertain at corners, reflecting surfaces, holes, etc. Another option are the laser sensors, which are very accurate, and less dependent on environmental conditions. Their high price is their main drawback. The last sensor usually considered in mobile robots are cameras, they may provide much more information than others, but they also demand lot of processing power and good algorithms to extract it.

Our Pioneer robot, displayed in left part of figure 3, is equipped with a ring of sixteen sonars, eight frontal and another eight rear, it carries a laptop under GNU/Linux where the control programs run. The platform also carries a small webcam which is attached to the laptop through USB port. The processing of the images got by the camera is done in the laptop. The platform itself contains also a microprocessor, devoted to the low level calculi like collecting encoder and sonar data, and closing PIDs feedback control loops.

The software environment available for programming the Pioneer robot are the Saphira[8] and ARIA[1] suites. Both provide a client server approach to access robot devices from C and C++ programs. They also offer multitasking and networking interfaces. We have chosen ARIA because is licensed as free software under GNU GPL license.

Works currently in progress using the Pioneer robot are:

- Development of a generic architecture to build robot controllers. This architecture is inspired in biological ideas and based in the dynamic construction of hierarchies of schemas [5].
- Implementation of various reactive behaviors for robot navigation.
- Development of localization and navigation algorithms using information from wireless communication network.

The second robot that is available to graduate students in their research works is the Aibo robot shown in right side of figure 3. This is the newest member of our robotic family. The reasons for starting to use this robot is that, as in the Pioneer case, this robot has become a major commercial success. Actually it is a bestseller, and it can be considered the first robot really sold at large scale. Its programming environment was opened last summer, which helped us to make our mind about acquiring a unit in order to evaluate its feasibility as a research tool. The major goals with it are to start working on legged robots, and to study the robot-human interaction problems. Besides, we are considering its possible use in the RoboCup environment, where Aibo robots have their own category.

The robot itself is controlled by a 384Mhz. RISC processor, has 64Mb of memory, temperature, infrared, acceleration sensors, a color camera, and 20 degrees of freedom. The programming environment is based on the OPEN-R operating system[6] whose API was opened for public domain last summer.

4 Conclusions and future lines

This paper has presented the environments we have chosen to teach robotics at the *Universidad Rey Juan Carlos*. We have briefly shown the methodology that we employed in teaching this subject to computer science students at different levels, describing the tools, both hardware and software, that we use. Additional information about the courses, such as the detailed contents, slides, etc. can be found in the web pages of the course[7].

We are really proud of the results of the polls made both by the university itself and by ourselves. The numerical results shown by the official poll place this course among the top ones from the point of view of the students. Another clear indicator of the quality of the course is the fact that this is an elective course, and the last two years (only has been offered three times) the number of students applications has been higher than the available seats. The only objection made by the students is the high amount of time employed to test and debug the practical assignments due to poor performance of the LEGO sensors.

[6]http://www.aibo.com/openr
[7]http://gsyc.escet.urjc.es/docencia/asignaturas/robotica

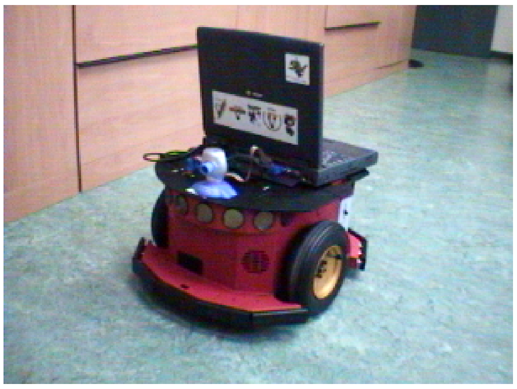
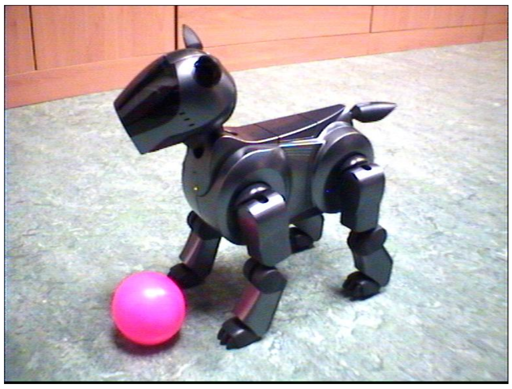

Figure 3: Pioneer and Aibo robots at the Rey Juan Carlos University

Concerning future plans, at the undergraduate level we are considering to use another free operating system available for the LEGO platform. This one is a Java(TM) based operating system. The main reason is the major activity in this project, as well as the added value of using Java as programming language by the students. For the FSP students, we are thinking in working with the same robots that are currently used by the PhD candidates.

References

[1] ActivMedia. Aria reference manual. Technical report, ActivMedia Robotics, 2002.

[2] Ronald C. Arkin. *Behavior based robotics.* MIT Press, 1998.

[3] Dave Baum. *Dave Baum's Definitive Guide to LEGO Mindstorms.* Apress, USA, 1999.

[4] Thomas Braünl and Birgit Graf. Autonomous mobile robots with onboard vision and local intelligence. In *Proceedings of Second IEEE Workshop on Perception for Mobile Agents*, 1999.

[5] José M. Cañas and Vicente Matellán. Dynamic schema hierarchies for an autonomous robot. In Miguel Toro Francisco J. Garijo, José C. Riquelme, editor, *Advances in Artificial Intelligence, Iberamia 2002*, volume LNAI-2527, pages 903–912. Springer Verlag, 2002.

[6] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Life*, pages 19–24, 1995.

[7] Jonathan B. Knudsend. *The Unofficial Guide to LEGO MINDSTORMS Robots.* O'Reilly & Associates USA, 1999.

[8] Kurt Konolige and Karen L. Myers. The Saphira architecture for autonomous mobile robots. In David Kortenkamp, R. Peter Bonasso, and Robin Murphy, editors, *Artificial Intelligence and Mobile Robots: case studies of successful robot systems*, pages 211–242. MIT Press, AAAI Press, 1998. ISBN: 0-262-61137-6.

[9] F. Martin. Ideal and real systems: A study of notions of control in undergraduates who design robots. In Y. Kafi and M. Resnick, editors, *Constructionism in Practice: Rethinking the Roles of Technology in Learning.* MIT Press, 1994.

[10] Markus L. Noga. Legos: Open-source embedded operating system for the lego mindstorms. http://www.noga.de/legOS/.

[11] R. Simmons, J. Fernández, R. Goodwin, S. Koenig, and J. O'Sullivan. Lessons learned from Xavier. *IEEE Robotics and Automation Magazine*, 7(2):33–39, June 2000.