Hindawi
Journal of Robotics
Volume 2018, Article ID 2312984, 15 pages
https://doi.org/10.1155/2018/2312984

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

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.

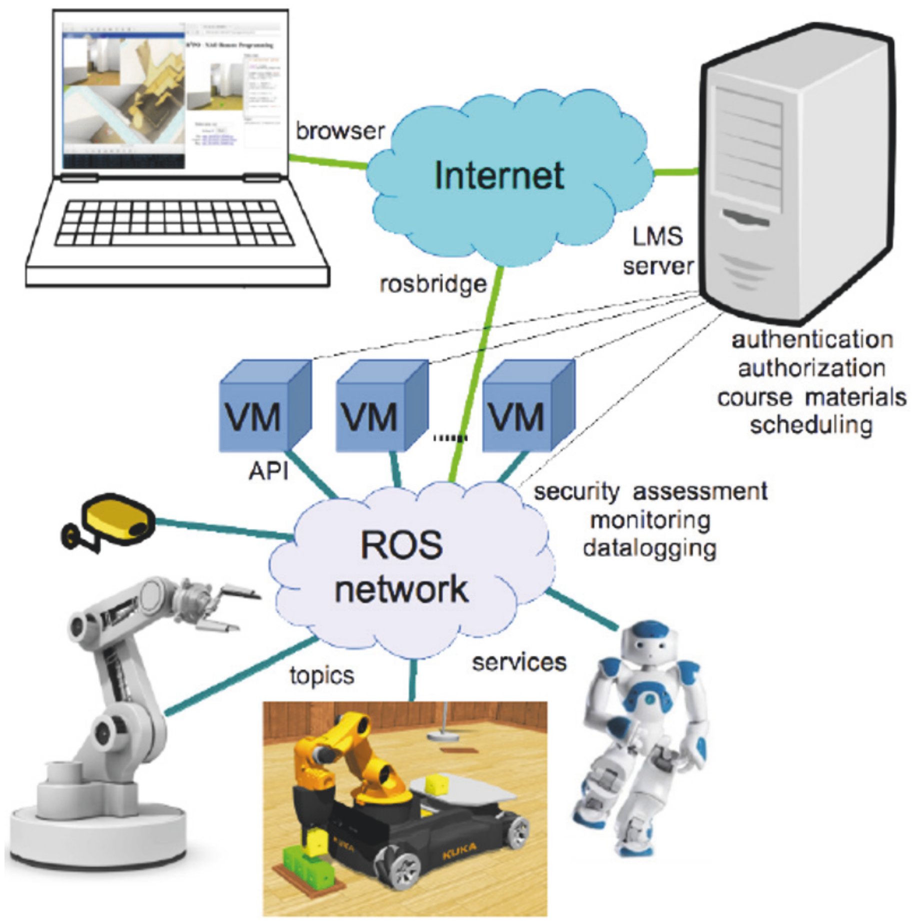


Figure 1: Overall architecture of the Robot Programming Network: the user is connected to Internet via a browser and is granted access to the LMS server. The user's code is run on a Virtual Machine, where it uses a secure API for interacting with the ROS modules of the network, through the available 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.

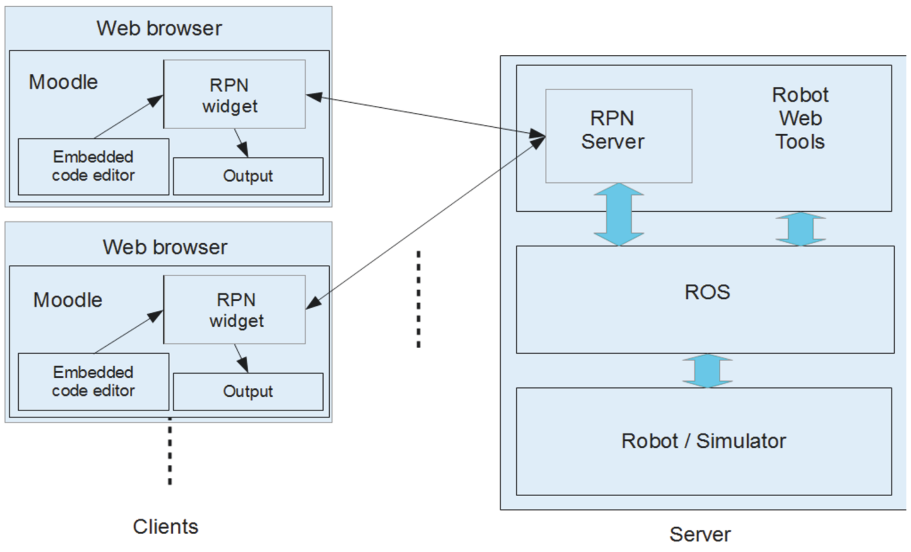


Figure 2: Block diagram of the system.

(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.

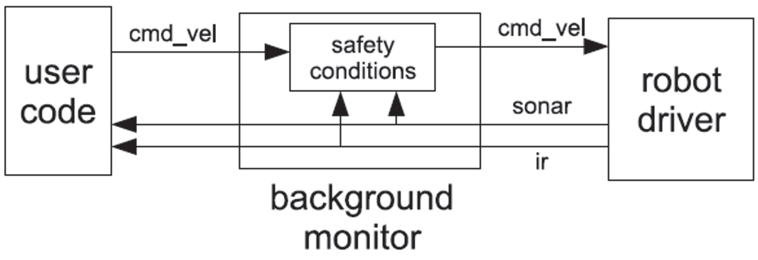


Figure 3: Example of safety monitoring for a mobile robot.

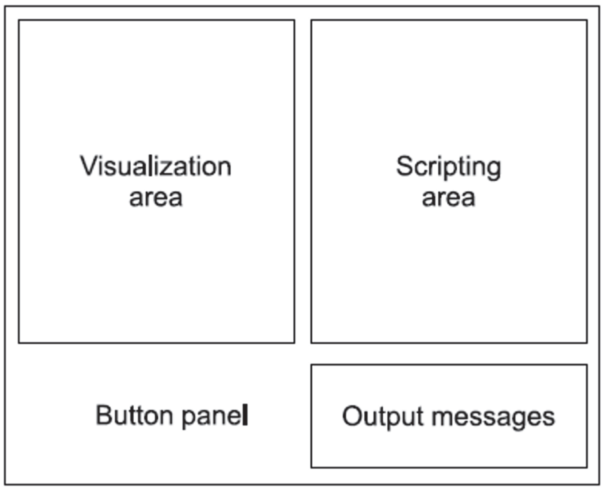


Figure 4: Layout of the user interface.

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.

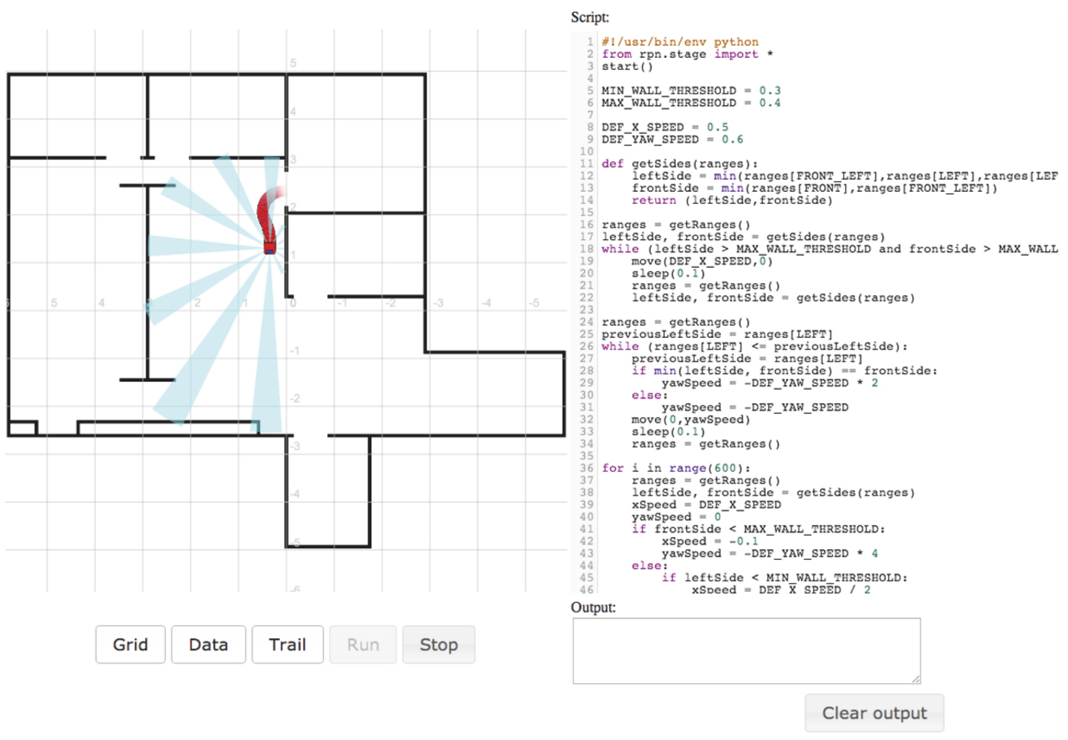
3. Courses and Activities

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, Catalonian, Arabic, and English), with English being the main one.

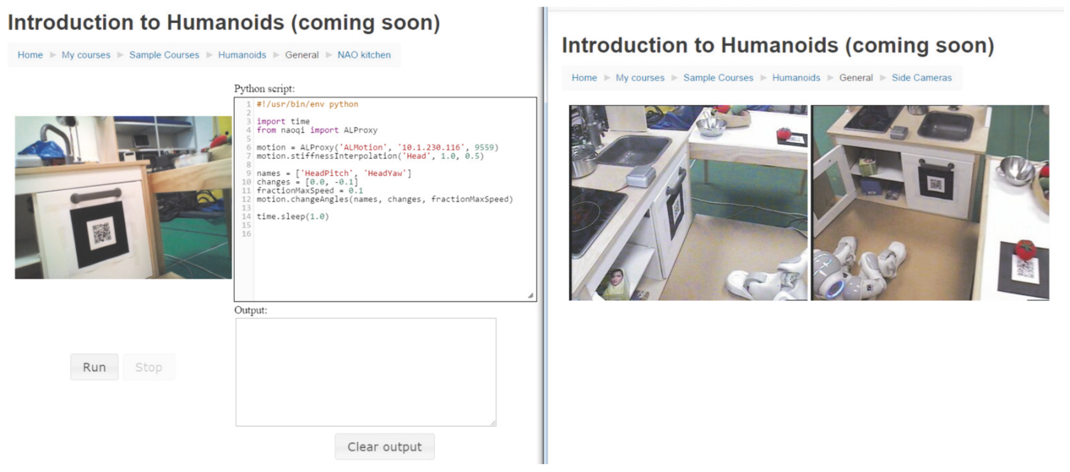
Given the complexity of the field and the myriad of possibilities we had, we decided to limit them. First of all, we believed that Python (https://www.python.org/) was the most adequate language to the task of teaching programming and at the same time controlling robots without the unnecessary complications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



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

MIN_WALL_THRESHOLD = 0.3
MAX_WALL_THRESHOLD = 0.4

DEF_X_SPEED = 0.5
DEF_YAW_SPEED = 0.6

def getSides(ranges):
    leftSide = min(ranges[FRONT_LEFT],ranges[LEFT],ranges[LEF
    frontSide = min(ranges[FRONT],ranges[FRONT_LEFT])
    return (leftSide,frontSide)

ranges = getRanges()
leftSide, frontSide = getSides(ranges)
while (leftSide > MAX_WALL_THRESHOLD and frontSide > MAX_WALL
    move(DEF_X_SPEED,0)
    sleep(0.1)
    ranges = getRanges()
    leftSide, frontSide = getSides(ranges)

ranges = getRanges()
previousLeftSide = ranges[LEFT]
while (ranges[LEFT] <= previousLeftSide):
    previousLeftSide = ranges[LEFT]
    if min(leftSide, frontSide) == frontSide:
        yawSpeed = -DEF_YAW_SPEED * 2
    else:
        yawSpeed = -DEF_YAW_SPEED
    move(0,yawSpeed)
    sleep(0.1)
    ranges = getRanges()

for i in range(600):
    ranges = getRanges()
    leftSide, frontSide = getSides(ranges)
    xSpeed = DEF_X_SPEED
    yawSpeed = 0
    if frontSide < MAX_WALL_THRESHOLD:
        xSpeed = -0.1
        yawSpeed = -DEF_YAW_SPEED * 4
    else:
        if leftSide < MIN_WALL_THRESHOLD:
            xSpeed = DEF_X_SPEED / 2
```

Figure 5: Programming environment of the course "Introduction to Mobile Robots," control buttons and simulation.


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

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

import time
from naoqi import ALProxy

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

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

time.sleep(1.0)
```

Figure 6: Left subwindow, programming environment with a vision through the robot camera. Right subwindow, images of the environment of the robot through two external cameras, from the HUMABOT 2014 challenge.

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.

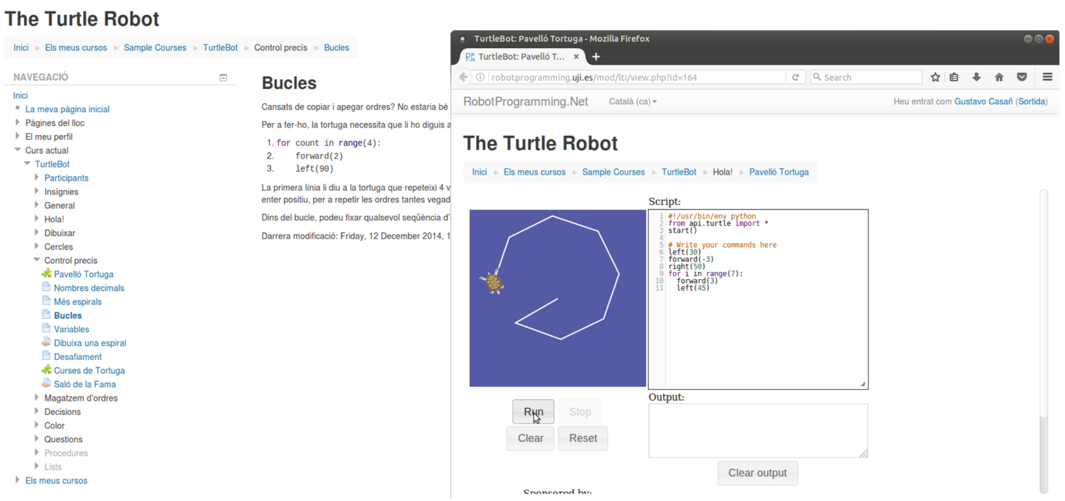
The Many Levels of Inquiry [35] clearly outlines four levels of inquiry. Our methods can be considered to be in levels 1 (Confirmation Inquiry, in which the teacher has taught a particular topic and then develops questions and a procedure that guides students through an activity where the results are 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.

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



Bucles

Cansats de copiar i apegar ordres? No estaria bé

Per a fer-ho, la tortuga necessita que li ho diguis a

```
for count in range(4):
    forward(2)
    left(90)
```

La primera línia li diu a la tortuga que repeteixi 4 v
enter positiu, per a repetir les ordres tantes vegad

Dins del bucle, podeu fixar qualsevol seqüència d'

Darrera modificació: Friday, 12 December 2014, 1

TurtleBot: Pavelló Tortuga - Mozilla Firefox
TurtleBot: Pavelló T...
robotprogramming.uji.es/mod/lti/view.php?id=164
Search
RobotProgramming.Net
Català (ca)
Heu entrat com Gustavo Casañ (Sortida)
The Turtle Robot
Inici ▸ Els meus cursos ▸ Sample Courses ▸ TurtleBot ▸ Hola! ▸ Pavelló Tortuga
Script:
Output:
Run
Stop
Clear
Reset
Clear output

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

# Write your commands here
left(30)
forward(-3)
right(50)
for i in range(7):
  forward(3)
  left(45)
```

Figure 7: Course documentation page of the course (left) and programming page (right).

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.

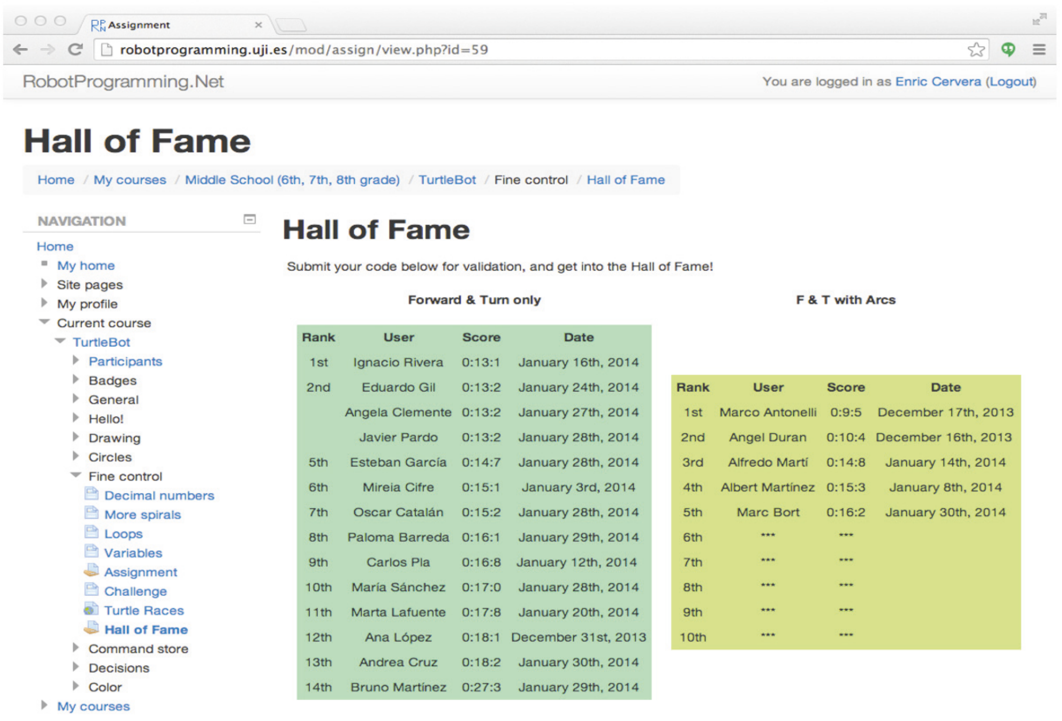
Finally it is important to remark the security of the system. Each student has to identify himself or herself with some type of email account (although we facilitate the procedure through the use of google and Facebook accounts) before entering into the system. This creates a problem, as in some countries (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.



Hall of Fame

Submit your code below for validation, and get into the Hall of Fame!

Forward & Turn only

Rank	User	Score	Date
1st	Ignacio Rivera	0:13:1	January 16th, 2014
2nd	Eduardo Gil	0:13:2	January 24th, 2014
	Angela Clemente	0:13:2	January 27th, 2014
	Javier Pardo	0:13:2	January 28th, 2014
5th	Esteban García	0:14:7	January 28th, 2014
6th	Mireia Cifre	0:15:1	January 3rd, 2014
7th	Oscar Catalán	0:15:2	January 28th, 2014
8th	Paloma Barreda	0:16:1	January 29th, 2014
9th	Carlos Pla	0:16:8	January 12th, 2014
10th	María Sánchez	0:17:0	January 28th, 2014
11th	Marta Lafuente	0:17:8	January 20th, 2014
12th	Ana López	0:18:1	December 31st, 2013
13th	Andrea Cruz	0:18:2	January 30th, 2014
14th	Bruno Martínez	0:27:3	January 29th, 2014

F & T with Arcs

Rank	User	Score	Date
1st	Marco Antonelli	0:9:5	December 17th, 2013
2nd	Angel Duran	0:10:4	December 16th, 2013
3rd	Alfredo Martí	0:14:8	January 14th, 2014
4th	Albert Martínez	0:15:3	January 8th, 2014
5th	Marc Bort	0:16:2	January 30th, 2014
6th	***	***	
7th	***	***	
8th	***	***	
9th	***	***	
10th	***	***	

Figure 8: Image of the Hall of Fame with the students' classification.

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.

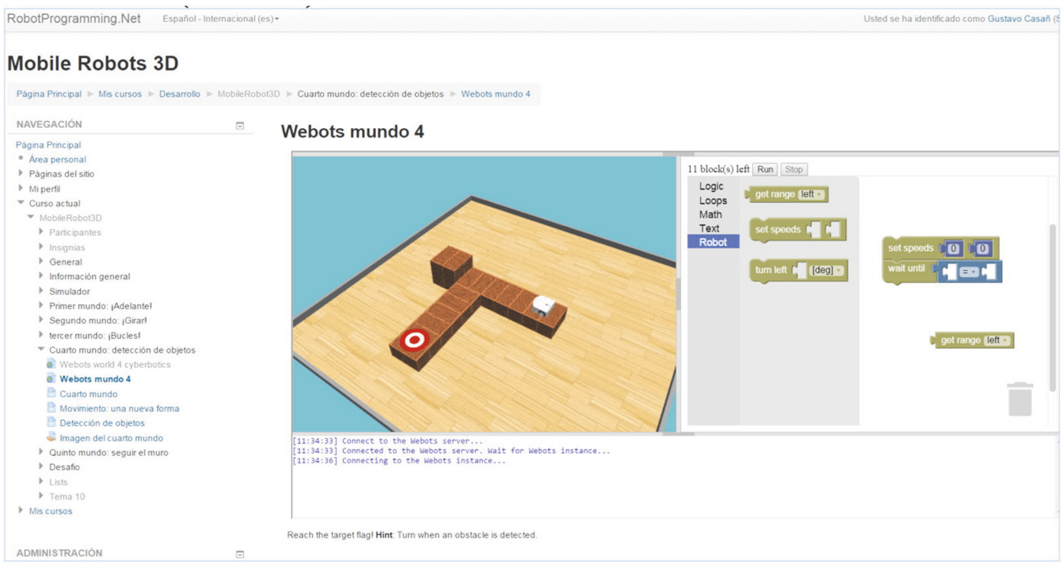
3.2.2. Introduction to Mobile Robots. The second course [38, 41] was developed surrounding a more powerful and realistic simulator for mobile robots, Stage [42], but it shares a similar organization and aspect with the Turtle Robot. The Stage simulator is readily available in ROS and it has been integrated 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).

Figure 9: Several badges created for this course.



RobotProgramming.Net Español - Internacional (es) Usted se ha identificado como Gustavo Casañ

Mobile Robots 3D

Página Principal ► Mis cursos ► Desarrollo ► MobileRobot3D ► Cuarto mundo: detección de objetos ► Webots mundo 4

NAVEGACIÓN

- Página Principal
  - Área personal
  - Páginas del sitio
  - Mi perfil
  - Curso actual
    - MobileRobot3D
      - Participantes
      - Insignias
      - General
      - Información general
      - Simulador
      - Primer mundo: ¡Adelante!
      - Segundo mundo: ¡Girar!
      - tercer mundo: ¡Bucles!
      - Cuarto mundo: detección de objetos
        - Webots world 4 cyberbotics
        - Webots mundo 4
        - Cuarto mundo
        - Movimiento: una nueva forma
        - Detección de objetos
        - Imagen del cuarto mundo
      - Quinto mundo: seguir el muro
      - Desafío
      - Lists
      - Tema 10
  - Mis cursos

ADMINISTRACIÓN

Webots mundo 4

11 block(s) left Run Stop

```
[11:34:33] Connect to the Webots server...
[11:34:33] Connected to the Webots server. Wait for Webots instance...
[11:34:36] Connecting to the Webots instance...
```

Reach the target flag! **Hint**: Turn when an obstacle is detected.

Figure 10: Mobile robots 3D environment.

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=kONe0zC5HnY&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.uji.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 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 OpenBadges 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.

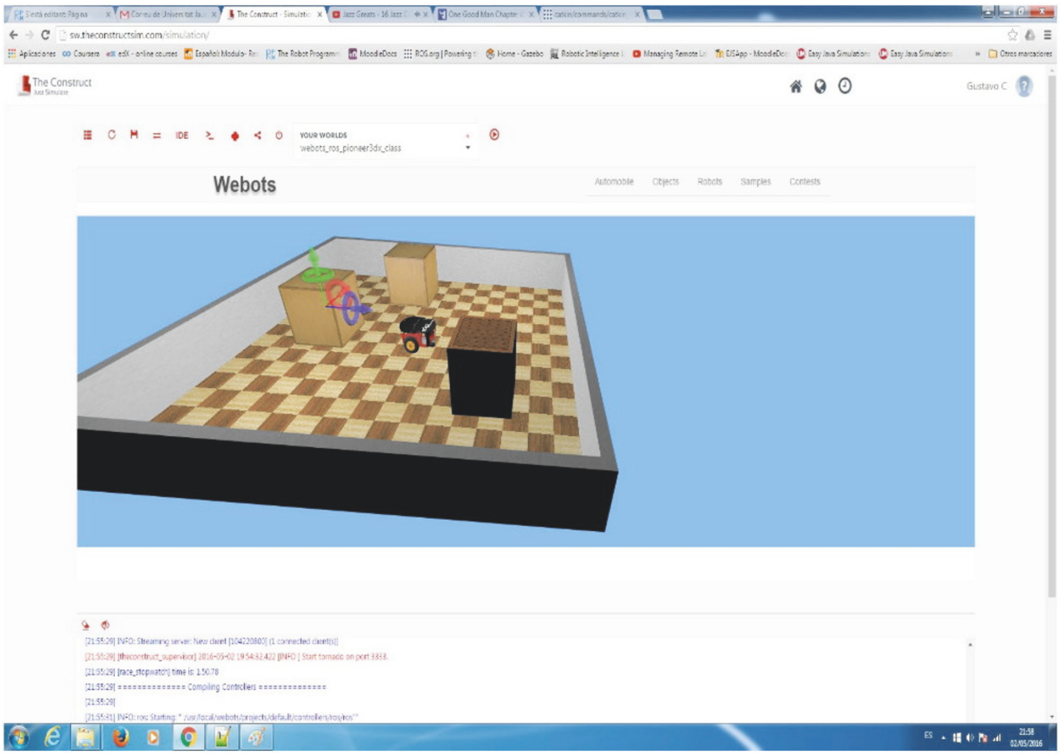
Thus, we created a version of our Mobile Robots course using their system and two different simulators, Gazebo (http://gazebosim.org/) [49] and Webots. The robots used were a Pioneer model (Figure 11) and a Kobuki.

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].

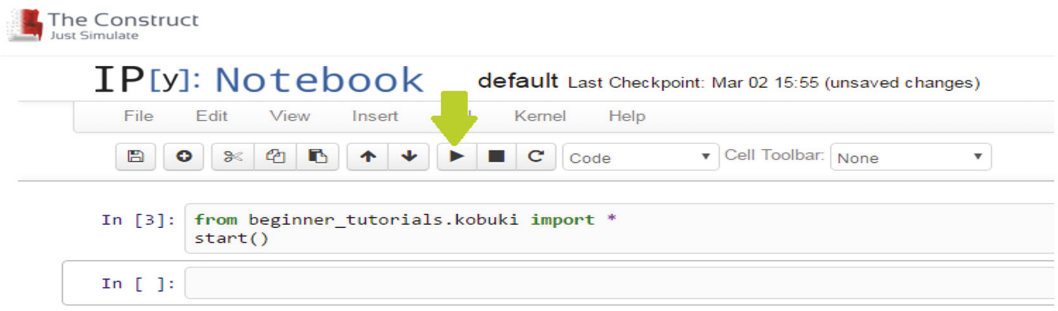
3.3. Evaluation. The evaluation of each course was made through the use of a final questionnaire (we can see an example in Table 2) to the students and the results have already been published in previous papers like [37, 38, 41, 43, 52]. The number of students in the individual experiences was limited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



```
[21:55:29] INFO: Streaming server: New client [104220800] (1 connected client(s))
[21:55:29] [theconstruct_supervisor] 2016-05-02 19:54:32,422 [INFO ] Start tornado on port 3333.
[21:55:29] [race_stopwatch] time is: 1.50.78
[21:55:29] ============== Compiling Controllers ==============
[21:55:29]
[21:55:31] INFO: ros: Starting: " /usr/local/webots/projects/default/controllers/ros/ros""
```

Figure 11: Pioneer robot simulation in the Construct.



```
In [3]: from beginner_tutorials.kobuki import *
        start()

In [ ]:
```

Figure 12: Jupyter Notebook. The green arrow marks the run button.

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."

Learning compared with traditional methods
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?
Ease of use
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?
Quality of the virtual robot
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?
Suitability in learning of relevant concepts
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?
Satisfaction degree
In general, do you feel satisfied with the practical experiences through the Internet?

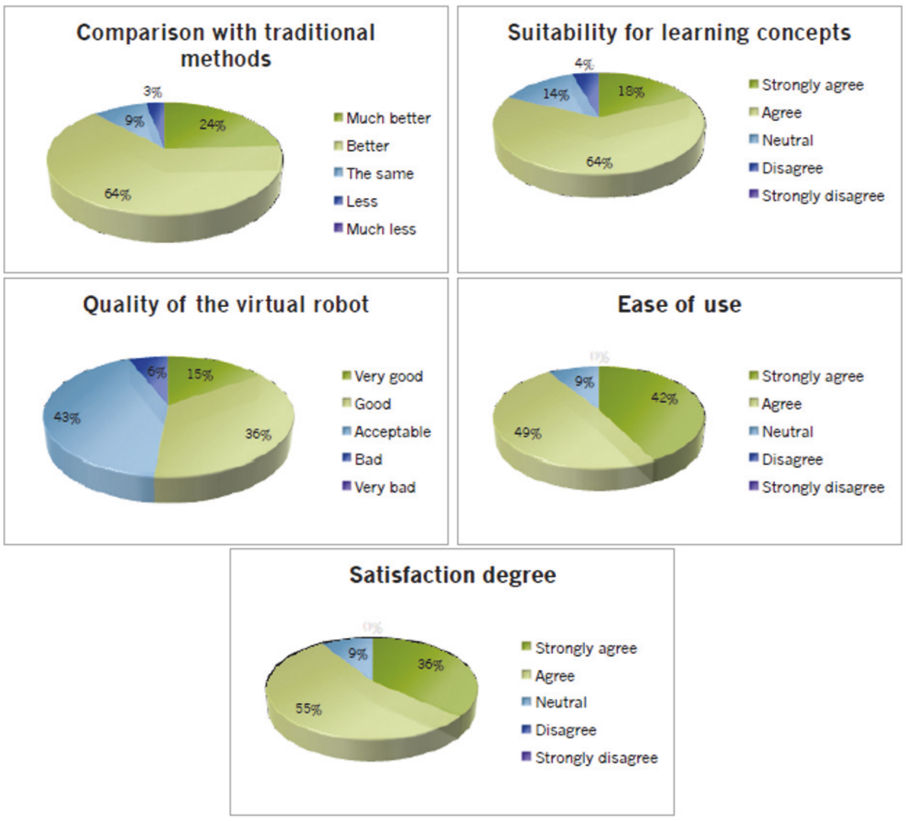


Figure 13: Results of the Turtle Robot course students questionnaire, from [37].

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.

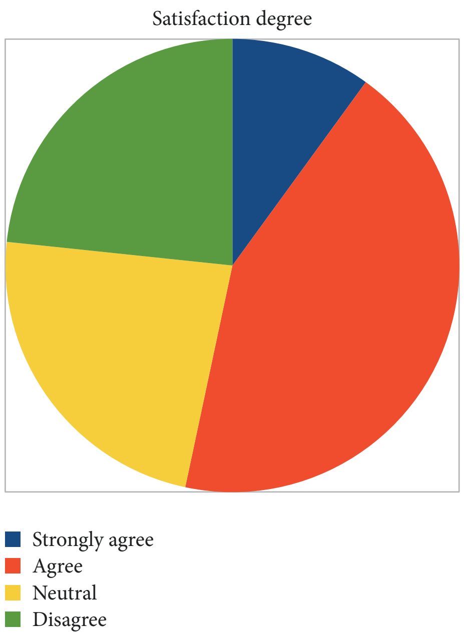


Figure 14: Results of the Turtle Robot course students questionnaire, from [52].

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 [19]. 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/perfils/futurs/base/accions/grau/connecta/&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/RobotProgrammingNetwork/ and a YouTube channel https://www.youtube.com/channel/UCeWlJxX52YHjf62xAoWfLeg), as a further method to increase visibility of the project.

Abbreviations

AENUI:	Asociación de Enseñantes Universitarios de la Informática
API:	Application Programming Interfaces
CEMRA:	Creation of Educational Materials for Robotics and Automation
IEEE-RAS:	Institute of Electrical and Electronics Engineers Robotics and Automation Society
LTI:	Learning Tools Interoperability
MOOC:	Massive Online Open Courses
RASPEBRAS:	Robotic and Automation Society Summer School on Experimental Methodology, Performance Evaluation and Benchmarking in Robotics

RMS:	Robot Management System
ROS:	Robot Operating System
RPN:	Robot Programming Network
VM:	Virtual Machine.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This project was made possible thanks to IEEE Robotics and Automation Society (IEEE-RAS http://www.ieee-ras.org/) through a CEMRA grant (Creation of Educational Materials for Robotics and Automation). Thanks are due to to the Construct and Cyberbotics. Thanks are also due to Ministerio de Economía y Competitividad (DPI2011-27846) and Generalitat Valenciana (PROMETEOII/2014/028) at Jaume I University (P1-1B2011-54).

References

[1] J. G. Wilhelm and P. J. Wilhelm, "Inquiring minds learn to read, write, and think: Reaching all learners through inquiry," *Middle School Journal*, pp. 39–46, 2010.

[2] E. M. Furtak, T. Seidel, H. Iverson, and D. C. Briggs, "Experimental and quasi-experimental studies of inquiry-based science teaching. A meta-analysis," *Review of Educational Research*, vol. 82, no. 3, Article ID 0034654312457206, pp. 300–329, 2012, https://doi.org/10.3102/0034654312457206.

[3] C. Bonwell and J. Eison, Active Learning: Creating Excitement in the Classroom AEHE-ERIC Higher Education Report No. 1, Jossey-Bass, Washington, D.C., USA, ISBN 1-878380-08-7, 1991.

[4] M. A. Albanese and S. Mitchell, "Problem-based learning: a review of literature on its outcomes and implementation issues," *Academic Medicine*, vol. 68, no. 1, pp. 52–81, 1993.

[5] J. Johnson, "Children, robotics, and education," *Artificial Life and Robotics*, vol. 7, no. 1-2, pp. 16–21, 2003.

[6] V. Dagdilelis, M. Sartatzemi, and K. Kagani, "Teaching (with) robots in secondary schools: Some new and not-so-new pedagogical problems," in *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies, ICALT 2005*, pp. 757–761, Kaohsiung, Taiwan, July 2005.

[7] F. Barreto and V. Benitti, "Exploring the educational potential of robotics in schools: A systematic review," *Computers & Education*, vol. 58, no. 3, pp. 978–988, 2012.

[8] K. Schilling, H. Roth, and O. J. Rösch, "Mobile mini-robots for engineering education," *Global Journal of Engineering Education*, vol. 6, no. 1, pp. 79–84, 2002.

[9] S. Djenic, R. Krneta, and J. Mitic, "Blended learning of programming in the internet age," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 247–254, 2011.

[10] C. C. Ratcliff and S. E. Anderson, "Reviving the turtle: Exploring the use of logo with students with mild disabilities," *Computers in the Schools*, vol. 28, no. 3, pp. 241–255, 2011, https://doi.org/10.1080/07380569.2011.594987.

[11] K. Asanovic, R. Bodik, J. Demmel et al., "A view of the parallel computing landscape," *Communications of the ACM*, vol. 52, no. 10, pp. 56–67, 2009.

[12] N. Tillmann, M. Moskal, J. de Halleux et al., "The future of teaching programming is on mobile devices," in *Proceedings of the Proc. 17th ACM annual conference on Innovation and Technology in Computer Science Education, ITiCSE'12*, pp. 156–161, Haifa, Israel, 2012.

[13] E. Wang, "Teaching freshmen design, creativity and programming with LEGOs and Labview," in *Proceedings of the 31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education.*, pp. F3G–11-15, Reno, NV, USA.

[14] F. Mondada, M. Bonani, X. Raemy et al., "The e-puck, a robot designed for education in engineering," in *9th Conference on Autonomous Robot Systems and Competitions*, vol. 1, no. 1, pp. 59–65, 2009.

[15] J. Trevelyan, "Lessons learned from 10 years experience with remote laboratories," in *Proceedings of International Conference on Engineering Education and Research, iCEER*, pp. 687–697, Bouzov Castle, Check Republic, 2004.

[16] V. Djalic, P. Maric, D. Kosic, D. Samuelsen, B. Thyberg, and O. Graven, "Remote laboratory for robotics and automation as a tool for remote access to learning content," in *Proceedings of the 2012 15th International Conference on Interactive Collaborative Learning, ICL 2012*, Villach, Austria, September 2012.

[17] M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino, "A remote lab for experiments with a team of mobile robots," *Sensors*, vol. 14, no. 9, pp. 16486–16507, 2014.

[18] C. D'Angelo, D. Rutstein, C. Harris, R. Bernard, E. Borokhovski, and G. Haertel, "Simulations for STEM Learning: Systematic Review and Meta-Analysis," in *Proceedings of the SRI International*, Menlo Park, CA, USA, 2014.

[19] T. De Jong, M. C. Linn, and Z. C. Zacharia, "Physical and Virtual Laboratories in Science and Engineering Education," *Science*, vol. 340, no. 6130, 2013.

[20] T. de Jong, S. Sotiriou, and D. Gillet, "Innovations in STEM education: the Go-Lab federation of online labs," *Smart Learning Environments*, vol. 1, no. 1, 2014.

[21] M. Quigley, K. Conley, B. Gerkey et al., "ROS," *An open-source robot operating system*, 2009.

[22] S. Osentoski, G. Jay, C. Crick, B. Pitzer, C. Du Hadway, and O. C. Jenkins, "Robots as web services: Reproducible experimentation and application development using rosjs," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation, ICRA 2011*, pp. 6078–6083, chn, May 2011.

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

[24] M. Waibel, M. Beetz, J. Civera et al., "Robo earth," *IEEE Robotics and Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[25] M. C. Viegas, G. Alves, A. Marques et al., "The VISIR+ Project Preliminary results of the training actions," *Online Engineering & Internet of Things*, pp. 375–391, 2018.

[26] A. Mejías and J. M. Andújar, "Interaction of Real Robots with Virtual Scenarios through Augmented Reality: Application to Robotics Teaching/ Learning by Means of Remote Labs," *International Journal of Engineering Education*, vol. 9, no. 3, 2013.

[27] B. Pitzer, S. Osentoski, G. Jay, C. Crick, and O. C. Jenkins, "PR2 Remote Lab: An environment for remote development and experimentation," pp. 3200–3205.

[28] R. Marín, P. J. Sanz, and A. P. Del Pobil, "The UJI Online Robot: An Education and Training Experience," *Autonomous Robots*, vol. 15, no. 3, pp. 283–297, 2003.

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

[30] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, “ROS and Rosbridge: Roboticists out of the loop,” in *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI'12*, pp. 493-494, USA, March 2012.

[31] M. Dougiamas and P. C. Taylor, “Moodle, Using learning communities to create an open source course management system,” in *Proceedings of World conference on educational multimedia, hypermedia and telecommunications*, pp. 171–178, 2003.

[32] M. Alier, M. J. Casañ, and J. Piguillem, “Moodle 2.0: Shifting from a learning toolkit to a open learning platform,” *Communications in Computer and Information Science*, vol. 73, pp. 1–10, 2010.

[33] M. Casini, F. Chinello, D. Prattichizzo, and A. Vicino, “RACT: A remote lab for robotics experiments,” in *Proceedings of the 17th IFAC World Congress*, Seoul , korea, July 2008.

[34] I. Matijevics, “Local and remote laboratories in the education of robot architectures,” in *Intelligent Engineering Systems and Computational Cybernetics*, pp. 27–36, Springer, 2009.

[35] H. Banchi and R. Bell, “The Many Levels of Inquiry,” *Science and Children*, vol. 46, no. 2, pp. 26–29, 2008.

[36] K. M. Kapp, *The Gamification of Learning and Instruction*, Pfeiffer, Game-Based Methods and Strategies for Training and Education, 2012.

[37] E. Cervera, P. Martinet, R. Marin et al., “The Robot Programming Network,” *Journal of Intelligent and Robotic Sytems*, vol. 81, no. 1, pp. 77–95, 2015.

[38] E. Cervera and G. A. Casañ, “Robot Programming Network: un sistema distribuido para el aprendizaje de la programación de robots,” *ReVisión*, vol. 8, no. 1, ISSN 1989-1199, 2015.

[39] P. B. Lawhead, M. E. Duncan, C. G. Bland et al., “A road map for teaching introductory programming using LEGO⌐ mindstorms robots,” *ACM SIGCSE Bulletin*, vol. 35, no. 2, pp. 191–201, 2003.

[40] R. R. Murphy, “Competing for a robotics education,” *IEEE Robotics and Automation Magazine*, vol. 8, no. 2, pp. 44–55, 2001.

[41] G. A. Casañ and E. Cervera, “Introducción a la Programación de Robots Móviles: un curso en 3D con simuladores,” *ReVisión*, vol. 9, no. 3, ISSN 1989-1199, 2016.

[42] R. Vaughan, “Massively multi-robot simulation in stage,” *Swarm Intelligence*, vol. 2, no. 2-5, pp. 189–208, 2008.

[43] G. A. Casañ and E. Cervera, “RPN: aprendizaje de la programación de robots mediante bloques en un entorno 3D,” *ReVisión*, vol. 8, no. 3, ISSN 1989-1199, 2014.

[44] O. Michel, “Webots: professional mobile robot simulation,” *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[45] D. Gouaillier, V. Hugel, P. Blazevic et al., “Mechatronic design of NAO humanoid. Proc. IEEE International Conference on Robotics and Automation, ICRA'09,” in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 769–774, ICRA'09, Kobe, Japón, 2009.

[46] S. Van Noort and A. Visser, “Validation of the dynamics of an humanoid robot in USARSim,” in *Proceedings of the 11th Workshop on Performance Metrics for Intelligent Systems*, pp. 190–197, March 2012.

[47] G. A. Casañ, E. Cervera, A. del Pobil, and G. A. Casañ, “Online teaching of humanoid robots,” in *Human Robot Interaction Education Workshop*, Portland, Oregon, 2015.

[48] G. A. Casañ, E. Cervera, O. Michel, R. Tellez, and G. A. Casañ, “NAO Race Competition,” *ReVisión*, vol. 8, no. 3, ISSN 1989-1199, 2015.

[49] N. Koening and A. Howard, “Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154, Sendai, Japón, 2004.

[50] V. V. Klndratenko, C. P. Steffen, and R. J. Brunner, “Accelerating scientific applications with reconfigurable computing: getting started,” *Computing in Science & Engineering*, vol. 9, no. 5, pp. 70–77, 2007.

[51] D. Pritchard and T. Vasiga, “CS Circles: An in-browser python course for beginners,” in *Proceedings of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE 2013*, pp. 591–596, usa, March 2013.

[52] G. A. Casañ and P. Ramo, “Una experiencia en la enseñanza en secundaria de robótica y programación con recursos remotos: el curso Tortuga del RPN,” *ReVisión*, vol. 10, no. 3, ISSN 1989-1199, 2017.