A ROS-based open web platform for Intelligent Robotics education

David Roldán[1], Sakshay Mahna[2], and José M. Cañas[1]

[1] RoboticsLabURJC, Universidad Rey Juan Carlos, Spain,
david.roldan@urjc.es
[2] Indian Institute of Technology, Ropar, India

Abstract. This paper presents the new release of the Robotics Academy learning framework and the open course on Intelligent Robotics available on it for anyone. The framework hosts a collection of practical exercises of robot programming for engineering students and teachers at universities. It has evolved from an open tool, which the users had to install on their machines, to an open web platform, simplifying the user's experience. It has been redesigned with the adoption of state-of-the-art web technologies and DevOps that make it multiplatform and scalable. The web browser is the new frontend for the users, both for source code editing and for the monitoring GUI of the exercise execution. All the software dependences are already preinstalled in a container, including the Gazebo simulator. The proposed web platform provides a free and nice way for teaching Robotics following the learn-by-doing approach. It is also a useful complement for remote educational robotics.

Keywords: Web-based robotics, simulation, remote educational robotics

1 Introduction

In the last decade robotics has experienced a large growth in the number of applications available in the market. We usually think of robots as the robotic arms used in the industrial sector and automated assembly processes. However, today robots have appeared in many areas of daily life such as food processing or warehouse logistics [1]. Moreover, recently robots have been used in the homes to carry out real life tasks for people, such as vacuum cleaning. This shows how robots can successfully address real life tasks. In many areas, robots are progressively being included, with technologies such as automatic parking or driver-assistance systems, not to mention aerial robots that have also become really popular.

Robotics in Education tries to strengthen the learning skills of future engineers and scientists by using robot-based projects. Both in schools and colleges presenting robots in the classroom will give students a more interesting vision of science and engineering, and they will be able to observe directly the practical application of theoretical concepts in the fields of mathematics and technology.

From the perspective of the University, robot-based projects help to enhance key abilities since their origin, for instance those identified with mechatronics [2].

The fast technological changes and the subsequent movements in education require the development and use of educational methodologies and opportunities with a moderate economic effort. Many institutions answer to this challenge creating distance education programs. Web-based robotics distance education is a growing field of engineering education. Web based applications allow students to code robotic projects without having to run special software in their computers nor the robot physically near them [3, 4].

When learning online, the robotic projects take place using a robotic framework and a teaching tool where the robot behavior is programmed using certain language [4]. The most common way to interact with robots remotely is through virtual and remote laboratories [5]. According to Esposito's work [6], MATLAB (62%) and C (52%) are the dominat choices, with a minority using free open-source packages such as the Robot Operating System (ROS) (28%) or OpenCV (17%). ROS stands out for having a large and enthusiastic community. ROS enables all users to leverage an ever-increasing number of algorithms and simulation packages, as well as providing a common ground for testing and virtual commissioning [7]. The best ROS compatible simulator is Gazebo [8]. It provides 3D physics simulation for a large variety of sensors and robots, and it can be customized for the user needs. Gazebo is bundled into the full installation package of ROS, making it widely and easily available. Many robot manufacturers offer ROS packages specifically designed for Gazebo support. Therefore, ROS and Gazebo seem suitable tools to be used in web environments to offer robotics education at no cost.

In this paper we present the latest release of Robotics Academy learning framework.

On its exercises the student is intended to develop the intelligence of some robots for several robotics application. Robotics Academy is open source and uses Python, ROS, Gazebo and Docker for supplying an easy method to create robotics or computer vision applications.

2 Robotics Engineering Education

In educational robotics at universities there are two areas that are usually taught: industrial robots and mobile robots [9]. The main topics of knowledge are manipulation control techniques, inverse kinematics, trajectory calculations, local and global navigation, position control, perception, mapping and self-location.

Today, the proggressive digitalization of the world has transformed how the knowledge is created and consumed. In this sense, web platforms are gaining popularity exponentially, since they allow users to access content wherever and whenever as long as they have an Internet connection. Some leading universities already provide online robotic courses such as Artificial Intelligence for Robotics from Standford University, Autonomous Mobile Robots from ETH Zurich or the

Art of Grasping and Manipulation in Robotics from the University of Siena. They see the potential of online learning in robotics engineering.

In addition, many authors and developers have joined forces to create web platforms for robotics. We may find in the literature examples of robotics web platforms such as RobUALab [10]. This platform has been programmed in Java and thanks to web technologies it allows to teleoperate, experiment and programm a robotic arm. With it, the students can test diferent configurations, evaluate pairs, both Cartesian and articulation trajectories and program the arms with a list of comands.

Another example is the work of Peidró et al. [11]. They presented a virtual and remote laboratory of parallel robots consisting of a simulation tool and a real robot. The virtual tool helps the students to simulate the inverse and forward kinematic problems of three parallel robots in an intuitive and graphical way. On the remote lab, the students can experiment with a real parallel robot and let them explore phenomena that arise in the motion of the real robot and do not happen in the simulation.

Fabregas et al. [5] presented the Robots Formation Control Platform, a web based tool for simulation and real experimentation with mobile robots with pedagogical purposes. The platform provides an interactive simulator to develop advanced experiment of formation control with multi-robots systems, an experimental environment to develop laboratory sessions with real mobile robots and a low-cost robotic platform. It also allows the students to conduct experiments with mobile robots in a dynamic environment by chaning the configuration without having to reprogramm the simulation itself.

One of the main issues regarding robotics platforms is that they do not follow a standard in the robotic infrastructure used and many times they do not use open hardware or software [12]. It was not until recently when ROS has been taken into account in the development of educational robotics platforms. One interesting educational robotics platform is EUROPA, an extensible one, based on open hardware and software (ROS) which includes friendly user interfaces for control and data acquisition and a simulation environment [13]. By using ROS, they provide the opportunity for both teachers and students to work with advanced simulation and visualization tools. EUROPA allows easy programmability based on Python. In spite of all the advantages of the EUROPA platform, its functionality is limited to only one robot and the computer where the software is installed.

3 Robotics Academy: from open tool from open web platform

This educational robotics framework hosts a *collection of practical exercises of robot programming* for engineering students and teachers at universities. It follows the learn-by-doing paradigm and puts the focus in the programming of the main robotics algorithms: for perception, navigation, control, localization, mapping, etc. It supports Python as robot programming language and it is strongly

based on the Gazebo simulator. In addition, it is open source, its source code is publicly available at GitHub [3].

Each exercise is a robot application, which is designed as the *combination of the student's code and a template with auxiliary code*. The template includes many functionalities that are required to execute that robot application but are not the focus of the learning target. There is one specific template for each exercise.

The first major release was based on ICE middleware [1]. The second one was completely migrated to ROS middleware and two robotics courses were developed for it and provided [14, 9]. This second release included a ROS-node as the template for each exercise. The student was required to include her programming in a separate Python file and to install all the software dependences needed for each exercise. The framework included installation recipes for all those dependencies, which are typically third party software such as ROS middleware for sensor and actuator drivers, PX4 for drones, OpenCV for image processing, MoveIt for industrial robotics, etc.

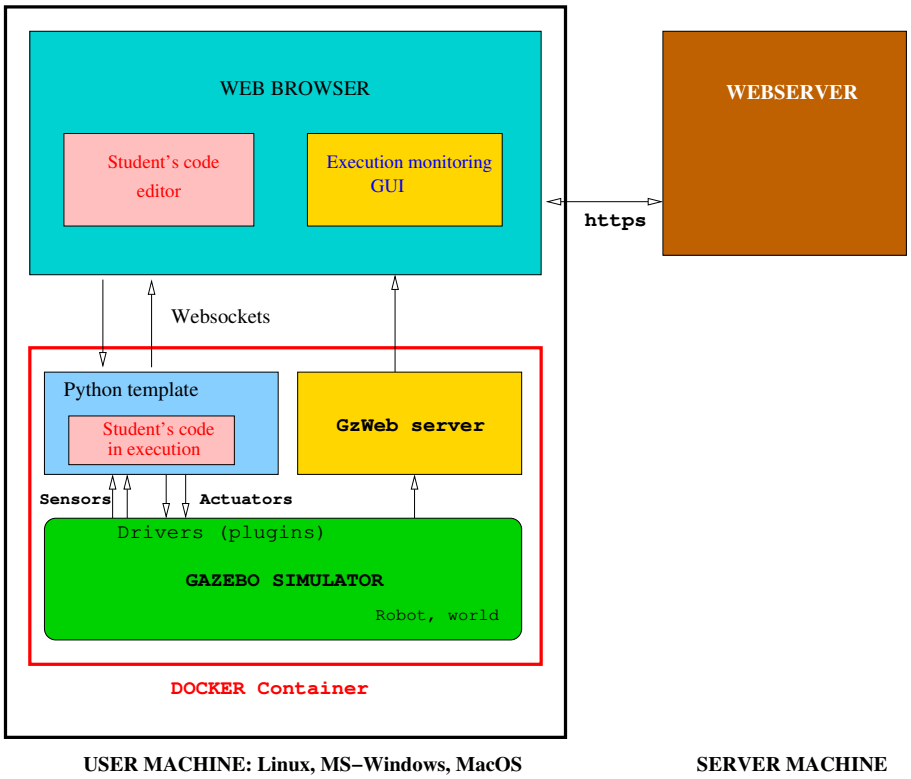


Fig. 1: New design of Robotics Academy learning framework

[3] https://github.com/JdeRobot/RoboticsAcademy

The third major release [15] has been significantly refactored following the design in Fig. 1 and delivered as a web platform. It is available and ready to use at the web[4], which makes it a nice choice for distance engineering learning. It persistently stores the student's code for the exercises, who may access them and run them anytime anywhere. Its main improvements are described in the following subsections.

3.1 Robotics Academy Docker Image

The robotics software is inherently complex, includes many pieces and so it usually has many dependencies. For instance, the software access to robot sensors and actuators is provided by drivers. Sometimes the focus for the Robotics students is on the drivers themselves, but when focusing on the algorithms the students just need to install and use them. They are included in common robotics frameworks such as ROS and Gazebo simulator.

In order to run RoboticsAcademy exercises many software pieces are required and so, the student had to install all of them. The installation recipes delivered in the second framework release were not enough to significantly simplify the experience for the users. They still had to manually install all the needed packages on their local operating system, and this was an entry barrier. In the presented release a Docker container is provided[5] which includes all the dependencies of exercises already preinstalled inside. It is called RADI (Robotics Academy Docker Image) and it may be easily installed and run in most operating systems, as all of them support this de-facto DevOps standard. This approach follows some of the ideas in ROSLab [16] which also uses ROS as underlying middleware and takes benefit of Jupyter Notebooks and Docker containers for providing reproducible research environments.

The burden of installing all the dependencies is removed from users and moved to the RoboticsAcademy developers who build that container. This way the learning curve is greatly reduced for final users, who skip most of the details. They are only required to install a Docker image, which is a simple and standard step. In addition, this container works both in Linux, MS-Windows and MacOS, expanding so the number of potential users of the framework.

The robot simulator and the student's code both run inside that Docker container, in the user's machine. This design allows the framework to scale up to many users, as the main computing costs are already distributed.

3.2 New web-based exercise templates

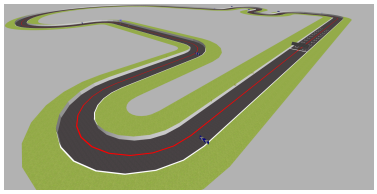
The new exercise templates have two parts, as shown in Fig.1: a Python program inside the RADI (`host.py`) and web page inside the browser (`exercise.html`). They are connected to each other through two websockets. The browser holds the code editor and sends the user's source code for an exercise to the `host.py`,

[4] https://unibotics.org
[5] https://hub.docker.com/r/jderobot/robotics-academy/tags

where it will be run. The `host.py` sends debugging information to the browser, where it will be displayed.

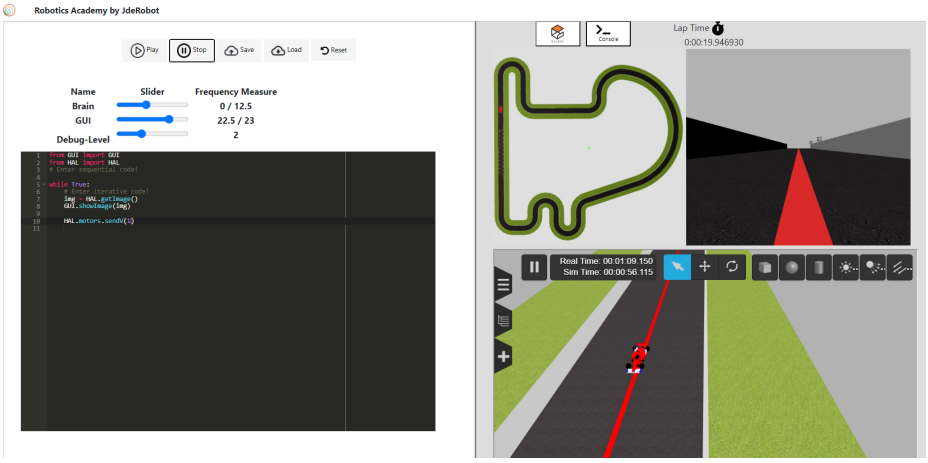
The student just programs the robot's brain in the browser. From the user's point of view the template provides two programming interfaces: (a) HAL-API for reading the robot sensor measurements and for commanding orders to the robot actuators; and (b) GUI-API which includes several useful methods for debugging, for showing print messages in the browser or displaying the processed images.

The Python program `host.py` is connected to the Gazebo simulator for getting the sensor readings and setting the motor commands. In addition, it runs the source code of the robot's brain received from the browser, which typically executes perception and control iterations in an infinite loop.



(a) Gazebo environment

(b) Autonomous F1 Car



(c) Exercise GUI

Fig. 2: Visual Follow Line Exercise

The browser is the only Graphical User Interface (GUI). The webpage of each exercise, `exercise.html`, is divided in two vertical columns, as shown in Fig.2c. The left column is for an inline text editor and the buttons for executing the code, stopping it, etc.. The right column typically includes a view of the simulated world to see the generated robot behavior. It also includes a debugging

console for text messages and several exercise-specific illustrative widgets to show debugging information such as the processed images. The widgets use state-of-the-art web technologies from html5 such as 2D canvas, 3D graphics, websockets, etc.

3.3 Webserver

The core of the web platform is a Django based webserver. It shows the different exercises available to the users so they may choose which one try to solve. The currently available exercises are shown in Fig.3. Once the user chooses an exercise the webserver will show the template corresponding to that exercise, its simulation will be executed in the Docker container that the user has instantiated on his computer, as shown in Fig.1.

The webserver also supports a second option: the Docker containers are instantiated and run on a backend computer farm, and so the user is not required to install anything in his/her computer. The webserver distributes the user's simulation into one of the available computers in the farm. The IP of the farm computers remain hidden for the user, since all of the required communications (sending code, receiving images, sensor information, etc) occur through the webserver itself.

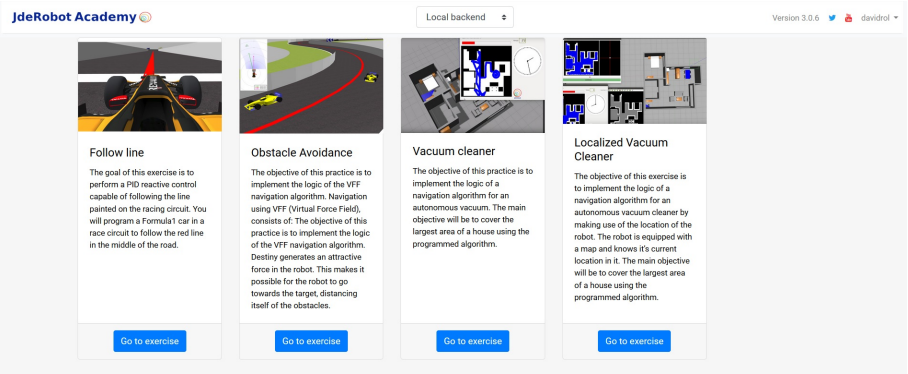


Fig. 3: Available exercises in Robotics Academy

In order to manage the computer farm and the running Docker containers the webserver provides an administration panel as shown in Fig.4. The panel offers all the functionalities needed to monitor execution, add a new computer, to delete a container, to restart a container and to run a new container.

Through the webserver, the user is able to save and load the code written for each exercise. When the code is saved, the webserver stores it in an Amazon S3 service. Amazon S3 is an object storage service that offers scalability, data availability, security and performance. Everytime the user loads an exercise, the last version of the code is retrieved from Amazon S3.

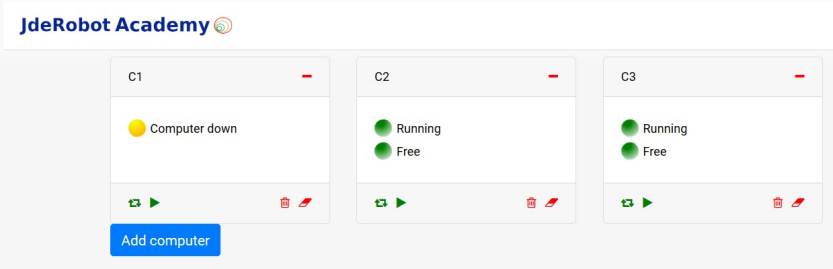


Fig. 4: Admin panel with the computers in the farm with a running RADI

4 Intelligent Robotics Course

Robotics-Academy is currently in use in two university courses on Intelligent Robotics at Universidad Rey Juan Carlos. Both of them follow a 'learn by doing' paradigm and focus on robotics algorithms. Solutions to the exercises involve the use of PID Controllers, Local and Global Navigation, random as well as Optimal Coverage algorithms. In each exercise, the robot to be used, the environment and the task to be carried out are detailed. Reference solutions have been developed and published as illustrative videos, so the students may know in advance the expected robot behavior and results for each exercise. Some Theory related to the exercise task and programming hints are also available for the students.

4.1 Visual Follow Line exercise

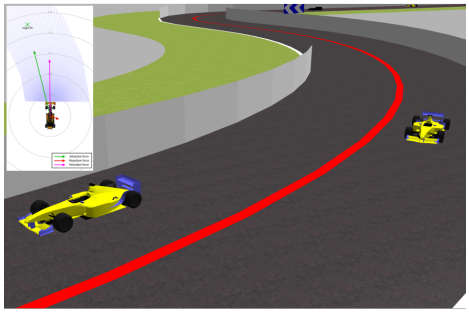
The challenge of this exercise is to program an autonomous F1 Car to complete a lap of the circuit in the shortest possible time by following a red colored line drawn throughout the circuit. The car is equipped with an onboard front camera. Regarding movement, an intermediate command interface has been provided: forward speed V and angular speed W. Fig. 2a and Fig. 2b show the Gazebo environment and the F1 car used.

This exercise is designed to teach basic reactive control, including PID controllers as well as introducing students to basic image processing, for example color filters, morphological filters or segmentation of the red line from the track. The solution involves segmenting the red line and making the car follow it using a PID based control. The GUI of this exercise includes a bird's eye view widget to know the current position of the car inside the track, as shown in Fig. 2c.

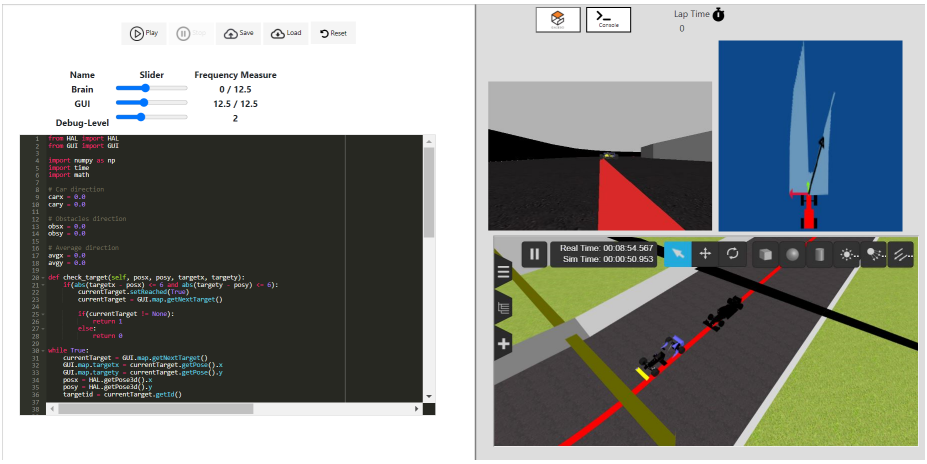
4.2 Obstacle Avoidance exercise

In this exercise, the user has to program an Autonomous Car to complete a lap of the circuit in the minimum possible time while avoiding the obstacles present in the way. The car has a laser sensor in front of it. The car may be given commands for translational speed V and angular speed W. The Gazebo scenario consists of a race track similar to Visual Follow Line exercise, with stationary

obstacles present at different places on the track. Fig. 5a shows the GUI, the robot and simulated scenario.



(a) Gazebo scenario



(b) Exercise GUI

Fig. 5: Obstacle Avoidance Exercise

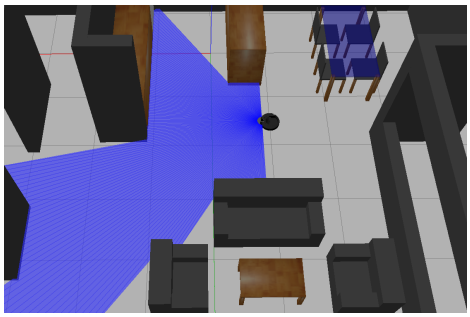
The solution of this exercise involves the use of Local Navigation Algorithms, such as Virtual Force Field (VFF). A sequence of waypoints is provided to the students for a successful completion of the circuit. Navigation using the VFF algorithm consists of obstacles generating a repulsive force and the current waypoint generating an attractive force. This makes it possible for the robot to go towards the nearest waypoint, distancing itself of the obstacles and addressing towards the vector sum of the forces. A debugging widget specific to this vector sum algorithm is provided in the GUI, as shown in Fig.5b.

4.3 Vacuum Cleaner exercise

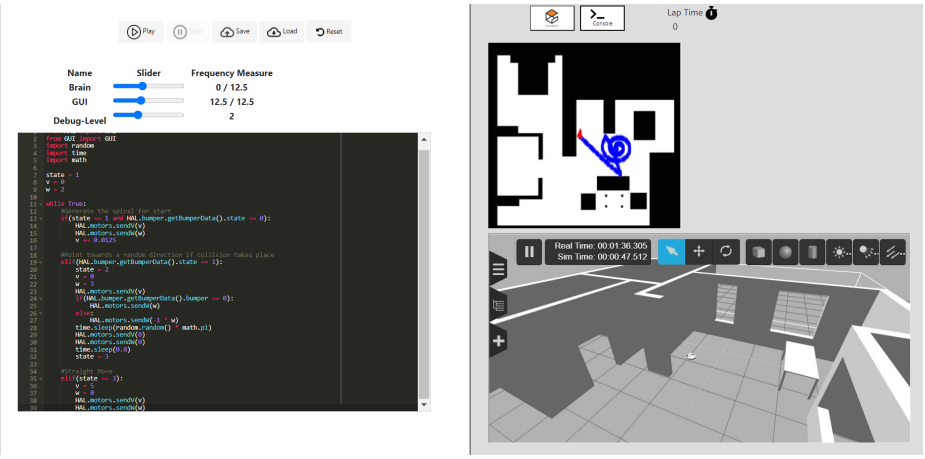
The challenge of this exercise is to program a Roomba vacuum cleaner robot to clean an apartment. The robot is intended to visit the whole area of the apart-

ment, in the shortest possible time. It can softly bump into walls, chairs etc, as real vacuum cleaner robots do. The robot has motors which can be commanded in translational speed V and rotational speed W. It includes a laser sensor and 3 bump sensors (left, middle and right). For instance, the *getLaserData()* method is used to read the laser sensor measurements. Fig.6a shows the Gazebo environment of the exercise.

Random Wandering algorithms may be developed to solve this exercise. For instance, initial spiral and bump and go behavior may be proposed as base solutions. This exercise includes a debugging widget as shown in Fig. 6b, which displays the path followed and the area swept by the robot.



(a) Gazebo scenario



(b) Exercise GUI

Fig. 6: Vacuum Cleaner Exercise

4.4 Vacuum Cleaner with Localization exercise

This exercise is similar to the previous one. The student is provided with the same Roomba vacuum cleaner robot to clean an apartment. However, the robot

is equipped with an additional pose estimation sensor, which is used to estimate the 3D position of the robot. Its position estimations are available using a Python API call to *getPose3d()* method.

This exercise has been developed to teach planning algorithms. As self localization is provided, efficient foraging algorithms, such as boustrophedon motion and A* search algorithm, are available choices . They minimize the revisiting of places and sweep the apartment more systematically and efficiently.

5 Conclusions

The lessons learnt with the previous release of the learning framework have motivated the transition from an open tool to an open web platform. First, the installation of the software dependencies has been greatly simplified as they are now all pre-installed in the Docker container. This makes simpler the use of the framework.

Second, the web browser is the only frontend for the student, who edits the robot program there and monitors its execution there as well. The new exercise templates are composed of a webpage inside the browser and a Python program inside the Docker container, mutually connected. The Python program is also connected to the robot simulator and runs the user's code for the robot's brain.

A webserver has also been developed and the new platform is ready to use by anyone at a web site, for free. The framework is truly multiplatorm as the container technology and the web browser used technologies are widely supported in the main operating systems.

Beyond the major framework upgrade, the exercises of the Intelligent Robotics course have been successfully migrated to the web platform and are ready to use by Robotics teachers and students around the world. An instance of this course is currently taking place with twenty students of the Rey Juan Carlos University.

Future lines for the new web platform: (a) include support for real robots; (b) introduce more educative contents such as computer vision course and industrial robotics course; (c) upgrade the infrastructure dependencies to ROS2 and Ignition simulator; and (d) perform an in-depth and empirical study with quantitative evidence of its impact in the learning process of real students.

Acknowledgements

This research was partially funded by the Community of Madrid within three projects: "UNIBOTICS2.0: Plataforma web educativa de programación de robots y visión artificial" inside a multiannual agreement with URJC, Encouragement of Young Ph.D. Students Investigation; "RoboCity2030—Madrid Robotics Digital Innovation Hub" (Ref. S2018/NMT-4331) ; and by the Madrid Regional Government through the project eMadrid-CM(S2018/TCS-4307). The later is also co-financed by the Structural Funds (FSE and FEDER). The authors also thank Google for funding the JdeRobot non-profit organization in its calls for Google Summer of Code 2015, 2017, 2018, 2019, and 2020.

References

1. Cañas, J. M., Martín, A., Perdices, E., Rivas, F., Calvo, R.. Entorno Docente Universitario para la Programación de los Robots. Revista Iberoamericana de Automática e Informática industrial, 15(4), 404-415 (2018) doi:10.4995/riai.2018.8962
2. Curto, B., and Moreno, V. (2016). Robotics in education. Journal of Intelligent and Robotic Systems, 81(1), 3-4.
3. Cheng, H. H., Chen, B., and Ko, D. (2009). Control System Design and Analysis Education via the Web. In Web-Based Control and Robotics Education (pp. 39-59). Springer, Dordrecht.
4. Bacca-Cortés, B., Florián-Gaviria, B., García, S., and Rueda, S. (2017). Development of a platform for teaching basic programming using mobile robots. Revista Facultad de Ingeniería, 26(45), 61-70.
5. Fabregas, E.; Farias, G.; Dormido-Canto, S.; Guinaldo, M.; Sanchez, J.; Dormido, S. Platform for teaching mobile robotics. J. Intell. Robot Syst. 2016, 81, 131–143
6. Esposito, J.M. The state of robotics education: Proposed goals for positively transforming robotics education at postsecondary institutions. IEEE Robot. Autom. Mag. 2017, 24, 157–164.
7. Erős, E., Dahl, M., Hanna, A., Götvall, P. L., Falkman, P., and Bengtsson, K. (2020). Development of an Industry 4.0 Demonstrator Using Sequence Planner and ROS2. In Robot Operating System (ROS) (pp. 3-29). Springer, Cham.
8. Marian, M., Stîngă, F., Georgescu, M. T., Roibu, H., Popescu, D., & Manta, F. (2020, October). A ROS-based Control Application for a Robotic Platform Using the Gazebo 3D Simulator. In 2020 21th International Carpathian Control Conference (ICCC) (pp. 1-5). IEEE.
9. Cañas, J. M., Martín-Martín, D., Arias, P., Vega, J., Roldán-Álvarez, D., García-Pérez, L., Fernández-Conde, J.. Open-Source Drone Programming Course for Distance Engineering Education. Electronics, 9(12), 2163 (2020). doi:10.3390/electronics9122163
10. Jara, C., Candelas, F., Pomares, J., Torres, F., 2013. Java software platform for the development of advanced robotic virtual laboratories. Computer Applications in Engineering Education 21, 14–30. DOI: 10.1002/cae.20542
11. Peidró, A., Reinoso, Ó., José, A. G., Marín, M., and Payá, L. (2016). A simulation tool to study the kinematics and control of 2RPR-PR Parallel Robots. IFAC-PapersOnLine, 49(6), 268-273.
12. Casan, G. A., Cervera, E., Moughlbay, A. A., Alemany, J., & Martinet, P. (2015, May). ROS-based online robot programming for remote education and training. In 2015 IEEE International Conference on Robotics and Automation (ICRA) (pp. 6101-6106). IEEE.
13. Karalekas, G., Vologiannidis, S., & Kalomiros, J. (2019, September). EUROPA—A ROS-based Open Platform for Educational Robotics. In 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 452-457). IEEE.
14. Cañas, J. M., Perdices, E., García-Pérez, L., Fernández-Conde, J. (2020). A ROS-Based Open Tool for Intelligent Robotics Education. Applied Sciences, 10(21), 7419 (2020). doi:10.3390/app10217419
15. Mahna, S., Roldán, D., Cañas, J.M., JdeRobot Robotics Academy: web based templates. ROSWorld2020 (2020). https://vimeo.com/480550758
16. Cervera, E., del Pobil, Ángel P., ROSLab: Sharing ROS Code Interactively With Docker and JupyterLab. IEEE Robotics Autom. Mag. 26(3): 64-69 (2019)