Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 20 (2023) 124–136

www.revista-riai.org

Un laboratorio remoto de código abierto y bajo coste para el brazo robótico educativo Dobot Magician

Jesús Chacon*, Daniela Goncalves, Eva Besada, Jose Antonio López-Orozco

Departamento de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid, Plaza de las Ciencias 1, 28040, Madrid, España.

To cite this article: Chacón, J., Goncalves, D., Besada, E., López-Orozco, J. 2023. A low-cost open-source remote laboratory for the educational robot arm Dobot Magician. Revista Iberoamericana de Automática e Informática Industrial 20, 124-136. https://doi.org/10.4995/riai.2023.17477

Resumen

Este artículo presenta el laboratorio remoto diseñado en la Universidad Complutense de Madrid (UCM) para dar acceso remoto, a través de Internet, al robot educativo Dobot Magician. El software del laboratorio remoto está formado por el servidor web *ReNoLabs*, que gestiona el acceso al laboratorio, despliega sus páginas web, y sirve como pasarela de comunicación entre el software de control que interactúa directamente con el robot y la interfaz web de la experiencia. *ReNoLabs* está programado en Node.js, el software de control en Python y la interfaz web en Easy JavaScript Simulations (EJsS). EJsS también se utiliza para gestionar de forma centralizada el laboratorio, al haber ampliado su funcionalidad mediante un *"Plugin"*. Además, el software anterior, creado con herramientas software gratuitas, se ejecuta sobre una Raspberry Pi y la interfaz web de la experiencia puede integrarse, si así se desea, en un sistema de gestión de aprendizaje general como Moodle. Finalmente, el artículo también presenta un par de ejemplos de uso del laboratorio remoto.

Palabras clave: Robótica Educativa, Laboratorios Remotos, Brazo Robótico, Programación de Robot, EJsS

A low-cost open-source remote laboratory for the educational robot arm Dobot Magician

Abstract

This article presents the remote laboratory designed at the Complutense University of Madrid (UCM) to provide remote access, through the Internet, to the educational robot Dobot Magician. The software of the remote lab consists of the *ReNoLabs* web server that manages the access to the lab, displays its web pages, and serves as a communication gateway between the control software that interacts directly with the robot, and the web graphical interface of the experience. *ReNoLabs* is programmed in Node.js, the control software in Python and the graphical interface in EJsS. EJsS is also used to centrally manage the lab, after expanding its functionality through a *"Plugin"*. In addition, the above software, created with free software tools, runs on a Raspberry Pi and the web interface of the experience can be integrated, if desired, into a general learning management system such as Moodle. Finally, the article also presents a couple of practical examples of the use of the remote laboratory.

Keywords: Robotics Education, Remote Laboratory, Robotic Arms, Robot Programming, EJsS

1. Introducción

Las prácticas de laboratorio permiten a los alumnos de Ciencias Experimentales e Ingeniería enfrentarse a problemas reales, que 1) ilustran aspectos relevantes de las clases teóricas y 2) les transmiten habilidades y métodos de trabajo útiles para su futuro laboral (Ma and Nickerson, 2006). Sin embargo, estos beneficios se encuentran limitados, en numerosas ocasiones, por 1) el coste económico que implica la puesta a punto de múltiples puestos de laboratorio donde los alumnos puedan rea-

* Autor para correspondencia: jeschaco@ucm.es

lizar simultáneamente la misma experiencia y 2) por el número de experiencias que pueden realizarse durante el horario correspondiente a las sesiones de prácticas (Gomes, 2009).

Los laboratorios remotos, que dan acceso a los dispositivos del laboratorio a través de Internet, mitigan los inconvenientes anteriores, al aumentar la disponibilidad, accesibilidad y seguridad del material de las prácticas. Más en detalle, permiten que el estudiante, desde su casa, interactúe y modifique el comportamiento del sistema bajo estudio, observe y mida la evolución de sus variables de interés, y pueda repetir o llevar a cabo experiencias adicionales a las disponibles durante los horarios de los laboratorios presenciales. También conviene destacar diferentes estudios que muestran que los laboratorios presenciales y remotos pueden aportar beneficios educativos similares (de la Torre et al., 2016; Faulconer and Gruss, 2018; Kostaras et al., 2011; Nickerson et al., 2007) y resaltar que las restricciones impuestas en el ámbito educativo durante la pandemia de la COVID-19 ha estimulado, de forma repentina y sin precedentes, la creación de experiencias remotas similares a las de numerosos laboratorios presenciales pre-pandémicos (Gamage et al., 2020; Bhute et al., 2021). Este trabajo recoge el fruto del esfuerzo realizado en este contexto por un grupo de investigadores de la Universidad Complutense de Madrid (UCM) para que los alumnos de *Robótica* de diversas titulaciones de sus Facultades de Informática y Ciencias Físicas puedan interactuar, remotamente, con alguno de los dispositivos de sus laboratorios presenciales.

La *Robótica* es un área tecnológica relevante para el sector industrial y de servicios, que durante los últimos 20 años ha ido ganando protagonismo en el ámbito educativo ya que promueve los procesos prácticos de aprendizaje, motiva y despierta el interés de los estudiantes, y puede ser utilizado con fines didácticos en la enseñanza primaria, secundaria y universitaria (Merdan et al., 2020). La importancia de la *Robótica Educativa* es tal, que es considerada una subdisciplina de la *Robótica*, centrada en la concepción, creación y puesta en funcionamiento, con fines didácticos, de objetos tecnológicos que reproducen los procesos y herramientas robóticas que son utilizados de forma habitual en nuestro entorno social, productivo y cultural (Papadakis and Kalogiannakis, 2020).

La incorporación progresiva de currículo de *Robótica* en todos los niveles educativos se encuentra favorecida por la aparición de un amplio abanico de plataformas comerciales desarrolladas para este fin, entre las que podemos mencionar los robots manipuladores de Dobot (Dobot Webpage, 2022), Cyton (Cyton Webpage, 2022), Owi (Owi Webpage, 2022) y Scorbot (Scorbot Webpage, 2022); los robots humanoides NAO (Nao Webpage, 2022) y Pepper (Pepper Webpage, 2022); o los robots móviles sobre ruedas de Lego Mindstorms (Mindstorms Webpage, 2022) y Moway (Moway Wepbage, 2022). Entre sus ventajas se encuentra su comercialización como soluciones educativas completas, que incluyen, además del hardware y el software necesario para que los alumnos puedan interactuar con ellos y programar su comportamiento, material didáctico adicional para ser utilizado directamente por todos los agentes educativos. Sin embargo, sus herramientas software no suelen admitir la interacción con los robots a través de Internet, motivo por el que una parte de la comunidad educativa ha desarrollado laboratorios remotos propios sobre alguna de estas plataformas (Jara et al., 2011; Chaos et al., 2013; Angulo et al., 2017; Losada-Gutiérrez et al., 2020). Este trabajo presenta uno de estos desarrollos, en concreto, el realizado en la UCM para el brazo robótico Dobot Magician.

Los laboratorios remotos de robótica educativa comparten necesidades, características y elementos similares a los desarrollados en otras áreas. Más concretamente, desde el punto de vista software necesitan, tal y como se esquematiza en la Figura 1, una *interfaz gráfica de la experiencia*, que permite que los alumnos interactúen con el robot y monitoricen su comportamiento desde sus computadores o dispositivos inteligentes; un *programa de control*, que traduce y ejecuta sobre el robot las ordenes enviadas por el alumno desde la interfaz gráfica; y uno o varios *servidores*, que controlan el acceso de los alumnos al laboratorio y sirven de pasarela de comunicación entre la interfaz gráfica y el programa de control. Además, sus necesidades hardware incluyen los robots para los que han sido diseñados, y los equipos sobre los que se despliegan el programa de control y los servidores involucrados en el laboratorio. A este respecto, la Tabla 1 muestra las características más relevantes de los elementos involucrados en diferentes laboratorios remotos que han sido descritos en los trabajos de la literatura que se indican en la primera columna, organizados según el tipo de robot para el que han sido desarrollados y su fecha de publicación. Los espacios en blanco se deben a que no existe información suficiente en la publicación para indicar el hardware/software que soporta el elemento correspondiente, aunque este exista. Además, para facilitar la comparación de las características de los elementos de los diferentes laboratorios, se utiliza cursiva para resaltar los elementos similares el nuestro (mostrado en la última fila). En ella se observa que 1) los laboratorios remotos presentan bastante variedad respecto al tipo, plataforma y programa de control del robot, que 2) la información del servidor no siempre está disponible, y que 3) que la interfaz gráfica de las experiencias más antiguas/modernas son habitualmente applets/páginas web desarrollados con EJS/EJsS.

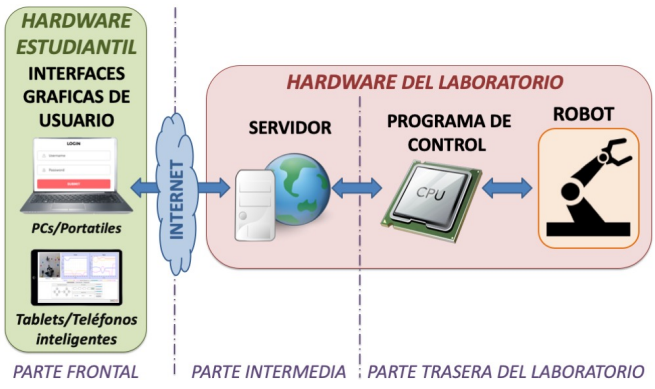


Figura 1: Relación entre los elementos de un laboratorio remoto genérico.

La propuesta de este trabajo, caracterizada en la última fila de la tabla, comparte con otros trabajos la herramienta de desarrollo de la interfaz gráfica de la experiencia (EJsS) y utiliza una Raspberry Pi, un ordenador monoplaca de bajo-coste, para desplegar el software de control (programado en Python) del Dobot Magician y el servidor de páginas web de laboratorio. Este último, programado en Node.js, es una versión actualizada del servidor *ReNoLabs*, utilizado para desarrollar los laboratorios remotos de Control de Sistemas de la UCM (Bermudez-Ortega

Tabla 1: Laboratorios remotos de robótica educativa

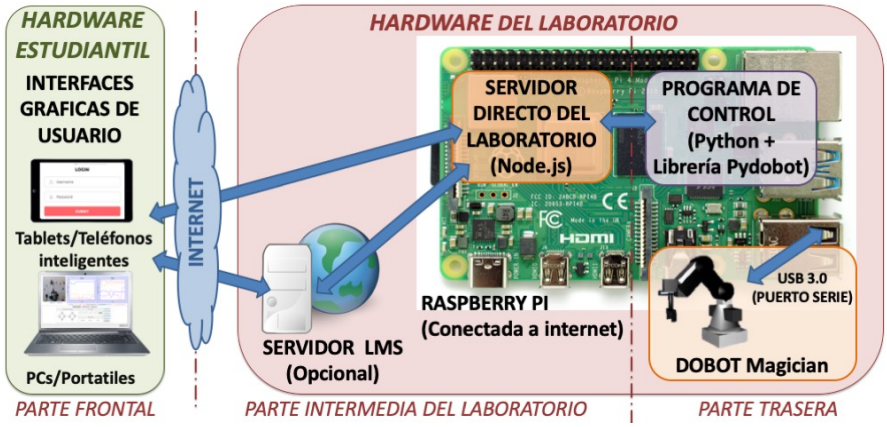
Tipo de robot	Trabajo	Robot	Programa de control	Servidor	Interfaz Gráfico
Móvil	Chaos et al. (2013)	LEGO NTX	LabView	Matlab+JIL	EJS applet
	Fabregas et al. (2016)	Moway	Visual C# Gateway		EJS applet
	dos Santos Lopes et al. (2017)	Propio	Arduino	Java+*Node.js*	
Manipulador	Marín et al. (2005)	Mentor	Mentor		Java applet
	Jara et al. (2011)	Scorbot	Scorbot		EJS applet
	Vagaš et al. (2016)	Almega	Arduino	Moodle	EJS applet
	Jiménez et al. (2018)		Arduino		Aplicación C
	Saenz et al. (2020)	Propio	Arduino	RIP Server	*EJsS webpage*
	Este trabajo	Dobot	Python + Rasp	*Node.js* (+ Moodle)	*EJsS webpage*

et al., 2016a, 2017). En su versión actual, el servidor no solo gestiona el control de acceso al laboratorio y sirve como pasarela entre su interfaz gráfica y el programa de control del robot, sino que también incorpora nuevos protocolos estandarizados de comunicación y da soporte a la gestión centralizada del software del laboratorio desde EJsS. Por lo tanto, EJsS, además de seguir siendo utilizado como la herramienta con la que se diseña la interfaz gráfica de la experiencia, permite, gracias al uso de un *"Plugin"* (Aizpuru-Rueda et al., 2019; Chacón et al., 2021) especialmente desarrollado para este fin, configurar diferentes aspectos del laboratorio remoto del Dobot.

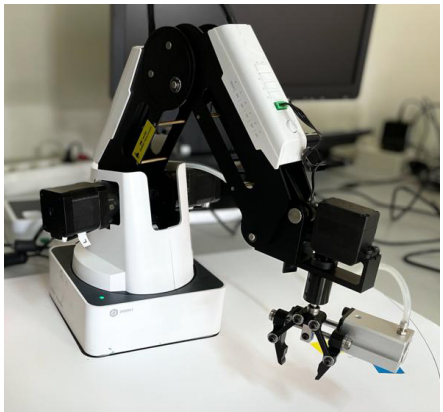
Las principales ventajas de nuestro laboratorio frente a la mayoría de laboratorios existentes son 1) la posibilidad de trabajar con el Dobot mediante elementos interactivos o el lenguaje de programación secuencial de instrucciones *Advanced Control Language* (ACL); 2) una flexibilidad de uso que permite desplegarlo de forma autónoma o integrado en un sistema de gestión de aprendizaje (LMS, del inglés Learning Management System); 3) un ciclo de desarrollo/despliegue/mantenimiento completamente integrado en una única herramienta de código abierto; y 4) una versatilidad que favorecerá su futura adaptabilidad a otros brazos robóticos. En este punto también conviene recalcar dos aportaciones adicionales de este trabajo, que además de presentar el nuevo laboratorio remoto del Dobot Magician, mejora la plataforma ReNoLabs mediante la incorporación un nuevo componente de conexión con ZeroMQ y el intérprete del lenguaje de programación ACL.

Finalmente, cabe destacar que las características más relevantes de la primera versión del laboratorio remoto que se presenta en este artículo han sido publicadas en (Goncalves-López-Medrano et al., 2021). La versión del laboratorio remoto del Dobot Magician que se describe a continuación presenta dos novedades: una interfaz gráfica renovada y el módulo de programación de secuencias de instrucciones. Además, en este artículo se describe el laboratorio remoto con mayor detalle, se presentan nuevas experiencias educativas realizables con él, y se recogen los primeros resultados de su uso por parte de los docentes de las asignaturas.

La organización de este artículo es la siguiente. En la Sección 2 se introducen las características más relevantes de los elementos software (Python, Node.js, EJsS y diferentes librerías de comunicación) y hardware (el robot manipulador y la Raspberri Pi) sobre los que se sustenta el laboratorio. En la Sección 3 se describe el funcionamiento de los elementos software del laboratorio remoto del Dobot, haciendo especial hincapié en aquellos desarrollados específicamente para este laboratorio remoto (i.e. en el programa de control del Dobot, el servidor del laboratorio, la interfaz gráfica de la experiencia y la máquina virtual para la programación del Dobot mediante ACL. En la Sección 4 se presentan dos ejemplos prácticos de uso del laboratorio para alumnos de diferentes niveles educativos: una práctica relacionada con el estudio de la cinemática de los robots manipuladores y otra relacionada con la programación de comportamientos automáticos. En la Sección 5 se presenta un análisis preliminar de la viabilidad del laboratorio remoto como

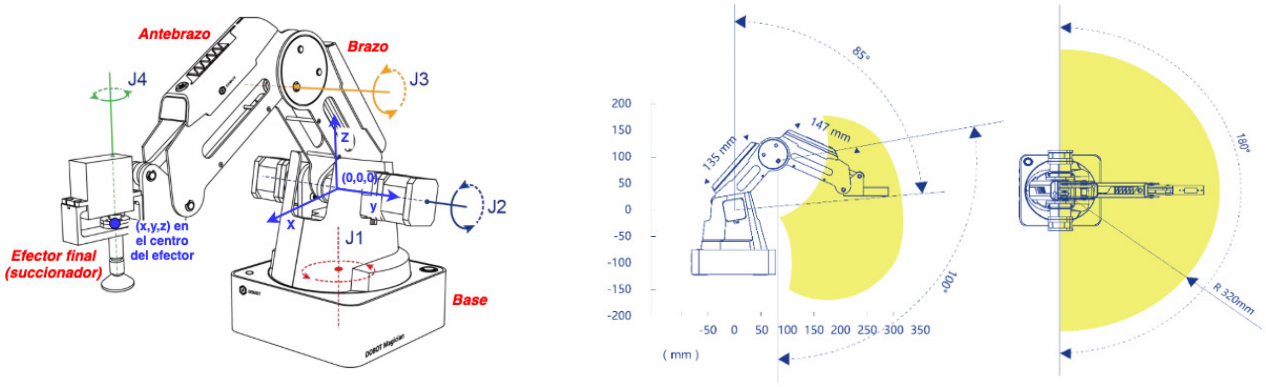


(a) Relación entre los elementos del laboratorio remoto



(b) Imagen del Dobot Magician (con pinza)

Figura 2: Elementos de nuestro laboratorio remoto



(a) Elementos, ejes articulares y ejes cartesianos

(b) Espacio de trabajo

Figura 3: Características del Dobot Magician, imágenes tomadas de (Dobot Webpage, 2022).

herramienta educativa. Finalmente, en la Sección 6 se extraen las conclusiones del trabajo realizado y se exponen una serie de actividades futuras.

2. Herramientas que dan soporte al laboratorio remoto

Nuestro laboratorio remoto está formado por los elementos hardware y software que se muestran en la Figura 2(a), y que incluyen el brazo robótico Dobot Magician; el programa de control que interactúa directamente con él y que está escrito en Python; el servidor de páginas web del laboratorio programado en Node.js; la Raspberry Pi donde se ejecutan el programa de control y el servidor anteriormente mencionados, y a cuyo puerto USB se conecta el Dobot; la interfaz gráfica de las experiencias que se ejecuta en los equipos de los estudiantes; y opcionalmente, un servidor adicional sobre el que se ejecuta el sistema de gestión de aprendizaje (LMS, del inglés Learning Management System) general de la asignatura, en el que se puede integrar, si así se desea, este laboratorio remoto.

En esta sección se describen las características más relevantes del hardware y software anteriormente mencionado.

2.1. *El robot manipulador Dobot Magician*

El Dobot Magician (Dobot Webpage, 2022), mostrado en la Figura 2(b), es un pequeño brazo robótico que está formado por una base, un brazo, un antebrazo y, opcionalmente, un efector final (a elegir entre pinza o succionador). Además, cuenta con tres articulaciones de revolución (denominadas J1, J2, J3 en la Figura 3(a) y que le proporcionan el rango de trabajo observable en la Figura 3(b)) y opcionalmente, cuando se le conecta el efector final, con una articulación de revolución adicional (denominada J4). La posición 3D del efector final se mide respecto el origen de coordenadas de sus ejes cartesianos que, tal y como se muestra en la Figura 3(a), se sitúa a la altura de la segunda de las articulaciones. Finalmente, también permite la conexión de señales digitales o analógicas externas, por lo que se le puede conectar algún sensor para la toma de decisiones durante los experimentos realizados con el brazo robótico.

El software interno del Dobot permite moverlo de tres formas diferentes. Primero, el modo *jogging* lo desplaza incrementalmente según cada una de las direcciones del eje cartesiano o de los ejes articulares. Segundo, el modo *PTP* (del inglés *Point-To-Point*) lo desplaza hasta el punto destino seleccionado, incrementando los valores de los ejes articulares (*MOVJ - MOVe Joint*), siguiendo una linea recta (*MOVL - MOVe Line*), o uniendo tres lineas recta de forma que la primera y tercera aumenten/disminuyen la altura del efector final para que, mediante el salto (*JUMP*) definido por ellas, el Dobot pueda esquivar un objeto. Tercero, el modo *ARC* (arco) lo desplaza siguiendo un arco definido por la posición inicial, un punto intermedio y el punto destino. Finalmente, conviene indicar que todos estos modos de funcionamiento son accesibles desde una aplicación de control interactiva propia y desde una Interfaz de Programación de Aplicaciones (API), ambas proporcionadas por el fabricante con el objeto de que el Dobot, conectado a otro equipo a través de un puerto USB 3.0, pueda ser manipulado por estudiantes de diferentes niveles educativos. Además, el desarrollo de nuevas aplicaciones de control del Dobot se ve favorecido por su accesibilidad desde diferentes lenguajes/entornos de programación (e.g. C, Python, ROS, Qt, Matlab, Labview, Android, Blockly).

2.2. *La Raspberry Pi: el soporte computacional principal*

La Raspberry Pi (Raspberry PI Webpage, 2022) es un ordenador monoplaca, originalmente desarrollado para promover la Informática entre estudiantes de primaria y secundaria en todo el mundo. Sin embargo, su bajo coste, tamaño reducido y conectividad (que, según la versión, puede incluir puertos USB 2.0 y 3.0, Ethernet, HDMI y hasta 40 pines de propósito general) la ha convertido en un sistema embebido de referencia para muchas aplicaciones, entre las que se incluyen diferentes laboratorios remotos (Bermudez-Ortega et al., 2015, 2016a; Filipović et al., 2017; Carballo et al., 2018; Letowski et al., 2019; Fukumoto et al., 2021).

Para el laboratorio remoto del Dobot es preferible usar el modelo B con 8 MB de RAM de la Raspberry Pi 4, ya que, aunque tiene un coste superior (en concreto de unos 100€) a la de otras versiones de la placa, sobre ella se ejecutará el programa para controlar al robot y el servidor principal del laboratorio. Además, incluye un puerto USB 3.0 al que conectar el puerto USB 3.0 del Dobot.

2.3. *Python: el lenguaje de programación del controlador*

Python (Python Webpage, 2022) es un lenguaje de programación interpretado, que ha ganado popularidad en los últimos

años, ya que al estar distribuido bajo una licencia de código abierto y poder ser ejecutado en múltiples plataformas, existe una amplia comunidad de programadores que desarrollan y comparten su librerías de forma altruista para un variado conjunto de aplicaciones.

Entre dichas librerías está `pydobot` (Pydobot Webpage (2022)), que permite acceder a una pequeña parte de la funcionalidad de la API proporcionada por el fabricante del Dobot Magician para controlar sus movimiento, obtener la posición y velocidad de sus ejes cartesianos y articulaciones, y trasladarlo a su configuración inicial (*home*).

Por lo tanto, Python y `pydobot` son el lenguaje de programación y la librería utilizados para escribir el programa de control con el que se interactúa, directamente a través del puerto USB 3.0 de la Raspberry Pi, con el Dobot. Además, la limitada funcionalidad de `pydobot` se ha resuelto ampliando su número de funciones para incorporarle la funcionalidad que el programa de control de nuestro laboratorio remoto necesita. A modo de resumen, las funciones más relevantes, originales y nuevas, se muestran en la Tabla 2, junto el tipo de movimiento con el que se encuentran relacionadas y el tipo de orden al que pertenecen, que puede ser de *Ejecución Inmediata* (EI), lo que implica que son ejecutadas por el Dobot instantáneamente cuando son recibidas (con independencia de si éste está realizando otra acción), o de *Ejecución Encolada* (EE), lo que implica que son almacenadas por el Dobot en una lista FIFO (*First-In First-Out*) para ser procesadas, de forma sucesiva, según su orden de llegada.

Tabla 2: Funciones principales de `pydobot`.

Versión	Orden	Modo	Tipo de orden
Original	getPose		EI
	setSpeed		EE
	setGrip		EE
	setSuck		EE
	wait		EE
	moveToPos	*	EE
Nueva	moveToAngleMod	*	EE
	moveToPointMod	*	EE
	increaseAngle	JOG	EE
	increasePos	JOG	EE
	moveSingleJoint	MOVJ	EE
	moveSingleCartesian	MOVL	EE
	setSpeedJog	JOG	EI
	setSpeedAccPTP	MOVJ, MOVL	EI
	getSpeedAcc	MOVJ, MOVL	EI
	home		EE
	getInput		EI
	setOutput		EI
	stop		EI
	startQueue		EI

* Modo de movimiento seleccionable según argumento.

2.4. *Node.js: el entorno de ejecución del servidor principal*

Node.js (Node.js Webpage, 2022) es un entorno multiplataforma de código abierto para la ejecución de aplicaciones de red. Los servidores web desarrollados para Node.js son computacionalmente ligeros, eficientes y escalables, ya que este entorno está soportado por una arquitectura de entrada/salida no-bloqueante basada en eventos y en el motor V8 de Google, que favorece el intercambio intensivo y en tiempo real de gran cantidad de datos. Un motivo adicional, y no por ello menos importante que los anteriores, que justifica la elección de Node.js para desarrollar y ejecutar nuestro servidor principal es que sus aplicaciones se programan en JavaScript, el mismo lenguaje que utilizamos para programar, por medio de EJsS, la interfaz gráfica de la experiencias del laboratorio.

La primera versión de nuestro servidor, llamado *ReNoLabs* y presentada en (Bermudez-Ortega et al., 2016a), permitía el control de acceso de los usuarios; el despliegue de las páginas web del laboratorio (entre las que se encuentra la interfaz gráfica de la experiencia); la comunicación ente dicha interfaz y la aplicación de control del sistema objeto del laboratorio; y el registro de datos experimentales y eventos. Su versión actual incorpora, además, la funcionalidad necesaria para desplegar y poner a punto, desde EJsS, el laboratorio remoto. Finalmente, cabe indicar que, durante el desarrollo del laboratorio remoto para el Dobot Magician, la funcionalidad del servidor ha sido nuevamente ampliada, tal y como se detallará en la Sección 3.2, con el objeto de mejorar la comunicación entre el programa de control del Dobot y el servidor.

2.5. *EJsS: herramienta para diseñar la interfaz gráfica de la experiencia y gestionar el laboratorio*

Easy Java/JavaScript Simulations (EJS/EJsS, EJsS Webpage (2022)) es una herramienta de código abierto, inicialmente desarrollada para facilitar la creación de simulaciones físicas interactivas a usuarios con conocimientos limitados de programación. Para lograrlo, EJsS permite 1) que sus usuarios definan, de una forma gráfica e interactiva, por una parte, el modelo de la simulación, y por otra, la interfaz gráfica o vista de la simulación, y 2) que a continuación generen la aplicación (applet o página web) correspondiente.

Una aplicación alternativa de EJsS, utilizada en este trabajo y en el campo de la Automática, es desarrollar con ella la interfaz gráfica de las experiencias de laboratorios remotos, sustituyendo el modelo de simulación por llamadas a una serie de rutinas que la permiten actuar como la aplicación cliente del servidor del laboratorio (Bermudez-Ortega et al., 2015, 2016a,b; Galan et al., 2018; Sánchez-Herrera et al., 2020).

Además, a partir de la versión 6.0 de EJsS es posible extender su funcionalidad e interfaz gráfica por medio de *"Plugins"* (Chacón et al., 2021), de forma que la herramienta se adapte a las necesidades de sus usuarios. Explotando esta propiedad, hemos desarrollado un *"Plugin"* para centralizar, desde EJsS, la configuración y gestión de diferentes aspectos de los laboratorios remotos basados en *ReNoLabs*. Por lo tanto, usaremos dicho *"Plugin"* para facilitar la puesta a punto del laboratorio del Dobot Magician.

Más concretamente, y tal y como se puede ver en la Figura 4, el *"Plugin"* de *ReNoLabs* añade la pestaña de laboratorios remotos al menú principal de EJsS. A través de dicha pestaña es posible acceder a diferentes paneles, en los que se indican los parámetros de conexión (IP y puerto de acceso) del servidor, se

configura el tipo de controlador, o se gestiona la lista de usuarios del laboratorio. Además, con el nuevo elemento de la barra de botones (resaltado dentro del cuadrado cian) se despliega un menú (también resaltado en cian) desde el que se puede indicar cual es el servidor del laboratorio sobre el que desplegar el entorno gráfico web de la experiencia, diseñado en EJsS. Finalmente, el *"Plugin"* carga una instancia de la clase `Lab`, que debe ser utilizada durante la definición de la funcionalidad y el aspecto gráfico de la interfaz web de la experiencia desarrollada con EJsS para para poder acceder desde dicha interfaz a la información que ésta intercambia con el programa de control del Dobot Magician a través del servidor *ReNoLabs*.

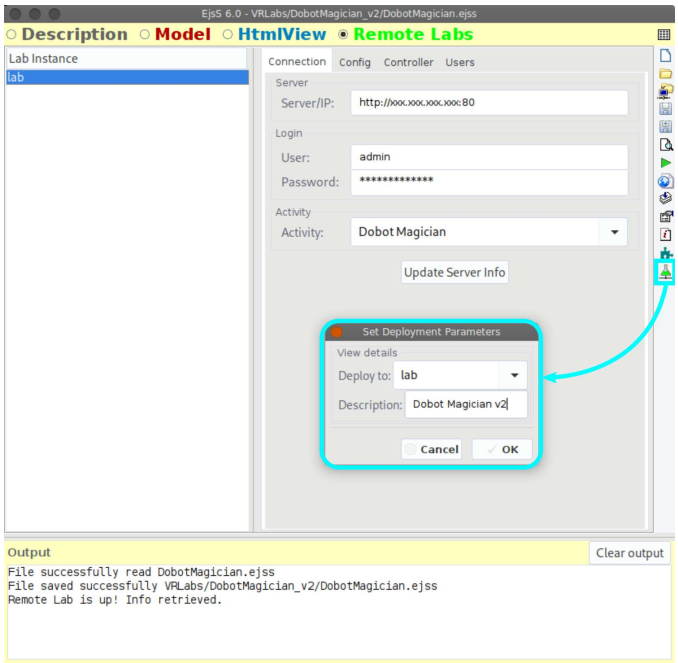


Figura 4: EJsS con funcionalidad añadida por el *Plugin* de *ReNoLabs*.

2.6. *Herramientas de comunicación*

Para establecer las comunicaciones entre las tres capas software del laboratorio (i.e. el programa de control del Dobot Magician, el servidor *ReNolabs* y la interfaz web de la experiencia) se utilizan los siguientes protocolos y librerías.

La librería de comunicación `ZeroMQ` es utilizada para intercambiar información entre el servidor *ReNoLabs* y el programa de control, porque 1) proporciona un modelo de comunicaciones de entrada/salida asíncrono de alto rendimiento; porque 2) soporta los patrones de comunicación publicador-subscriptor y petición-respuesta que resultan convenientes para nuestro caso; y porque 3) su robustez y popularidad está respaldada por una amplia comunidad de desarrolladores/usuarios (ZeroMQ Webpage, 2022). Por último, destacar que la funcionalidad de `ZeroMQ` se encuentra implementada sobre los principales lenguajes de programación, entre los que se incluyen Python y Node.js, lo que ha agilizado la programación de las comunicación entre el programa de control del Dobot Magician (programado en Python) y el servidor *ReNoLabs* (programado en Node.js).

Para la transferencia de información entre la interfaz web de la experiencia y el servidor se utiliza un protocolo de comunicación específico de *ReNoLabs*. Se escoge este protocolo porque está basado en el intercambio de mensajes en formato JSON (JavaScript Object Notation) sobre una conexión WebSocket o HTTP de sondeo largo, al estar soportado por la librería `socket.io` y por su capacidad de elegir la conexión más adecuada (según los navegadores involucrados, la existencia de proxy y firewalls, etc.). Además, conviene indicar que esta librería es una opción de comunicación eficiente, fiable y escalable (Socket.io Webpage, 2022) para comunicaciones bidireccionales y en tiempo real entre servidores Node.js (en nuestro caso, *ReNoLabs*) y diferentes tipos de clientes (en nuestro caso la interfaz de la experiencia).

Finalmente, destacar que utilizamos diferentes herramientas de comunicación entre cada pareja de elementos, ya que ZeroMQ facilita la comunicación entre nuestro servidor y programas de control implementados sobre un amplio abanico de lenguaje, mientras que `socket.io` favorece que las comunicaciones entre la interfaz gráfica de las experiencias y el servidor se realicen a través de www.

3. El funcionamiento del laboratorio remoto

En esta sección se describen las características más relevantes de los elementos que se han desarrollado específicamente para el laboratorio remoto del Dobot Magician. Todo el software e instrucciones necesarios para desplegar este laboratorio están disponibles en `https://gitlab.com/jcsombria/jj-aa-2021` y `https://github.com/jcsombria/ReNoLabs`.

3.1. *Programa de control*

El software de control del laboratorio, que reenvía al Dobot, a través del puerto USB 3.0 de la Raspberry Pi, las instrucciones que el usuario elige y envía al servidor desde el entorno gráfico de la experiencia, está programado en Python.

Su funcionalidad se organiza, tal y como se muestra en la Figura 5, en dos hilos de ejecución independientes.

El primer hilo, correspondiente al programa principal, se encarga de 1) establecer la comunicación con el dobot; 2) abrir dos sockets de comunicación con el servidor; 3) lanzar el segundo hilo de ejecución y a continuación; y 4) recibir, interpretar y reenviar al Dobot las instrucciones que la interfaz web de la experiencia envía al servidor cada vez que los alumnos interactúan con alguno de sus elementos. Más concretamente, cuando el estudiante solicita desde la intefaz gráfica que el robot lleve a cabo una acción, pulsando para este fin alguno de sus botones interactivos o escribiendo un programa en ACL, la interfaz traduce la acción o acciones solicitadas en una orden o secuencia de órdenes que son enviadas a través del servidor al programa de control, que es el responsable de interpretarlas y ejecutarlas sobre el Dobot. Además, para que en un futuro sea posible utilizar la misma herramienta sobre diferentes robot, el proceso de traducción de acciones a instrucciones está encapsulado en un módulo de JavaScript, cuyas características más relevantes se detallan en la sección 3.4.

[1] En nuestro caso utilizamos $Ts = 100ms$. Este valor ha sido ajustado experimentalmente para obtener un compromiso razonable entre rendimiento, visualización y sobrecarga de las comunicaciones.

El segundo hilo, por su parte, se encarga de 1) leer periódicamente[1] los valores de la posición del efector final y de los ángulos de las articulaciones del Dobot, y de 2) enviárselos al servidor para que éste pueda reenviarlos a la interfaz gráfica de la experiencia.

La funcionalidad del controlador se ha divido en dos tareas, para poder intercambiar información con el servidor de dos formas diferentes. El envío al servidor de las posiciones del robot se realiza periódicamente mediante la publicación de esta información a través de uno de los sockets de comunicación. La recepción/respuesta a las instrucciones que el servidor le reenvía, se realiza de forma asíncrona mediante el patrón petición-respuesta. Finalmente, se intercambian mensajes de texto fácilmente interpretables y separables (del tipo "`identificador:valor1,valor2,...,valorN`"), ya que existen librerías de manipulación de `strings` en la mayoría de los lenguajes de programación.

Figura 5: Esquema del software de control

Por otra parte, conviene indicar que como el servidor y el programa de control se ejecutan sobre la misma Raspberry PI, se ha utilizado una IP local (127.0.0.1) al inicializar ambos sockets. Sin embargo, sería posible distribuir ambas aplicaciones en dos Raspberry PIs (u otros dispositivos de computo) reconfigurando ambas IPs de acceso. Esta posibilidad aporta un grado de libertad adicional al laboratorio remoto del Dobot, que puede ser útil, por ejemplo, si se disponen de varios brazos robóticos, controlados desde diferentes Raspberry PIs, a los que se desee acceder a través de un único servidor del laboratorio.

Finalmente, cabe recordar que toda la interacción del programa de control con el Dobot está soportada por nuestra versión mejorada de la librería `pydobot` y, por lo tanto, por la funcionalidad de la API proporcionada por el fabricante.

3.2. *Servidor del laboratorio*

El servidor del laboratorio, *ReNoLabs*, tiene múltiples funciones, que pueden ser organizadas, tal y como la Figura 6 esquematiza, en dos grandes grupos: aquellas relacionadas con la gestión y configuración del laboratorio desde EJsS y aquellas relacionadas con las páginas web del laboratorio. Antes de explicarlas brevemente, cabe indicar que la funcionalidad genérica asociada a las páginas web del laboratorio ya se encontraba disponible en las versiones del servidor descritas en Bermudez-Ortega et al. (2016a, 2017), mientras que una primera versión de parte de la funcionalidad asociada a la configuración y gestión del laboratorio fue esbozada en Aizpuru-Rueda et al. (2019). En cualquier caso, la funcionalidad del servidor *ReNoLabs* utilizado en el Dobot Magician ha sido refinado en los últimos años.

3.2.1. *Servidor web del laboratorio*

El servidor de páginas web de nuestro laboratorio incluye: 1) una página de acceso al laboratorio donde los alumnos introducen su usuario y contraseña antes de poder navegar por el resto de las páginas web del laboratorio; 2) una página de ayuda en la que los instructores pueden describir las características más relevantes del laboratorio y de cada experiencia; 3) una página de acceso a los ficheros con registros temporales del estado (i.e. posición del efector final y de los ángulos de las articulaciones) del Dobot y 4) la interfaz gráfica de la experiencia con la que los estudiantes interactúan con el Dobot, a través del servidor y del programa de control.

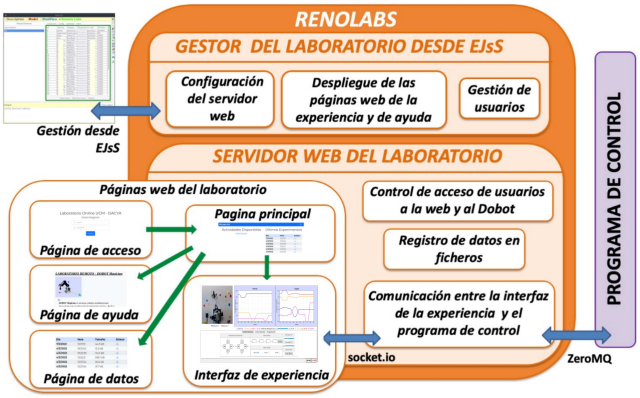


Figura 6: Esquema de la funcionalidad del servidor.

Además de las páginas web, el servidor incorpora la funcionalidad requerida para asegurar que éstas cumplen con su misión correspondiente. Para ello, por una parte asegura que únicamente acceden a todas las páginas web del laboratorio los usuarios registrados en una base de datos y que a la interfaz gráfica de la experiencia del Dobot no acceden dos alumnos de forma simultánea. Por otra, almacena, en un fichero diferente por cada alumno y sesión, los valores que le son enviados periódicamente desde el programa de control del Dobot. Finalmente, establece las comunicaciones necesarias con la interfaz gráfica de experiencias y con el programa de control de Dobot, para que estos intercambien, a través del servidor, las instrucciones provenientes de la interfaz gráfica de la experiencia y el estado del Dobot monitorizado desde el programa de control.

También conviene resaltar que la estructura que *ReNoLabs* establece para intercomunicar la pagina web de la experiencia y el programa de control del Dobot permite utilizar los protocolos de comunicación más adecuados para cada caso. Así, aunque la primera versión del laboratorio remoto del Dobot Magician utiliza una implementación *ad-hoc* de las comunicaciones entre el servidor y el programa de control, su versión actual, presentada en este artículo, utiliza una nueva versión más general, capaz de comunicar la pagina web de la experiencia con cualquier

controlador que utilice la librería de comunicación ZeroMQ y se adhiera al protocolo de comunicaciones servidor-controlador estándar utilizado por *ReNoLabs*.

Por último, en aquellos casos en los que en la institución donde se despliegue el laboratorio remoto dispongan de Moodle, es posible incorporar directamente la interfaz gráfica de la experiencia a dicho LMS. En este caso es necesario que la interfaz incluya los elementos gráficos de acceso a usuarios disponibles en el "*Plugin*" de EJsS para *ReNoLabs*, para que el usuario pueda proporcionar sus credenciales del laboratorio desde la interfaz gráfica sin necesidad de que el equipo docente tenga que solicitar la instalación de un Plugin de Moodle en la plataforma LMS de su institución.

3.2.2. Gestión y configuración del laboratorio remoto desde EJsS

Tal y como se ha explicado en la Sección 2.5, existe un *"Plugin"* de EJsS para que los profesores gestionen y configuren desde dicha herramienta los laboratorios remotos de *ReNoLabs*. Para que la funcionalidad que dicho *"Plugin"* incorpora a EJsS sea aplicable sobre el servidor del laboratorio es necesario que este último de respuesta a las labores de gestión y configuración que está siendo realizada, desde EJsS, por los profesores. Por este motivo, el servidor del laboratorio incorpora la funcionalidad necesaria para actualizar los ficheros en los que 1) se configuran diferentes aspectos (p.e. dirección IP, puerto de comunicación) del servidor web del laboratorio, 2) se define la interfaz web de la experiencia y la página web de ayuda, y 3) se recoge información (p.e. nombre, contraseña, tipo) de los usuarios potenciales del laboratorio.

Además, tanto la página web de ayuda como la interfaz web de la experiencia se definen en EJsS. Para la primera se usan las utilidades del panel de Descripción de dicha herramienta. Para la segunda se utiliza la Vista HTML y el elemento *Lab*, incorporado por el *"Plugin"* de *ReNoLabs* a EJsS para aglutinar las rutinas de comunicación entre la interfaz web de la experiencia y el servidor *ReNoLabs*. De esta forma, tras configurar los datos de conexión del laboratorio remoto en EJsS, la interfaz de la experiencia puede recibir automáticamente los datos enviados por el servidor y enviar las órdenes solicitadas por sus usuarios. Aun más, la instancia de *Lab* mantiene una lista con los valores de las variables que sincroniza con el servidor y usa una memoria intermedia para adaptar las velocidades de recepción de datos y de visualización.

3.3. Interfaz gráfica de la experiencia

En secciones anteriores hemos explicado como crear la interfaz web de la experiencia y como desplegarla en el servidor. En esta sección procederemos a describir el aspecto gráfico (representado en las imágenes que aparecen en la Figura 7) y la funcionalidad de su versión actual.

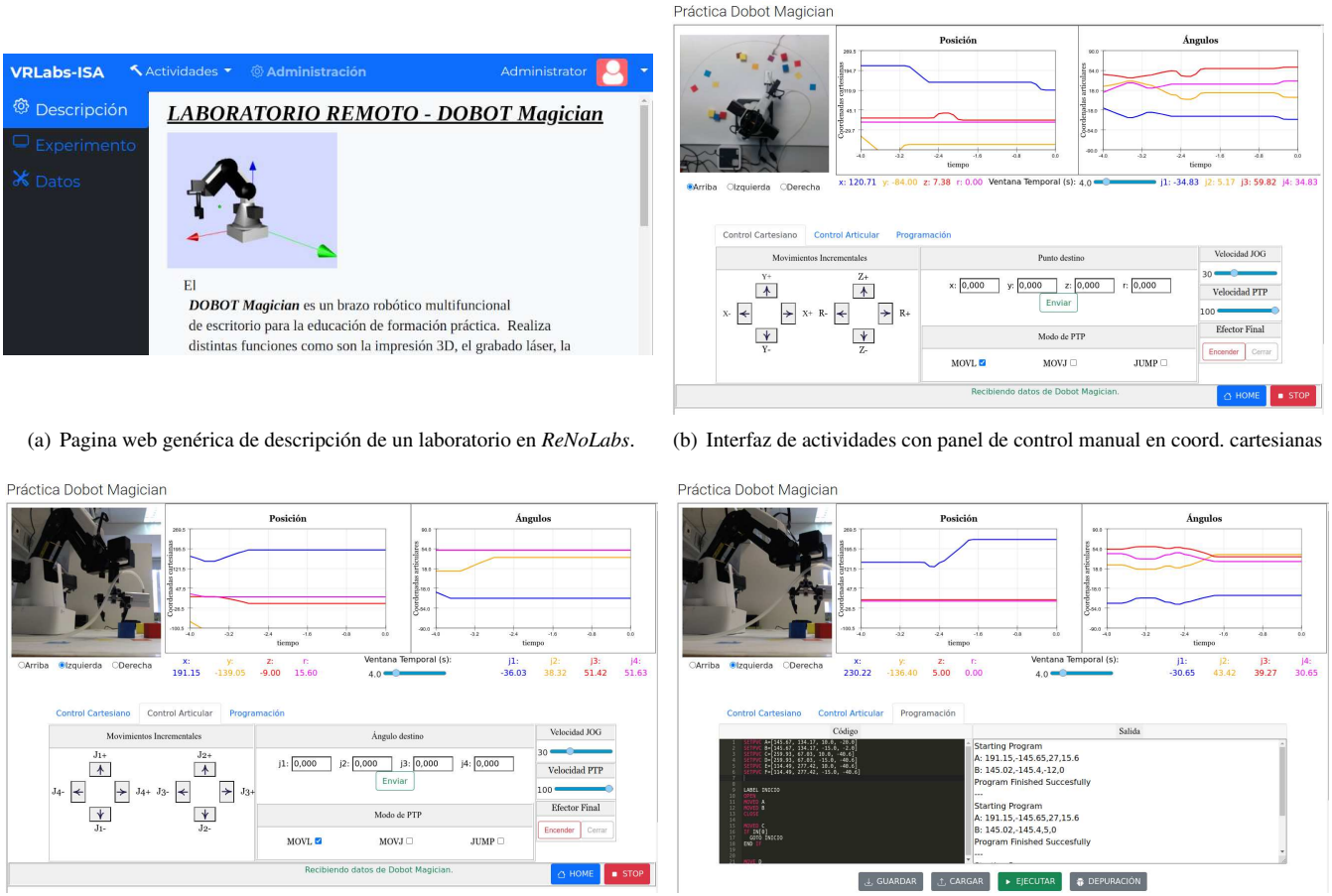
Más concretamente, el servidor de laboratorio *ReNoLabs* alberga la interfaz web que se muestra en la Figura 7(a) y que proporciona: una barra de navegación principal (en azul, en la parte superior) para que el alumno seleccione la actividad que desea realizar y otras configuraciones del usuario y una barra de navegación *en-laboratorio* (a la izquierda, en gris oscuro) para que el alumno seleccione si desea mostrar la descripción de la actividad, desplegar la interfaz gráfica del experimento o descargar alguno de los archivos de datos que ha generado en algún experimento previo.

Dentro de la interfaz de la actividad del Dobot, que se muestra en las Figuras 7(b), 7(c) y 7(d), se mantiene el patrón estándar de diseño. En la parte superior izquierda de las tres figuras se muestra una vista del Dobot durante la realización de la experiencia, proporcionada bien por una cámara cenital o por dos cámaras laterales - una a cada lado del Dobot-, para que el alumno pueda seleccionar la más adecuada en cada caso. En la parte superior derecha de las tres figuras, como complemento a la vista del Dobot, se presentan dos conjuntos de curvas que respectivamente muestran la evolución temporal de las posición del Dobot en coordenadas cartesianas y articulares. Por otra parte, los elementos de control se sitúan en la parte inferior de las tres figuras, organizados en tres paneles de operación diferentes: control manual cartesiano, control manual articular y control por programa. Mas concretamente en la parte inferior de la Figura 7(b) se muestra el panel del modo de control manual cartesiano, que permite al alumno enviar directamente y de forma interactiva instrucciones de movimiento incrementales (mediante cursores) o absolutas (indicando el punto de destino) en los ejes cartesianos (x, y, z, r), elegir el modo de desplazamiento del Dobot, la velocidad de movimiento y la situación del efector final. El panel de control manual en los ejes articulares, que se muestra en la parte inferior de la Figura 7(c), es análogo al anterior, salvo por el hecho de que tal y como indica su nombre, las instrucciones de movimiento incrementales y absolutas se dan en los ejes articulares (J1, J2, J3, J4) en vez de en los ejes cartesianos (x, y, z, r). El panel de control por programa, que se muestra en la parte inferior de la Figura 7(d), permite al usuario definir y enviar al Dobot una serie de instrucciones en el lenguaje de programación de robots ACL. Finalmente, en la parte inferior derecha de la interfaz web de actividades del Dobot, que se muestra en las Figuras 7(b), 7(c) y 7(d), se proporcionan un botón *home* para llevar al Dobot a su posición inicial y un botón *stop* para que el alumno pueda realizar una parada de emergencia del Dobot.

Por último, queremos destacar que gracias a los tres paneles de control disponibles, los alumnos pueden realizar un amplio conjunto de actividades, como son el familiarizarse con el comportamiento y los movimientos básicos del Dobot mediante los controles disponibles en los paneles manuales, o definir movimientos complejos, repetitivos, y que incluyan una cierta lógica de control, mediante la escritura de un programa en ACL. El soporte computacional de esta última utilidad, que no está disponible entre las capacidades proporcionadas por el fabricante del Dobot, se detalla en la siguiente sección.

3.4. Máquina virtual de programación en ACL

El laboratorio remoto presentado en Goncalves-López-Medrano et al. (2021) permitía controlar de forma manual el movimiento del brazo robótico a través de los elementos interactivos que en la actualidad se organizan en los paneles de control manual cartesiano y articular, y que anteriormente se aglutinaban en un único panel de control. Aunque esta forma de proceder es indudablemente útil para que los alumnos tengan una primera toma de contacto con el Dobot, para hacer las prácticas más interesantes y, en cierto modo, acercar la experiencia del alumno a un entorno de trabajo más cercano al in-



(a) Pagina web genérica de descripción de un laboratorio en *ReNoLabs*.

(b) Interfaz de actividades con panel de control manual en coord. cartesianas

(c) Interfaz de actividades con panel de control manual en coor. articulares

(d) Interfaz de actividades con panel de control por programa

Figura 7: Interfaz web del laboratorio remoto del Dobot.

dustrial, en la versión del laboratorio remoto del Dobot que se presenta en este trabajo hemos añadido el panel que permite realizar la programación del movimiento mediante un lenguaje que incorpora un subconjunto de las órdenes del ACL. Se ha elegido este lenguaje de programación, clásico y bien conocido, ya que es el utilizado por el brazo SCORBOT ER-V Plus, disponible en el laboratorio de los autores y con el que los alumnos siguen realizando prácticas presenciales de Robótica.

Las ordenes implementadas de ACL para del Dobot son las que se presentan en la Tabla 3. Información adicional sobre su significado puede encontrarse en (ACL Webpage, 2022). Para que puedan ser ejecutadas sobre el Dobot Magician, hemos desarrollado una máquina virtual que se ejecuta en el cliente web, dentro de la aplicación del laboratorio generada por EJS, y que se encarga de 1) compilar el programa ACL que el alumno ha escrito en el panel de control de programación a un código intermedio (*bytecode*) y de 2) ir traduciendo y ejecutando las ordenes del código intermedio en instrucciones soportadas por la API del Dobot. De esta manera, el programa introducido por el alumno en el transcurso de la práctica se puede ejecutar totalmente o paso a paso, convirtiendo las instrucciones ACL en órdenes inteligibles por el Dobot.

Más concretamente, para procesar y analizar léxica y sintácticamente el programa en ACL utilizamos la librería *peggy.js* (PeggyJS Webpage, 2022), que está basada en el formalismo Parsing Expression Grammar (PEG). El funcionamiento de nuestra implementación es, en líneas generales, el siguiente: la máquina virtual se encapsula en un objeto de tipo `ACLVirtualMachine`, que proporciona en su interfaz los métodos necesarios para compilar el código en un programa de código intermedio y almacenarlo en un objeto de la clase `Program`. Las instrucciones de código intermedio son objetos de tipo `Instruction` y se corresponden con ordenes directamente ejecutables por la máquina virtual. Para enlazarlas con las operaciones del sistema que hay que controlar, se utiliza un objeto de tipo `Controller` que encapsula, en nuestro caso, la interfaz de instrucciones del Dobot Magician. Por lo tanto, la máquina virtual encapsula todo lo necesario para conseguir la ejecución del código ACL, incluyendo la compilación y almacenamiento del programa compilado, las variables utilizadas por el programa, el registro de salida y los errores de ejecución.

Finalmente, cabe mencionar que, aunque hemos considerado más conveniente que la máquina virtual se ejecute en el cliente web, podría también ser desplegada en el servidor, ya que éste también utiliza el lenguaje de programación Javascript. Además, conviene destacar que, aunque en este trabajo se utiliza la máquina virtual para controlar el Dobot, la arquitectura de clases implementadas hace que sea fácilmente extensible a otro hardware. También queremos mencionar que sería igualmente viable (aunque posiblemente más trabajoso) utilizar el mismo procedimiento para dar soporte a otros lenguajes de programación distintos de ACL.

Tabla 3: Resumen de instrucciones ACL implementadas.

Movimiento	E/S	Posiciones
MOVE pos	SET IN[n]=val	HERE var
MOVED pos	SET OUT[n]=val	TEACH var
SPEED val	**Programa**	SETPV var
OPEN	DELAY var	SETPVC var
CLOSE	ABORT	SETP var
HOME	STOP	SETP var2=var1
Control de Flujo		**Usuario**
IF v1 oper v2	FOR v1=v2 TO v3	PRINT VAR
	...	PRINTLN var
ELSE	ENDFOR	SHOW din
		SHOW dout
ENDIF	LABEL n	READ "string"var
	GOTO n	GET var

4. Ejemplos de experiencias realizables

Nuestro laboratorio remoto es utilizado como complemento de las prácticas presenciales que realizan los alumnos de Ingeniería Electrónica en Comunicaciones de la Universidad Complutense de Madrid. A través del laboratorio remoto pueden reproducir y verificar, sobre el robot real y en cualquier momento, los resultados obtenidos durante la sesión presencial del laboratorio. Además, pueden realizar de forma remota prácticas más complejas o avanzadas que, por las limitaciones propias del laboratorio presencial (p.e. horario, disponibilidad de un único Dobot Magician), no pueden hacer de forma presencial.

Las prácticas, que están diseñadas para que los alumnos detecten sus errores de forma progresiva, se organizan en dos fases. En primer lugar, los alumnos diseñan el experimento de forma *local* mediante Matlab, definiendo, especificando y verificando todas las operaciones, funciones y pasos que debe ejecutar el Dobot para poder realizar la tarea encomendada. En segundo lugar, una vez completada la primera fase, el alumno utiliza el laboratorio *remoto* para comprobar sobre el robot real que el diseño realizado es válido, analizando si el comportamiento del Dobot es correcto y el esperado. Si esto no ocurre, pueden corregir sus cálculos y volver a comprobar su nueva propuesta de solución en el Dobot. En esta segunda fase se puede utilizar el laboratorio remoto en cualquier horario en el que no esté siendo utilizado de forma simultanea por otro alumno.

A continuación mostramos dos experiencias que siguen el esquema anterior. La primera experiencia, que denominaremos *Práctica de Cinemática del Dobot*, está orientada a que los alumnos se ejerciten sobre principios básicos de Robótica, como son la cinemática directa, los parámetros de Denavit-Hartenberg, y la definición de los ejes y movimientos necesarios para llevar a cabo una operación concreta. La segunda experiencia, que denominaremos *Práctica de Programación del Dobot*, está dirigida a la programación de una tarea más compleja para el robot, en la que se incluya la detección automática de fallos durante su realización.

Para realizar ambas prácticas, al alumno se le proporcionan las dimensiones de un conjunto de cubos de diferentes colores (rojo - *CRj*, amarillo - *CAm*, y azul - *CAz*) que deberá mover desde/hasta diferentes posiciones fijas que se sitúan a ambos lados y en la posición central del robot, y que se marcan, tal y como se muestra en la Figura 8, mediante las siguientes etiquetas: *BAm*, *BRj*, *BAz*, *BMa*, *BNa*, *BMr*.

4.1. Práctica de Cinemática del Dobot

En esta práctica, como trabajo *local*, el alumno debe obtener los parámetros de Denavit-Hartenberg del robot a partir de la información mostrada en la Figura 3. Una vez obtenida la tabla de parámetros, debe calcular las matrices ${}^{i-1}A_i$ e implementar en Matlab una función para la cinemática directa del robot. Finalmente, debe conectarse al laboratorio remoto y verificar que el Dobot Magician se comporta de la misma forma que predicen las funciones que ha implementado en Matlab.

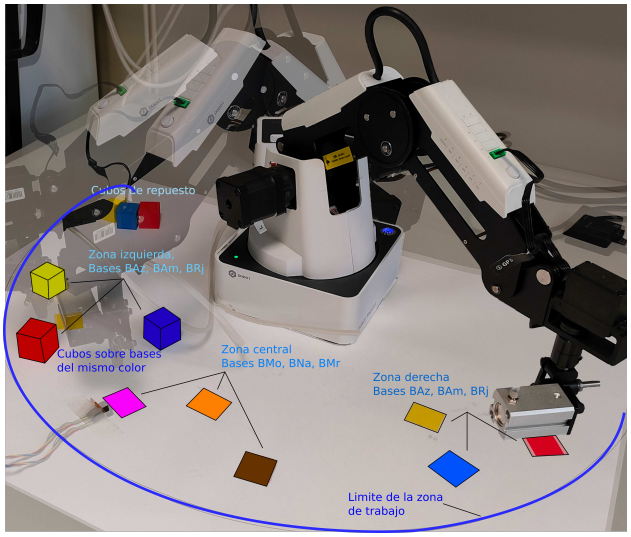
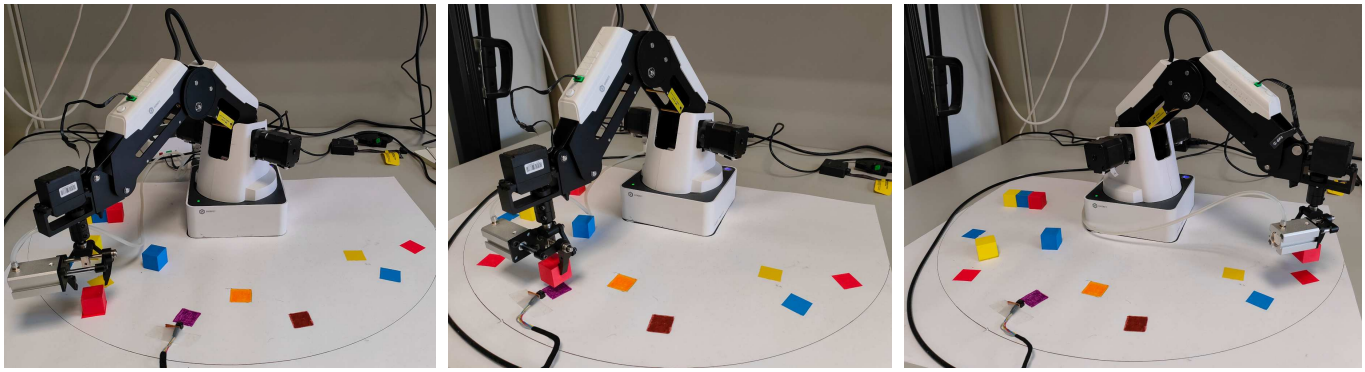


Figura 8: Posiciones fijas de los bloques en los experimentos.

Una vez comprobado que conoce el funcionamiento del Dobot, debe diseñar el movimiento necesario para llevar a cabo una operación de montaje sencilla con los elementos disponibles en la Figura 8. Esta actividad tiene como finalidad que: 1) el alumno aprenda a manejar diferentes ejes, puntos adecuados de aproximación y despegue; y que 2) calcule la trayectoria más adecuada para evitar colisiones del Dobot durante la operación. A cada alumno se le propone: (1) una posición inicial (derecha o izquierda) y configuración de cubos diferente (p.e. *CRj* sobre *BAm*, *CAm* sobre *BRj*, y *CAz* sobre *Baz*); y (2) un movimiento a otra posición (en el lado opuesto) con una configuración de colores diferente. Para resolverlo, el alumno debe tener en cuenta las posiciones iniciales y finales de los cubos para definir la secuencia de pasos (puntos) por los que el Dobot debe pasar y las acciones de su efector final (pinza). A continuación, después de verificar, mediante simulación, que la secuencia de pasos obtenida permite realizar la operación con éxito, el alumno se conectará de nuevo al laboratorio remoto para ejecutar la secuencia de acciones que ha elegido mediante el uso de los elementos interactivos de los paneles de control manual.

Finalmente, cabe destacar que el alumno puede encontrarse los bloques en cualquier posición, por lo que lo primero que debe hacer es situar los bloques en su posición inicial, usando los cursores de movimiento incremental de la interfaz gráfica de la experiencia y los bloques de repuesto (por si alguno ha quedado fuera del espacio de trabajo). Una vez haya situado los cubos



(a) Posición inicial (b) Posición central (c) Posición final

Figura 9: Imágenes del Dobot durante la práctica de programación.

en su posición inicial correcta, ejecutará paso a paso, mediante los campos de posiciones absolutas, la secuencia de movimientos calculados, para comprobarán de forma experimental que el Dobot puede llevar a cabo la operación planteada con éxito.

4.2. Práctica de Programación del Dobot

En este caso se sigue un procedimiento de montaje que incluye un paso adicional intermedio. Es decir, además de ofrecer al alumno una configuración inicial y final de los bloques a la derecha o izquierda del Dobot, durante el desplazamiento de los bloques a la posición final éstos deben pasar por un punto situado en la zona central (p.e. en la posición marrón *BMr*), acercando el cubo a la superficie de trabajo sin abrir la pinza. En este punto central hay un sensor capaz de detectar si el cubo está situado o no sobre él. La finalidad del ejercicio es que el alumno programe el comportamiento del Dobot con un lenguaje orientado a robots (en nuestro caso, ACL), incluyendo en su código una etapa de sensorización en la que se verifica de forma automática (i.e. sin la supervisión del usuario) que la operación de ensamblaje se está llevando a cabo correctamente, y en caso de que ocurra un error, actuar de forma adecuada.

El alumno primero diseñará el movimiento a llevar a cabo y lo verificará mediante simulación con Matlab. A continuación, escribirá un programa en ACL que se ocupe de 1) realizar los movimientos necesarios y de 2) comprobar si los cubos son detectados en la posición *Bmr*. En caso afirmativo, el programa puede continuar con la operación de ensamblaje. En caso negativo, la operación es incorrecta (debido a diferentes causas, como no llevar el cubo a dicha posición o no haberlo agarrado adecuadamente), por lo que debe mostrarse un mensaje de error y colocar el Dobot en posición de reposo a espera de nuevas órdenes. Finalmente, una vez diseñado el código, el alumno se conectará al laboratorio remoto, colocará los bloques en la posición inicial pedida, copiará el código en el panel de programación de la interfaz web de actividades del Dobot, y lo ejecutará para que comprobar si el Dobot realiza los movimientos adecuados. En caso de que exista un error de sintaxis en el programa, aparecerá un mensaje de error en el panel de programación de la interfaz web para que el alumno lo corrija antes para poder ejecutar el programa. En caso de que el Dobot realice pasos imprevistos o movimientos inadecuados, el alumno deberá modificar y volver a ejecutar el programa, hasta que el Dobot realice la operación solicitada de forma correcta.

Por último, cabe indicar que para documentar ambas prácticas, el alumno debe 1) representar gráficamente las posiciones cartesianas y articulares del Dobot a lo largo de cada actividad (utilizando para este fin la información almacenada en los ficheros de Datos experimentales del servidor *RenoLabs*) y 2) proporcionar imágenes del Dobot y de los cubos en las que se pueda observar: a) la situación del Dobot al comenzar la experiencia, b) la disposición del Dobot y de los cubos en la configuración inicial, c) la situación de todos los elementos en varias diferentes situaciones intermedias, y d) la situación de todos los elementos en la configuración final. En la Figura 9 se muestran alguna de las figuras relevantes para documentar una experiencia realizada durante la práctica de Programación del Dobot.

5. Estudio de viabilidad del laboroto remoto

Para evaluar la viabilidad del laboratorio remoto como herramienta educativa hemos realizado unas pruebas preliminares de su usabilidad en las que han participado 5 docentes de asignaturas relacionadas con la Robótica y 5 alumnos que habían cursado previamente una asignatura de Robótica.

Se reproduce la situación en la que nuestros alumnos utilizarán este laboratorio remoto, creando un seminario virtual en Moodle LMS e incluyendo en él: 1) un guión de prácticas, 2) un enlace para acceder a la actividad en *ReNoLabs*, y 3) un cuestionario de opinión. El cuestionario, tanto se diseña para recabar información que nos permita analizar la adecuación del laboratorio remoto como complemento educativo de las asignaturas de Robótica de la UCM, tanto desde el punto de vista del docente (Tabla 4) como del alumno (Tabla 5). El cuestionario incluye preguntas evaluables mediante una escala *Likert* (Liddell and Kruschke, 2018) con cinco niveles del 1 (totalmente en desacuerdo con la afirmación) al 5 (totalmente de acuerdo), y preguntas abiertas que recogen información adicional sobre los problemas técnicos detectados por los usuarios y sugerencias de posibles mejoras del laboratorio.

Teniendo en cuenta el número de participantes, consideramos interesante hacer un análisis cualitativo de los resultados. Los respuestas indican que, en general, hay un alto grado de satisfacción con la interfaz del laboratorio, tanto desde un punto de vista técnico (se reportaron pocos problemas) como desde un punto de vista docente. Todos los participantes están de acuerdo o muy de acuerdo con la afirmación de que la práctica captura

Tabla 4: Encuesta para el profesor

Pregunta	*Respuesta*				
	1	2	3	4	5
En mis cursos de robótica los alumnos realizan prácticas con un manipulador		20 %	20 %	20 %	40 %
Es viable integrar el laboratorio remoto en uno de mis cursos/asignaturas de robótica.				40 %	60 %
Para usar el laboratorio en mis cursos necesitaría modificar o añadir funcionalidad.	20 %	40 %	40 %		
En su estado actual, el laboratorio remoto es adecuado para ser usado por los alumnos.				60 %	40 %
Sugerencias para mejorar la actividad	pregunta abierta				

Tabla 5: Encuesta para el alumno

Pregunta	*Respuesta*				
	1	2	3	4	5
La práctica de forma remota captura la esencia de la práctica presencial.				20 %	80 %
La actividad en Moodle proporciona toda la información necesaria para realizar la práctica.				100 %	
El laboratorio remoto facilita la realización de la tarea propuesta en el guion de la práctica					100 %
Considero necesario recibir formación en el uso del laboratorio para realizar la práctica.			40 %	20 %	40 %
Realizar la práctica de manera presencial no me habría ayudado a comprender mejor la tarea.			40 %	40 %	20 %
He experimentado problemas técnicos durante el uso del laboratorio remoto	pregunta abierta				
Creo que el laboratorio remoto sería más útil si incluyera ...	pregunta abierta				

la esencia del laboratorio presencial y por tanto podría utilizarse en su lugar.

El principal problema de usabilidad detectado está relacionado con el campo de visión que ofrecen las cámaras del volumen de trabajo del Dobot Magician, ya que hay ángulos en los que las imágenes muestran ambigüedades que dificultan el correcto posicionamiento del brazo. Nos parece especialmente interesante una sugerencia recibida en este sentido, que apuntaba la posibilidad de situar una cámara en el efector final. Además, durante las pruebas se produjeron algunas paradas de servicio temporales, que están siendo analizadas actualmente. Una posible solución al problema podría estar ligado a las sugerencias que inciden en la necesidad de aumentar la información sobre el estado del Dobot proporcionada por la interfaz gráfica. Por último, algunos usuarios sugieren colocar unos ejes virtuales sobre las imágenes reales del Dobot para facilitar el posicionamiento.

Para poder estudiar el desarrollo de la actividad, hemos registrado en Moodle las interacciones de los voluntarios a través del plugin EjsS de Learning Analytics (Esquembre et al., 2019). El análisis del proceso de realización de la práctica nos permite extraer las siguientes conclusiones de interés:

- El número medio de sesiones[2] por voluntario es 2. Esto nos indica que se han realizado las prácticas según lo previsto, utilizando la primera sesión para familiarizarse con el movimiento del robot y la segunda para programarlo.
- El tiempo medio de duración de la primera sesión es 8 minutos y el de la segunda sesión es 20 minutos. Esto es esperable, ya que la programación del Dobot en ACL conlleva una mayor dificultad que el control manual.
- El panel de control cartesiano registra un uso mayor que el articular (los voluntarios realizaron aproximadamente el doble de acciones con el primero). Esto puede ser debido a que resulta más intuitivo el control cartesiano o podría estar condicionado porque el panel por defecto es el cartesiano. Para confirmarlo o desmentirlo, en el futuro aleatorizaremos el panel mostrado al inicio.

Finalmente cabe destacar que aunque son resultados preliminares, la experiencia ha resultado muy positiva y confirma la viabilidad de la actividad práctica, que se incluirá en un curso real de robótica en el próximo curso académico. Evidentemente, existen multitud de aspectos a tener en cuenta para garantizar el correcto funcionamiento. Entre ellos, los que hemos previsto o detectado han sido subsanados mediante diversas medidas para intentar minimizar las paradas de servicio, como delimitar un volumen de trabajo seguro o utilizar bloques blandos de goma espuma.

6. Conclusiones

En este artículo se presenta un laboratorio remoto para el robot educativo Dobot Magician, que puede ser desplegado, siempre que se disponga de dicho brazo robótico, por el coste de una Raspberry PI 4 (100€), ya que toda su funcionalidad se encuentra soportada por este ordenador monoplaca y herramientas software de código abierto (EJsS, Node.js y Python). Además, conviene destacar que este laboratorio remoto utiliza el servidor de prácticas remotas *ReNoLabs*, motivo por el que la puesta a punto de su interfaz gráfica y la gestión de sus usuarios pueden ser realizadas desde EJsS, siempre y cuando se incorpore al mismo la funcionalidad del *Plugin* específicamente desarrollado para *ReNoLabs*. Finalmente, resaltamos que la nueva versión del laboratorio remoto que se describe en este artículo, presenta una interfaz gráfica de experiencias renovadas que facilita la distinción de las actividades que los alumnos pueden realizar con el Dobot y que incorpora un nuevo panel para programar el comportamiento del Dobot mediante ACL.

[2]Descartando las de duración menor a 1 minuto, que consideramos desconexiones fortuitas

Como trabajo futuro se plantean las siguientes posibilidades. Por una parte, y de forma inminente, deseamos utilizar este nuevo laboratorio remoto durante este curso en las asignaturas de Robótica. Por otra, se plantea la posibilidad de aprovechar las capacidades habituales de EJsS para modelar el funcionamiento del Dobot y permitir que los alumnos realicen simulaciones interactivas de este brazo robótico, desde el mismo entorno con el que actúan sobre el robot real. Finalmente, al disponer en los laboratorios presenciales del brazo SCORBOT ER-V Plus, se está planteando la posibilidad de reutilizar parte de los desarrollos del laboratorio remoto del Dobot Magician para dar también acceso remoto a este otro brazo robótico industrial.

Agradecimientos

Este trabajo ha sido realizado con el apoyo del programa de Proyectos de Innovación Educativa y Mejora de la Calidad Docente de la Universidad Complutense de Madrid (concretamente a través de los proyectos 2019/20-139 y 2021/22-39).

Referencias

ACL Webpage, 2022. http://www.theoldrobots.com/book45/ACL28-Ctrl-B.pdf, Accessed: 2022-03-15.

Aizpuru-Rueda, I., Besada-Portas, E., Chacón, J., Lopez-Orozco, J. A., 2019. Despliegue automático de laboratorios remotos extendiendo las capacidades de EJsS. In: XL Jornadas de Automática.

Angulo, I., García-Zubía, J., Hernández-Jayo, U., Uriarte, I., Rodríguez-Gil, L., Orduña, P., Martínez Pieper, G., 2017. Roboblock: A remote lab for robotics and visual programming. In: 4th Experiment@International Conference.

Bermudez-Ortega, J., Besada-Portas, E., de la Torre, L., Lopez-Orozco, J. A., de la Cruz, J. M., 2016a. Lightweight Node.js & EJsS-based web server for remote control laboratories. In: IFAC Symposium on Advances in Control Education.

Bermudez-Ortega, J., Besada-Portas, E., Lopez-Orozco, J. A., Bonache-Seco, J., de la Cruz, J. M., 2015. Remote web-based control laboratory for mobile devices based on EJsS, Raspberry Pi and Node.js. In: IFAC Workshop on Internet Based Control Education.

Bermudez-Ortega, J., Besada-Portas, E., Lopez-Orozco, J. A., Chacon, J., de la Cruz, J. M., 2016b. Developing web & TwinCAT PLC-based remote control laboratories for modern web-browsers or mobile devices. In: 2016 IEEE Conference on Control Applications.

Bermudez-Ortega, J., Besada-Portas, E., Lopez-Orozco, J. A., de la Cruz, J. M., 2017. A new open-source and smart device accessible remote control laboratory. In: 4th Experiment@ International Conference.

Bhute, V. J., Inguva, P., Shah, U., Brechtelsbauer, C., 2021. Transforming traditional teaching laboratories for effective remote deliveryâa review. Education for Chemical Engineers 35, 96–104.

Carballo, J. A., Bonilla, J., Roca, L., Berenguel, M., 2018. New low-cost solar tracking system based on open source hardware for educational purposes. Solar Energy 174, 826–836.

Chacón, J., Besada-Portas, E., Carazo-Barbero, G., López-Orozco, J. A., 2021. Enhancing EJsS with extension plugins. Electronics 10 (3).

Chaos, D., Chacón, J., Lopez-Orozco, J. A., Dormido, S., 2013. Virtual and remote robotic laboratory using ejs, matlab and labview. Sensors 13 (2).

Cyton Webpage, 2022. https://robots.ros.org/cyton-gamma/, Accessed: 2022-03-15.

de la Torre, L., Sanchez, J., Dormido, S., 2016. What remote labs can do for you. Physics Today 69.

Dobot Webpage, 2022. https://www.dobot.cc, Accessed: 2022-03-15.

dos Santos Lopes, M. S., Pacheco-Gomes, I., Trindade, R. M. P., da Silva, A. F., de C. Lima, A. C., 2017. Web environment for programming and control of a mobile robot in a remote laboratory. IEEE Trans. on Learning Tech. 10 (4).

EJsS Webpage, 2022. http://fem.um.es/Ejs, Accessed: 2022-03-15.

Esquembre, F., García Clemente, F. J., Chicón, R., Wee, L. K., Kwang, L., Tan, D., 10 2019. Easy java/javascript simulations as a tool for learning analytics. In: 10th International Conference on Applied Innovations in IT, (ICAIIT).

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.

Faulconer, E. K., Gruss, A. B., 2018. A review to weigh the pros and cons of online, remote, and distance science laboratory experiences. The International Review of Research in Open and Distributed Learning 19 (2).

Filipović, F., Petronijević, M., Mitrović, N., Banković, B., 2017. Affordable virtual laboratory for remote control of variable speed drives. In: Int. Conf. on Information, Communication and Energy Systems and Tech. (ICEST).

Fukumoto, H., Yamaguchi, T., Ishibashi, M., Furukawa, T., 2021. Developing a remote laboratory system of stepper motor for learning support. IEEE Transactions on Education 64 (3), 292–298.

Galan, D., Isaksson, O., Rostedt, M., Enger, J., Hanstorp, D., de la Torre, L., 2018. A remote laboratory for optical levitation of charged droplets. European Journal of Physics 39.

Gamage, K. A. A., Wijesuriya, D. I., Ekanayake, S. Y., Rennie, A. E. W., Lambert, C. G., Gunawardhana, N., 2020. Online delivery of teaching and laboratory practices: Continuity of university programmes during COVID-19 pandemic. Education Sciences 10 (10).

Gomes, L., 2009. Current trends in remote laboratories. IEEE Transactions on Industrial Electronics 56.

Goncalves-López-Medrano, D. A., Chacón, J., López-Orozco, J. A., Besada-Portas, E., 2021. Laboratorio remoto para el robot educativo Dobot Magician. In: XLII Jornadas de Automática.

Jara, C. A., Candelas, F. A., Puente, S. T., Torres, F., 2011. Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. Computers & Education 57.

Jiménez, R., Sanchez, O. A., Mauledeox, M., 2018. Remote lab for robotics applications. International Journal of Online and Biomedical Eng. 14 (1).

Kostaras, N., Xenos, M., Skodras, A., 2011. Evaluating usability in a distance digital systems laboratory class. IEEE Transactions on Education 54 (2).

Letowski, B., Lavayssière, C., Larroque, B., Luthon, F., 2019. An open source remote laboratory network based on a ready to use solution: LABOREM. In: Int. Conf. of Education, Research and Innovation (ICERI).

Liddell, T. M., Kruschke, J. K., 2018. Analyzing ordinal data with metric models: What could possibly go wrong? Journal of Experimental Social Psychology 79, 328–348.

Losada-Gutiérrez, C., Espinosa, F., Santos-Pérez, C., Marrón-Romera, M., Rodríguez-Ascariz, J., 2020. Remote control of a robotic unit: A case study for control engineering formation. IEEE Transactions on Education 63 (4).

Ma, J., Nickerson, J. V., 2006. Hands-on, simulated, and remote laboratories: A comparative literature review. ACM Computing Surveys 38 (7).

Marín, R., Sanz, P. J., Nebot, P., Wirz, R., 2005. A multimodal interface to control a robot arm via the web: A case study on remote programming. IEEE Transactions on Industrial Electronics 52 (6).

Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., Obdržálek, D. (Eds.), 2020. Robotics in Education. Springer.

Mindstorms Webpage, 2022. https://www.lego.com/es-es/themes/mindstorms, Accessed: 2022-03-15.

Moway Wepbage, 2022. http://moway-robot.com, Accessed: 2022-03-15.

Nao Webpage, 2022. https://aliverobots.com/robot-nao/, Accessed: 2022-03-15.

Nickerson, J. V., Corter, J. E., Esche, S. K., Chassapis, C., 2007. A model for evaluating the effectiveness of remote engineering laboratories and simulations in education. Computers & Education 49 (3).

Node.js Webpage, 2022. https://nodejs.org/es/, Accessed: 2022-03-15.

Owi Webpage, 2022. https://owirobot.com/, Accessed: 2022-03-15.

Papadakis, S., Kalogiannakis, M., 2020. Handbook of Research on Using Educational Robotics to Facilitate Student Learning. IGI Global.

PeggyJS Webpage, 2022. https://peggyjs.org/, Accessed: 2022-03-15.

Pepper Webpage, 2022. https://aliverobots.com/robot-pepper/, Accessed: 2022-03-15.

Pydobot Webpage, 2022. https://github.com/luismesas/pydobot, Accessed: 2022-03-15.

Python Webpage, 2022. https://www.python.org/, Accessed: 2022-03-15.

Raspberry PI Webpage, 2022. https://www.raspberrypi.org/, Accessed: 2022-03-15.

Saenz, J., de la Torre, L., Chacón, J., Dormido, S., 2020. Learning planar robots with an open source online laboratory. In: 21th IFAC World Congress.

Scorbot Webpage, 2022. https://intelitek.com/scorbot-er-4u-educational-robot/, Accessed: 2022-03-15.

Socket.io Webpage, 2022. https://socket.io/, Accessed: 2022-03-15.

Sánchez-Herrera, R., Marquez, M. A., , Andújar, J. M., 2020. Easy and secure handling of sensors and actuators as cloud-based service. IEEE Access 8.

Vagaš, M., Sukop, M., Varga, J., 2016. Design and implementation of remote lab with industrial robot accessible through the web. Applied Mechanics and Materials 859.

ZeroMQ Webpage, 2022. https://zeromq.org/, Accessed: 2022-03-15.