CEA Comité Español de Automática

XLV JORNADAS AUTOMÁTICA MÁLAGA 2024

UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Introduction to Control Education with the Unibotics web framework

García-Pérez, L.[a,*], Martín-Martín, D.[b], Cañas Plaza, J.M.[c], Chacón, J.[a], Roldán, D.[c]

[a]*Dpto. de Arquitectura de Computadores y Automática, Facultad de C.C. Físicas, Universidad Complutense, Plaza de las Ciencias 1, 28040 Madrid, Spain.*
[b]*Área de Tecnología Electrónica, Escuela Superior de Ciencias Experimentales y Tecnología, Universidad Rey Juan Carlos. 28933 Móstoles, Spain*
[c]*Dpto. de Sistemas Telemáticos y Computación, Universidad Rey Juan Carlos 28942 Madrid, Spain*

To cite this article: García-Pérez, L., Martín-Martín, D., Cañas, J.M., Chacón, J., Roldán, D. 2024.Introduction to Control Education with the Unibotics web framework.
Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10933

Abstract

This paper presents the Unibotics platform, a simulation framework for teaching robotics in university courses. It allows students to program robots from the web browser in Python, using current technologies such as ROS middleware and Gazebo robotics simulator. Unibotics provides an interface that hides the initial complexity of installing and dealing directly with ROS, Gazebo or MoveIt and allows students to focus on the algorithmic aspects of robotics. In addition, this article describes two teaching experiences of using Unibotics with real higher education students to introduce robot control. First, with Computer Science students in an elective Robotics course at U. Complutense Madrid. Second, with students of "Automation and Industrial Robotics" course of Industrial Electronics and Automation Engineering degree at U. Rey Juan Carlos. Both experiences have been successful and they will be continued in the next course.

Keywords: Internet based teaching of control engineering, E-learning in control engineering, Robotics technology, Robotic manipulators, Teaching industrial robotics.

1. Introduction

In the current landscape of engineering education, robotics emerges as a fundamental discipline that integrates knowledge of mechanics, electronics, computer science and automatic control. Teaching robotics to engineering students not only prepares future professionals to face contemporary technological challenges Shibata et al. (2021), but also fosters critical skills such as problem solving, logical thinking and creativity.

One of the crucial aspects of robotics education is the teaching of control of robotic systems Shibata et al. (2021). Control is the heart of robotics, as it enables robots to perform precise and complex tasks by manipulating their actuators in response to sensor feedback. This mastery of control is essential for students to understand not only how to design and build robots, but also how to ensure that they operate safely and efficiently in real-world environments.

Despite the importance of individual practices with complex real robots, these activities are notoriously difficult to implement in an educational environment. Real robots represent a significant investment in terms of cost and maintenance. In addition, the complexity of these systems requires constant supervision and a high level of technical knowledge on the part of instructors, which can be a challenge in classrooms with many students. These difficulties often limit students' ability to interact directly with real robots, which can restrict their practical understanding and their ability to apply abstract theories to real-world situations.

In this context, it is vital to explore and develop innovative pedagogical methods and tools that facilitate access to robotics for engineering students. This may include the use of advanced simulators Tellez (2017); Lopez-Nicolas et al. (2009), educational robotics platforms and hybrid approaches Dakeev et al. (2022) that combine theory with practice. In this way, it is possible to provide comprehensive training that prepares students for the demands of the labour market and the technological needs of today's society.

There are several private web online robotics training platforms; all of them use virtual laboratories. Robot Ignite

*Autor para correspondencia: liagar05@ucm.es

Academy from The Construct is based on ROS, Gazebo and Jupyter, and provides several short courses. Riders.ai provides online robotics courses with real-world applications, including drones, and exciting competitions that students can participate in. Users only need a browser to access the integrated development environment where they edit code and run simulations. RobotBenchmark from Cyberbotics provides free access to a series of robotics exercises based on Webots simulations which are run in the cloud.

Robots Formation Control Platform (RFCP) is a web-based interactive environment for experimentation with mobile robots Fabregas et al. (2016). RFCP has an interactive multi-robot simulator, an experimental environment for working remotely with real robots, namely the low-cost moway educational robots.

Another noteworthy initiative is the Robot Programming Network (RPN) Cervera et al. (2016); Casañ and Cervera (2018). It extends existing remote robot laboratories with the flexibility and power of writing ROS code in a Web browser and running it on the remote robot on the server side with a single click.

An additional web learning framework is Unibotics, which was born offline, then became ROS-based Roldán-Álvarez et al. (2021) and then online Roldán-Álvarez et al. (2023). It currently includes morethan 20 exercises and several courses, for instance about Intelligent Robotics and about Drones.

In this paper we give an overview of the platform and its application in two different teaching environments: in the optional subject of Robotics in the Computer Science degree at the Complutense University (UCM) and in the subject of Automation and Industrial Robotics of the Bachelor's degree in Industrial Electronics and Automation Engineering, at the Rey Juan Carlos University (URJC). It was used by more than 50 real students in total.

2. Unibotics framework

Unibotics is a robot programming web platform with engineering higher education contents Roldán-Álvarez et al. (2023). It allows the robot programming from the web browser and uses state-of-the-art robotics tools such as ROS middleware and Gazebo simulator.

Currently ROS (Robot Operating System) middleware Quigley et al. (2015); Macenski et al. (2022) has become the de facto worldwide standard in the robot programming community. It proposes a distributed component-based paradigm for robotics applications, they are composed of several ROS nodes which interoperate among them through typed messages called ROS topics. The ROS ecosystem includes a large community, several tools (Rviz, ROSbags, etc.), a collection of drivers and software pieces ready to reuse (Navigation stack, MoveIt stack for industrial robots, MAVROS for drones, etc...). It has fostered the software reuse in robotics.

ROS is increasingly being incorporated into more and more university courses. However, starting to work with ROS requires a considerable effort that is not feasible in many university robotics courses, either because robotics is an elective subject or because students do not have the necessary computer skills to tackle it in a single semester.

Using Unibotics, almost no installation is required from Windows, Linux and MacOS computers. More than 20 exercises are currently available on different robotics topics: service robotics, autonomous driving, drones, computer vision, mobile robots and industrial manipulators.

The platform follows the BBS approach: robot Brain, robot Body and Scenario. For each exercise Unibotics already provides the Scenario, the robot Body and the task to be solved (such as cleaning a full room with a vacuum cleaner). The robot Brain has to be programmed in Python language by the students, and it is connected to the robot Body which includes sensors and actuators that are usable through ROS topics.

The main components of Unibotics are: (1) the Robotics Academy Docker Image, which includes all the robotics software already preinstalled; (2) the HTML web page for each exercise, which is used as the frontend for robot programming editor and as the Graphical User Interface of execution monitoring; and (3) the web backend with the webserver that interconnects everything and the databases used to store the usage information. The web page of each exercise is provided by the webserver, it includes an inline text editor for the student and connects to the docker image where the robot application developed by the user is run.

2.1. Robotics Academy Docker Image

The robotics software is inherently complex, it includes many components, dependencies, libraries, etc. Before developing any robot application, many dependencies need to be installed in the student environment. Usually, when we want to develop software to use robot sensors and actuators, we need first to install their drivers. Sometimes robotics students are focused in learning how the drivers work, but most times the focus is on robotics algorithms and their implementation. In this second case, it is really common for the students to spend some sessions installing all the dependencies. In educational settings, time is a valuable resources and teachers usually want the students to work as much as possible without wasting time on non-educational purposes. Usually some level of expertise is required in order to install the robotics dependencies locally and not all of them are available in all Operating Systems.

To avoid installation issues and to reduce the time-to-start-robot-programming for the students we created the Robotics Academy Docker Image (RADI). This RADI includes all the dependencies needed for the exercises that will be provided by the Unibotics platform. It can be easily installed in Windows, Linux or MacOS, providing this way support to many operative systems (OS). With the RADI the user only needs to install Docker and run the container. It manages all the required robotics software inside a Linux based container. Both the robot simulator and the students' code run inside it, in the user's computer. This approach follows some of the ideas in ROSLab Cervera and Del Pobil (2019) which uses ROS, Jupyter Notebooks and Docker containers for providing reproducible research environments.

This way the learning curve is reduced for final users, who only need to focus on programming the algorithms. In this scenario, the user will need to log in the platform through any browser, run an exercise and all the connections with the RADI will be established in order to launch the dependencies of that exercise and allow the user to send code to the robot

directly. In addition, being the users' computers the ones running the simulation, the number of users can grow without affecting the performance of Unibotics platform itfself. The simulator runs headless inside the container, regardless the operating system of that computer. The Gazebo simulated world is displayed at the browser through an `gzclient` and a VNC connection with a VNCserver, which also runs inside the container. Once an exercise is requested, the **exercise.py** script of that exercise starts running. That script is connected to the Gazebo simulator to get the sensor readings and to set the motor commands. It runs the source code of the robots' brain received from the browser. The RADI and the web platform are connected through several websockets.

2.2. *Exercise frontend*

Each exercise has its own HTML webpage. Through the browser the user can edit the code and send it to the RADI in order for the robot to execute it. The RADI will send back to the browser debugging and visual information, where it will be displayed.

The web page of each exercise is divided in three main parts as it is shown in Figure 1. In the left side an inline text editor is be used to write the robot program. The right side is a view of the simulated world: the scenario. There is also a debugging console so that the user can print text messages to debug the exercise code. Some exercises have specific widgets, for example of camera outputs, which also help students in the development of the code. Then in the top area of the web page there is a toolbar which is used to implement the basic operations of the platform: saving the current code, loading the code into the robot, running the simulation, resetting it, showing and hiding some GUI widgets and two buttons to evaluate the efficacy of the code and the programming style.

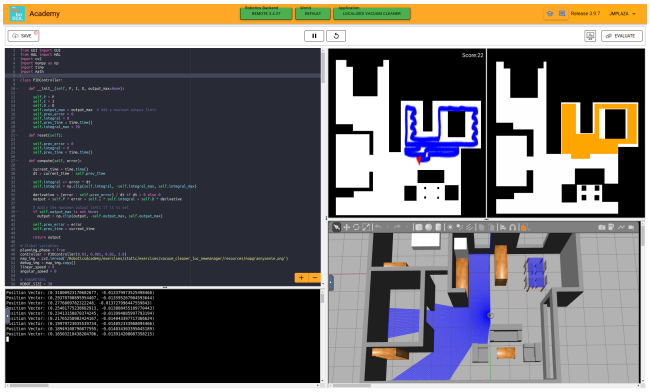

Figure 1: Example of the Vacuum Cleaner exercise web page

2.3. *Webserver backend*

The webserver provides the web pages of the available exercises and also to records information about user activity in the platform. It has been developed using the Python-based Django framework, Fig. 2. The structural information of the platform and the users' interactions are stored in a PostgreSQL database. In addition, the users' code is stored in the Amazon Simple Storage Service (Amazon S3). The webserver loads the user's code each time the user enters into an exercise, and it saves the code automatically every 5 minutes or when the user orders it.

Typically the RADI runs in the user computer. Another possibility, which is the one used in the URJC is the use of a backend computer farm with 120 computers running the RADI so the students do not have to install anything at all in their computers, therefore allowing them to directly focus on programming robots. Once a student connects to an exercise, the webserver connects the student browser automatically with one of the available computers. The IP address of the computer to which the student is connected to remains hidden, since all the required communications (sending code, receiving images, etc) occur through the webserver itself.

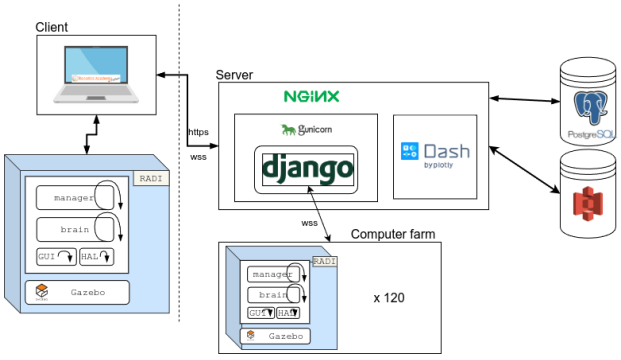

Figure 2: Unibotics architecture

3. PID control using a visual follow line exercise

The Robotics course at UCM's Faculty of Computer Science is an optional course offered to third and fourth year students of Software Engineering, Computer Engineering and Video Game Development. The aim of the course is to introduce some of the basic concepts of robotics to students with no previous knowledge of robotics, control neither sensors. As an optional subject offered at different degrees, students may come from a wide variety of backgrounds. Coming from computer science backgrounds, all students have programming skills but no knowledge at all about control theory, and only some of them have basic knowledge of electronics.

The Robotics course is taught in the autumn term with two weekly sessions of 1h40m each. One of these sessions is taught in the laboratory, Fig. 3. Traditionally, the practice of this subject has only focused on assembling and programming a do-it-yourself small mobile robot. Working with the robot is one of the main attractions of the subject and students appreciate learning how to assemble and program the hardware, despite the heavy workload.

29 students were enrolled in the Autumn 2023-24 course. In this semester the robot assembly and programming exercises were completed with 3 programming exercises inside the Unibotics framework: Formula 1 "Follow Line", "Basic Vacuum Cleaner" and the "Obstacle Avoidance". These exercises allow students to program more complex robotic behaviours than those allowed by the hardware they have for the physical robot.

Figure 3: UCM Robotics Class

One of the first concepts introduced in the course is closed-loop control. As these are students who have no previous training in control, signal processing or linear systems, the aim is not so much for them to learn control as for them to understand in broad terms what closed-loop control means and the need for feedback. Without control, there is no way to approach autonomous behaviour in robotics.

The visual feedback exercise "Follow Line" inside Unibotics, Fig. 4, has been used for the didactic purpose of introducing students to the concepts of feedback control and PID controllers and their use in robotics. The students' task in this exercise is to program an algorithm that will allow the simulated Formula 1 car to complete the circuit without going off the track. For this purpose, the track is marked with a red line, so that simple visual feedback can be used to perform the control.

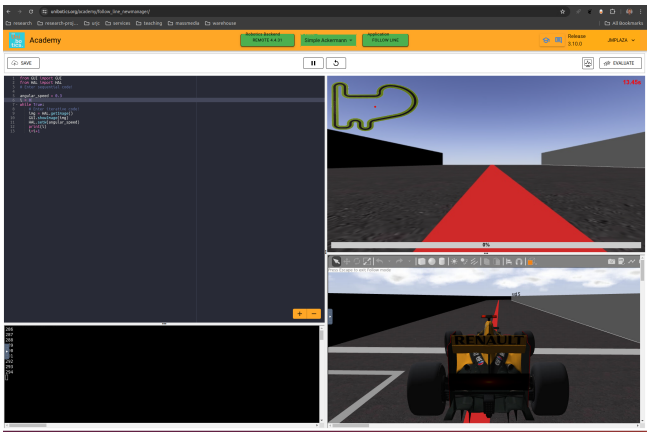

Figure 4: Unibotics Follow Line Exercise interface

Before starting the practice in a theoretical lesson, the basic concepts of closed-loop control, setpoint, setpoint tracking error and PID controller had been worked on in the classroom. Since the students had no prior knowledge of machine vision algorithms and the aim was to work on the control part, the students were provided with the necessary image processing code. Using the supplied functions, the image is captured, segmented to obtain the red line and the position in image coordinates of the centre of the line is returned.

The students' task is to use these functions to obtain feedback about the relative position of the car with respect to the red line marking the centre of the lane and using that feedback design, implement and test a PID controller for the robot angular velocity ω.

$$u = K_p \cdot e + K_i \int_0^t e dt + K_d \cdot \frac{d}{dt} e(t) \tag{1}$$

For this, a basic tuning process is suggested:

1. Program an error-proportional control. Find a value of the constant K_p that makes the vehicle oscillate around the red line.
2. Add an integral term and adjust the constant K_i so that the robot stops oscillating around the centre line
3. Add a derivative term to the controller, adjusting the K_d constant so that the response is smooth.

All the students managed to program a controller that made their robots complete the circuit. 7 of them also successfully tested the controller with other available circuits in this Unibotics exercise. One student also implemented a proportional-to-slice controller to vary the linear velocity in addition to the one requested for the angular velocity. The most common problem encountered when correcting practice is that some students have not actually programmed the requested PID controller, but have made a simple case-based control.

In general, students were very positive about being able to program more complex robots in a simple way. Some of the students' comments along these lines were: "easy implementation of test code without having to install ROS", "the platform allows you to learn to program and simulate a wide variety of things. The tutorials are very useful" or "the platform is great". Negative comments focus on the difficulty of controlling the cameras and the map.

From the teacher's point of view, we consider the practices carried out with Unibotics an essential complement to the course. In addition, the tutorials and teaching units developed on the platform greatly facilitate the work of preparing classes. However, given the students' limited prior training in robotics, it is necessary to dedicate more face-to-face time in class. Next year we will maintain the practices with Unibotics but we will dedicate more time to them. Specifically, in the case of the FollowLine practice, more time will be devoted in the theory class to explain the fundamentals and to carrying out some more examples.

4. Pick and place with industrial robotic manipulators using ROS and MoveIt

The Unibotics platform has also been used for teaching an introductory seminar on industrial robotic manipulators with ROS and Moveit. The seminar was attended by 24 students (see Fig. 5). It was conceived as a complementary activity to the subject of *Automation and Industrial Robotics*, taught at the third year of the Bachelor's degree in Industrial Electronics and Automation Engineering, at the Rey Juan Carlos University.

In the first part of this subject students learn the principles and applications of industrial process automation, as well as the operation of Programmable Logic Controllers (PLC), their

hardware architectures and their programming fundamentals. In the second part, the course reviews the morphology, configuration and applications of an industrial robot, as well as the different components of an complete industrial robotic station: end effectors, power, control and communication subsystems, safety considerations, etc. The subject also covers the study of the direct, inverse and differential kinematics of the robotic manipulator.

In addition, several laboratory sessions are carried out. In them, students learn to program real industrial hardware, including SIEMENS SIMATIC S7-1200 PLCs, FESTO MPS203 production lines and both traditional (ABB IRB120) and collaborative (ABB CRB15000) industrial robots, with both static tools and grippers. All the software programming is done using proprietary industrial frameworks (SIEMENS Totally Integrated Automation Portal and ABB Robotstudio).

Figure 5: Seminar on industrial manipulators with Unibotics, URJC

However, the official syllabus of the course does not cover programming using tools and simulators based on open source software, as ROS, MoveIt or Gazebo. The advantages of this alternative approach are multiple and complementary to what has been seen in the course, since it allows: i) the use of conventional programming languages (C++ and/or Python), ii) apply the code to almost any commercial hardware brand, iii) use several trajectory planning algorithms, iv) a fast and easy integration of computer vision and/or artificial intelligence tools, and v) the extension to mobile robotic manipulators.

Therefore, the goal of the seminar was to complete a pick and place exercise with a simulated ABB IRB120 6-axis angular robotic arm and a Robotiq 2f-85 two-finger articulated gripper, classifying different objects initially placed in a static conveyor (see Fig. 6). The objects could be classified by color or shape using four colored drawers. To complete the task, students were provided with a Python API including key components to command the robot with pose messages to make its end effector frame move to the desired pose via inverse kinematics, to generate MoveIt grasp messages, open and close the gripper, send the robot back to the Home pose, etc. In addition, the ROS 3D graphical visualizer (RViz) could be used to preview the complete robot trajectory before each motion execution.

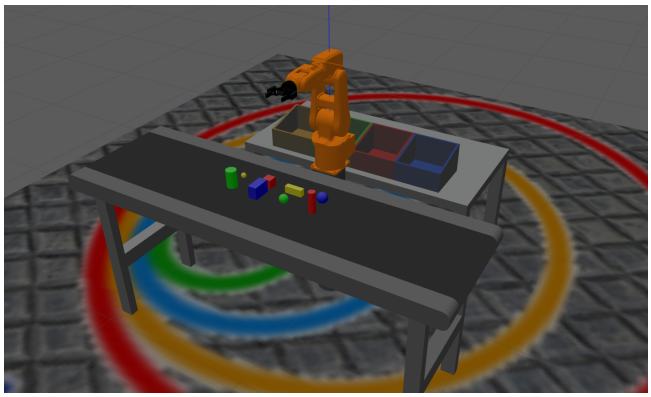

Figure 6: Gazebo world of the pick and place exercise in Unibotics

Regarding the seminar results, it is worth noting that the use of the Unibotics platform allowed a much more efficient use of the time available, as none of the required software components had to be installed. In general, all the students showed great interest in this open source programming alternative and, though only a few of them completed the exercise, they all indicated that they would continue to work on it on their own and become a little more familiar with the use of ROS and MoveIt for programming industrial robots.

5. Conclusions

In this article we have presented the Unibotics platform, a simulation framework for teaching robotics in university courses. Unibotics allows web-based programming and offers the possibility for students to deal with complex environments and robots. Together with the brief description of the platform, two very different didactic experiences are presented. The first case explains the development of an introductory control practice for Computer Science students at the Complutense University in a robotics elective course using the Unibotics line-following exercise. The second experience is an introductory seminar on programming industrial robots with ROS and MoveIt as part of the subject *Automation and Industrial Robotics*, taught at the third year of the Bachelor's degree in Industrial Electronics and Automation Engineering, at the Rey Juan Carlos University.

In both cases the use of Unibotics has allowed students to acquire basic robotics knowledge, i.e. PID controller and programming using ROS and MoveIt, which without the web framework provided by Unibotics would have been impossible to address in both courses. In addition, the use of the simulation environment has allowed the students to work individually, which would otherwise have been completely impossible due to the cost of the robots that have been programmed.

We currently have three lines of work in progress. On the one hand, the improvement of the Unibotics platform, migrating all practices to ROS2 and generating new scenarios, robots and exercises that expand its possibilities. Second, we are working on including gamification aspects into Unibotics, such as robotics competitions Fernández-Ruiz et al. (2022), and on including an intelligent tutor to help the teacher in classroom management answering text-based student questions. Third, we are trying to expand the community of users

to other Spanish universities. A large community of open source developers contributes to keep alive and constantly improves this platform that we consider to be very useful for both control students and teachers.

Acknowledgments

This work is partially supported by INSERTION projec t (PID2021-27648OB-C33) of the Knowledge Generation program of the Ministry of Science and Innovation, and by the UNIBOTICS-GAM project Ref. TED2021-132632B-I00, from Proyectos de Transición Ecológica y Transición Digital 2021 call Spain. Authors also appreciate the help of Google for improving RoboticsAcademy through the Google Summer of Code program since 2017.

References

Casañ, G. A., Cervera, E., 2018. The experience of the robot programming network initiative. Journal of Robotics 2018.

Cervera, E., Del Pobil, A. P., 2019. ROSlab: Sharing ROS code interactively with docker and jupyterlab. IEEE Robotics and Automation Magazine 26 (3), 64–69.
DOI: 10.1109/MRA.2019.2916286

Cervera, E., Martinet, P., Marin, R., Moughlbay, A. A., Del Pobil, A. P., Alemany, J., Esteller, R., Casan, G., 2016. The robot programming network. Journal of Intelligent & Robotic Systems 81 (1), 77–95.

Dakeev, U., Pecen, R. R., Yildiz, F., Basith, I. I., Obeidat, S. M., Sowell, L. E., May 2022. Development of virtual reality robotics laboratory simulation. In: 2022 ASEE Zone IV Conference. No. 10.18260/1-2–44729. ASEE Conferences, Vancouver, https://peer.asee.org/44729.

Fabregas, E., Farias, G., Dormido-Canto, S., Guinaldo, M., Sánchez, J., Dormido Bencomo, S., 2016. Platform for teaching mobile robotics. Journal of Intelligent & Robotic Systems 81, 131–143.

Fernández-Ruiz, R., Palacios, D., Cañas, J., Roldán, D., 2022. Automatic Competitions in the Unibotics open online robot programming web. Proceedings of Robot2022.

Lopez-Nicolas, G., Romeo, A., Guerrero, J. J., 2009. Simulation tools for active learning in robot control and programming. In: 2009 EAEEIE Annual Conference. pp. 1–6.
DOI: 10.1109/EAEEIE.2009.5335490

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7 (66), eabm6074.

Quigley, M., Gerkey, B., Smart, W. D., 2015. Programming Robots with ROS: a practical introduction to the Robot Operating System. " O'Reilly Media, Inc.".

Roldán-Álvarez, D., Cañas, J. M., Valladares, D., Arias-Perez, P., Mahna, S., 2023. Unibotics: open ros-based online framework for practical learning of robotics in higher education. Multimedia Tools and Applications.
DOI: 10.1007/s11042-023-17514-z

Roldán-Álvarez, D., Mahna, S., Cañas, J. M., 2021. A ROS-based Open Web Platform for Intelligent Robotics Education. International Conference on Robotics in Education (RiE). Springer, pp. 243–255.

Shibata, M., Demura, K., Hirai, S., Matsumoto, A., 2021. Comparative study of robotics curricula. IEEE Transactions on Education 64 (3), 283–291.
DOI: 10.1109/TE.2020.3041667

Tellez, R., 2017. A thousand robots for each student: Using cloud robot simulations to teach robotics. In: Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R. (Eds.), Robotics in Education. Springer International Publishing, Cham, pp. 143–155.