2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

ROS-based Online Robot Programming for Remote Education and Training*

Gustavo A. Casañ[1], Enric Cervera[1], Amine A. Moughlbay[2], Jaime Alemany[1] and Philippe Martinet[2]

Abstract— **RPN (Robotic Programming Network) is an initiative to bring existing remote robot laboratories to a new dimension, by adding the flexibility and power of writing ROS code in an Internet browser and running it in the remote robot with a single click. The code is executed in the robot server at full speed, i.e. without any communication delay, and the output of the process is returned back. Built upon Robot Web Tools, RPN works out-of-the-box in any ROS-based robot or simulator. This paper presents the core functionality of RPN in the context of a web-enabled ROS system, its possibilities for remote education and training, and some experimentation with simulators and real robots in which we have integrated the tool in a Moodle environment, creating some programming courses and make it open to researchers and students (`http://robotprogramming.uji.es`).**

I. INTRODUCTION

Online robots and remote laboratories have been around for nearly two decades, with considerable success [1]. With cross-platform middleware [2] apparition and the adoption of new powerful World Wide Web standards [3], we may well be approaching a new golden era for web robots.

The availability of such platforms will surely increase the productivity of the robotics research community, yet they will also become invaluable as educational resources, for teachers, students and interested public. Sophisticated robot platforms could be made accessible worldwide, being the only cost for the user the price of an Internet connection.

Nowadays, there already exists a myriad of web-enabled robots, in theory ready to be remotely controlled, their sensors and outputs visualized. An awesome example of such a system is the PR2 Remote Lab [4], which enables a large community of researchers to use a state-of-the-art yet expensive platform.

However, to our knowledge, most existing systems lack the fundamental capacity of allowing remote users to easily write and execute a program *as if it were running on the real robot*. Usually, the interface only makes it possible to control the elements of the robot. In some cases, scripting capabilities for executing a limited set of commands are provided [5].

In this paper, we present a system that allows users to *remotely program* a ROS-enabled robot or simulator and explain how we are testing it through a Learning Management System, a Moodle system[1] in which we have created several courses and one challenge. ROS is a flexible framework for writing robot software. It collects tools, libraries, and conventions to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. With the advent of cheap robotic kits, teaching with robots has become popular, specially to ease the learning process of introductory programming courses [6], [7], [8], [9]. Robots provide entry level programming students with a physical model to visually demostrate concepts and ideas.

The programs consist of *fully-functional ROS scripts*[2], which are executed as ROS nodes in the server. As such, they can access all ROS topics and services, without remote communication overhead during execution. The output of the process is returned back to the user's browser, and a bag of recorded topics is readily available to download for further analysis. The server connects to a Moodle External Tool, which allows the user to interact with IMS LTI-compliant learning resources and activities [10].

The rest of this paper is organized as follows: Section II describes some related work on web-based robot laboratories. An overview of the RPN tools is presented in Section III. Thorough experimental work with simulators and robots is described in Section IV. Finally, conclusions and future works are outlined in Section V.

II. RELATED WORK

Practically from its conception between the late 1980s and early 1990s, the Internet was realized to allow remote users to interact with and monitor robots [1].

After being online for over ten years, the Telerobot of the University of Western Australia has become one of the most popular remote laboratories, and similar systems have proliferated since then [5], [11], [12], [13], [14], [15], [16].

The PR2 Remote Lab [4], [13] represents a milestone in online robot systems. Previous attempts focused on simple experiments and online learning, and did not build upon shared robot middleware frameworks. This laboratory uses Robot Web Tools [17], a collection of open-source modules and tools for building web-based robot apps, allowing web applications to interface with robots running ROS.

Another milestone is the RoboEarth project [18]. A more ambitious system, it consists of a network and database

*This paper describes research done at the Robotic Intelligence Laboratory, with support in part by Ministerio de Economa y Competitividad (DPI2011-27846), by Generalitat Valenciana (PROMETEOII/2014/028), by Universitat Jaume I (P1-1B2014-52), and by IEEE RAS under a CEMRA grant (Creation of Educational Materials for Robotics and Automation).

[1]Enric Cervera, Gustavo A. Casañ and Jaime Alemany are with Robotic Intelligence Laboratory, Jaume-I University, 12071 Castelló, Spain {ecervera, al079333}@uji.es

[2]Amine A. Moughlbay and Philippe Martinet are with IRCCyN, Ecole Centrale de Nantes, France {Amine.Abou-Moughlbay, Philippe.Martinet}@irccyn.ec-nantes.fr

[1]http://www.moodle.org
[2]http://www.ros.org/wiki/rospy

repository where robots can share information and learn from each other about their behavior and their environment.

Robot benchmarking also benefits from the availability of common online platforms for the development and testing of algorithms [19]. To do so, easy means of executing robot programs should be available. A REST-based architecture has been proposed in [20] and demonstrated in [21] with remote experiments on visual servoing.

While most existing remote laboratories have proven highly successful and invaluable for spreading the use and knowledge of online robots, RPN aims to fill a gap, by providing a simple, seamless way of executing a *real* program for a remote user.

III. DESCRIPTION OF THE SYSTEM

RPN consists of a simple scripting interface, with a text box and submit button built upon Robot Web Tools, which can be accesed as a Moodle LTI resource. A similar capability is available in the PR2 Remote Lab, yet they differ in a crucial aspect: PR2 scripting is done in Javascript, and it runs on the client side; RPN scripting is done in Python (but it could be done in any other ROS-supported scripting language), and it runs on the server side.

As a result, RPN has these fundamental differences:

1) The script is executed as a true ROS node on the server, with access to any topic or service.
2) The remote communication delay is only present during the transmission of the code, not during its execution.
3) The code is stored in the server, together with its output and a bag of the topics, readily available for downloading.

Flexibility comes at the expense of security, though. The remote script is allowed to access the inner parts of the system. To improve security we have taken the step of creating a Virtual Machine (VM) for each student and thus the code does not execute in the real machines, reducing risks. Additional security policies have been enabled, as provided by Robot Web Tools [3]: protected topics and services are necessary when there are critical services that the client should not interfere with; key authorization allows the system to limit access to specific, trusted users.

Fig. 1 depicts a block diagram of the system. RPN server side is built upon Robot Web Tools [3], [17] for communicating with the clients. In addition, it also communicates directly with ROS for dynamically starting new processes, i.e. executing the client programs. It is also responsible of launching *rosbag*[3] for data logging.

Communication is exclusively performed through ROS. Consequently, RPN does not rely on any particular detail of the underlying hardware or software, and it can cope with any ROS-enabled system.

The client side of RPN consists of an HTML5, Javascript-enabled web browser, which runs the Javascript widgets.

[3]http://www.ros.org/wiki/rosbag

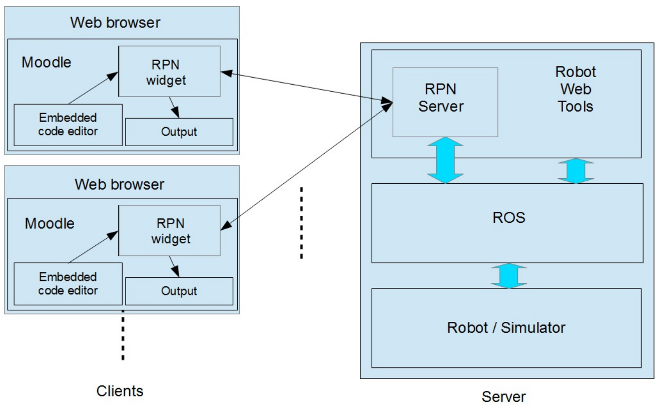


Fig. 1. Overview of the system

The program source code is typed in a user-friendly, syntax-highlighting, embedded editor[4]. Very little additional input is required: a text field with the file name for archiving purposes, and a selectable button which enables or disables *rosbag* recording.

A single click on the *run* button is enough to launch the processing of the code, and trigger the whole interaction process between client and server. Fig. 2 depicts a more detailed view of such interaction between the client and server sides. Program code is written in the embedded browser editor, on the client side. Upon pressing the *run* button, a goal is generated by the RPN client. This goal consists of an identifier for the program, the code itself as a string, and an additional parameter which indicates whether a bag of ROS topics should be recorded or not.

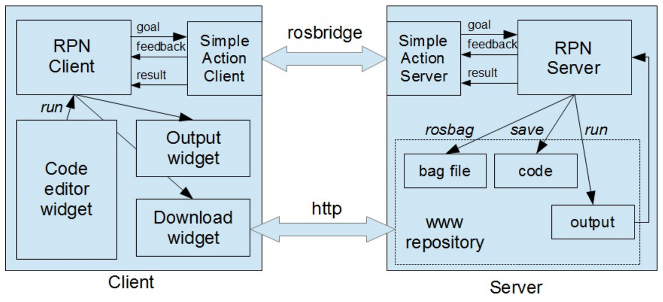


Fig. 2. Detailed interaction between client and server

This information is sent from the client to the server through *rosbridge*[5]. Rosbridge provides a JSON API to ROS functionality for non-ROS programs. Of the variety of front ends that interface with rosbridge, we employ the Websocket server [22] which allows web browsers to interact with ROS, in the same way as used in the PR2 Remote Lab [4].

Communication between client and server is implemented with the *actionlib* ROS package[6]. First, the RPN server checks the goal: if a bag is required, a *rosbag record* process

[4]http://codemirror.net/
[5]http://www.ros.org/wiki/rosbridge_suite
[6]http://www.ros.org/wiki/actionlib

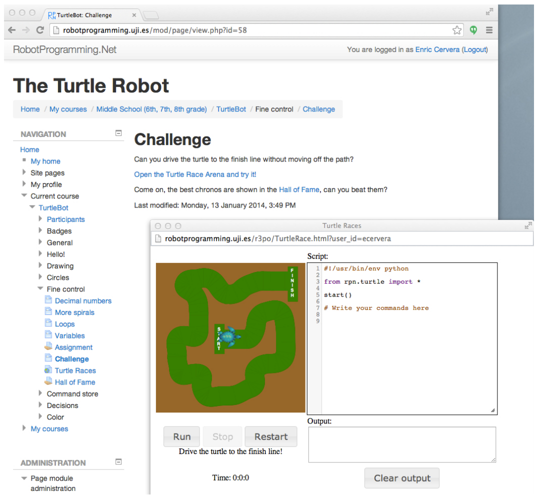
is launched. Second, the string of code is saved to a local file inside a *sandbox* ROS package, with the proper format, extension, and permissions.

Next, the code is executed with the *rosrun*[7] tool, with its standard output redirected to the RPN server. Upon finalization of the user process, the *rosbag* process is stopped; the output of the process is both saved to a file, and returned back to the client as result.

Upon reception of the result, the RPN client updates the output and download widgets. The output of the process is thus displayed in the browser window, and the resulting bag file can be downloaded in a straightforward way. Both the source code and the output are also stored in the server, and can be downloaded for archiving purposes.

Additional feedback can be provided thanks to the tight integration of RPN with Robot Web Tools: existing widgets allow a 2D map, a 3D robot model, or a MJPEG stream coming from a remote camera to be visualized [17].

RPN can also be integrated with RMS (Robot Management System)[8], a remote lab management tool designed to control ROS enabled robots from the web.



Can you drive the turtle to the finish line without moving off the path?

Open the Turtle Race Arena and try it!

Come on, the best chronos are shown in the Hall of Fame, can you beat them?

Last modified: Monday, 13 January 2014, 3:49 PM

```
#!/usr/bin/env python

from rpn.turtle import *

start()

# Write your commands here
```

Drive the turtle to the finish line!

Time: 0:0:0

Fig. 3. Remote programming with TurleSim: a challenge in the course (back window) and the correcponding programming window (front window) with the 2D world with the Turtle, buttons and the code window.

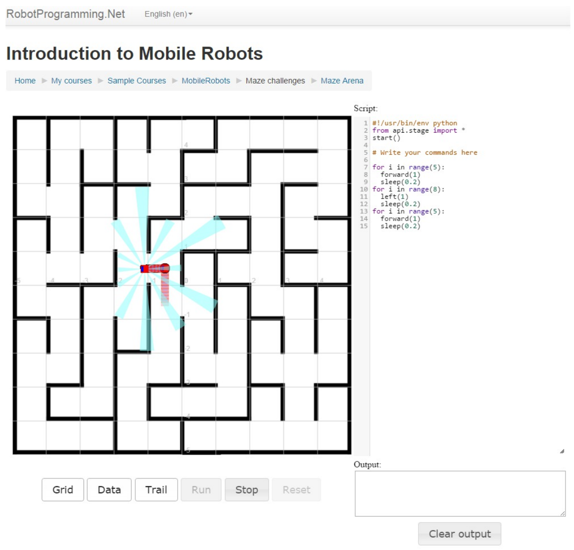
IV. EXPERIMENTAL SETUP

RPN has been tested on different ROS setups, consisting of both simulators and real robots, and some of them have been used to create open robotic courses (`http://www.robotprogramming.uji.es`).

A. Programming Simulators

Two simulators have been used to organize the corresponding programming courses: Turtlesim provides a simulated 2D turtle (in Fig. 3) which is controlled by a ROS topic containing its linear and angular velocity. Simple introductory programs (in Python) can be used for producing different

[7]`http://www.ros.org/wiki/rosbash#rosrun`
[8]`http://www.ros.org/wiki/rms/`


RobotProgramming.Net
English (en)
Introduction to Mobile Robots
Home ▸ My courses ▸ Sample Courses ▸ MobileRobots ▸ Maze challenges ▸ Maze Arena
Script:
Output:
Grid
Data
Trail
Run
Stop
Reset
Clear output

```
#!/usr/bin/env python
from api.stage import *
start()

# Write your commands here

for i in range(5):
  forward(1)
  sleep(0.2)
for i in range(8):
  left(1)
  sleep(0.2)
for i in range(5):
  forward(1)
  sleep(0.2)
```

Fig. 4. Remote programming with MobileSim: the code written in the right window has been sent to the server and is being executed by MobileSim (left window). The output is returned back and displayed in the top left window in the browser.

figures. Mobilesim is similar to the Turtle simulator, with the main difference that this mobile robot has sensors which provide information about the distance between the robot and the nearest obstacle (wall). This allow the development of more complex tasks, like exploring a house or a labyrinth.

The code is directly executable as a ROS script. In RPN, the code is typed in the text editor, and it can be readily executed remotely on the ROS system. Fig. 4 depicts a snapshot of such execution: the window belongs to the client machine that runs the RPN client-side in a Chrome browser.

The resulting trajectory can be seen in the MobileSim window of the server, as images of the simulation are sent to the client browser by an *mjpeg_server*[9] ROS node. The images are captured from the simulator window by *gstreamer*[10], and made available as a ROS topic by *gscam*[11] and them shown as part of Moodle.

The output of the console, consisting of ROS log messages, is presented in the browser too, in the right botton window.

If an error occurs during execution, the abnormal condition is detected and the standard error output of the process is also returned back to the browser, and displayed in the output window. Such feedback includes, as usual, the line number and the characteristics of the error. The debugging process is thus as simple as if the code were running locally.

Should the script hang, the user can abort its execution with the *stop* button. A timeout have also be set in the server to prevent lengthy executions (such as an accidental infinite loop).

This setup is easily transportable to any other ROS-based simulator (e.g. Stage or Gazebo). The server runs all

[9]`http://www.ros.org/wiki/mjpeg_server`
[10]`http://gstreamer.freedesktop.org/`
[11]`http://www.ros.org/wiki/gscam`

the simulator processes, and the client script controls the robot, and comunicates with Moodle to display the feedback. The simulator window is shown in the client browser, and cameras view can also be added to the user interface in other windows.

B. *Programming a Humanoid Robot*

In the third experiment, a robotics challenge, RPN is used for remotely programming a NAO humanoid [23]. Communication with the robot is possible because the ROS NAO driver[12] connects to a custom module running the robot middleware (NAOqi), which has been developed for USARSim [24].

To demonstrate RPN, we have created an artificial environment, a kitchen (as can be seen in Fig. 6), as part of the HUMABOT Challenge 2014[13]. The system user interface (in Fig. 5) is similar to the previous simulators, but we create a window in which we can see the NAO's camera view, and all NAO's sensors are available through programming. We have also added several cameras, which allow to have a 360 degrees view of the robot. Once finished the challenge, we plan to open the environtment like a programming course.

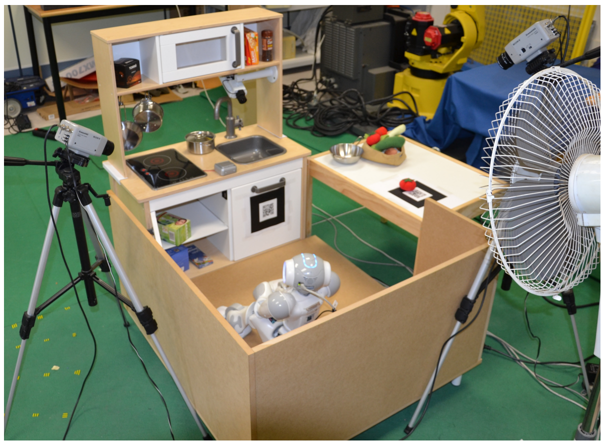


Fig. 6. Kitchen environment for the HUMABOT Challenge.

C. *Opportunities for Education and Training*

RPN is useful in any programming course that uses ROS-enabled robots or simulators. It allows students to work out of the laboratory, *anywhere* with a laptop and an Internet connection. In the Moodle environment we have created the students only have to sign up in any of the courses we have open to the public and begin to learn. The general system security is improved by the use of Moodle system and its own security. As we must be careful when working with real robots, the EJApp booking system[14] is being adapted to control robot access to one student/researcher at a time.

But the benefits far outweigh the problems: there is no need for the local installation and setup of cumbersome simulators, since a simple browser is sufficient (an inherently cross-platform solution), and accessing from different systems is made easier. From a social point of view, users in different countries can access robot equipments that otherwise would not be available, and to facilitate the access we have made versions of the courses in different languages (English, Spanish, Catalonian and Arabic).

And we strongly share the belief [25] that robots can be extremely useful in learning programming skills, and their associated problem solving and reasoning counterparts. The results in a preliminary study with 23 subjects, undergraduate first-year students from engineering degrees with none or little previous experience in programming, were encorauging. They were presented a version of the *Turtle Robot* course on the RPN platform, consisting of programming a simulated turtle-like robot.

The students were asked to solve different tasks and submit their code to the site. And after the end of the activity, they were asked to fill a questionnaire (ease of use of the system, overall satisfaction, ...) about their experience with the course. Overall, the satisfaction degree was complete, with 91% of the students answering positively (strongly agree or agree), and a significant part which complain about the minimalist interface. A complete analysis of the results can be found in an accepted journal paper of Dr. Cervera yet not published ("The Robot Programming Network", in *Journal of Intelligent and Robotic Systems*).

Access to robotic competitions [26], [27], [28], [29] would also benefit from a framework where participants can instantly check their code in their browsers, without the need to replicate the environment locally. As we have previously explained, right now our system is being employed (and tested) as part of the HUMABOT Challenge 2014[15]. Thus, the participants can test remotely their programs on a kitchen environment we have created previously to the competicion. There has not been simultaneus access problems to the NAO robot, maybe because there are less than ten teams enrolled in the competition.

To remark that the system is not dependant on Moodle, and it can be used with any LTI compliant education system. We made a successful test with Eliademy[16], creating a version of the *Turtle Robot* course.

V. CONCLUSIONS AND FUTURE EXTENSIONS

We have presented a web-based extension for ROS-based remote laboratories and online robots, which allows the rapid, seamless execution of a real program in the system, launched by a client user in a remote browser. Also, we have presented the learning environment and programming courses we have created using Moodle and this extension.

Development is in a preliminary stage, and it must yet be tested for a large number of users, althought Moodle has been used for millions of students without problems, which gives garanties about that part of the learning system. Moodle

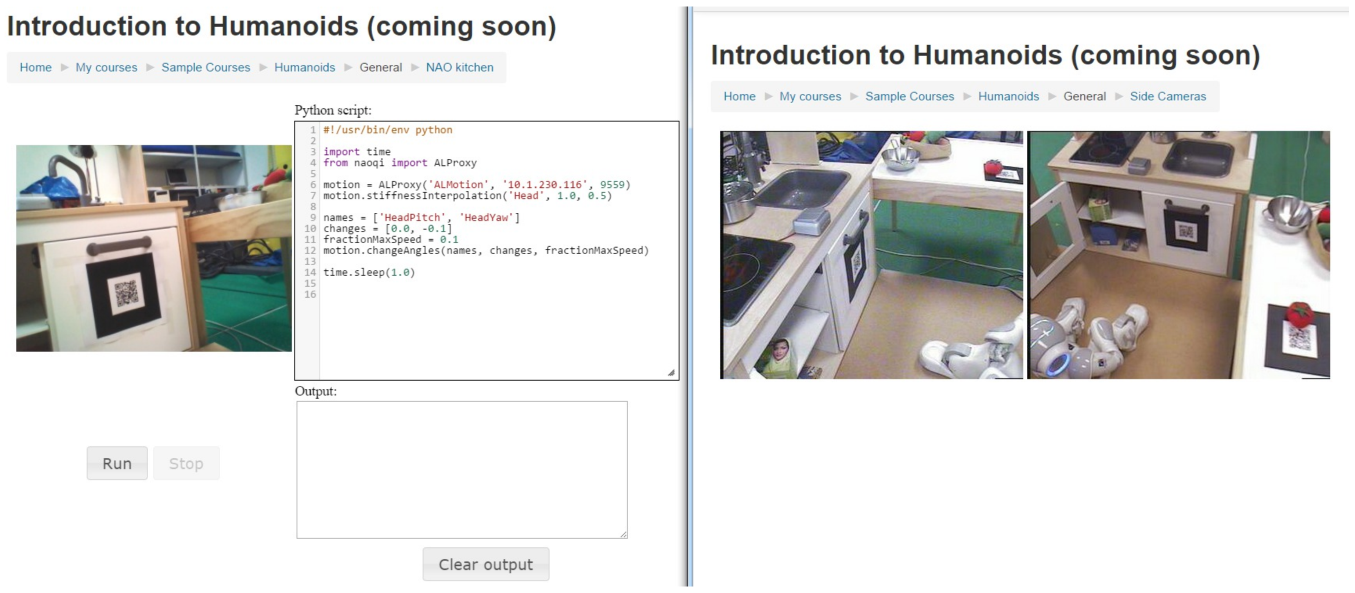
[12]`http://www.ros.org/wiki/Robots/Nao`

[13]`http://www.irs.uji.es/humabot/`

[14]`https://moodle.org/plugins/view.php?plugin=mod_ejsappbooking`

[15]`http://www.irs.uji.es/humabot/`

[16]`https://eliademy.com/es`


Introduction to Humanoids (coming soon)
Home ▶ My courses ▶ Sample Courses ▶ Humanoids ▶ General ▶ NAO kitchen
Python script:
Output:
Run
Stop
Clear output
Introduction to Humanoids (coming soon)
Home ▶ My courses ▶ Sample Courses ▶ Humanoids ▶ General ▶ Side Cameras

```python
#!/usr/bin/env python

import time
from naoqi import ALProxy

motion = ALProxy('ALMotion', '10.1.230.116', 9559)
motion.stiffnessInterpolation('Head', 1.0, 0.5)

names = ['HeadPitch', 'HeadYaw']
changes = [0.0, -0.1]
fractionMaxSpeed = 0.1
motion.changeAngles(names, changes, fractionMaxSpeed)

time.sleep(1.0)


```

Fig. 5. Remote programming with the NAO humanoid: we can see two browser windows, in the left one we can see the written code that has been sent to the server and executed by NAO, and the real time view of the robot's camera; in the right window we can see real time images of the robot actions through two side cameras.

also controls users authentication and time-share facilities. A thorough study of system vulnerabilities to malicious code must also be carried out, but the system has proved to be fairly robust.

Another extension is the addition of feedback messages during the execution of the script, a possibility already available in the *actionlib* ROS package. The client would then periodically update the output in the browser window, thus providing the user with a more interactive experience.

The system is restricted to scripting languages, i.e. languages which do not need compilation. The current implementation is limited to the Python language, but any scripting language with ROS support could be used, e.g. Lisp or Lua[17]. We have began to explore the possibility of using Blockly[18], a block programming language editor, which can generate Python code and thus it can be easly integrated in the system. The blocks programming paradigm is proving to be more attractive to the younger students, increasing the potential users of the system.

Special attention should be devoted to Matlab, a leading programming language for science and engineering. Some experimental work for interfacing ROS and Matlab is available[19]. A Matlab remote programming environment for ROS might become a winner in robotics education.

For research purposes, a more ambitious approach may also be implemented, as proposed in the PR2 Lab [4]: remote users could provide a link to their code within an SVN repository; the code would be automatically downloaded, compiled and executed; users could interactively monitor the results of the code or algorithms remotely, debugging and performing experiments.

[17]`http://www.ros.org/wiki/Client%20Libraries`
[18]`https://developers.google.com/blockly/`
[19]`http://www.ros.org/wiki/groovy/Planning/Matlab`

ACKNOWLEDGMENT

This project relies on open-source software packages, too numerous to list: we sincerely thank all the software developers and users for their lively support and contributions.

REFERENCES

[1] J. Trevelyan, "Lessons learned from 10 years experience with remote laboratories," in *Engineering Education and Research (iNEER), International Conference on*, 2004, pp. 1562–3580.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.

[3] S. Osentoski, G. Jay, C. Crick, B. Pitzer, C. DuHadway, and O. C. Jenkins, "Robots as web services: Reproducible experimentation and application development using rosjs," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2011, pp. 6078–6083.

[4] B. Pitzer, S. Osentoski, G. Jay, C. Crick, and O. C. Jenkins, "PR2 Remote Lab: An environment for remote development and experimentation," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2012, pp. 3200–3205.

[5] R. Marín, P. J. Sanz, and A. P. Del Pobil, "The UJI online robot: An education and training experience," *Autonomous Robots*, vol. 15, no. 3, pp. 283–297, 2003.

[6] V. Dagdilelis, M. Sartatzemi, and K. Kagani, "Teaching (with) robots in secondary schools: some new and not-so-new pedagogical problems," in *Advanced Learning Technologies, 2005. ICALT 2005. Fifth IEEE International Conference on*, July 2005, pp. 757–761.

[7] E. Wang, "Teaching freshmen design, creativity and programming with legos and labview," in *Frontiers in Education Conference, 2001. 31st Annual*, vol. 3. IEEE, 2001, pp. F3G–11.

[8] A. Gage and R. R. Murphy, "Principles and experiences in using legos to teach behavioral robotics," in *Frontiers in Education, 2003. FIE 2003 33rd Annual*, vol. 2. IEEE, 2003, pp. F4E–23.

[9] P. B. Lawhead, M. E. Duncan, C. G. Bland, M. Goldweber, M. Schep, D. J. Barnes, and R. G. Hollingsworth, "A road map for teaching introductory programming using lego© mindstorms robots," *ACM SIGCSE Bulletin*, vol. 35, no. 2, pp. 191–201, 2003.

[10] M. Caeiro-Rodríguez, M. Manso-Vázquez, and L. Anido-Rifón, "Design of flexible and open learning management systems using IMS specifications. the Game Tel experience," *Journal of Research and Practice in Information Technology*, vol. 44, no. 2, p. 151, 2012.

[11] V. Djalic, P. Maric, D. Kosic, D. Samuelsen, B. Thyberg, and O. Graven, "Remote laboratory for robotics and automation as a tool for remote access to learning content," in *Interactive Collaborative Learning (ICL), 15th International Conference on*, 2012, pp. 1–3.

[12] P. Orduna, L. Rodriguez-Gil, D. Lopez-de Ipina, and J. Garcia-Zubia, "Sharing the remote laboratories among different institutions: A practical case," in *Remote Engineering and Virtual Instrumentation (REV), 9th International Conference on*, 2012, pp. 1–4.

[13] S. Osentoski, B. Pitzer, C. Crick, G. Jay, S. Dong, D. Grollman, H. B. Suay, and O. C. Jenkins, "Remote robotic laboratories for learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 449–461, 2012.

[14] I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez, "Remote laboratories for education and research purposes in automatic control systems," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 547–556, 2013.

[15] M. Kulich, J. Chudoba, K. Kosnar, T. Krajnik, J. Faigl, and L. Preucil, "SyRoTek – distance teaching of mobile robotics," *Education, IEEE Transactions on*, vol. 56, no. 1, pp. 18–23, 2013.

[16] L. Furler, A. S. Malik, F. Meriaudeau, and V. Nagrath, "An auto-operated telepresence system for the NAO humanoid robot," in *Communication Systems and Network Technologies (CSNT), International Conference on*, 2013, pp. 262–267.

[17] B. Alexander, K. Hsiao, C. Jenkins, B. Suay, and R. Toris, "Robot Web Tools [ROS topics]," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 4, pp. 20–23, 2012.

[18] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *Robotics Automation Magazine, IEEE*, vol. 18, no. 2, pp. 69–82, 2011.

[19] F. Bonsignorio, J. Hallam, and A. del Pobil, "Defining the requisites of a replicable robotics experiment," in *RSS2009 Workshop on Good Experimental Methodologies in Robotics*, 2009.

[20] R. Esteller-Curto, E. Cervera, A. P. Del Pobil, and R. Marin, "Proposal of a REST-based architecture server to control a robot," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), IEEE International Conference on*, 2012, pp. 708–710.

[21] R. Esteller-Curto, A. P. Del Pobil, E. Cervera, and R. Marin, "A test-bed Internet based architecture proposal for benchmarking of visual servoing techniques," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), IEEE International Conference on*, 2012, pp. 864–867.

[22] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, "ROS and Rosbridge: Roboticists out of the loop," in *Human-Robot Interaction (HRI), Seventh Annual ACM/IEEE International Conference on*, 2012, pp. 493–494.

[23] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "Mechatronic design of NAO humanoid," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2009, pp. 769–774.

[24] S. van Noort and A. Visser, "Validation of the dynamics of an humanoid robot in USARSim," in *ACM Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, 2012, pp. 190–197.

[25] T. R. Flowers and K. A. Gossett, "Teaching problem solving, computing, and information technology with robots," *Journal of Computing Sciences in Colleges*, vol. 17, no. 6, pp. 45–55, 2002.

[26] N. Labhart, B. S. Hasler, A. Zbinden, and A. Schmeil, "The ShanghAI Lectures: A global education project on artificial intelligence," *Journal of Universal Computer Science*, vol. 18, no. 18, pp. 2542–2555, 2012.

[27] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, "Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 6. IEEE, 1999, pp. 739–743.

[28] S. Balakirsky, R. Madhavan, and C. Scrapper, "NIST/IEEE Virtual Manufacturing Automation Competition: from earliest beginnings to future directions," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*. ACM, 2008, pp. 214–219.

[29] M. Veloso and P. Stone, "Video: Robocup robot soccer history 1997–2011," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2012, pp. 5452–5453.