Libre software environment for robot programming

Vicente Matellán Olivera, Jesús María González Barahona,
José Centeno González, and Pedro de las Heras Quirós
{vmo,jgb,jcenteno,pheras}@gsyc.escet.urjc.es
Grupo de Sistemas y Comunicaciones (GSyC)
Dept. Ciencias Experimentales e Ingeniería. Universidad Rey Juan Carlos
C/ Tulipán s/n 28933-Móstoles (Madrid) -SPAIN

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 considered as which is the most appropriate robot, and which are the right tools for learning how to control it. In this paper we describe the tools we have chosen for teaching robotics, focusing on the environment that support the practical assignments. We also analyze the reasons that make us choose each of the tools, making special emphasis on the *Libre* [1] requirement that we have imposed to every tool we are using. Finally, we present the results and opinions we have got from our students, and the lessons we have learned by using the *Libre* software approach.

1 Environmental description

GSyC group (*Grupo de Sistemas y Comunicaciones* [2]) is currently in charge of teaching the robotic-related subjects at *Universidad Rey Juan Carlos (URJC)*. The *curricula* offered by our department are mainly focused on computer engineering, which means that the orientation of the robotics-related subjects offered by us have to be more focused on programming issues related to mobile robots, than in the automation problems. In this environment, the subjects related to systems interacting with the real world that our group is currently teaching are "Robotics", and "Critical Systems Programming" in the undergraduate level.

The main goal of this paper is the description of the practical assignments of the robotics course (named *Robótica*) teached by GSyC, as well as the lab organization. The orientation of the description will be mainly focused in the influence the *libre* software requirement we imposed to the software tools used.

GSyC group has been compromised with the *libre* software for long time [10], and we have used it in teaching for some time, mainly in computer networks and operating systems. Our experiences in these subjects encouraged us to extend this approach to the robotics field when the department decided to encharged our group the teaching of this subject.

In the design of the curse syllabus we decided to follow a constructionistic approach [9], where students not only has to learn the basic concepts of robot control, but to really build and control real robots. In this way , the textbook that we are currently recommending is the Ronald C. Arkin one [1].

This is the environment of the *Robótica* course, additional information about the course, such as the detailed contents, slides, etc. can be found in the web pages of the course [3]. Next section describe the process that lead us to choose LEGO Mindstorm as the most appropriate platform. Then, we analyze the *libre* software tools for robot programming that constitutes the core of this paper, as well as other software tools that have helped us in the subject. Last, we summarized the opinions of students and ours about this teaching environment.

2 Hardware platform

Concerning the hardware platform, we have previous experiences using the Khepera mini-robot [6] for research purposes in the RoboCup competition [5]. Apart from research aspects, this initiative has generated interesting experiences for teaching robotics [8].

[1] *Libre* word will be used in this paper to avoid the usual misunderstanding between "free" as in "free beer" (*gratis* in Spanish), and *free* as in "free speech" (*libre* in Spanish)

[2] http://gsyc.escet.urjc.es

[3] http://gsyc.escet.urjc.es/docencia/asignaturas/robotica

In this way, our first idea was to use the robot we were used to. However, this robot was to expensive to be used as class material for every student in our class, so we decided to try with other robots.

First option we considered RugWarrior[4] robot. This a Motorola 68HC11 based robot widely used in robotics courses. However, after letting some students work with it, we concluded that this was more appropriate for electronic oriented courses, where the goal was to show the students robot architecture concepts: hardware problems, operating system considerations, low level control, etc. than for robot control programming.

Another two options we considered were the Handyboard [4], a Motorola 68HC11-based controller board designed for experimental mobile robotics work, but it is also more oriented to teach electronic or architectural concepts than software control; and the EyeBot [5], and it was so expensive as Khepera.

We finally choose the LEGO Mindstorm [6] platform. The main reason was that this is a very flexible, and at the same time it is a completely end-user product. It does not require any soldering, and the building blocks are intuitive and well known for all the students.

The Mindstorm are sold as complete kits made up by more than 1600 LEGO pieces. Each kit includes two motors, two contact sensors, one light sensor, the micro-processor block known as the "intelligent brick" or just the brick, a graphic software developing environment, and the equipment for download the software into the robot.

The students were grouped in teams which received a complete Mindstorm kit, and additionally each group, made up by two students, received another light sensor, as well as, a rotation sensor in order to allow them to develop more sophisticated behaviors.

The software environment included in the LEGO Mindstorm 1.5 kit is a graphical programming language designed for kids. This is a very limited development systems, and which is more important, it does not fulfill the *libre* requirement. However, Mindstorm robotic platform is one of robots that has got more attention from the open source community, which has resulted in different options that were analyzed in the next section.

[4] http://el.www.media.mit.edu/projects/handy-board/
[5] http://www.ee.uwa.edu.au/ braunl/eyebot/
[6] http://www.legomindstorms.com

3 *Libre* software tools used

Once the robot had been decided, the programming environment has to be chosen. That environment includes the operating systems that will run onboard (into the robot), the development platform (a PC under GNU/Linux), the cross compilation system, the debugging tools, and the simulators. In this section we will analyze the *libre* software tools we have used in the *Robótica* course for these purposes, and some other generic tools to give information to the students, or to keep in touch with them.

3.1 Robot programming tools

The practical assignments of the *Robótica* course comprise two parts, first but less important, building the robots; and second, programming the robots. 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 [7] has probed to be enough.

There are different options for programming the LEGO Mindstorm robots [3]. The first one is the child-oriented graphical language provided with the kit. This is a very intuitive tool, but quite limited, and it is a proprietary tool which does not fulfill the *libre* requirement. This means that we had to look some other options, mainly NQC and legOS, that are described in the next two sections.

The option we recommend to the students was legOS operating system, rationale for this decision is also given in section 3.1.1. The use of this operating systems requires also the installation of a cross compilation system, based on `binutils` and `egcs`

In that environment, that is Mindstorm brick and LegOS, it is sometimes difficult to debug a program, so we install two different simulators for our students: Legosim, and emulegos. The differences between them are also studied in sections 3.2.1, and 3.2.2.

3.1.1 NQC

NQC [2] 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 programmable brick (from the Mindstorm set). It is the simplest option to the drag and drop icon programming that Mindstorm provides. NQC is *libre* software, released under the Mozilla

[7] http://el.www.media.mit.edu/groups/el/projects/constructopedia/

Public License (MPL), however it uses the standard operating system developed by LEGO, which means that this options scarcely fulfills the *libre* requirement. However, that was not the main reason to choose LegOS. NQC has been designed to be very simple, and accessible to people with few knowledges of programming, this means that there are strong limitations, the main ones from our point of view are:

- Subroutines do not admit parameters.
- There only global variables.
- Subroutines do not return values.
- The number of variables is strongly limited, just 32 variables.
- There are no data structures, neither static, nor dynamic.

This made us discard NQC because our students are supposed to have strong programming abilities, and they are supposed to use them in this course to generate sophisticated behaviors for the robots.

3.1.2 LegOS

LegOS is an open-source embedded operating system designed for the LEGO Mindstorm brick, mainly designed by Markus Noga [7]. Compared to the standard software, the one from LEGO, LegOS offers vastly superior performance and flexibility. As a summary, version 0.2.0, LegOS features include:

- Dynamic loading of programs and modules.
- Full IR packet networking.
- Preemptive multitasking.
- Dynamic memory management.
- Drivers for all RCX subsystems.
- 16 MHz native mode speed.
- Access to the 32k RAM.
- Full use of the programming language chosen, for instance C (pointers, data structures, etc.).

The development environment is made up by a computer to compile on and communicate with the Mindstorm, both to download the program, and to debug programs. LegOS-0.2.0 works pretty reliably under RedHat and SuSE versions, as well as, under Debian 2.1 that was the distribution used in the lab.

LegOS is just the operating system, obviously programming languages are needed. In the course we used C, but any other languages whose compiler supports cross compilation can be used, for instance, most people use C++ instead of C.

Various other tools are available, including simulators as LegoSim or EmuLegos, web-based compilers (see section 3.2.3 working directly from your browser, etc. Some of them are described in the following sections.

3.2 LegOS related tools

3.2.1 LegoSim

LegoSim is a UNIX-based Simulator for LegOS. The main differences with Emulegos is that the graphical user interface (GUI) is separate from the simulator. This separation also let the GUI serve as a control unit, not just a simulator interface, allowing for example to connect it to other RCXs via IR, or to run the GUI on a different machine than the simulator.

The GUI is a Java applet that closely resembles the RCX, while the simulator is just a library. Both components are coupled by Perl scripts. The simulator library is a replacement for LegOS that can be linked with any LegOS application. LegOS tasks are mapped onto POSIX threads, and input and output of any devices is simulated as a string on stdin/stdout following a grammatical specification.

LegoSim is distributed under Mozarilla MPL license, and it was designed designed and implemented by Frank Mueller, Thomas Röblitz, and Oliver Bühn [8].

3.2.2 EmuLegOS

EmuLegOS [9] is a LegOS emulator design to provide a more comfortable environment where to test and debug programs. EmuLegOS is in essence a set of C++ code that can be compiled and linked together with any LegOS code, generating an application that emulates the behavior of the code as it were running on the actual RCX. The API layer emulates the LegOS routines. Most of LegOS is actually implemented, including multithreading and IR support.

Its visual interface (see Figure 1) allows the user to configure the sensors and interact with them while the program is running, for instance simulating external

[8] http://www.informatik.hu-berlin.de/ mueller/legosim/
[9] http://www.geocities.com/~marioferrari/emulegos.html

Figure 1: emulegos simulator

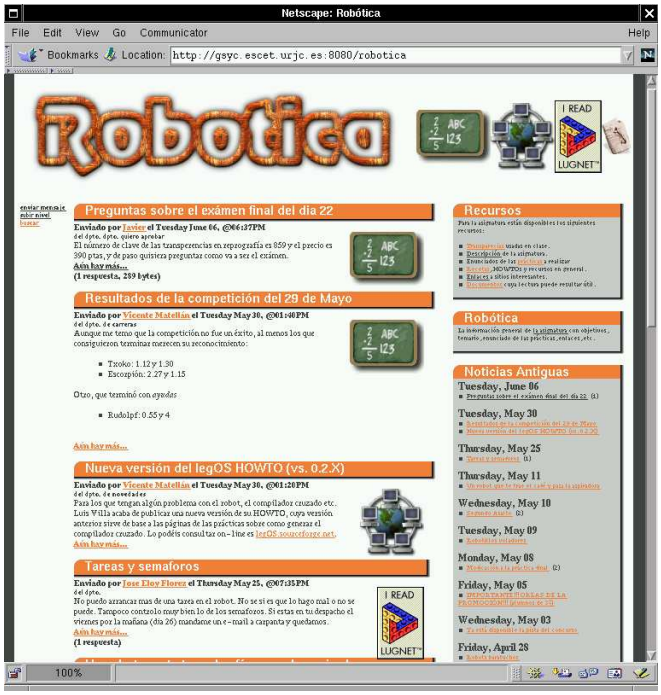

Figure 2: Weblog of the *Robótica* laboratory

events. The interface also shows the current status of three motors virtually attached to the A, B, and C ports.

EmuLegOS also provides *Real world* emulation support providing a place where to put code to mimic some of the mechanical features of the robot, such as a rotation sensor that turns while a motor is running, or a touch sensor that closes after n seconds a motor started.

The main utility of simulators is the possibility of debbuging. In both cases (EmulegOS and LegoSim), all the debugging facilities of the development environment are available, as the LegOS program is compiled and run inside it.

3.2.3 WebLegos

Web-LegOS is an HTML-based interface for compiling programs written to the LegOS operating system for the Lego Mindstorm RCX controller.

Using Web-LegOS is simple, it is possible to cut and paste the source file into a text box of a web page, choose the language, select the way we want to receive the s-record file that will be generated, and just click the "Compile" button. Once the s-record file has been obtained, it is possible to download it into the local RCX by whatever means.

Web-LegOS currently uses the 03/30/99 snapshot of LegOS, but is in no way affiliated with the LegOS project.

3.3 Alumni information tools

Apart from the previous tools, specific for robotics programming, some other software tools has been use in order to promote students cooperation, as well as, the relationship between the students and their advisors. In this way, an e-mail list is created for each subject taught by GSyC (robot@gsyc.escet.urjc.es is the one for robotics subject), another one exists to communicate to/among the professor in the group (gsyc-profes@gsyc.escet.urjc.es), internal news groups are also created for each subject (news://es.urjc.robotica), and web pages (http://gsyc.escet.urjc.es/docencia) were students can find the information related to the subjects: description, material used in classes, practical assignments, usual links, etc. We use `mailman` [10] as the e-mail lists manager software, `xxx` [11] as the news software, and `Apache` [12] as the web server. All of them running on a Debian GNU/Linux [13] operating system over a personal computer.

Specially interesting is the weblog concept. The idea is to give the people involved in a particular area the opportunity to provide pieces of news interesting to the rest of the community. One of the

[10] http://www.mailman.org
[11] http://www.xxxx.org
[12] http://www.apache.org
[13] http://www.debian.org

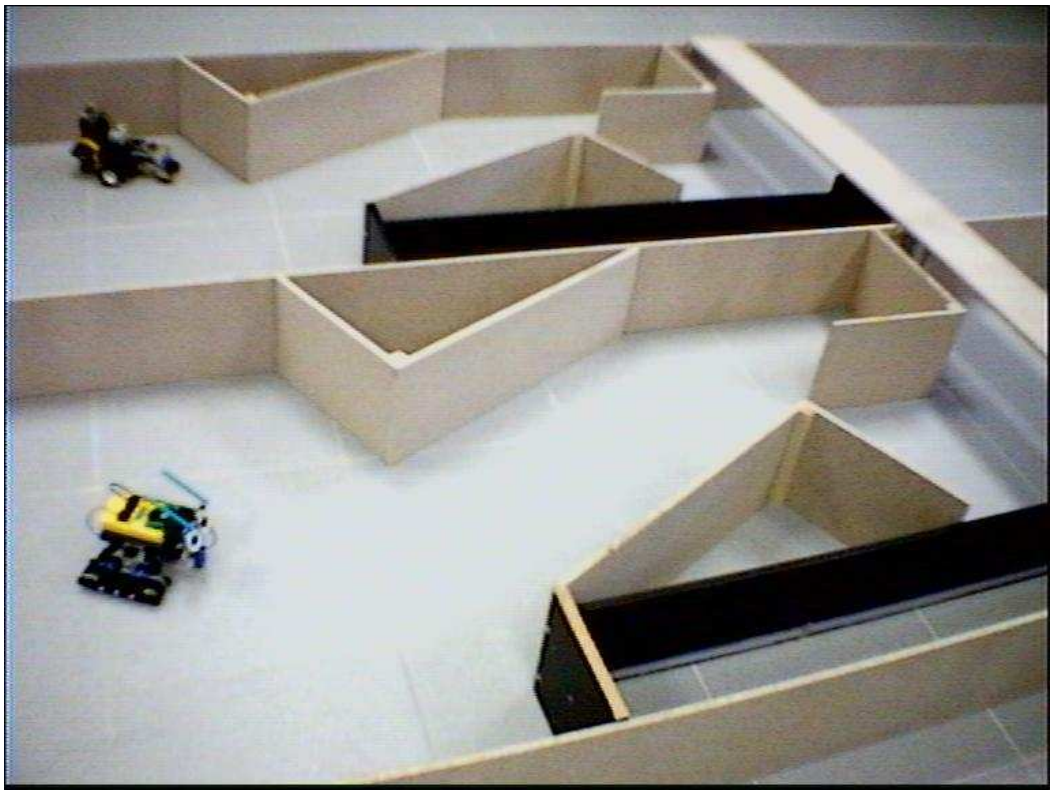

Figure 3: Practical assignment maze

most widely known weblogs is SlashDot [14] the weblog of the geek community. The look and feel of this kind of tools can be seen in Figure 2, and the one used in *Robótica* subject can be accessed in `http://gsyc.escet.urjc.es/foros/robotica`.

The software used to built the weblog for the Robótica subject was SquishDot [15], a Zope [16] developed by Digital Creations [17]

4 Evaluation of the course

There were three practical assignments

The third one was organized as a competition, where two robots have to find the exit of a non-known maze in parallel. Figure 3 shows the field where the competition took place.

[14] http://slashdot.org
[15] http://www.squishdot.org
[16] http://www.zope.org
[17] http://www.digital.com

5 Conclusions

This paper has presented the advantages that provides the use of *libre* software in robotics teaching. We have shown the methodology that we employed in teaching this kind of subjects to computer science students as well as the tools, both hardware and software, that we have used.

Summarizing, using *just* free software can be considered the major innovative aspect described in the paper. As a secondary contribution, we have described in this paper the auxiliary tools we have set up to help in the teaching process, mainly Internet based tools (for instance weblogs). We have also analyzed the opinions given by our students, as well as the evaluation of those opinions.

References

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

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

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

[4] **Joseph L. Jones, Anita M. Flynn, and Bruce A. Seiger**. *Mobile Robots: Inspiration to Implementation (2nd Edition)*. A. K. Peters. Wellesley, Massachusetts (USA). 1998.

[5] **Using ABC^2 in the RoboCup Domain**. *Vicente Matellán, Daniel Borrajo y Camino Fernández*. In *RoboCup-97: Robot Soccer World Cup I*, Editor Hiroaki Kitano, pag. 475-483. Ed. Springer Verlag 1998. ISBN: 3-540-64473-3.

[6] **Francesco Mondada, F. P. Ienne**. *Mobile Robot Miniaturization: A tool for investigation in control algorithms*. Proceedings of the Third International Symposium on Experimental Robotics. Kyoto, Japan, 1993.

[7] **Markus L. Noga**. *LegOS: Open-source embedded operating system for the LEGO Mindstorms*. `http://www.noga.de/legOS/`.

[8] **Henrik Hautop Lund**. *Robot Soccer in Education*. Advanced Robotics Journal, special issue on RoboCup.

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

[10] **J.M. Barahona, J. Centeno, P. de las Heras, V. Matellán, and F.J. Ballesteros**. *Libre software in CS practice teaching (The experience at Carlos III University)*. IEEE Software, Vol. 17, No. 3, May/June 2000.