



HAL
open science

The Robot Programming Network

Enric Cervera, Philippe Martinet, Raul Marin, Amine A Moughlbay, Angel P del Pobil, Jaime Alemany, Roger Esteller, Gustavo Casañ

► **To cite this version:**

Enric Cervera, Philippe Martinet, Raul Marin, Amine A Moughlbay, Angel P del Pobil, et al.. The Robot Programming Network. *Journal of Intelligent and Robotic Systems*, 2016, 81 (1), pp.77-95. 10.1007/s10846-015-0201-7 . hal-02369471

HAL Id: hal-02369471

<https://hal.science/hal-02369471v1>

Submitted on 19 Nov 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The Robot Programming Network

An Online Distributed System for Practical Robotics Learning

Enric Cervera · Philippe Martinet ·
Raul Marin · Amine A. Moughlbay ·
Angel P. del Pobil · Jaime Alemany ·
Roger Esteller · Gustavo Casañ

Received: date / Accepted: date

Abstract The Robot Programming Network (RPN) is an initiative for creating a network of robotics education laboratories with remote programming capabilities. It consists of both online open course materials and online servers that are ready to execute and test the programs written by remote students. Online materials include introductory course modules on robot programming, mobile robotics and humanoids, aimed to learn from basic concepts in science, technology, engineering, and mathematics (STEM) to more advanced programming skills. The students have access to the online server hosts, where they submit and run their programming code on the fly. The hosts run a variety of robot simulation environments, and access to real robots can also be granted, upon successful achievement of the course modules. The learning materials provide step-by-step guidance for solving problems with increasing level of difficulty. Skill tests and challenges are given for checking the success, and online competitions are scheduled for additional motivation and fun. Use of standard robotics middleware (ROS) allows the system to be extended to a large number of robot platforms, and connected to other existing tele-laboratories for building a large social network for online teaching of robotics.

Keywords Remote laboratories · Robot programming · Online learning

E. Cervera, R. Marin, A. P. del Pobil, J. Alemany, R. Esteller, G. Casañ
Robotic Intelligence Lab
Jaume-I University of Castelló
Spain
E-mail: eervera@uji.es

P. Martinet, A. A. Moughlbay
IRCCyN
Ecole Centrale de Nantes
France

1 Introduction

Remote laboratories and online robotic systems have been around for nearly two decades, with considerable success [45]. With the advent of cross-platform middleware [38] and the adoption of new powerful World Wide Web standards [33], we may well be approaching a new golden era for web-based laboratories.

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

Nowadays, there already exists a myriad of web-enabled intelligent systems, ready to be remotely controlled, their sensors and outputs visualized. An awesome example is the PR2 Remote Lab [36], which enables a large community of researchers to use a state-of-the-art yet expensive platform.

However, to our knowledge, most systems are built ad-hoc with their customized solutions for management and development. The lack of a standardized remote laboratory framework and the dilemma between offering capabilities and maintaining security prevents the widespread extension of the access to such systems. 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 [28].

In this paper, we present a system that allows users of a Virtual Learning Environment to seamlessly work with web-based laboratories consisting of real robots or 2D/3D simulators. User programs consist of fully-functional source code written on any of the supported programming languages (Python, Lisp, Matlab). The code is executed in the remote laboratory, thus it can access all the available information and services, without any additional remote communication overhead during execution. Upon finishing, the output of the process is returned back to the user's browser, and the generated data is readily available to download for further analysis.

The distributed nature of the framework is the key factor for its scalability, allowing the extension to a large number of learning environments and remote laboratories. Security is emphasized through user authentication services, adequate Application Programming Interfaces (API) and the use of Virtual Machines (VM) for the execution of the user's code.

2 Motivation and Related Work

The RPN project (www.robotprogramming.net) aims to bring together three components (robots, Internet, and programming), which, up to now, have been grouped into pairs with considerable success. Our aim is to advance a step further, combining the potential of all three technologies into a unique learning framework (Fig. 1).

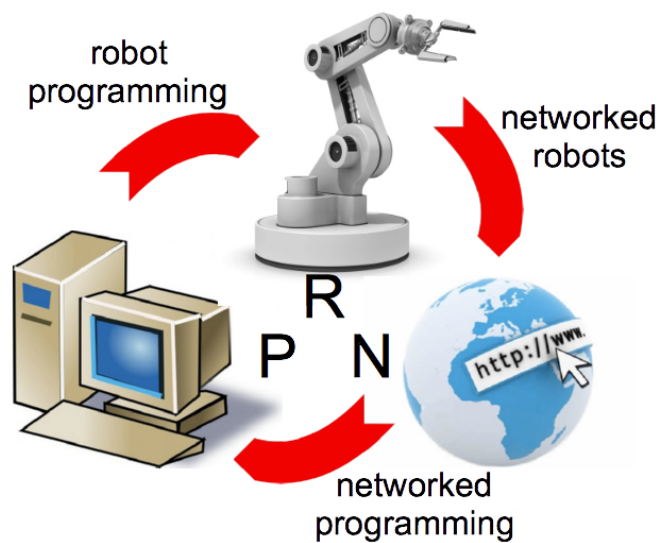


Fig. 1 The RPN concept: online programming of robotic systems.

2.1 Networked Robots

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 and autonomous systems [45].

With the major achievement of being online for over ten years, the Telerobot of the University of Western Australia (UWA) has become one of the most popular remote laboratories, and similar systems have proliferated since then [28, 13, 32, 34, 41, 24, 20, 23].

The PR2 Remote Lab [36, 34] 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 [3], a collection of open-source modules and tools for building web-based robot apps, allowing web applications to interface with a variety of robots running ROS.

Another milestone in the development of online robotics is the RoboEarth project [47]: it is a more ambitious system, which consists of a network and database 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 [6]. To do so, easy means of executing robot programs should be available. A REST-based architecture has been proposed in [17] and demonstrated in [18] 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, in our opinion a gap has not yet been filled: the need for providing a simple, seamless, and secure way of executing real programs for users of a remote laboratory.

2.2 Robot Programming

With the advent of cheap robot kits, teaching with robots has become increasingly popular not only in universities but in high schools, and it has raised a large interest among the educational community to assess its benefits and drawbacks. Robots have been used to ease the learning process of introductory programming courses [12]. Inexpensive robot kits are claimed as a cost- and time-effective means of reinforcing behavioral robotics principles to students of different disciplines (computer science, engineering, psychology) with limited programming skills [21].

With robotic design contests becoming increasingly common, it is claimed [31] that competitions can be an important tool for fostering intellectual maturity, as defined by the Perry Model [35]. A competition involves a clearly defined yet open-ended problem, with many possible solutions. Students are encouraged to work collaboratively in teams, and the goals provide the contextual aspect of applying knowledge.

Using robots in the introductory computer science curriculum has attracted lots of attention in recent years [26] [48]. This approach is meaning to challenge the Computer Science teaching community to move from the premise that computation is calculation to the idea that computation is interaction. Robots provide entry level programming students with a physical model to visually demonstrate concepts or ideas traditionally taught using abstractions.

Robots may add another benefit, since they could become an attractor to Computer Science studies. Number of undergraduates declaring a computer science major is dropping steadily in the last years [27]. Women, always a minority in the field, have become even scarcer than before. Use of robots in introductory computer science has been proposed as a means to fight the enrollment decline [5]. Some experiences report that student enrollment has grown over 2 fold since the introduction of robots [48].

2.3 Networked Programming

In recent years there has been a proliferation of educational websites focusing on interactive online programming. Some of them are MOOCs (massive open online courses) like those of the companies Coursera (coursera.org) and Udacity (udacity.com). Others come without university partnerships or certification processes, like Codecademy (codecademy.com) or Khan Academy (www.khanacademy.org/cs).

The use of Internet and multimedia has brought new opportunities for learning programming. Either replacing or in addition to classroom lessons, it

offers the use of learning material with interactive simulations, and the use of applications for self-checking of the acquisition of knowledge [14].

Online programming environments work directly in the user's browser, without the need of downloading and installing a compiler. The choice of programming language offers a wide range including full-featured Javascript, Ruby, or Python [37], and simpler languages targeted to young students like Logo (turtleacademy.com) or graphical languages like Scratch (scratch.mit.edu) [40].

Most environments require the use of a PC or laptop to write code, but recent initiatives are directed towards computer programming being done directly on the mobile devices [44].

Other proposals make use of new technologies such as three-dimensional virtual worlds, for better effectiveness in the learning of programming [19].

3 The RPN Framework

Our approach aims to bring together the advantages of online programming and networked robots: the appeal of using robots makes learning programming more attractive, while, on the other hand, the possibility of programming the robot provides deeper knowledge about its functioning. The widespread availability of networked robots, through Internet or an academic private network, allow the students to share resources, thus lowering equipment and maintenance costs.

3.1 Hardware and Software Architecture

The overall architecture of RPN is shown in Fig. 2. It is built upon two networks: the Internet (or a local academic network) for the students to access, and a local ROS network which connects the robot systems (either simulators or real robots) and other devices like video cameras.

ROS (Robot Operating System - www.ros.org) [38] is a framework for robot software, consisting of tools, libraries and conventions for a wide variety of robotic platforms. By choosing ROS as the core component of RPN, we gain access not only to a number of different robots, simulators, and vision systems, but also to a large library of robot behaviors which can be readily used for providing high-level functionality to the user, or running in the background for monitoring, data logging or security purposes.

There is a bridge between both networks, consisting of a module which translates the information from and two different languages: ROS topics and services on the robot side, and web data structures on the student side. This module, called `rosbridge` [3], can both read ROS topics and publish them through the web, and write ROS topics with information provided by the web clients [33].

Though the ROS network is accessible at Internet through `rosbridge`, access must be authenticated and authorized by the system, centralized in a Learning

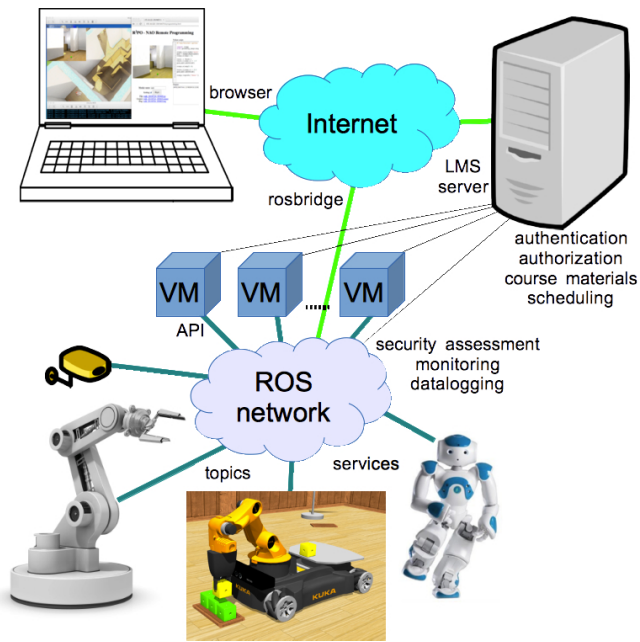


Fig. 2 Overall architecture of the Robot Programming Network: the user is connected to Internet via a browser, and is granted access to the LMS server. The user's code is run on a Virtual Machine, where it uses a secure API for interacting with the ROS modules of the network, through the available ROS topics and services.

Management Server (LMS). The user must first sign in with a recognized user account, or log into the system with an identification provided by other web service (e.g. Gmail or Facebook).

The following list explains the thorough steps from the moment the student types the password until the robot moves:

1. The student is first authenticated and a secure session is started in Moodle.
2. Some courses are freely available to enroll; in others, access is granted by teachers upon request.
3. Once enrolled in a course, the student browses through the Moodle pages, where links to the robots and simulators are shown.
4. When the student clicks on such a link, the server connects to a Moodle External Tool, which allows the user to interact with IMS LTI-compliant learning resources and activities [8].
5. The Moodle External Tool provides the user account information to the LTI-compliant module. After checking authentication, this module connects to the ROS system of the robot or simulator through a secure rosbridge connection.
6. Once the connection is established, the student can use the browser to control the ROS system. In our case, the student writes a program in a

text field, which is submitted to a server process that executes the code in the robot or simulator.

7. The server receives the source code, and launches a new ROS process for the execution of that code. The new process will publish the necessary topics to make the robot move.
8. Both the output of the process and the message errors (if any) will be redirected back to the student's browser window, for monitoring and debugging purposes.
9. Finally, when the student leaves the web page, the connection with the ROS server is automatically closed.

There is no need to modify the Moodle platform for running our system, since support for LTI-compliant materials is already included [4], but it is necessary to add some interfacing code in PHP, in order to build the bridge with the ROS server. The current version only works in one direction (passing the authentication information to ROS) but, since the LTI protocol is defined in both ways, in the future we plan to add feedback to Moodle from ROS, e.g. sending grades to Moodle assignment based on the performance of the robot task.

3.2 Security

Security policies must be established, as in other web laboratories [9,29]: the LMS server is also responsible of the access policy to the shared resources, by storing a database of time slots, where users can book the facilities for a determined amount of time. Only booked users have full access to the system, while others can be monitoring or analyzing the system, in a read-only mode.

The student's code (Python, Ruby, Lua, Matlab, and Lisp can be supported) is not executed directly in the real machines, but instead it runs on a Virtual Machine (VM). Virtualization provides both safety and control of resources. Malicious code has only access to the virtualized system, without any possibility of intrusion into sensitive processes, like those controlling the robot hardware or the RPN system itself. In addition, a VM is allowed to use a fixed number of processors and a maximum amount of RAM memory, thus preventing an overload of the system. In critical cases, the VM can be reset, or directly deleted and restarted to a safe state.

In addition, the code is not allowed to publish directly to the topics that control the robot hardware; instead, it is redirected to similar topics which are filtered by background modules that monitor the state of the robot and either retransmit or block the user commands depending on the safety conditions, e.g. danger of collision. Fig. 3 depicts an example for a mobile robot: the user code does not publish directly to the command velocity topic (`cmd_vel`) of the robot driver. Instead, the topic is read by the background monitor, which reads also the information topics from the robot driver, consisting of the sonar and infrared sensor data. Based on the sensor values, the monitor process determines the safety of the commanded motion, and forwards the values to

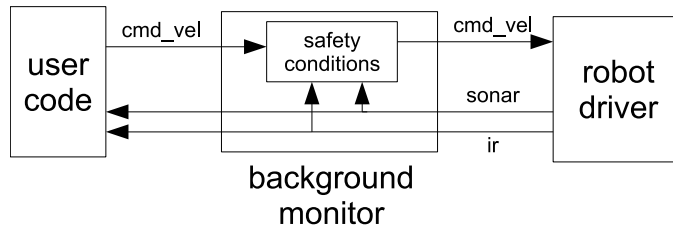


Fig. 3 Example of safety monitoring for a mobile robot.

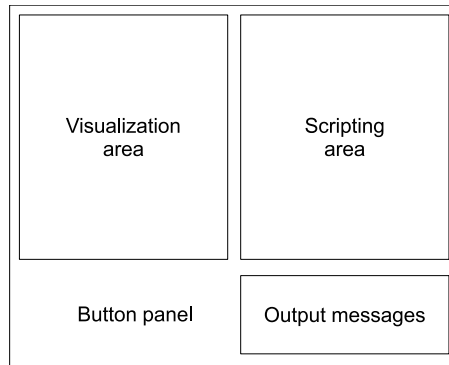


Fig. 4 Layout of the user interface.

the robot driver. This monitoring process is transparent to the user, by using the dynamic remapping capabilities of ROS.

3.3 Scalability

According to the statistics published in the Moodle home page [30], the largest sites in the world currently have up to 1,000,000 users. So the scalability of Moodle is not a problem at all, provided that the appropriate hardware (processing power, bandwidth) is available. Our current system is experimental, thus it runs on a single computer. When the number of user increases, we plan to migrate to a Moodle cloud system. Of course a bottleneck is the number of real robots available, but since our system can connect to ROS systems all over the Internet, we aim to grow a distributed network community of robots, thus the workload can be distributed among online robots on different remote laboratories.

3.4 User Interface

The generic user interface is intentionally kept very simple for clarity and ease of use (Fig. 4). It consists of four window areas: the top left side is the visualization area, where the system displays the simulated setup, or video



Fig. 5 Turtle simulator.

feedback from live cameras; the top right side is the scripting area, where the user types the source code of the program to be run into the system; the left bottom side consists of a simple button panel, for running or stopping the program; finally, to the right bottom side, there is another output area for system messages (compilation errors, console output, etc).

Nevertheless this basic interface can be customized or expanded with additional components, depending on the available equipment of the remote system (cameras) or the visualization needs (2D/3D).

In the following subsections, several examples of functioning user interfaces are thoroughly explained, each of them controlling a different robot setup, namely the ROS Turtle and Stage simulators, the Syrotek mobile robot laboratory [25], and the NAO humanoid robot.

3.4.1 Turtle Simulator

This is a simple 2D simulator without physics, initially designed for teaching ROS concepts, but also suitable for teaching programming concepts or an introduction to mobile robots. It resembles the Logo turtle [39], but the notion of time (even simulated) makes a significant difference is: the velocity of the turtle can be controlled, thus the execution of the code is not immediate, but progressive.

In Fig. 5, the visualization area shows the turtle and the trail path that it as followed. The web code is subscribed to the turtle position topics, and as it moves, new position values are received and the trajectory and turtle position on the browser window are updated.

The turtle moves with linear and angular velocities, allowing the user to program curved trajectories. Additionally, the color of the path is selectable, thus colorful patterns can be drawn.

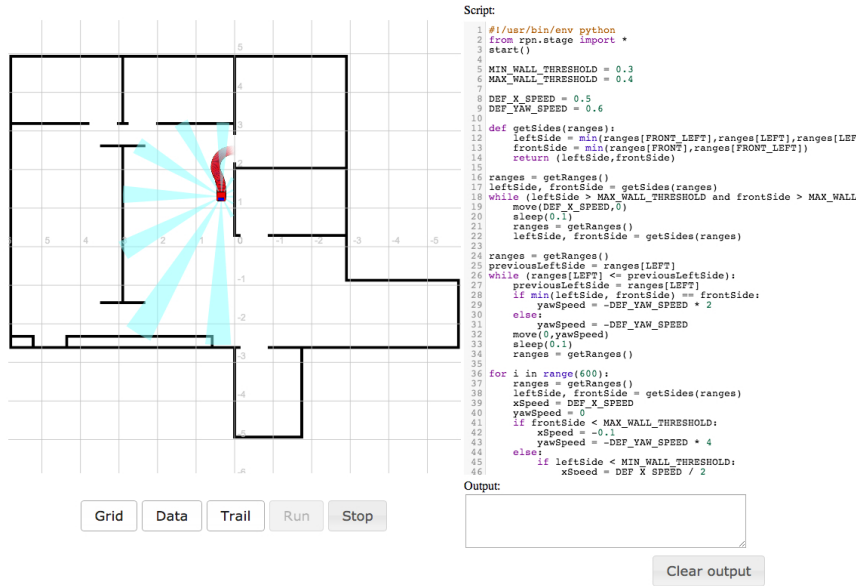


Fig. 6 Mobile Robot 2D Simulator.

A small piece of code is shown in the scripting area. The top five lines are automatically added to include the API module and call an initialization function; the user commands are written below.

In the figure, the user calls the API function $leftArc(a, r)$ which moves the turtle during one second along an arc trajectory of a degrees and radius r . Internally, the function computes the linear and angular velocities and publishes them into the corresponding topics for moving the turtle.

The panel button consists of four buttons for running and stopping the program, clearing the trajectory path, and resetting the turtle position to the window center respectively. Each button triggers the corresponding action, which communicates with the ROS system. For example, the *run* action reads the code from the script area, and calls a ROS service in the network, which analyzes the code and launches a new ROS node in a virtual machine to execute the user code.

The user can abort the execution with the *stop* button. Otherwise, either the code will end by itself, due to an error or to successful execution, or it will be stopped by a timeout that can be set by the system administrator to prevent excessive running times. It can also be aborted for security reasons, if an unauthorized access to system resources is detected.

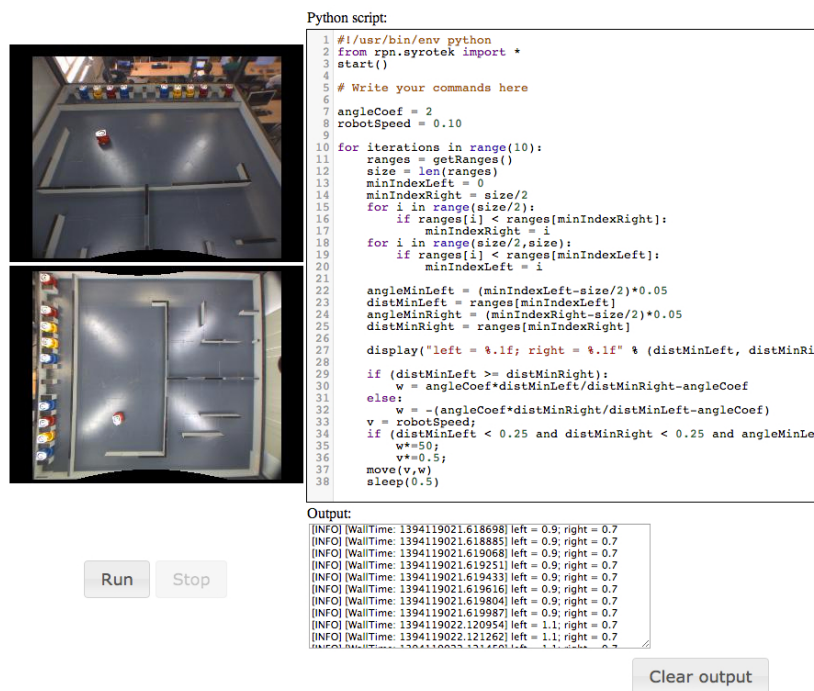
3.4.2 Mobile Robot 2D Simulator

A more powerful and realistic simulator for mobile robots is also available. The Stage simulator is readily available in ROS and it has been integrated in

our framework. Fig. 6 depicts the user interface for this simulator: the visualization area shows the environment and the robot, with optional displaying of the robot's trail path and the range sensors; the scripting area contains a sample code for wall following. As in the previous setup, the first lines are automatically added, for including the API library, and starting the initial setup. The rest of the code consists of arbitrary code (functions, variables, control loops) for performing the task.

Most of the code is devoted to the wall following algorithm. Only two API functions are used: *move(v,w)*, which moves the robot with the given linear and angular velocities, and *getRanges()*, which returns the current readings of the range sensors. Both functions communicate with the underlying ROS topics defined by the simulator module.

The button panel includes the buttons for starting and stopping the user program, as well as some buttons for the main displaying options (grid, sensors, trail).



Python script:

```

1 #!/usr/bin/env python
2 from rpn.syrotek import *
3 start()
4
5 # Write your commands here
6
7 angleCoeF = 2
8 robotSpeed = 0.10
9
10 for iterations in range(10):
11     ranges = getRanges()
12     size = len(ranges)
13     minIndexLeft = 0
14     minIndexRight = size/2
15     for i in range(size/2):
16         if ranges[i] < ranges[minIndexRight]:
17             minIndexRight = i
18     for i in range(size/2, size):
19         if ranges[i] < ranges[minIndexLeft]:
20             minIndexLeft = i
21
22     angleMinLeft = (minIndexLeft-size/2)*0.05
23     distMinLeft = ranges[minIndexLeft]
24     angleMinRight = (minIndexRight-size/2)*0.05
25     distMinRight = ranges[minIndexRight]
26
27     display("left = %.1f; right = %.1f" % (distMinLeft, distMinRi
28
29     if (distMinLeft >= distMinRight):
30         w = angleCoeF*distMinLeft/distMinRight-angleCoeF
31     else:
32         w = -(angleCoeF*distMinRight/distMinLeft-angleCoeF)
33     v = robotSpeed;
34     if (distMinLeft < 0.25 and distMinRight < 0.25 and angleMinLe
35         w*=50;
36         v*=0.5;
37     move(v,w)
38     sleep(0.5)

```

Output:

```

[INFO] [WallTime: 1394119021.618698] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.618885] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.619068] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.619251] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.619433] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.619616] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.619804] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119021.619987] left = 0.9; right = 0.7
[INFO] [WallTime: 1394119022.120954] left = 1.1; right = 0.7
[INFO] [WallTime: 1394119022.121262] left = 1.1; right = 0.7
[INFO] [WallTime: 1394119022.121570] left = 1.1; right = 0.7

```

Run Stop Clear output

Fig. 7 Syrotek mobile robot Internet laboratory. The source code in the script is implementing a wandering behavior, using the range sensors for obstacle avoidance.

3.5 Mobile robot laboratory

Besides simulation, the RPN system can be seamlessly integrated with ROS-based robotic systems, even those which are already set up. Thus, we have performed a successful connection with the Syrotek system [25]. The SyRoTek (System for robotic e-learning) is an online laboratory set up at the Intelligent and Mobile Robotics Division (IMR) of the Faculty of Electrical Engineering, Czech Technical University, which allows users to remotely (via internet) control a multi-robot platform in a dynamic environment.

For the integration with RPN, we have adapted the existing camera widgets to the visualization area of the user interface, as depicted in Fig. 7, and we have developed a simple API, based on that of the mobile robot simulator, for managing the topics that control the real robots.

As a result, a user can simply program a robot behavior with the script interface. This is far simpler and more straightforward than the standard development method, where the user must connect to a terminal of the Syrotek server, upload the source code, and launch it. The provided API also contributes to lowering the difficulty, since it hides some implementation details, by providing the user with the same functions (e.g. *move*, *getRanges*) that have already been presented in the simulator.

3.5.1 Humanoid Robot

Humanoid robots are also supported, both in simulation and real platforms. Fig.8 depicts a snapshot of a realistic NAO humanoid 3D simulator based on USARSim [1][2]. The humanoid executes the user's code in the simulator and feedback from the simulated cameras is provided.

The user interface for programming a real humanoid robot is shown in Fig. 9. The robot is a NAO humanoid manufactured by Aldebaran Robotics [22], a widespread platform for teaching and research.



Fig. 8 3D simulation of a humanoid robot.

Being a real robot, the visualization area is now endowed with live camera images: the main image is the robot internal camera, and the three top images are provided by external network cameras from different points of view in the environment (a small-scale kitchen). In addition, remote audio from a microphone is available through a stream server.

Landmarks (QR codes in filled black squares) have been added to the environment to facilitate its perception by the robot, and its localization with respect to the robot's camera. Such landmarks are not intended to be processed by the user program; instead, the background processes track the codes and provide the system with a 3D pose of the landmark model with respect to the (calibrated) camera [11].

As in previous examples, the API library is imported, and the starting function is called, in the first lines of the code, which are added automatically. Later, there are two API functions: *moveHead(pitch, yaw)*, which moves the head of the robot for the given angles, and *talk(string)*, which calls the robot speech synthesizer service for converting the text input to speech.

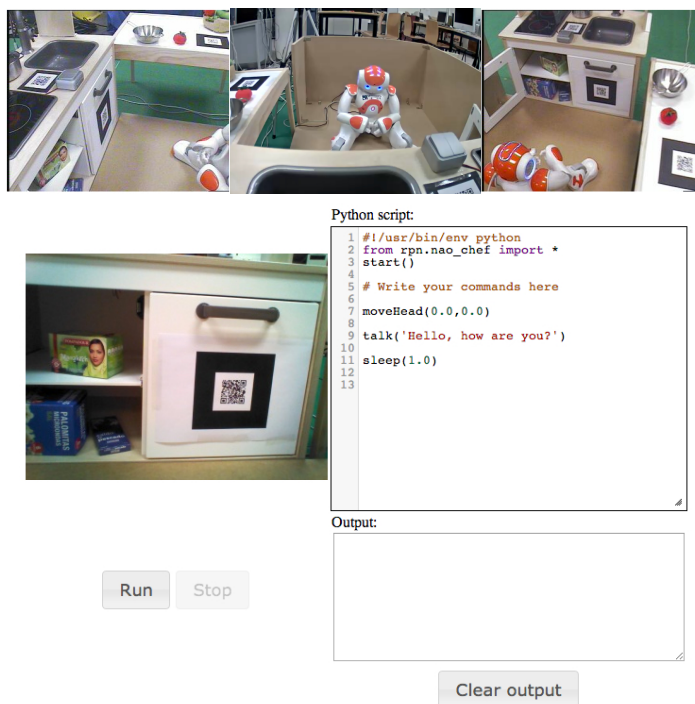


Fig. 9 Humanoid robot.

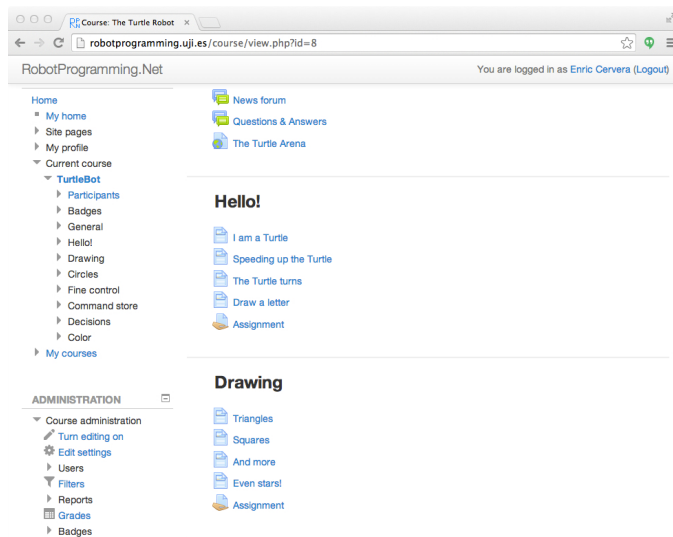


Fig. 10 Main web page of a course in the LMS, with some introductory sections, each consisting of four document pages and one assignment.

4 The Learning Management System

The Learning Management System (LMS) consists of the Moodle software package (www.moodle.org) [15] for the administration, documentation, tracking, reporting and delivery of robotics courses. In this section, the LMS features are described in the context of a simple course that has been carried out in the RPN platform.

4.1 The Turtle Robot

This course is inspired in the Logo turtle [39]: it is targeted to young students (12-14 years) and its aim is to learn the basic concepts of programming: statements, variables, control flow, procedures and functions. It consists of seven units with a few web documentation pages and assignment on each. A competition is also included, where students are asked to program the robot to describe a trajectory in a circuit in the fastest time without going off the path.

Fig.10 depicts the web interface and the structure of three sections, with the inner documents. The sections are not numbered, but they are arranged in increasing difficulty order.

A typical documentation page is shown in Fig.11. The main web page contains the information about the task to solve and some instructions about the solution. The user is asked to program the task on the simulator in the pop-up window.

At the end of each section, an assignment is proposed to summarize the presented contents. The student is asked to solve a programming problem,

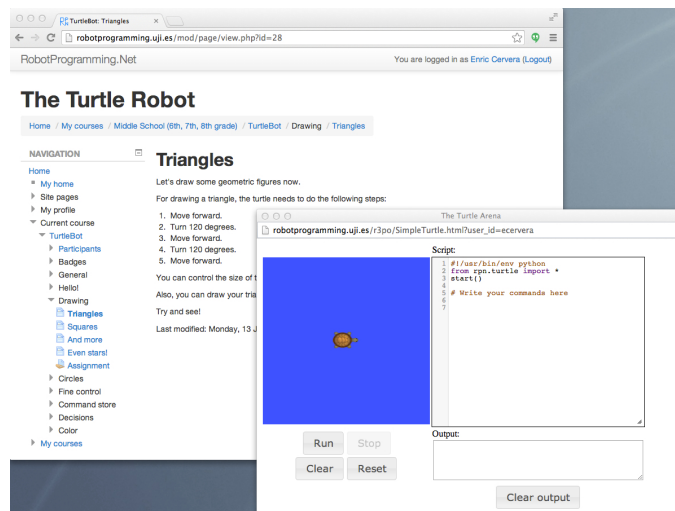


Fig. 11 Documentation page of the Turtle Robot course; information is shown in the main page, and the simulator is available in the pop-up window.

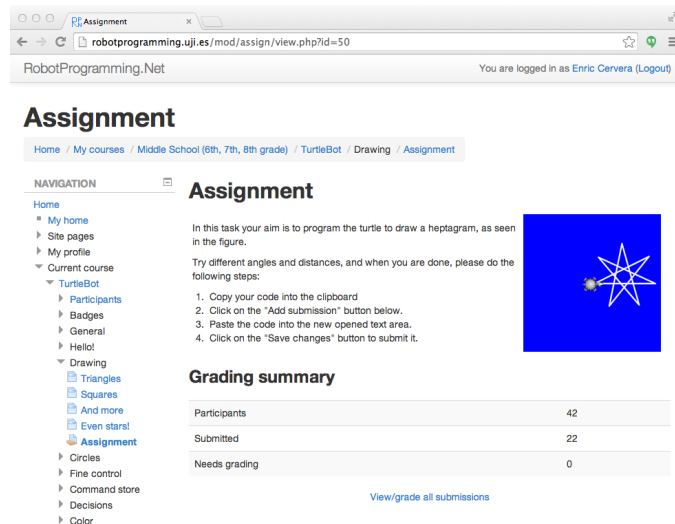


Fig. 12 Assignment page of the Turtle Robot course; the problem is presented, including a figure for information.

and to submit the solution for the teacher to review. Fig. 12 depicts a typical assignment page.

The course includes a competition challenge, in order to increase the motivation of the students. It consists of programming the turtle for completing a circuit, in the minimum time, without going off the path. The web page for the challenge is shown in Fig. 13

The screenshot shows a web browser window with the URL `robotprogramming.uji.es/mod/page/view.php?id=58`. The page title is "The Turtle Robot" and it is part of a "Challenge" section. The main content area contains the challenge text: "Can you drive the turtle to the finish line without moving off the path?" and "Come on, the best chronos are shown in the Hall of Fame, can you beat them?". A pop-up window titled "Turtle Races" is open, showing a simulator arena with a green maze and a turtle. The simulator includes a "Script" editor with Python code, an "Output" field, and "Run", "Stop", and "Restart" buttons. The time is 0:0:0.

Fig. 13 Challenge page of the Turtle Robot course; the competition is explained in the main window, and the simulator arena is shown in the pop-up window.

A Hall of Fame with the best chronos is kept in the course, as depicted in Fig. 14, which shows the result of the pilot experiment that will be presented later.

5 Pilot Teaching Experience

In order to test the RPN platform in real conditions, a pilot study was carried out. The subjects of the study were undergraduate first-year students from engineering degrees (electrical, mechanical, chemistry). With none or little previous experience in programming, they had followed an introductory course in computer science, including the basics of programming in Matlab. Upon completion of this course, they were presented 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. The experiment was not compulsory, but some extra credit was awarded, thus 23 students did take part in the experience. The activity was presented at the end of the course period and it was scheduled for completion during a month.



Fig. 14 Hall of Fame page of the Turtle Robot course.

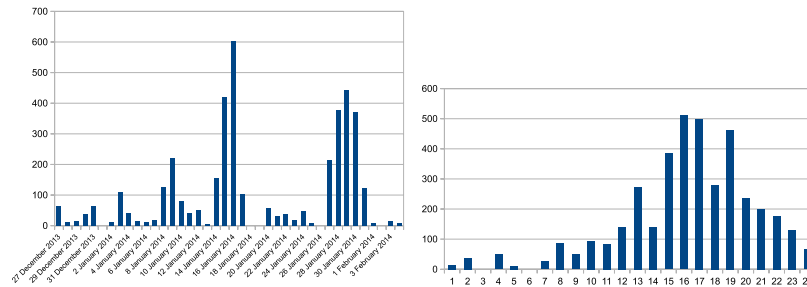


Fig. 15 Number of accesses to the course web versus dates (left), and number of accesses to the course web versus time of the day (right).

5.1 Course traffic analysis

The LMS logs all the student activity, thus allowing the teacher to view statistics and analyze the trajectory of the whole group, or individual students.

The activity during the two first weeks was low. As seen in Fig. 15, which depicts the number of accesses to the site, there are hardly 100 hits/day during that period. A reminder was sent to the students, and its effect on the course activity is significant, with 400 and 600 hits in the following days. Finally, the 'deadline' effect produced a last significant peak in the activity, in the last days of the period. The site remained open and active after the deadline, and some hits were registered but only the assignments submitted in due time were considered for the presented results.

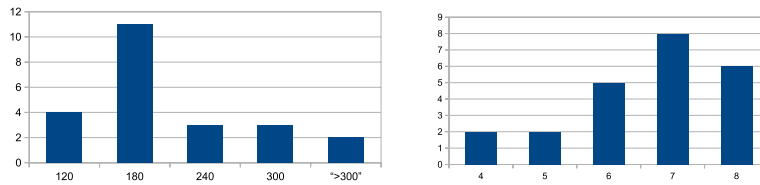


Fig. 16 Histogram of the number of accesses (left) and the number of completed tasks (right) per student.

As for the time of the connections, they concentrated on the afternoon, between 3pm and 7pm, as depicted in Fig. 15 (all the users were local students, i.e. they were presumably working in the same time zone).

The course materials consisted on 26 document pages and 8 assignment pages. The students could browse freely through the materials, but we assume that most of them followed the logical order from the simple initial tasks to the more complex tasks which followed. The number of accesses to the materials ranged between 100 and 300 per student. Fig. 16 depicts the histogram of the number of accesses, with a strong peak in 180 hits, and the histogram of completed assignments. In the latter, most of the students completed 6 or more tasks.

5.2 Data logging

The system recorded all the programs submitted by the students, and the data generated during their execution. When a user logged into the simulator, the system started to record the data. When the user disconnected, recording was stopped. The total amount of recorded data during the pilot test was approximately 2GB of uncompressed files (ROS bags) for 554 sessions. The number of executed programs amounted for 4171 scripts. Of those executions, syntax errors were detected in 728 scripts (17.45%).

The recorded log data can either be downloaded by the user, or analyzed by the teacher. Fig. 17 depicts the recorded trajectories of the turtle robot during one continuous session of a student. The session lasted for 50 minutes; during that time the student run 106 scripts. In this session the robot was programmed for traveling from the starting position to the finish line in the minimum time, while keeping inside the green path.

The trajectories show the progressive adjustment of the trajectories for completing the path with the given constraints. The velocity profile of the robot can also be analyzed: Fig. 18 depicts the linear and angular velocity of the robot during the session. In the first programs, velocities are lower, and they are increased in further executions. The zoom on three selected runs depicts the profile of a typical run, consisting of alternating translation and rotation motions.

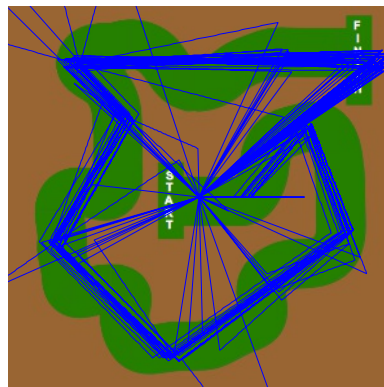


Fig. 17 Recorded trajectories of the turtle robot during a student's session.

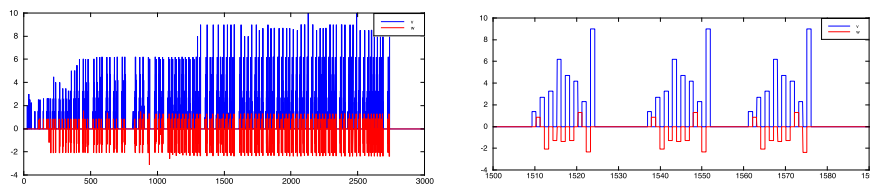


Fig. 18 Velocity profile (blue: linear, red: angular) of the turtle robot during a student's session: all the runs (left) and three selected runs (right).

5.3 Student's Feedback

After the end of the activity, the students were asked to fill a questionnaire about their experience with the robot programming system. The questions are listed in Table 1 and they refer to the perceived differences in learning with respect to traditional methods, the ease of use of the system, the quality of the web, the suitability in learning concepts, and the overall satisfaction degree [23].

The results of the questionnaire are very positive and encouraging. As seen in Fig. 19, 88% of the students believed that the learning experience was better or much better than traditional methods. An almost similar percentage (82%) agreed that the system was suitable for leaning the presented concepts.

The ease of use of the system was confirmed by 91% of the students (strongly agree or agree), but the quality of the experience needs to be improved: though still 51% of the answers were positive (very good or good), there is a significant amount of students (43% acceptable, 6% bad) who complain about the quality of the web, and the response time. The minimalist design of the user interface, as well as the use of plain 2D simulation, may need an enrichment to provide the users with a more challenging experience.

Table 1 Questionnaire for students' feedback.

<i>Learning compared with traditional methods</i>
Did The Turtle Robot help you to visualize the theoretical concepts to be learned?
How would you rate the outcome of your learning using The Turtle Robot if compared with "traditional methods"?
Did The Turtle Robot enhance your ability to understand the theoretical concepts about programming in a new way?
<i>Ease of use</i>
Did you find easy the use of The Turtle Robot?
Did you think that the course was well structured and organized?
Were you able to use The Turtle Robot by following the instructions provided?
<i>Quality of the virtual robot</i>
In which grade will you score to the quality of the virtual robot and its simulation?
In which grade will you score to the quality of the remote connection?
Was the response time of the remote laboratory suitable?
<i>Suitability in learning of relevant concepts</i>
Did the Turtle Robot help you for understanding the concepts of structured programming (conditions, loops) of the lectures?
In which grade do you think that the Turtle Robot can be used for learning programming?
<i>Satisfaction degree</i>
In general, do you feel satisfied with the practical experiences through the Internet?

Overall, the satisfaction degree was complete, with 91% of the students answering positively (strongly agree or agree).

6 Discussion and Future Roadmap

This paper has presented a recent initiative for robotics education, which aims to the creation of a distributed network of robot programming laboratories and their associated teaching materials.

Compared to existing robotic tele-laboratories, the user interface is simplified and tied to a Learning Management System, which guides the students through lessons and practical exercises with progressive level of complexity. Competitions and challenges can be designed and included in the framework.

The current system supports the Python programming language, but more options are in the pipeline, since ROS supports client libraries in Lisp, Lua and Ruby, and third party vendors provide additional languages like Matlab.

Support for compiled languages (C++, Java) could be considered, but an analysis of benefits and costs needs to be carried out. The advantage of higher speed may not outweigh the increased difficulties for compilation in the server, nor the potential security issues of executing compiled instead of interpreted code.

More promising seems the integration of graphical languages like Scratch [40] for courses oriented to young students. This language is entirely browser-based, and some implementations already communicate with real robots [46].

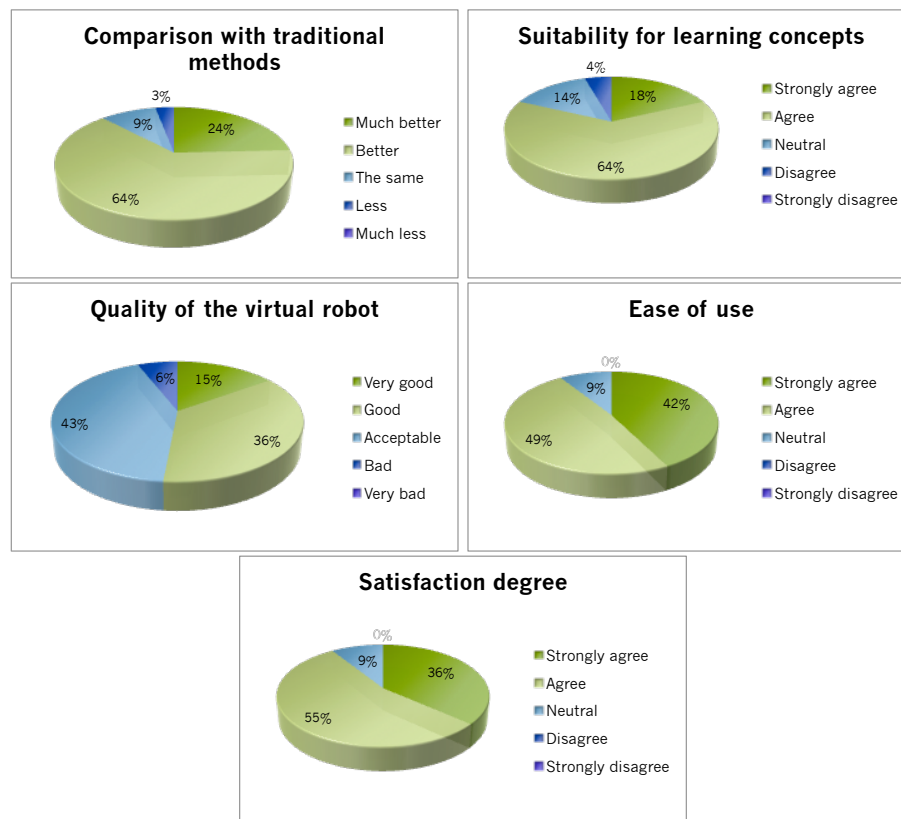


Fig. 19 Results of the questionnaire.

Several examples with simulators and real robots have been presented. As seen with the Syrotek laboratory [25], the system can be extended with any other ROS-based robotic system with Internet access. We are currently working on the connection to the Wurzburg University Mobile Robot Tele Experiments [42].

It would be also very interesting to be able to interact with different kind of standard industrial robots with different standard programming languages. The ROS Industrial framework [16] is a strong candidate for interfacing to such manufacturing robots, and provide a seamless interface for online education and training on industrial platforms.

The results of a pilot sample course have been presented. We plan to open progressively the platform to the general public, and use the presented course materials for teaching introductory courses to mobile robotics and humanoids.

Interconnection with realistic 3D simulators is feasible too: Gazebo, Webots, or USARSim are supported in ROS. However, there is a need for visualization clients for browsers, an effort that is slowly but steadily progressing, with the increasing adoption of the WebGL standard [10]. 3D simulation setups

are increasingly popular for robot competitions like Robocup [43] or FIRST [7]. The availability of an online setup would lower the level of difficulty for newcomers, and increase the user base for potential new participants in the real competitions.

We would like to encourage the robotics community to promote the use of RPN by students and hobbyists. Access is free, and there is no need of registration, when access is provided by other identification platforms (Gmail, Facebook). New courses are being prepared, and prospective teachers are welcome to propose new ideas. All the materials in the site are open and can be re-used for non-commercial purposes.

Existing ROS-based online robots are specially welcome to link to the network, or to build similar sites in their own facilities. The source code of the project is open and available. Non-ROS facilities could explore the possibility of adding a ROS layer for opening new possibilities of user access.

Future versions of the RPN platform will be migrated to the cloud, for serving a large number of users. We dream of connecting to any number of ROS systems running on different laboratories all around the world, through Internet. Such a distributed system would ensure robustness and availability, since it would not depend on a centralized system. We would be very happy to provide guidance and support for expanding such a network and creating a worldwide infrastructure for learning robotics anywhere, at anytime.

Acknowledgements The authors thank Miroslav Kulich and Libor Přeucil for their help in the integration with Syrotek. Support of IEEE RAS through the CEMRA program (Creation of Educational Material for Robotics and Automation) is gratefully acknowledged. This paper describes research done at the Robotic Intelligence Laboratory. Support for this laboratory is provided in part by Ministerio de Economía y Competitividad (DPI2011-27846), by Generalitat Valenciana (PROMETEOII/2014/028) and by Universitat Jaume I (P1-1B2011-54).

References

1. Alemany, J., Cervera, E.: Design of high quality, efficient simulation environments for USARSim. In: Proceedings of the IASTED International Conference on Robotics, pp. 226–233 (2011)
2. Alemany, J., Cervera, E.: Appealing robots as a means to increase enrollment rates: a case study. In: Proceedings of the 3rd International Conference on Robotics in Education, pp. 15–19 (2012)
3. Alexander, B., Hsiao, K., Jenkins, C., Suay, B., Toris, R.: Robot Web Tools [ROS topics]. *Robotics & Automation Magazine, IEEE* **19**(4), 20–23 (2012)
4. Alier, M., Casañ, M.J., Piguillem, J.: Moodle 2.0: Shifting from a learning toolkit to an open learning platform. In: *Technology Enhanced Learning. Quality of Teaching and Educational Reform*, pp. 1–10. Springer (2010)
5. Bergin, J., Lister, R., Owens, B.B., McNally, M.: The first programming course: ideas to end the enrollment decline. *ACM SIGCSE Bulletin* **38**(3), 301–302 (2006)
6. Bonsignorio, F., Hallam, J., del Pobil, A.: Defining the requisites of a replicable robotics experiment. In: *RSS2009 Workshop on Good Experimental Methodologies in Robotics* (2009)
7. Buckhaults, C.: Increasing computer science participation in the first robotics competition with robot simulation. In: *Proceedings of the 47th Annual Southeast Regional Conference*, p. 19. ACM (2009)

8. Caeiro-Rodríguez, M., Manso-Vázquez, M., Anido-Rifón, L.: Design of flexible and open learning management systems using IMS specifications. the Game Tel experience. *Journal of Research and Practice in Information Technology* **44**(2), 151 (2012)
9. Casini, M., Chinello, F., Prattichizzo, D., Vicino, A.: Ract: A remote lab for robotics experiments. In: *Proceedings of the 17th IFAC World Congress*. Seoul (Korea) (2008)
10. Chen, B., Xu, Z.: A framework for browser-based multiplayer online games using webgl and websocket. In: *Multimedia Technology (ICMT), 2011 International Conference on*, pp. 471–474. IEEE (2011)
11. Comport, A.I., Marchand, E., Pressigout, M., Chaumette, F.: Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *Visualization and Computer Graphics, IEEE Transactions on* **12**(4), 615–628 (2006)
12. Dagdilelis, V., Sartatzemi, M., Kagani, K.: 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*, pp. 757–761 (2005)
13. Djalic, V., Maric, P., Kosic, D., Samuelsen, D., Thyberg, B., Graven, O.: Remote laboratory for robotics and automation as a tool for remote access to learning content. In: *Interactive Collaborative Learning (ICL), 15th International Conference on*, pp. 1–3 (2012). DOI 10.1109/ICL.2012.6402174
14. Djenic, S., Krneta, R., Mitic, J.: Blended learning of programming in the internet age. *Education, IEEE Transactions on* **54**(2), 247–254 (2011)
15. Dougiamas, M., Taylor, P.: Moodle: Using learning communities to create an open source course management system. In: *World conference on educational multimedia, hypermedia and telecommunications*, vol. 2003, pp. 171–178 (2003)
16. Edwards, S., Lewis, C.: Ros-industrial—applying the robot operating system (ros) to industrial applications. In: *IEEE Int. Conference on Robotics and Automation, ECHORD Workshop* (2012)
17. Esteller-Curto, R., Cervera, E., Del Pobil, A.P., Marin, R.: 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*, pp. 708–710 (2012)
18. Esteller-Curto, R., Del Pobil, A.P., Cervera, E., Marin, R.: 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*, pp. 864–867 (2012)
19. Esteves, M., Fonseca, B., Morgado, L., Martins, P.: Improving teaching and learning of computer programming through the use of the second life virtual world. *British Journal of Educational Technology* **42**(4), 624–637 (2011)
20. Furler, L., Malik, A.S., Meriaudeau, F., Nagrath, V.: An auto-operated telepresence system for the NAO humanoid robot. In: *Communication Systems and Network Technologies (CSNT), International Conference on*, pp. 262–267 (2013)
21. Gage, A., Murphy, R.R.: Principles and experiences in using legos to teach behavioral robotics. In: *Frontiers in Education, 2003. FIE 2003 33rd Annual*, vol. 2, pp. F4E–23. IEEE (2003)
22. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. In: *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 769–774 (2009)
23. Jara, C.A., Candelas, F.A., Puente, S.T., Torres, F.: Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. *Computers & Education* **57**(4), 2451–2461 (2011)
24. Kulich, M., Chudoba, J., Kosnar, K., Krajnik, T., Faigl, J., Preucil, L.: SyRoTek – distance teaching of mobile robotics. *Education, IEEE Transactions on* **56**(1), 18–23 (2013). DOI 10.1109/TE.2012.2224867
25. Kulich, M., Chudoba, J., Kosnar, K., Krajnik, T., Faigl, J., Preucil, L.: Syrotekdistance teaching of mobile robotics. *Education, IEEE Transactions on* **56**(1), 18–23 (2013)
26. Lawhead, P.B., Duncan, M.E., Bland, C.G., Goldweber, M., Schep, M., Barnes, D.J., Hollingsworth, R.G.: A road map for teaching introductory programming using lego® mindstorms robots. *ACM SIGCSE Bulletin* **35**(2), 191–201 (2003)
27. Lester, B.: Robots’ allure: can it remedy what ails computer science? *Science (New York, NY)* **318**(5853), 1086–1087 (2007)

28. Marín, R., Sanz, P.J., Del Pobil, A.P.: The UJI online robot: An education and training experience. *Autonomous Robots* **15**(3), 283–297 (2003)
29. Matijevics, I.: Local and remote laboratories in the education of robot architectures. In: *Intelligent Engineering Systems and Computational Cybernetics*, pp. 27–36. Springer (2009)
30. Moodle.org: Moodle statistics. <http://moodle.net/stats/>. Accessed: 2014-09-18
31. Murphy, R.R.: competing for a robotics education. *Robotics & Automation Magazine, IEEE* **8**(2), 44–55 (2001)
32. Orduna, P., Rodriguez-Gil, L., Lopez-de Ipina, D., Garcia-Zubia, J.: Sharing the remote laboratories among different institutions: A practical case. In: *Remote Engineering and Virtual Instrumentation (REV)*, 9th International Conference on, pp. 1–4 (2012). DOI 10.1109/REV.2012.6293178
33. Osentoski, S., Jay, G., Crick, C., Pitzer, B., DuHadway, C., Jenkins, O.C.: Robots as web services: Reproducible experimentation and application development using rosjs. In: *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 6078–6083 (2011)
34. Osentoski, S., Pitzer, B., Crick, C., Jay, G., Dong, S., Grollman, D., Suay, H.B., Jenkins, O.C.: Remote robotic laboratories for learning from demonstration. *International Journal of Social Robotics* **4**(4), 449–461 (2012)
35. Pavelich, M.J., Moore, W.: Measuring maturing rates of engineering students using the perry model. In: *Frontiers in Education Conference, 1993. Twenty-Third Annual Conference. 'Engineering Education: Renewing America's Technology', Proceedings.*, pp. 451–455. IEEE (1993)
36. Pitzer, B., Osentoski, S., Jay, G., Crick, C., Jenkins, O.C.: PR2 Remote Lab: An environment for remote development and experimentation. In: *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 3200–3205 (2012)
37. Pritchard, D., Vasiga, T.: Cs circles: an in-browser python course for beginners. In: *Proceeding of the 44th ACM technical symposium on Computer science education*, pp. 591–596. ACM (2013)
38. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: An open-source robot operating system. In: *ICRA Workshop on Open Source Software*, vol. 3 (2009)
39. Ratcliff, C.C., Anderson, S.E.: Reviving the turtle: Exploring the use of logo with students with mild disabilities. *Computers in the Schools* **28**(3), 241–255 (2011)
40. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programming for all. *Communications of the ACM* **52**(11), 60–67 (2009)
41. Santana, I., Ferre, M., Izaguirre, E., Aracil, R., Hernandez, L.: Remote laboratories for education and research purposes in automatic control systems. *Industrial Informatics, IEEE Transactions on* **9**(1), 547–556 (2013). DOI 10.1109/TII.2011.2182518
42. Schilling, K., Roth, H., Rösch, O.J.: Mobile mini-robots for engineering education. *Global J. of Engng. Educ* **6**(1), 79–84 (2002)
43. Tadokoro, S., Kitano, H., Takahashi, T., Noda, I., Matsubara, H., Shinjoh, A., Koto, T., Takeuchi, K., Matsuno, F., Hatayama, M., et al.: The robocup-rescue project: A robotic approach to the disaster mitigation problem. In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4, pp. 4089–4094. IEEE (2000)
44. Tillmann, N., Moskal, M., de Halleux, J., Fahndrich, M., Bishop, J., Samuel, A., Xie, T.: The future of teaching programming is on mobile devices. In: *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pp. 156–161. ACM (2012)
45. Trevelyan, J.: Lessons learned from 10 years experience with remote laboratories. In: *Engineering Education and Research (iNEER), International Conference on*, pp. 1562–1580 (2004)
46. Uludag, S., Karakus, M., Turner, S.W.: Implementing it0/cs0 with scratch, app inventor forandroid, and lego mindstorms. In: *Proceedings of the 2011 conference on Information technology education*, pp. 183–190. ACM (2011)

-
47. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: RoboEarth. *Robotics Automation Magazine, IEEE* **18**(2), 69–82 (2011). DOI 10.1109/MRA.2011.941632
 48. Wang, E.: Teaching freshmen design, creativity and programming with legos and lab-view. In: *Frontiers in Education Conference, 2001. 31st Annual*, vol. 3, pp. F3G–11. IEEE (2001)