Designing a Myro-Compatible Robot for Education as Copyleft Hardware

Eduardo Grosclaude, Rafael Zurita, José Riquelme, Rodolfo del Castillo, and Miriam Lechner

Facultad de Informática, Universidad Nacional del Comahue,
Buenos Aires 1400, Neuquén, Argentina
{oso,rafa,jose.riquelme,rdc,mtl}@fi.uncoma.edu.ar
http://www.fi.uncoma.edu.ar

Abstract. The application of less conventional teaching techniques shows a growing trend. Studies have shown that the usage of educational robots improves learning in informatics. This work presents the architecture of a vision capable, low-cost robotic system designed and built to be used as an educational platform.
We opted for open architecture and copyleft hardware to make our development shareable with other institutions and agencies, while keeping in mind the costs and complexity associated with the creation of such a platform. By mixing in an embedded Linux system and a vision system, the robotic platform became more powerful and apt for other uses.
The robotic system described is used within our Facultad de Informática to teach Embedded Systems, to learn and test AI techniques, to teach Programming in an enticing way, and for dissemination extension activities about Computing.

1 Introduction

Since 2006, when **IPRE** (Institute for Personal Robots in Education) and other similar initiatives came to appear, educational robotic platforms have begun to make their way into introductory Computer Science courses. Using high-level programming languages to control these devices is highly motivating, since robots expose some aspects of computing that freshman students usually find difficult to grasp. The efforts initially have been directed to reduce the high drop-out rates observed in undergraduate Computer Science courses [**1**][**2**]. Many kinds of educational robots and programming tools were developed to this end. In particular **Myro** (My Robot), jointly developed by several institutions through IPRE, is among the most adopted frameworks. Myro implements a protocol of messages to be sent to (and answered by) a robot. Its main goals are to allow for an easy access to the whole robot's capabilities, and to be usable via any programming language. On this call, this framework has been ported to many languages, such as Python, C, C++ and Java [**3**][**4**].

Myro has also been integrated into Calico, a modern development environment, which enables many beginners' programming tasks within a simple-to-use

ambient. While Myro is a part of Calico, this tool's main focus dwells in providing a development environment for beginner Computer Sciences students [**5**].

Besides, Calico-Myro supports several types of educational robots, which allows for some variants to institutions adopting these learning tools [**6**].

In spite of these possibilities, the only easily reusable element is the control software, released under an Open Source license. However, the robot platforms supported are commercially licensed, and they must be purchased to be used [**7**][**8**][**9**][**10**].

Because of this, efforts have been set forth to develop robotic devices under free licenses supporting Myro-Calico, as few chances exist to purchase compatible robotic platforms in our region. In Argentina, a few universities use the commercial robot N6 from RobotGroup [**11**] which integrates **Arduino** free hardware. Even so, the rest of the platform still cannot be easily reproduced, as the remaining documentation needed to build these robots has not been released [**12**].

With this scenario in mind, we designed at our Department a low-cost, Myro-compatible robot, to be developed and released under hardware and software free licenses. This licensing would allow to locally use the robot and to share it to other institutions interested in building and improving such a platform.

The platform was designed not only to be used at first courses in Computer Sciences, but also for research purposes. In fact, the integrated vision capability offers a very important resource for areas like Artificial Intelligence and Computer Vision. This design does also overcome a usual shortcoming found in educational robotic platforms, which usually employ basic sensors driven by low-resources micro-controllers. To achieve vision capability, which depends on an integrated camera, some computational resources had to be thrown in for image processing, as well as a communications device to enable the transmission of video sequences.

The purpose of this article is, on one hand, to describe the architecture of a hardware-copylefted robot which supports Myro and provides a vision extension of the system. We also describe the results obtained from real usage learning experiences in first courses and as a research tool.

2 System description

2.1 Copyleft Hardware

The **Copyleft Software** concept has its roots in free software development, and requires every modifications or extensions of a program to be free software as well. **Copyleft Hardware** devices licensing shares the same philosophy and draws from the **FOSS** (Free and Open Source Software) movement mindset.

This methodology's main characteristics are [**14**]:

- Principles similar to those of Free Software's are applied to hardware design and production.

– All related documentation must be released under open licenses: **GPL**, **GFDL**, **CC-BY** and **CC-BY-SA**.

Usage and development of Copyleft Hardware allows to [**14**]:

– Learn from it without a need for reverse engineering. Be able to spot poor functionality and devise enhancements.
– Adapt design to new uses and environments.
– Reuse part of the design in other projects.
– Achieve product longevity by actualization, improvement, repair or modification.

The Copyleft Hardware development model has been followed in this project since the design stages, as it allows for distribution and modification of design and source code, with the only requirements that derivative work must be under a compatible license and original authors must be credited. Other institutions interested in using educational robots can freely take up building, modification and usage of this platform. To achieve this goal, boards schematics and design files have been released as Copyleft Hardware, and the source code to programs and tools have been released as Open Source software. Documentation related to robot design and construction, as well as tutorials for installation and utilization of libraries and programs, have also been published[1].

2.2 Hardware Architecture

Fig. 1 shows the robot's hardware architecture basic scheme.

The system is built around a **MIPS MR3020** [**16**] controller board (main features shown in **Fig. 2**). This board works as an interface between the user (the PC) and the robot; provides wireless communication over a 2.4GHz channel; controls a UVC video camera, webcam type, and provides communication with the **EasyDuino** [**17**] board.

During the selection process for the Linux embedded system hardware, several similar boards were considered. Some of them went under some reverse engineering study to allow experimentation with Linux, and to conduct laboratory tests in order to analyze and compare characteristics [**18**].

The EasyDuino board is a Copyleft Hardware clone of **Arduino UNO** having the same features as the original –except for programming, which is done only through a serial connection (same as used for communicating with MIPS MR3020). EasyDuino's main characteristics are shown in **Fig. 3**.

For the power stage, i.e. DC engines control, a Shield for EasyDuino having an **LD293** (H bridge) chip has been designed and developed. On the Shield were also installed three LEDs and a Buzzer mini-speaker to enable signaling to the user in several ways. The same board supports the battery pack, catering energy for the whole system. This interface's features are shown in **Fig. 4**.

The finished robot and the aforementioned hardware components can be seen in **Fig. 5**.

[1] Downloadable from [**15**] and [**21**]

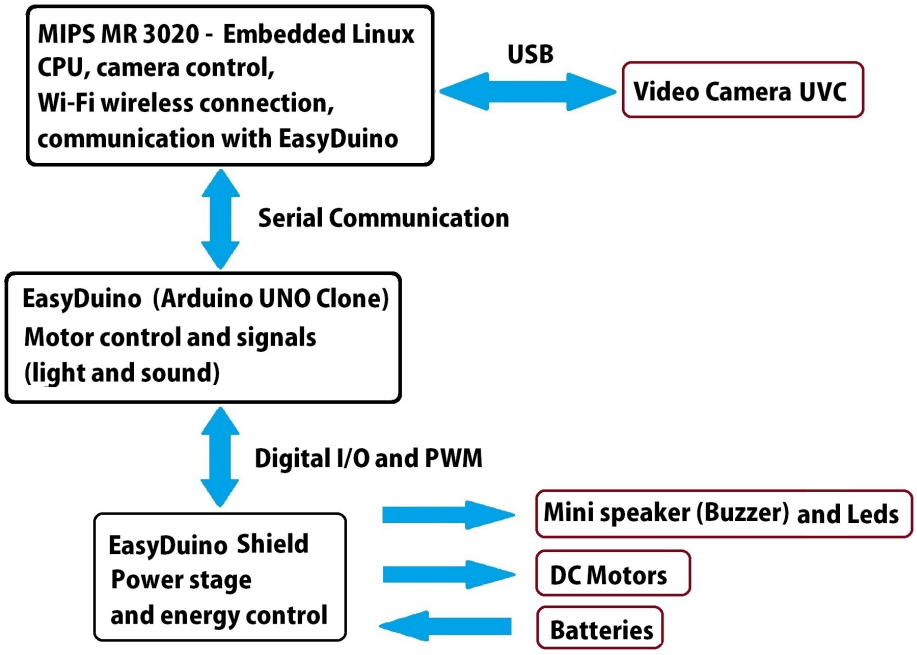

Fig. 1. System's hardware architecture schematics.

Fig. 2. Features of **MIPS** MR3020 controller board

.

2.3 Software Architecture

Software architecture, as depicted in **Fig. 6**, is structured over five clearly defined layers, distributed into the two components necessary for utilization of the educational robotic platform. Components shown in a red box were specifically

Fig. 3. Features of **EasyDuino** board.

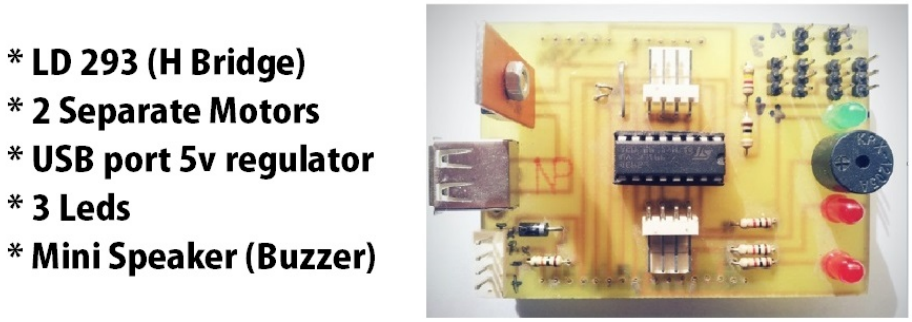

Fig. 4. Features of **EasyDuino**'s **Shield**.

developed for the project. Other components are open source software pieces which were set up, and in some cases, modified, for the robotic platform objectives.

In the remote component, i.e. PC, portable or mobile device, the following software levels are found:

- User programs.
- Myro and Vision libraries.
- Operating System, with TCP/IP communication capability.

The following software levels are found in the robot:

- Embedded Linux operating system, with TCP/IP, wireless, USB, UVC and serial drivers.
- Firmware for the **Atmega8** chip located in the EasyDuino board, with serial, engines, LEDs, and speaker drivers. To process the packets coming from the remote computer's Myro library, the software interpreter for Myro's protocol was also implemented into this firmware.

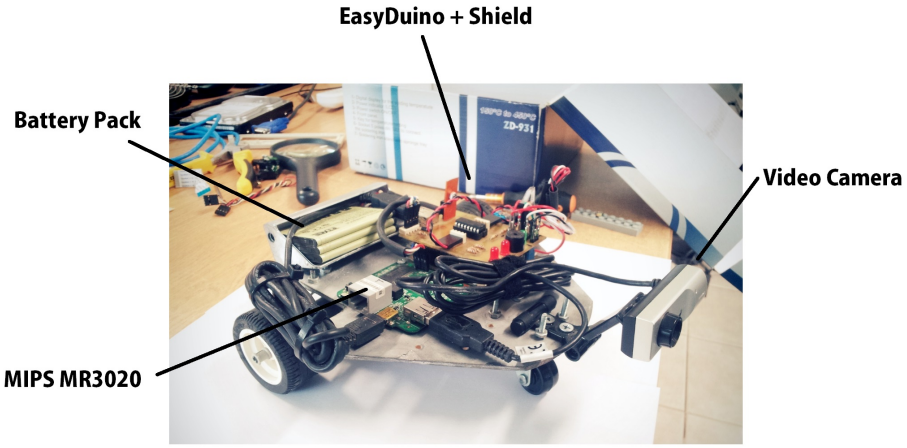

Fig. 5. Educational robot's hardware assembly.

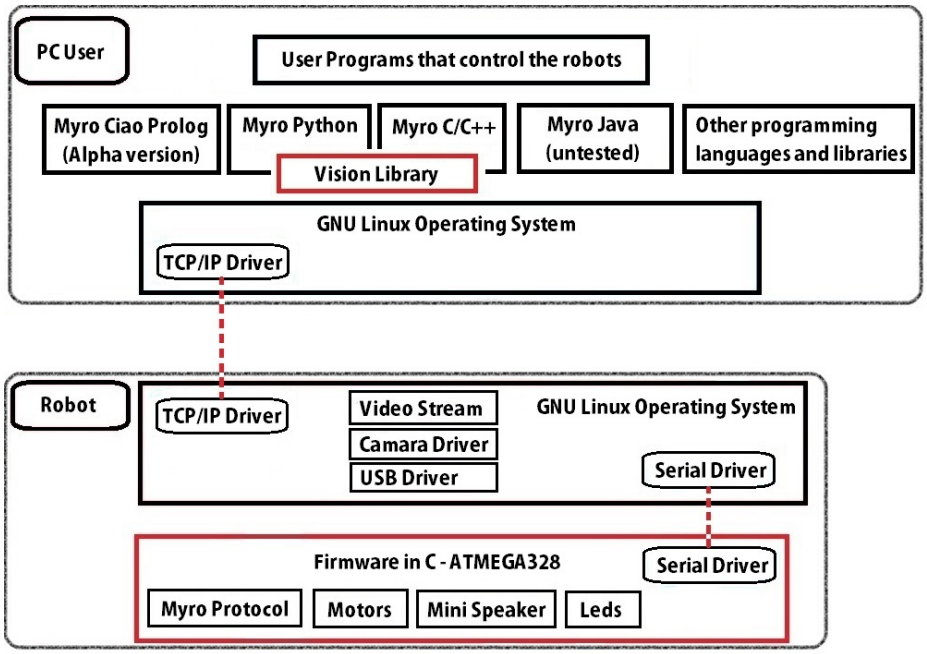

Fig. 6. Systems's software schematics.

User Software User programs using Myro's run-time library finally execute in a PC or remote portable computer to control the robots. As Myro library is

implemented in several different programming languages, the user has an array of choices when selecting a language for program development in this educational platform.

In this second level, we only needed to adapt Myro to TCP/IP communications, as the original version was developed to use robots through Bluetooth serial communications. In our implementation, we decided to use wireless communication, which provides a broader bandwidth and a broader distance range. These are required for vision streaming and for transmission of Myro's protocol packets. Myro's original communications modules were replaced to support TCP/IP, enabling the embedded Linux systems carried by the robots to perform wireless communications.

The vision component was implemented as a shared library, so as to be used by user programs with any existing Myro implementations.

Robot Software The robot shows two software layers: the Linux system embedded in the MIPS hardware, and the firmware developed for the EasyDuino board. The Linux system was built up from Openwrt, compiled and configured to be used as a communications interface and for vision control through a **UVC** (USB Video Class) camera. Myro's protocol packets received via wireless by the robot are routed by the Linux system to EasyDuino firmware through a serial link. In this way, several TCP/IP clients can make use of the robot independently. The wireless link is also used to send images from the robot's camera, and for regular system management.

The software developed for the EasyDuino board (firmware for the AVR Atmega8 micro-controller) has four principal modules: LEDs and speaker controller, engine controller, serial controller, and the interpreter module for Myro's protocol.

Whenever a packet is received from the Linux embedded system by the serial interface, it is routed to the module decodifying Myro packets. Once it is decodified and interpreted, the actuators pertaining to the received command (engines, LEDs, speaker, name management, etc.) are activated; and this same module sends a response packet, as defined by the protocol, through the serial interface.

Using an AVR Atmega micro-controller as an architectural component for the EasyDuino board allowed all robot software to be developed with open source tools, notably GNU GCC (GNU Compiler Collection).

2.4 Vision

Vision is controlled by the Linux system within the robot through the UVC driver and the embedded mjpg-streamer [**19**] application. An additional, C-language vision library was also developed to ask the camera for images and to process them. This library was integrated into Myro, so that user programs may use the robot's vision for environment sensing.

Whenever a user program uses the vision API, the vision library asks the robot for images to be processed. The robot's mjpg-streamer application then sends a video stream of 4 FPS with a 160x120 pixels resolution for every image.

The vision library receives these images and detects, using the cvblob library [**20**], objects by color or blobs, returning statistics about the elements identified. Then, the user program may call query functions to know size and location of the objects found. User programs can figure out, at pixel-level precision, how many pixels left or right from the robot's viewport center an interesting object has been found.

Using these recognition primitives based upon the objects' size (which tells about the distance) and location (which tells about the horizontal orientation regarding the robot's viewport), simple "object search" programs interacting with the real world can be implemented.

2.5 Using Robots

Myro software is used from the students' or teachers' PCs or netbooks. The robot is remotely controlled through programs written in Python, C, C++, Java or CiaoProlog. A small program of a few Python sentences is shown in **Fig. 7**. In this example, the robot will turn counterclockwise on its axis until a white object is found by its vision system. As can be seen, the API is easy to use, allowing to access the different capabilities in the platform (such as movement, vision, LED and speaker output devices) in a simple manner.

```
# python
>> from robofai import *
>> init()
>> detectar_objetos()
>> while porcentaje_color(BLANCO) < 90:
>>           izquierda(.3, 1)
>>           detectar_objetos()
>>
>> print "Encontramos el objeto color blanco. Festejamos!"
>> encender_led_izq()
>> beep()
>> apagar_led_izq()
```

Fig. 7. Color-searching algorithm example.

3 Results

In the second half of 2012 a first prototype was used in two dissemination talks about our courses, and in laboratory classes taught in *"Introducción a la programación"* within technical courses. These first real-world tests suffered from some expected inconveniences, especially regarding electronics, software and mechanical parts. Teachers in charge of those activities and projects, motivated by classroom innovation, contributed several improvements to software robustness. Some electronic problems got better software detection, and automatized actions were set in place to recover from problems. These changes can be viewed in the version history from the project repository [**21**].

Similar, substantial improvements were made during 2013, with contributions to the different software and hardware layers. That same year, the robot were used as a principal platform for the annual extension project *"Divulgando Computación con Robots en la Escuela Media"*. This project held meetings with secondary school students who were taught programming with robots there. The meetings sported a low rate of problems related to the robotic platform, and in all the cases they were solved during the same meeting [**22**].

Also during 2013, within *X Jornadas de Informática de la Universidad de la Patagonia Austral*, Río Gallegos, these educational robots were used as a testing tool in four university activities: an extension course and postgraduate course on Artificial Intelligence (*"Robótica Cognitiva"*), a programming workshop for university students, and a dissemination talk about university courses held during the Jornadas [**23**]. During 2014 they are being used as a principal platform in the annual extension project *"Con la Escuela Media Programamos robots y Conocemos más de Computación"*, in the initial Artificial Intelligence courses, and as a laboratory tool for the development of several undergraduate theses related to vision and Artificial Intelligence [**24**].

4 Conclusions

- Our design has allowed us to build a few educational robots at a low cost, enabling the use of this platform in several places at the same time for different purposes. Field usage has allowed us to improve hardware and software architectures, and we hope to keep making them better through the feedback from teachers and students, who report problems or ask for new features.
- Our experiencies have also awaken interest in local media [**25**] [**26**], broadening the reach of information about our courses in informatics, academic and extension activities.
- A Myro framework prototype for the CiaoProlog programming language is currently being implemented. This language is used in Artificial Intelligence classes and related undergraduate theses.
- Some tasks are still to be accomplished in order to comply with the Copyleft Hardware licensing definition. For instance, the printer circuit boards' schematics and files should be ported to formats supported by free software.

Fig. 8. Extension project "Con la Escuela Media Programamos Robots y conocemos más de Computación" 3rd meeting, Salón Azul, Universidad Nacional del Comahue, 06/24/14.

At the moment, they are in **eagle** format. Eagle is a non-cost software, but not a free software. More adequate formats are those generated by **Kicad** or **Geda**.

– Existing documentation, schematics, printed circuit boards designs, robot software, example user programs, installation tutorials, and every information related to the project can be obtained from our web site and from the git version repository. This fulfills most of the requirements of copyleft hardware and free software [**16**].

References

1. Tucker Balch, Jay Summet and Doug Blank, *Designing Personal Robots for Education: Hardware, Software, and Curriculum*, IEEE Pervasive Computing, Vol. 7, pp. 5-9, 2008.
2. Mikko Apiola, Matti Lattu and Tomi A., *PasanenCreativity and intrinsic motivation in computer science education: experimenting with robots*, ITiCSE '10 Proceedings of the fifteenth annual conference on Innovation and technology in computer science education Pages 199-203 - ACM New York, NY, USA 2010 - ISBN: 978-1-60558-820-9.
3. *Institute for Personal Robots in Education wiki:* `http://wiki.roboeducation.org`
4. Stefanie A. Markham and K. N. King, *Using personal robots in CS1: experiences, outcomes, and attitudinal influences*, Proceedings of the fifteenth annual conference on Innovation and technology in computer science education, pp. 204-208, (2010).

5. *Calico project website:*http://calicoproject.org/.
6. Douglas Blank, Jennifer S. Kay, James B. Marshall, Keith O'Hara and Mark Russo, *Calico: a multi-programming-language, multi-context framework designed for computer science education*, Proceeding SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education Pages 63-68 ACM New York, NY, USA 2012 - ISBN: 978-1-4503-1098-7.
7. *Calico-Myro Robots website:* http://calicoproject.org/Calico_Myro\#Robots.
8. *Scribbler Robot website:* http://wiki.roboteducation.org/Myro_Hardware.
9. *The Finch Robot website:* http://www.finchrobot.com/.
10. *Hummingbird Robot website:* http://www.hummingbirdkit.com/.
11. *Grupo RobotGroup website:* http://www.robotgroup.com.ar/.
12. Díaz Javier F., Banchoff Tzancoff Claudia M., Martin Sofía y López Fernando, *Aprendiendo a Programar con Juegos y Robots*, VII Congreso de Tecnología en Educación y Educación en Tecnología. Universidad Nacional del Noroeste (Buenos Aires, Argentina), Junio 2012 - http://hdl.handle.net/10915/19307/
13. Carlos I. Camargo, *El papel del Hardware Copyleft en la enseñanza de sistemas embebidos*, CASE (Congreso Argentino de Sistemas Embebidos) 2011.
14. Werner Almesberger, *Haciendo Hardware Copyleft*, Comunidad Qi Hardware - 2011 - http://downloads.qi-hardware.com/people/werner/fisl12_es.pdf
15. *Robot Educativo wiki:* http://labti.fi.uncoma.edu.ar/trac/wiki/RoboTitos
16. *MR3020 board openwrt website: http://wiki.openwrt.org/toh/tp-link/tl-mr3020*
17. *EasyDuino cloned repository:* https://github.com/zrafa/se_uncoma/robotitos/hardware/easyduino1
18. Rafael Zurita, Rodolfo del Castillo, Miriam Lechner y Eduardo Grosclaude, *Reverse-Engineering a Closed-Box Hardware and Software Linux Embedded System*, Libro de Trabajos de Foro Tecnológico y Pósters - Congreso Argentino de Sistema Embebidos, 14-16 de Agosto 2013, Buenos Aires - ISBN 978-987-9374-88-7.
19. *mjpg-streamer Project website:* http://sf.net/projects/mjpg-streamer/.
20. *Cvblob project website* http://code.google.com/p/cvblob/.
21. *Robot Educativo Frankestito website:* http://github.com/se_uncoma/.
22. *Proyecto de Extensión año 2013 "Divulgando Computación con Robots en la Escuela Media". Universidad Nacional del Comahue:* http://divulgando.fi.uncoma.edu.ar.
23. *Boletin Oficial de Noviembre de 2013 - Gobierno de Santa Cruz:* http://www.santacruz.gov.ar/boletin/13/noviembre13/12noviembre2013.pdf.
24. *Proyecto de Extensión año 2014 "Con la Escuela Media Programamos robots y Conocemos más de Computación". Universidad Nacional del Comahue:* http://robots.fi.uncoma.edu.ar.
25. *Diario La Mañana del Neuquén - Edición del 04/10/2013:* http://www.lmneuquen.com.ar/noticias/2013/10/4/aprendieron-a-programar-y-crearon-a-frankestito_202090.
26. *Programa de Televisión "Código Rupestre", capitulo "El lenguaje de los Robots":* http://www.codigorupestre.com.ar/?p=443.