Research Article

The Experience of the Robot Programming Network Initiative

Gustavo A. Casañ and Enric Cervera

Jaume I University, Castelló, Spain

Correspondence should be addressed to Gustavo A. Casañ; ncasan@uji.es

Received 7 December 2017; Revised 26 March 2018; Accepted 10 May 2018; Published 7 June 2018

Academic Editor: Gordon R. Pennock

Copyright © 2018 Gustavo A. Casañ and Enric Cervera. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Since its creation in 2014, the Robot Programming Network (RPN) has been an environment to learn and teach robotics to the general public, which has also allowed us to learn about education. In this paper, we aim to not only present the RPN and the active courses in the system but also show the evolution of this initiative in front of the changing e-learning environment and its possible future evolution towards Massive Online Open Courses (MOOCs) and cloud simulation.

1. Introduction

Teaching robotics and programming has nearly since the beginning employed some type of enquiry-based learning [1, 2], as this form of active learning [3] starts by posing questions, problems, or scenarios that the student has to think about or solve (in our case creating a robot or program to control the robot). In engineering, this approach is considered preferable than simply presenting established facts in a lecture mode or portraying a smooth path to knowledge. The main pedagogy employed is problem-based learning (PBL) [4], a student-centered pedagogy in which students learn about a subject through the experience of solving an open-ended problem found in trigger material. In the context of RPN, the learning experience is the creation of a program.

Using robotics for education has a long history [5–7], as they have been proven to be good tools for learning mechanics, electronics, and programming [8–13], among other technological disciplines. But, also from the beginning, most learning centres could not afford the price of a robot. Luckily, robots have been decreasing in price and increasing in capabilities in the last years (like the popular e-puck robot [14]) at the same time that robotic simulations have improved their usability and price.

One solution to the price problem has been sharing the hardware components using remote laboratories [15] and online robotic systems [16, 17]. Remote laboratories have been widely used for teaching engineering and the sciences [18–20]. With the advent of cross-platform middleware [21] and the adoption of new powerful World Wide Web standards [22], we may well be approaching a new golden era for web-based laboratories [23, 24], as sophisticated intelligent robotic platforms could be made accessible worldwide, with the only cost being an Internet connection for the user. Of course remote laboratories are not limited to robotics. Platforms like VISIR and VISIR+ are ongoing [25] and aim to spread the use of the tools developed to the general education public. Another interesting initiative is the inclusion of Virtual Reality technologies to the laboratories, making the systems cheaper than physic ones and more versatile. One example is presented in [26].

Nowadays, there are a myriad of web-enabled intelligent systems, ready to be remotely controlled, with their sensors and outputs visualized. An awesome example is the PR2 Remote Lab [27], which enables a large community of researchers to use a state-of-the-art yet expensive platform. But the work is orienting itself to create standards for online learning systems. For example, GOLC/IAOE (http://online-engineering.org/index.php) as part of the IEEE project P1876-Networked Smart Learning Objects for Online Laboratories (http://sites.ieee.org/sagroups-edusc/) is creating a standard that defines methods for storing and retrieving learning objects for remote laboratories. The standard will also define methods for linking learning objects
to design and implement smart learning environments for remote online laboratories. The European Union is funding several projects to improve the quality of online learning, like ENVISAGE (http://www.envisage-h2020.eu/abstract/), which has the objective of offering a solution towards optimizing the learning process in virtual labs and therefore maximizing their impact in education, or e-LIVES (https://e-lives.eu/), which aims to develop an e-learning implementation strategy. Another project is Next Lab (http://nextlab.golabz.eu/), which continues the project Go-Lab, which groups remote and virtual laboratories for different science domains (http://go-lab-project.eu/) and has also originated a private company (https://labsland.com/).

However, to our knowledge, most systems are built ad hoc with their customized solutions for management and development. The lack of a unique standardized remote laboratory framework and the dilemma between offering capabilities and maintaining security prevent 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, only scripting capabilities for executing a limited set of commands are provided to the users [28]. Products like the remote lab created by WebLab-Deusto (http://weblab.deusto.es/website/) have more general purpose and are not only oriented to robotics.

In this paper, we explore RPN (Cervera, 2015), a system that allows users of a Virtual Learning Environment to seamlessly work with web-based laboratories. We also explain how RPN has had different courses available during the last years and how it has evolved in orientation and characteristics, as RPN is not a closed system but is an evolving one.

First, we are going to explain some of the technical characteristics of RPN and its learning management system. After that, we will be giving a general description of the different courses, our collaborators in them, and the answer we obtained from the students and teachers.

Finally, we will be giving some conclusions and possible future work.

2. Robot Programming Network

RPN is an initiative to bring existing remote robot laboratories to a new dimension by adding the flexibility and power of writing ROS code (Robot Operating System, http://wiki.ros.org, [21]) in an Internet browser and running it in the remote robot with a single click. We aim to reduce the complexity of the interaction with the system not directly related to the human-robot interaction.

2.1. Hardware and Software Architecture. From the moment the student types the password until the robot moves, he or she has to follow a clear list of steps:

(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.

(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.

The overall architecture of RPN is shown in Figure 1. It is built upon two networks: the Internet (or a local academic network) for the students to access and a local ROS network that connects the robot systems (either simulators or real robots) and other devices like video cameras.

ROS 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 behaviours that can be readily used for providing high-level functionality to the user or running in the background for monitoring, data logging or, security purposes. ROS is an ongoing project with new characteristics included each day.

There is a bridge between both networks, consisting of a module that translates the information between two different languages: ROS topics and services on the robot side and web data structures on the student side. This module, called rosbridge [29], can both read ROS topics and publish them through the web (using its WebSocket transport layer [30], in the same way as used in the PR2 Remote Lab) and write ROS topics with information provided by the web clients [22].

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 [29]. RPN can also be integrated with RMS (Robot Management System, http://www.ros.org/wiki/rms/), a remote lab management tool designed to control ROS-enabled robots from the web.
Figure 1: 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.

Communication between client and server is implemented with the actionlib ROS package (http://www.ros.org/wiki/actionlib). First, the RPN server checks the goal: if a bag is required, a rosbag record process 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 (http://www.ros.org/wiki/rosbash#rosrun) tool, with its standard output redirected to the RPN server, which displays it in the browser window (bottom right subwindow in Figure 5).

Although the ROS network is accessible at Internet through rosbridge, access must be authenticated and authorized by the system, centralized in a Learning Management Server (LMS). The user must first sign in with a recognized user account or log into the system with an identification provided by another web service (e.g., Gmail or Facebook).

The Learning Management System (LMS) consists of the Moodle software package (www.moodle.org) [31] for the administration, documentation, tracking, reporting, and delivery of robotics courses. This package facilitates the creation of tests and the evaluation of students. Of course, different modules can be added to the system as needed, like one included to be able to use Mozilla OpenBadges (https://support.mozilla.org/es/products/open-badges).

The code (Python, Ruby, Lua, Matlab, and Lisp can be supported) is executed in the robot server at full speed, that is, without any communication delay, and the output of the process is returned back. Built upon Robot Web Tools [22, 29], RPN works out-of-the-box in any ROS-based robot or simulator.

RPN consists of a simple scripting interface, with a text box and submit button built upon Robot Web Tools, which can be accessed as a Moodle LTI (Learning Tools Interoperability http://www.imsglobal.org/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 (or any other ROS-supported scripting language), and it runs on the server side.

As a result, RPN has these fundamental differences from most systems:

1. The script is executed as a true ROS node on the server, with access to any topic or service.
Figure 2: Block diagram of the system.

(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.

Consequently, RPN does not rely on any particular detail of the underlying hardware or software, and it can cope with any ROS-enabled system. Also, each student has its own Virtual Machine (VM) and thus the code does not execute in the real machines, reducing risks and increasing independence.

The client side of RPN consists of an HTML5, JavaScript-enabled web browser, which runs the JavaScript widgets (in Figure 5). The program source code is typed in a user-friendly, syntax-highlighting, embedded editor (http://codemirror .net/, right subwindow in Figure 5).

Figure 2 depicts a block diagram of the system. RPN server side is built upon Robot Web Tools for communicating with the clients. In addition, it also communicates directly with ROS for dynamically starting new processes, that is, executing the client programs. It is also responsible for launching rosbag (http://www.ros.org/wiki/rosbag) for data logging. Communication is exclusively performed through ROS.

There is no need to modify the Moodle platform for running our system, since support for LTI-compliant materials is already included [32], 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, for example, sending grades to Moodle assignment based on the performance of the robot task.

2.2. Security. Security policies must be established, as in other web laboratories [33, 34]: the LMS server is also responsible for 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. Thus, only registered users have full access to the system, while others can be monitoring or analyzing the system in a read-only mode. A completely open-access system is not supported, although projects like Go-Lab aim at this type of learning systems.

As previously explained, the student’s code is not executed directly in the real machines, but instead it runs on a VirtualBox (https://www.virtualbox.org/) 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 that 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, for example, danger of collision. Figure 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 the robot driver. This monitoring process is transparent to the user by using the dynamic remapping capabilities of ROS.
2.3. Scalability. According to the statistics published in the Moodle home page (http://moodle.net/stats), the largest sites in the world currently have up to 2,000,000 users. So the scalability of Moodle is not a problem at all, provided that the appropriate hardware (processing power and bandwidth) is available. Our current system is experimental; thus it runs on a single computer. When the number of users 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, the creation of a distributed network community of robots is technically possible. In this way, the workload could be distributed among online robots on different remote laboratories. Of course, the administrative and institutional issues cannot be ignored. Any type of sharing of resources among different institutions has to be carefully organized.

2.4. User Interface. The generic user interface is intentionally kept very simple for clarity and ease of use (Figure 4). It consists of four window areas: the top left side is the visualization area, where the system displays the simulated setup, or video 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 Figure 6, the left subwindow is the image from a camera (the robot). In the right subwindow, we can see the robot through the use of external video cameras in the environment we created.

A click on the run button is enough to launch the processing of the code and trigger the whole interaction process between client and server.

3. Courses and Activities

First, we are going to introduce some initial design decisions common to all the courses, and then we will talk briefly about each course, as they have been explained before.

3.1. Initial Decisions. Our own interest was in developing teaching tools for young and inexpert students, so we decided to begin the experience with generalist and basic courses. At the same time, given our international vocation and the expected age of most students, we decided to make the courses available in several languages (Spanish, Catalonia, Arabic, and English), with English being the main one.

Given the complexity of the field and the myriad of possibilities we had, we decided to limit them. First of all, we believed that Python (https://www.python.org/) was the most adequate language to the task of teaching programming and at the same time controlling robots without the unnecessary complications of other languages.

As previously indicated, from the complex field of enquiry-based learning, we decided to employ PBL. PBL is a student-centered pedagogy in which students learn about a subject through the experience of solving an open-ended problem found in trigger material. The PBL process does not focus on problem solving with a defined solution, but it allows for the development of other desirable skills and attributes. This includes knowledge acquisition, enhanced group collaboration, and communication. The process allows learners to develop skills used for their future practice. It enhances critical appraisal and literature retrieval and encourages ongoing learning in a team environment. The problems the students have to complete are programming tasks. The role of mediator is taken by the system and the teachers who answer questions.
Although the PBL tutorial process involves working in small groups of learners, in our system, the number of group activities is reduced, as it allows the courses to be permanently open to incorporation and each student works to his/her own rhythm.

The Many Levels of Inquiry [35] clearly outlines four levels of inquiry. Our methods can be considered to be in levels 1 (Confirmation Inquiry, in which the teacher has taught a particular topic and then develops questions and a procedure that guides students through an activity where the results are already known) and 3 (Guided Inquiry, in which the teacher provides only the research question for the students who are responsible for designing and following their own procedures).

Also, it is important to remark that we continue employing classical tests (mainly multiple selection tests) not only to evaluate the students but also to consolidate the knowledge they acquire.

Although most of the theory material was presented in written form, we added videos showing examples of robot behaviour to facilitate understanding and increase interest in the lessons.
Table 1: General characteristics of the courses. The column "Students" represents the total number of students inscribed in the course from the start of the course to 2017.

<table>
<thead>
<tr>
<th>Course</th>
<th>Programming language</th>
<th>Simulator</th>
<th>Real robot</th>
<th>Students</th>
<th>Level</th>
<th>Available</th>
<th>Organized course</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Turtle Robot</td>
<td>Python</td>
<td>2D</td>
<td>No</td>
<td>200</td>
<td>Beginner</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Introduction to Mobile Robots</td>
<td>Python</td>
<td>2D</td>
<td>No</td>
<td>60</td>
<td>Beginner</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mobile Robots 3D with Webots</td>
<td>Blockly</td>
<td>3D</td>
<td>No</td>
<td>20</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Introduction to Humanoid Robots</td>
<td>Python</td>
<td>-</td>
<td>Yes</td>
<td>10</td>
<td>Advanced</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Mobile Robots with the Construct</td>
<td>Python</td>
<td>3D</td>
<td>No</td>
<td>30</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

To make the courses more interesting for the students, we decided to use gamification techniques [36] that have proven an improved answer by the students and use as complex as possible simulation environments.

A progressive method was used to present the materials, from easier to more complex, with different tasks to solve at each point, always trying to use a discovery approach. In general, we had three different sections in each course: (1) movement of the robot, (2) use of the robot’s sensors, and (3) integration of movement and sensors to solve problems.

We also decided that it was important to be present in the media, looking for collaborators and students but also becoming a source of robotics information for the general public. Thus, we created a simple Facebook page in which we try to inform about the activity of the RPN.

Finally it is important to remark the security of the system. Each student has to identify himself or herself with some type of email account (although we facilitate the procedure through the use of google and Facebook accounts) before entering into the system. This creates a problem, as in some countries (like Spain) younger people are not supposed to have their own email account. When necessary, we created accounts manually in the systems for such students.

3.2. Courses. There have been five courses: the Turtle Robot, Introduction to Mobile Robots, Mobile Robots 3D, Introduction to Humanoid Robots, and Mobile Robots with the Construct. All of them are part of the RPN system and employ enquiry-based learning and include different degrees of gamification, from badges to competitions, but there are important differences among them.

In Table 1, we can see some of the general characteristics of the courses.

3.2.1. The Turtle Robot. This is a simple 2D simulator [37, 38] without physics, initially designed for teaching ROS concepts, but it is 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: the velocity of the turtle can be controlled; thus the execution of the code is not immediate but progressive.

In the right window of Figure 7, the visualization area shows the turtle. 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 colour of the path is selectable; thus colourful patterns can be drawn. When the user calls an API function, like leftArc(a, r) which moves the turtle during one second along an arc trajectory of a degree and radius r, internally, the function computes the linear and angular velocities and publishes them into the corresponding topics for moving the turtle.
This course is targeted at 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 typical documentation page is shown in Figure 7. 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 and to submit the solution for the teacher to review.

The course includes a competition challenge [40] in order to increase the motivation of the students, where students are asked to program the robot to describe a trajectory in a circuit in the fastest time without going off the path. A Hall of Fame with the best times is kept in the course, as depicted in Figure 8, which shows the result of the pilot experiment.

This course has changed progressively, not only adding a final evaluation test (multioption) and gamificating it more deeply, with the use of badges (Mozilla OpenBadges, as can be seen in Figure 9), but also including explanatory videos for an easier understanding of the materials and increased attractiveness to the potential students.

3.2.2. Introduction to Mobile Robots. The second course [38, 41] was developed surrounding a more powerful and realistic simulator for mobile robots, Stage [42], but it shares a similar organization and aspect with the Turtle Robot. The Stage simulator is readily available in ROS and it has been integrated in our framework. This simulator adds sensors and the possibility to control directly the engines of the robot (which requires more mathematical knowledge). Figure 5 depicts the user interface for this simulator, which is clearly similar, with buttons for optional displaying of the robot’s trail path and the range sensors. Also, some videos were added to the course to reinforce the learning experience (https://www.youtube.com/watch?v=zzTkerwN2Cg&feature=youtu.be).

This course also uses tasks with increasing levels of difficulty and a final test, with a badge reward, but it is for older students though (at least high-school level). The feedback from the students has been positive, yet most of them consider that the final task (a wall-following problem) is very difficult.

3.2.3. Mobile Robots 3D. This course [43] tries to simplify the programming part of a robotics course to the maximum through the use of Blockly (https://developers.google.com/blockly/).

Blockly is an open source project from Google to create a visual programming language (in which we manipulate the elements of the program graphically following a spatial grammar and not textually). It consists in a JavaScript library to create visual block editors that work in a web browser. It uses blocks that link together and can generate JavaScript, Python, or Dart code and allowed us to create our own specific blocks to control the robot. Blockly has been previously used successfully with educational purposes (https://blockly-games.appspot.com/) or in Hour of Code (https://hourofcode.com/es).
At the same time, we decided to use a 3D simulator, Webots [44], which provided us with a more realistic environment. Thanks are due to Cyberbotics (http://www.cyberbotics.com/about), the company that was generous to gift us with a free unlimited license of Webots that we installed in one of our servers.

The programming environment is thus deeply changed from previous courses, as can be seen in Figure 10.

Webots provided us with a unique simple robot with two motors that allow it to move through a plane. Also, the robot has seven distance sensors that inform it of its surroundings. As the simulator is realistic, the robot can fall down from the platform, it takes time to move, and the sensors have limitations.

Following gamification techniques, the course is organized in five worlds (task) of increasing complexity in which...
the user has to learn to use new blocks (instructions) related to characteristics of the robot. The course includes a competition with oneself, a race against the clock with the five worlds. When the student finishes the course, he/she receives the corresponding badge.

The first final satisfaction quiz completed by students motivated us to make one important change: we removed the final test, as it was felt to be strange in the gamificated environment of this course. Also, some young students (7–11 years old) followed the course without problems until arriving to the last world, which they found too complex. We improved the help system and added some videos (https://www.youtube.com/watch?v=kONEzC5HnY&feature=youtu.be), which seem to have eliminated the problem, but it is a known problem of enquiry-based learning [19] that it should be reinforced with the correct use of scaffolding (like more videos and manuals) to reduce the cognitive load of the user.

3.2.4. Introduction to Humanoid Robots/HUMABOT 2014. As part of the congress Humanoids 2014, there was a challenge, HUMABOT 2014 (http://www.irs.ujj.es/humabot/), in which an artificial environment, a kitchen (as can be seen in Figure 6, right subwindow, using the cameras added to the environment), was created. In this kitchen, the robot, the NAO humanoid robot [45], had different tasks to solve, like locating the tea box or moving the tomato to the pot. We have already seen how we can create a window in the system user interface (in Figure 6, left subwindow) in which we can see the NAO’s camera view.

As a help for the participants in the challenge, we created a copy of the challenge and connected it to the Internet through our system. Communication with the robot is possible because the ROS NAO driver (http://www.ros.org/wiki/Robots/Nao/) connects to a custom module running the robot middleware (NAOqi), which has been developed for USARSim [46]. They could connect to the course in RPN and prove their programs in the real environment without the need to create their own.

The development for the challenge was reused to create a course in humanoid robotics [47], but we also decided to include a simulator as a complement and alternative. Although there are a lot of possibilities for humanoid robot simulators available in ROS, like REEM-C, HUBO, TULip, or PR2, we selected NAO as both the simulator and the real robot are available. The students begin to program using the simulator (which they could download and use in their own computer) and when they have passed several tasks, they begin to use the real robot.

An example of the simulator can be seen in Figure 3, where the NAO robot is standing in a room (top left we can see the image from the robot camera).

There are two problems that we would like to solve:

1. Several users wanted to use the NAO at the same time, which is evidently impossible. A simple reservation system helped to improve users’ coordination.

2. The NAO simulator available in ROS is not reliable. It crashed after working continuously for several hours and sometimes it did not remove itself completely from the system. As we are not the creators of the simulator, we decided not to enter into that problem. Modifying the Virtual Machine to clean better after itself solved part of the problem.

As usual, in the course, we use innovative gamification techniques to increase the students’ interest: there are tasks (problems) that the students have to solve and completing these tasks grants them badges and a final Mozilla Open-Badges when they pass the final test and finish successfully the course to show their accomplishments.

Several months after HUMABOT, we finally removed it from public view. The simulator presented problems and we could not maintain the kitchen environment and a NAO robot dedicated permanently to the project in our installation. We have been studying alternatives as changing to another (cheaper) humanoid robot, like Bioloid, to be able to have several robots available at the same time, or changing simulators, maybe jumping to cloud through the Construct, but there has not been interest by the general public.

3.2.5. Mobile Robots with the Construct. After taking part in the organization of the summer school RASPEBRAS 2015 (Summer School on Experimental Methodology, Performance Evaluation and Benchmarking in Robotics http://www.ieee-raspebras2015.org/) and creating a bench mark [48] with the Construct (http://sw.theconstructsim.com/), a company dedicated to provide robotic simulations in the cloud, we arrived to an agreement to use their system for our courses.

As the integration between RPN and the Construct is not complete, the student has to have two different accounts, one in our system, which has the instructions and theory needed to complete the tasks, and another in the Construct (the free account offers up to 10 hours of simulation). The recommended work way is with two different tags in a web browser.

Thus, we created a version of our Mobile Robots course using their system and two different simulators, Gazebo (http://gazebosim.org/) [49] and Webots. The robots used were a Pioneer model (Figure 11) and a Kobuki.

The complete course [41] could be done in one of them, but adding a different choice gives the students the opportunity to experiment how their knowledge can be transferred from one robot to another.

The programming language continues being Python, but the Construct includes Jupyter Notebook ([50]http://jupyter.org/, in Figure 12), which provides a powerful but simple environment for programming [51].

3.3. Evaluation. The evaluation of each course was made through the use of a final questionnaire (we can see an example in Table 2) to the students and the results have already been published in previous papers like [37, 38, 41, 43, 52]. The number of students in the individual experiences was limited from 10 to 40.

In general, the answers to the questions related to the satisfaction degree, the learning process, and quality of the virtual robot were very positive answers (about 90%, as can
be seen in Figure 13 for the first experiences with the Turtle Robot course [37]). Partial results of a different experience with high-school students can be seen in Figure 14 (taken from [52]). It is clearly shown how the general satisfaction decreased to 60%. This was caused by different problems with the simulators and the connections in different courses and at different moments (how is explained in [52]).

It is also interesting to note that the students did not seem to perceive the gamification techniques (badges, points, increasing difficult tasks, forums, etc.) as something especially interesting or engaging, except in the case of the competitions with public results, in which some of them entered into a very competitive stage, with repeated tries to improve their results. More exhaustive testing is needed to get a definitive reason for this behaviour, but it seems as if we need more collaborative tasks to increase communication among the students and their competitiveness.

The number of students in the rest of the courses was very reduced, so results and conclusions cannot be considered significant, but they tend to be similar to those obtained with the Turtle Robot course: interesting but connections and simulators were not always reliable.

4. Conclusions and Future Work

In this paper, we have shown that RPN is a mature tool that has already shown its capabilities to help in teaching basic programming skills to students, although its development is ongoing, and it must yet be tested for larger (hundreds) number of users. A thorough study of system vulnerabilities to malicious code must also be carried out, but the system has been proven to be fairly robust. Although the Robotic Intelligence Laboratory continues being the main supporter of the RPN, two companies,
Table 2: Students’ questionnaire feedback for the Turtle Robot course. The answers are in a scale from one to five: “strongly agree,” “agree,” “neutral,” “disagree,” and “strongly disagree.”

<table>
<thead>
<tr>
<th>Learning compared with traditional methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did the Turtle Robot help you to visualize the theoretical concepts to be learned?</td>
</tr>
<tr>
<td>How would you rate the outcome of your learning using the Turtle Robot if compared with “traditional methods”?</td>
</tr>
<tr>
<td>Did the Turtle Robot enhance your ability to understand the theoretical concepts about programming in a new way?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ease of use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did you find easy the use of the Turtle Robot?</td>
</tr>
<tr>
<td>Did you think that the course was well structured and organized?</td>
</tr>
<tr>
<td>Were you able to use the Turtle Robot by following the instructions provided?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quality of the virtual robot</th>
</tr>
</thead>
<tbody>
<tr>
<td>In which grade will you score to the quality of the virtual robot and its simulation?</td>
</tr>
<tr>
<td>In which grade will you score to the quality of the remote connection?</td>
</tr>
<tr>
<td>Was the response time of the remote laboratory suitable?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suitability in learning of relevant concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did the Turtle Robot help you for understanding the concepts of structured programming (conditions, loops) of the lectures?</td>
</tr>
<tr>
<td>In which grade do you think that the Turtle Robot can be used for learning programming?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Satisfaction degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>In general, do you feel satisfied with the practical experiences through the Internet?</td>
</tr>
</tbody>
</table>

![Figure 13: Results of the Turtle Robot course students questionnaire, from [37].](image)

Cyberbotics and the Construct, are actively collaborating with us.

The initiative and the courses have woken up interest among high-school teachers, several of which have tested the courses with the stated intention of using them in the normal development of their classes [52]. Most of them have not finally applied the system, citing lack of familiarity with the subject and time and organization problems.

Some university teachers have also expressed interest, but in general they find the courses too easy for their students.
Given the opportunity to create their own courses (or versions of ours) using our simulators, they have preferred to use their own systems (usually some Moodle version provided by their university) and have a bigger control of the process.

In general, the students are interested in robots and controlling them, but they find programming a robot a challenge.

There are three general types of student, in decreasing numbers:

(1) Curious ones that enter into the courses but do not begin the assignments
(2) Half-way students who after completing several tasks of increasing difficulty leave the course
(3) Completion students who complete all the tasks and do the tests

In some cases, both teachers and students expected a more “traditional” MOOC, with video lectures that they could use directly in class.

The completion users have expressed a wish for more realistic robots, which has moved us from 2D to 3D simulators. At the same time, to compensate for the lack of programming experience, we have chosen simple programming languages and intuitive environments.

We also plan to improve our approach to enquiry-based learning including more reinforcing materials, as manuals and videos, as suggested by other authors, like [39]. This should help students to make the more difficult exercises easier and also finish the courses successfully.

In the latest developments, we have oriented the initiative to use the Construct simulator. The free tools it provides are a big improvement over other systems and free us from the necessity of our own servers.

At the same time, looking for a more attractive presentation and following the users recommendations, we have created a different type of course, a more traditional MOOC called Autonomous Mobile Robots (http://mooc.uji.es/enrol/index.php?id=22) that began in February 2017 and had a moderate success, with more than 100 students. In 2018, the MOOC will have its second edition and we hope it will be still more successful.

Computer science master students at Jaume I University will continue testing the courses and help us to improve them, becoming open access when they reach an acceptable level. At the end of 2017, we are still trying to improve the integration of the Construct system with ours and testing the 2018 MOOC, which will include more collaborative tasks.

Technically, a possible extension of the system 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 (Python), but any scripting language with ROS support could be used. Special attention should be devoted to Matlab, a leading programming language for science and engineering.

We plan to develop further courses about cooperative robots, robot manipulators, or drones.

As part of our educational initiative and aiming to attract students, we participate several times a year in robotics demonstrations oriented to young people and general public like “Conecta con la Ciencia” organized by Jaume I University (http://www.uji.es/perfiles/futurs/base/acciones/grau/connec-ta/&idioma=es) or DESTACA2014 (http://feriadestaca.es/) organized by Cátedra de Innovación Cerámica «Ciutat de Vila-real» and Vila-real City, with Fundación Globalis collaboration. These activities have guaranteed our presence in the local media. And finally we have made a conscious effort to have a presence in the media (through a Facebook page https://www.facebook.com/RobotProgrammingNet-work/ and a YouTube channel https://www.youtube.com/ channel/UeWljxSx52YHj62x AoWlf Eg), as a further method to increase visibility of the project.

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AENUI</td>
<td>Asociación de Enseñantes Universitarios de la Informática</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interfaces</td>
</tr>
<tr>
<td>CEMRA</td>
<td>Creation of Educational Materials for Robotics and Automation</td>
</tr>
<tr>
<td>IEEE-RAS</td>
<td>Institute of Electrical and Electronics Engineers Robotics and Automation Society</td>
</tr>
<tr>
<td>LTI</td>
<td>Learning Tools Interoperability</td>
</tr>
<tr>
<td>MOOC</td>
<td>Massive Online Open Courses</td>
</tr>
<tr>
<td>RASPEBRAS</td>
<td>Robotic and Automation Society Summer School on Experimental Methodology, Performance Evaluation and Benchmarking in Robotics</td>
</tr>
</tbody>
</table>
References


Submit your manuscripts at
www.hindawi.com