LABORATORIO REMOTO PARA EL ROBOT EDUCATIVO DOBOT MAGICIAN

Daniela A. Goncalves-López-Medrano, Jesús Chacón, José A. López-Orozco, Eva Besada-Portas
Departamento de Arquitectura de Computadores y Automática. Universidad Complutense de Madrid (UCM)
{dgoncalv,jeschaco,jalopez,ebesada}@ucm.es

Resumen

Este artículo presenta una metodología para dar acceso remoto, a través de una aplicación web, al robot educativo Dobot Magician. El procedimiento consiste en 1) ampliar la funcionalidad de ReNoLabs, el servidor utilizado en la Universidad Complutense de Madrid (UCM) para poner en marcha sus laboratorios de Control de Sistemas, y en 2) implementar un controlador para el robot en Python. Además, se despliega todo el software del laboratorio en una Raspberry PI y se utiliza EJsS, reforzado mediante un "Plugin", para gestionar de forma centralizada la puesta en marcha del laboratorio. Finalmente, también se presenta un ejemplo de aplicación práctica del laboratorio remoto.

Palabras clave: EJsS, Node.js, Raspberry Pi.

1. INTRODUCCIÓN

La realización de prácticas de laboratorio permite a los alumnos de Ciencias Experimentales e Ingeniería aplicar, sobre casos reales, los conocimientos adquiridos en las sesiones teóricas y adquirir habilidades y competencias útiles para adaptarlos al entorno laboral [26]. Sin embargo, para que diferentes alumnos puedan realizarlas de forma simultánea es conveniente que el laboratorio multiplique sus puestos, algo que no es posible cuando el hardware necesario para equiparlos tiene un elevado coste económico. Además, los laboratorios presenciales limitan el tiempo de acceso al hardware de los alumnos, y por lo tanto, el número de experiencias que pueden realizar con él [22].

Los inconvenientes anteriores pueden ser mitigados a través del desarrollo de laboratorios remotos, que dan acceso a través de Internet a los dispositivos/hardware del laboratorio. Además, su uso se encuentra avalado por diferentes estudios que muestran que aportan beneficios educativos similares a los laboratorios presenciales [17, 20, 25, 29]. Finalmente, destacar que el nuevo escenario educativo, con limitaciones de aforo adicionales impuestas por la pandemia de la COVID-19, ha acelerado el proceso de adaptación de las prácticas presenciales a sus variantes remotas [21].

En este contexto, los autores de este artículo han decido dar acceso remoto a uno de los brazos robóticos de los laboratorios presenciales de las titulaciones de Informática y de Ingeniería Electrónica de Comunicaciones de la Universidad Complutense de Madrid (UCM). Para lograrlo, han seguido una metodología *similar* a la que utilizan para desarrollar los laboratorios remotos de Control de Sistemas de la misma universidad [12, 14], en la que un servidor (programado en Node.js) sirve 1) de gestor de la pagina web de las experiencias y 2) de pasarela entre su interfaz gráfica (diseñada en Easy JavaScript Simulations, EJsS) y el controlador (definido en un lenguaje de programación estándar) de los dispositivos del laboratorio. Además, dicha metodología contempla la posibilidad de desplegar todo el software del laboratorio en una Raspberry PI y gestionarlo de forma centralizada desde un EJsS con funcionalidades añadidas mediante un *Plugin* [10, 13, 15].

Más concretamente, para este laboratorio, y tal y como se muestra en la Figura 1, el brazo robótico del laboratorio es el Dobot Magician [1], que es controlado a través del puerto USB de la Raspberry PI mediante un programa escrito en Python, con el que interactúa con una nueva versión de nuestro servidor Node.js, a la que se le han añadido protocolos estandarizados de comunicación.

En este artículo se describen todas las capas software desarrolladas para crear este nuevo laborato-

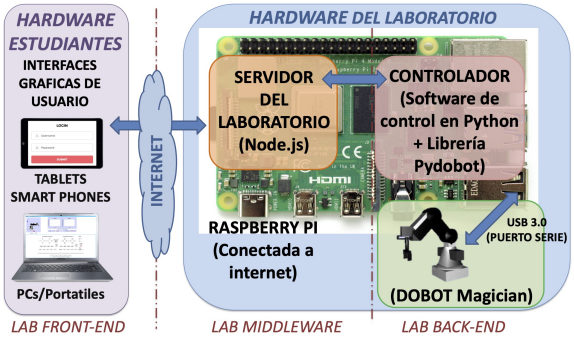


Figura 1: Elementos del laboratorio remoto

https://doi.org/10.17979/spudc.9788497498043.232

rio remoto de robótica educativa, la metodología seguida para obtenerlas y las herramientas que las sustentan. Además, también se contextualiza el trabajo y se presentan experiencias que pueden realizar los alumnos con este laboratorio remoto.

2. CONTEXTUALIZACIÓN

Esta sección plantea la importancia de la robótica y de sus laboratorios remotos en la enseñanza.

2.1. ROBÓTICA EDUCATIVA

En la actualidad, la robótica es un área tecnológica relevante en el sector industrial y de servicios, que extiende su protagonismo de forma progresiva al ámbito cotidiano y a la enseñanza. La robótica educativa se define como la concepción, creación y puesta en funcionamiento, con fines didácticos, de objetos tecnológicos que reproducen los procesos y herramientas robóticas usadas cotidianamente en nuestro entorno social, productivo y cultural [30]. Además, la robótica educativa promueve los procesos prácticos de enseñanza y aprendizaje, fomenta la motivación y el interés del estudiante, y es aplicable a todos los niveles de enseñanza (primaria, secundaria, universitaria, y formación profesional [28]).

El interés despertado por esta nueva forma de enseñanza ha promovido el desarrollo de numerosas plataformas robóticas educativas, entre las que podríamos mencionar los robots manipuladores Dobot [1] y Scorbot [8], los robots humanoides NAO [4] y Pepper [5], o los robots móviles sobre ruedas Lego Mindstorm [2] y Moway [3]. Estos robots comerciales constituyen soluciones educativas completas, incluyen hardware y software para que los alumnos interactúen con ellos y programen su comportamiento mediante diferentes lenguajes, y material didáctico adicional. Sin embargo, no suelen estar diseñados para interactuar con ellos de forma remota.

2.2. LABORATORIOS REMOTOS

Los laboratorios remotos para robótica educativa responden a las mismas necesidades que los desarrollados en otras áreas, tal y como muestran los ejemplos de laboratorios remotos que hay en la literatura para robots móviles [11, 16, 18, 19] y manipuladores [23, 27, 24, 31, 32]. Entre las características de estos últimos, resumidas en la Tabla 1 según la fecha de creación del laboratorio, destaca el uso de herramientas comunes para el desarrollo de la interfaz de la experiencia y para implementar el controlador de cada robot.

Tabla 1: Lab. remotos de brazos robóticos

	Interfaz experiencia	Controlador	Robot
[27]	Java	Mentor	Mentor
[23]	EJS	Scorbot	Scorbot
[32]	EJS	Arduino	Almega
[24]	C	Arduino + Rasp	
[31]	EJsS	Arduino	Propio
Este	EJsS	Python + Rasp	Dobot

La propuesta de este trabajo, recogida en la última fila de la tabla, comparte la herramienta común de desarrollo de la interfaz de la experiencia (EJsS) y utiliza una Raspberry PI no sólo para interactuar con el robot a través del controlador programado en Python, sino también para desplegar el servidor de páginas web de laboratorio. Además, es configurado de forma centralizada desde EJsS, una novedad de este trabajo que podría facilitar, modificando las capas de software adecuadas, el uso de la metodología sobre otras plataformas robóticas.

3. HERRAMIENTAS

En esta sección se presentan las herramientas hardware y software que han sido utilizadas y/o adaptadas para nuestro laboratorio remoto.

3.1. EL BRAZO ROBÓTICO DOBOT MAGICIAN

El brazo robótico DOBOT Magician [1], mostrado en la Figura 2, está formado por una base, un brazo trasero, un antebrazo y, opcionalmente, un efector final. Además, cuenta con tres o cuatro articulaciones de revolución (J1, J2, J3 y J4 - apareciendo la última cuando se instala un efector final) que le proporcionan el rango de trabajo mostrado en la Figura 3.

El software de control proporcionado por el fabricante permite moverlo de forma incremental

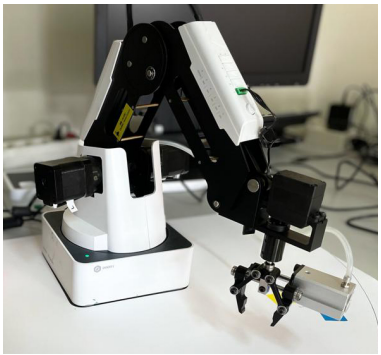
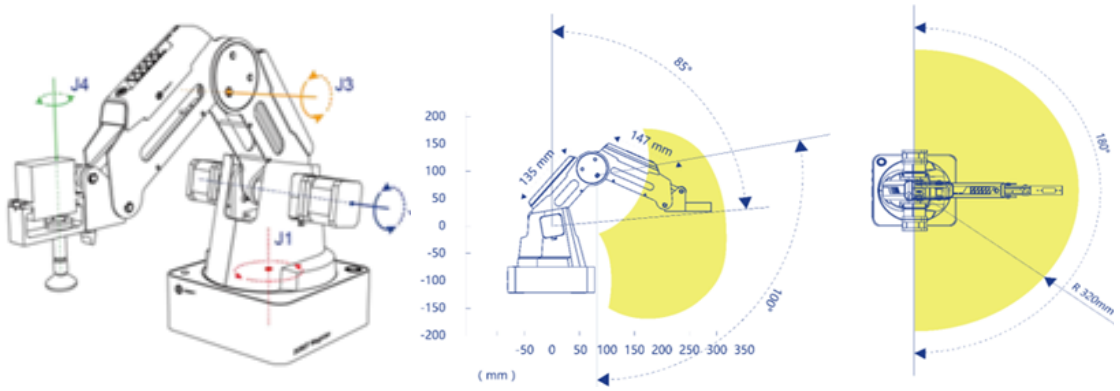


Figura 2: Imagen del Dobot Magician

https://doi.org/10.17979/spudc.9788497498043.232



(a) Articulaciones

(b) Espacio de trabajo

Figura 3: Caracterización del Dobot Magician (fuente de imágenes [1])

(modo *jogging*), hasta un punto destino (modo Point To Point - *PTP*) o siguiendo un arco (modo *ARC*). Además su comportamiento puede ser programado desde diferentes entornos de programación (e.g. C, Python, ROS, Qt, Matlab, Labview, Android, Blockly) y desde una aplicación de control propia, lo que le convierte en una herramienta ideal para diferentes niveles educativos.

3.2. RASPBERRY PI

La Raspberry PI es un ordenador de placa única, originalmente desarrollado para promover la Informática entre estudiantes de primaria y secundaria. Sin embargo, su bajo coste, tamaño reducido y conectividad la ha convertido en un sistema embebido de referencia para muchas aplicaciones.

En nuestro caso, es conveniente utilizar la Raspberry Pi 4, ya que constituye el soporte computacional de todo el laboratorio (i.e. en ella se ejecuta el controlador y el servidor) y porque dispone, al igual que el Dobot, de un puerto USB 3.0.

3.3. PYTHON

El Dobot Magician proporciona una API (Interfaz de Programación de Aplicaciones) que es accesible desde Python a través de la librería `pydobot` [6].

La API dispone de dos tipos de órdenes. Las de *ejecución inmediata* se procesan instantáneamente (cuando son recibidas) con independencia de si el Dobot está realizando otra acción. Las de *ejecución encolada* se almacenan en una lista FIFO (*First-In First-Out*) y se procesan según su orden de llegada. Las instrucciones soportadas por la API controlan el movimiento del brazo, obtienen la posición y velocidad de las articulaciones, lo trasladan a su configuración inicial y acceden a toda su funcionalidad.

Sin embargo, la librería `pydobot`, utilizada en el controlador del Dobot, únicamente implementa parte de la funcionalidad de la API. Por este motivo, también se ha ampliado en este proyecto el conjunto de funciones disponibles en esta librería.

3.4. SERVIDOR NODE.JS

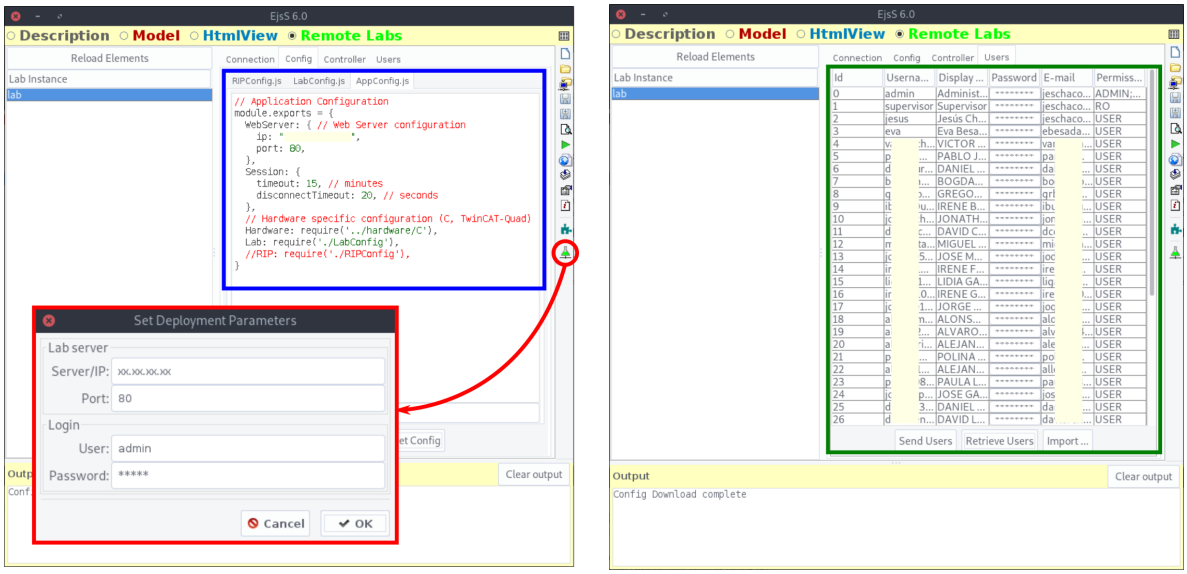
El servidor de laboratorios remotos *ReNoLabs* [12], desarrollado en Node.js, soporta la gestión del laboratorio (centralizada desde EJsS) y la creación de su infraestructura web, proporcionando entre otras las siguientes funciones: la gestión de usuarios y control de acceso, el alojamiento de la interfaz web de la experiencia, la comunicación ente dicha interfaz y el controlador, y el registro de datos experimentales y eventos.

3.5. EASY JAVASCRIPT SIMULATIONS

La herramienta EJsS se utiliza para crear la aplicación cliente que constituye el interfaz experimental del laboratorio remoto. Para facilitar el proceso, se utiliza la versión 6.0 de EJsS, ya que incorpora el soporte de *Plugins*, que es un nuevo mecanismo para extender la funcionalidad e interfaz de EJsS a las necesidades de sus usuarios [15].

El *Plugin* utilizado en este caso facilita la integración, en EJsS, del flujo de trabajo seguido para el diseño de laboratorios remotos. En concreto, y a modo de ejemplo, el botón rodeado por un círculo rojo en la Figura 4a permite desplegar la interfaz gráfica y la descripción desarrollada en EJsS sobre el servidor elegido, el Editor enmarcado en azul en la Figura 4a permite cambiar la configuración de despliegue en el servidor y modificar las variables usadas por el servidor para el intercambio de datos entre la interfaz experimental y el controlador, y el Editor enmarcado en verde en la Figura 4b permite gestionar los usuarios del laboratorio.

https://doi.org/10.17979/spudc.9788497498043.232



(a) Configuración del despliegue.

(b) Editor de usuarios

Figura 4: EJsS con funcionalidades añadidas por el *Plugin* del laboratorio

4. DESARROLLO

Esta sección describe las características más relevantes de los elementos desarrollados para nuestro laboratorio remoto, que estarán a disposición de la comunidad educativa, junto con las instrucciones necesarias para su despliegue, en el repositorio [7].

4.1. CONTROLADOR

La funcionalidad del controlador la constituyen dos tipos de tareas diferentes: 1) la lectura de la posición del efector final y de las orientaciones de la articulaciones del robot, y 2) la recepción y ejecución de las órdenes enviadas por el servidor.

Cada tarea se implementa en un hilo de ejecución independiente (el primero correspondiente al programa principal y el segundo ejecutado como respuesta a cada orden recibida). Esto se ha hecho así por dos motivos. El primero es la diferente naturaleza temporal de cada tarea: mientras que la lectura de datos se realiza periódicamente con una base de tiempos fija, la interpretación de las órdenes del servidor se hace de forma asíncrona. El segundo es el modelo de comunicaciones utilizado por cada tarea, que se discute en detalle en el siguiente apartado.

4.2. SERVIDOR

Desde el punto de vista de las comunicaciones, se establece una estructura en capas que desacopla a los componentes software del laboratorio: la pagina web de la experiencia (creada con EJsS), el servidor del laboratorio (llamado *ReNoLabs*) y el controlador del Dobot (progamado en Python).

Las comunicaciones entre el cliente EJsS y *ReNoLabs* son soportadas por el elemento *Lab* del *Plugin* desarrollado para facilitar la interacción de ambas herramientas. De esta manera, basta con configurar los datos de conexión en EJsS (ver Figura 4a) para que la aplicación reciba automáticamente los datos enviados por el servidor y sea capaz de enviar órdenes a discreción. La instancia del elemento *Lab* mantiene una lista de variables con los valores sincronizados con el servidor, permite utilizar una memoria intermedia para adaptar las velocidades de recepción de datos y de visualización, y proporciona métodos para realizar el envío de datos al servidor. Para la transferencia de datos utiliza el protocolo de comunicaciones específico de *ReNoLabs*, que está basado en JSON (JavaScript Object Notation) y que puede funcionar sobre una conexión HTTP o WebSocket.

La arquitectura modular de ReNoLabs permite definir las comunicaciones entre éste y el controlador del Dobot mediante la implementación de un objeto que actúa como interfaz entre ambos. Para este fin, se ha decidido utilizar la librería `ZeroMQ` por los siguientes motivos. En primer lugar, porque proporciona un modelo de comunicaciones de E/S asíncrono y de alto rendimiento. En segundo, porque soporta diferentes patrones de comunicaciones, entre ellos publicador-subscriptor y petición-respuesta, que resultan adecuados para este caso. En tercero, por su robustez, debida a su popularidad y a su amplia comunidad de desarrolladores. Por último, por la flexibilidad que aporta a

https://doi.org/10.17979/spudc.9788497498043.232

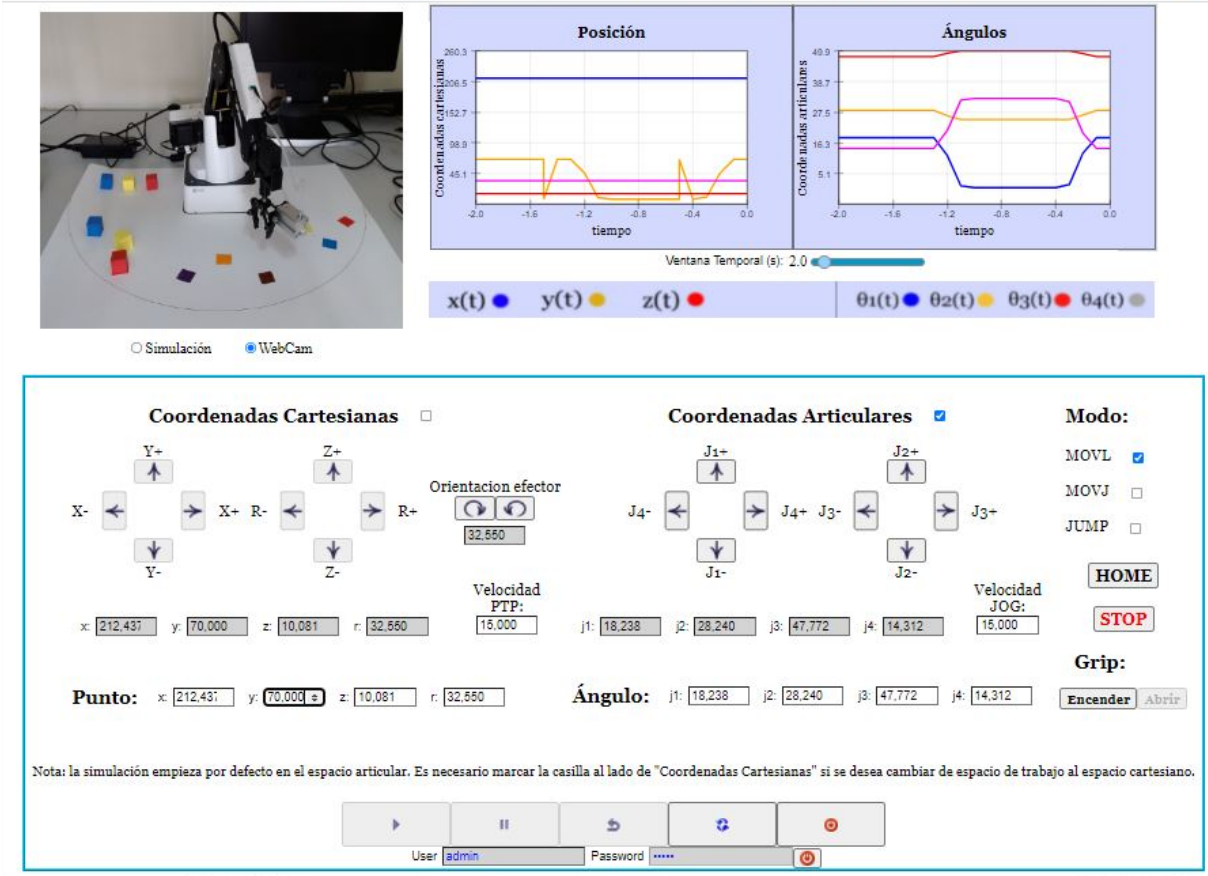


Figura 5: Interfaz experimental del laboratorio

la hora de integrar códigos de diferente naturaleza, al estar implementada en los principales lenguajes de programación (incluidos Python y Node.js).

Más concretamente, para establecer la comunicación entre el servidor y el controlador, se crean dos *sockets* TCP (Transmission Control Protocol) locales de comunicación, tal y como se muestra en el Código 1. Uno de ellos se inicializa para funcionar como petición-respuesta, y el otro para funcionar con el modelo publicador-subscriptor. En este caso, al estar ejecutando ambas aplicaciones en la Raspberry PI, se utiliza una IP local en la inicialización de los sockets. También sería posible utilizar dos dispositivos diferentes sin más que configurar las respectivas IP de acceso. Esta naturaleza distribuida aporta otro grado de libertad, que puede ser útil, por ejemplo, si se dispone de diferentes brazos robóticos, controlados desde diferentes Raspberry PIs, a los que se desee acceder a través de un único servidor del laboratorio.

Código 1: Uso de sockets ZeroMQ en Python

```
command = context.socket(zmq.REP)
notify = context.socket(zmq.PUB)
command.bind("tcp://127.0.0.1:5555")
notify.bind("tcp://127.0.0.1:5556")
```

También es importante resaltar una funcionalidad previa del servidor *ReNoLabs*, en concreto su soporte de RIP (Remote Interoperability Protocol), ya que permite la integración de este laboratorio remoto en otras plataformas. Más en concreto, posibilita la integración del laboratorio en *UNILabs* [9], la red universitaria de laboratorios interactivos formada por diversas universidades españolas, que está soportada por una plataforma basada en el sistema de gestión del aprendizaje Moodle LMS.

Finalmente, indicar que aunque para este laboratorio remoto se ha hecho una implementación *ad-hoc* de la comunicación entre el servidor y el controlador, el módulo añadido al servidor es genérico, y puede ser utilizado para comunicar con cualquier otro controlador que utilice la misma librería y se adhiera al protocolo de comunicaciones estándar utilizado por *ReNoLabs*.

4.3. INTERFAZ DEL LABORATORIO

La interfaz web para acceder al laboratorio sigue la estructura habitual de este tipo de aplicaciones, tal y como se observa en la Figura 5.

En la esquina superior izquierda se muestra la imagen del Dobot, obtenida por una cámara web que

https://doi.org/10.17979/spudc.9788497498043.232

está situada en el laboratorio y conectada a la Raspberry PI a través del puerto USB. La transmisión de video se realiza con la aplicación *mjpg-streamer*, que captura la imagen y la emite a través de una conexión HTTP. A la derecha de la imagen, dos gráficas muestran la evolución temporal de las variables de interés: la posición del efector final y los ángulos de las articulaciones del Dobot.

En la parte central, se disponen y organizan en varios paneles todos los elementos que permiten interactuar con el Dobot. A la izquierda, el panel de *Coordenadas Cartesianas* permite realizar un control del movimiento en el espacio cartesiano (XYZ+Roll), i.e. permite especificar cualquier punto del espacio de trabajo en el que se desea que se posicione el efector final del Dobot. En la parte central, el panel de *Coordenadas Articulares* permite hacer un control en el espacio articular, de forma que las coordenadas especificadas corresponden directamente a los ángulos requeridos para las articulaciones (J1-J4) del Dobot. En la zona derecha hay algunos botones adicionales que permiten (de arriba abajo): configurar el tipo de modo de movimiento punto a punto (en linea recta según los ejes XYZ, con un movimiento articular, y de forma similar al anterior pero con distancia de aproximación y despegue), mandar al robot a la posición de inicio, hacer una parada de emergencia, y controlar el efector final.

En la parte inferior, la interfaz proporciona una barra de control de ejecución y cuadro de autenticación (para introducir las credenciales necesarias para acceder al laboratorio). Cabe indicar que estos dos últimos grupos de elementos gráficos forman parte de los que el *Plugin* incorpora a los elementos visuales de EJsS.

5. EJEMPLO DE APLICACIÓN

Nuestro laboratorio remoto permite usar, a través de Internet, el Dobot Magician. Resulta un complemento ideal a las prácticas presenciales que realizan los alumnos sobre cinemática de robots manipuladores, ya que les permite verificar, sobre el robot real y en cualquier momento que tengan disponible, los resultados obtenidos durante la sesión presencial del laboratorio. De este modo, los alumnos pueden finalizar (o ampliar el número de experiencias relacionadas con) la práctica que comienzan a desarrollar en el laboratorio presencial y, que por las limitaciones propias de éste (horario restringido, un único Dobot) no todos pueden finalizar en una única sesión.

También permite que los alumnos detecten sus errores de forma progresiva. Para ello, los alumnos utilizan: 1) Matlab de forma local para llevar

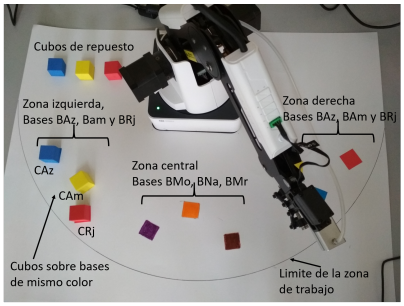


Figura 6: Disposición del entorno experimental

a cabo las operaciones, funciones y comprobaciones necesarias; y 2) el laboratorio remoto para verificar sobre el robot real si los cálculos realizados son correctos, comparando si el comportamiento del Dobot es el esperado. Si esto no ocurre, pueden corregir sus cálculos y volver a comprobarlos sobre el Dobot, en cualquier horario, en el que no esté siendo utilizado por otro alumno.

En concreto, en la práctica diseñada, los alumnos deben 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, calculan las matrices ${}^{i-1}A_i$ e implementan una función para la cinemática directa del robot. A continuación, deben verificar que su funcionamiento es correcto sobre el Dobot, que tienen a su disposición de forma remota.

Tras finalizar esta primera fase, deben diseñar el movimiento necesario para llevar a cabo una operación de montaje sencilla con los elementos disponibles en la Figura 6. Ésta tiene como finalidad que 1) el alumno aprenda a manejar diferentes ejes, puntos de aproximación y despegue adecuados; y que 2) calcule la trayectoria más adecuada para evitar colisiones del robot manipulador durante la operación. Para realizar esta parte de la experiencia al alumno se le proporcionan las dimensiones de los cubos (rojo - CRj, amarillo - CAm, y azul - CAz) que el robot debe colocar en una posición determinada desde una configuración inicial. Para simplificar el problema, las posiciones iniciales donde pueden estar situados los cubos están prefijadas (BAm, BRj y BAz) a ambos lados del robot. 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 final diferente. Los alumnos deben definir unos movimientos/posiciones relativos a las posiciones inicial y final de los cubos, obtener la secuencia de pasos (puntos) por los que debe pasar el robot y las acciones del efector final (pinza del Dobot).

https://doi.org/10.17979/spudc.9788497498043.232

Tras verificar mediante simulación que la secuencia de pasos obtenida permite realizar la operación con éxito, los alumnos se conectan de nuevo al laboratorio remoto para verificar que la secuencia es correcta. En esta última fase el alumno debe situar los bloques en su posición inicial (usando los cursores para movimiento manual y utilizando bloques de repuesto cuando sea necesario). Con los cubos en su posición inicial correcta, ejecuta paso a paso la secuencia de puntos calculada y verifica sobre el Dobot si ésta es correcta para llevar a cabo con éxito la operación planteada.

6. CONCLUSIONES

En este artículo se presenta un laboratorio remoto para el robot educativo Dobot Magician, que puede ser desplegado por menos de 100€ (gasto aproximado de una Raspberry PI 4), al haber sido desarrollado con herramientas software de código abierto (EJsS, Node.js y Python). Además, la metodología utilizada para desarrollarlo hace que su configuración se centralice en EJsS, la misma herramienta que permite al usuario modificar la interfaz gráfica del laboratorio (y adaptarla a cada experiencia concreta). También es relevante mencionar, que la metodología seguida ha sido aplicada, con algunas diferencias en los protocolos de comunicación y lenguaje de programación del controlador, en otros laboratorios. Y que el hecho de utilizar el servidor *ReNoLab* permite que el laboratorio desarrollado sea también compatible con el proyecto *UNILabs* [9] del Grupo de Educación en Control de CEA.

Como trabajos futuros se plantean los siguientes. Consideramos conveniente pulir la interfaz gráfica de la experiencia (reorganizando su información en pestañas) e incluir en ella un editor de secuencias de instrucciones. También deseamos utilizarlo el curso que viene en las asignaturas de Robótica.

Agradecimientos

Este trabajo ha sido financiado por los Proyectos de Innovación Educativa y Mejora de la Calidad Docente 2019-139 y 2021-39 de la UCM.

English summary

REMOTE LABORATORY FOR THE EDUCATIONAL DOBOT MAGICIAN

Abstract

This article presents a methodology to give remote access, through a web application, to the educational robot Dobot Magician. The procedure consists in 1) expanding the functionality of ReNoLabs, the server used at the University Complutense of Madrid (UCM) to set up its Control Systems laboratories, and in 2) implementing a robot controller in Python. In addition, it deploys all the laboratory software on a Raspberry PI and uses EJsS, enhanced by a Pluging, to centralize the management of the laboratory. Finally, an example of a practical application of the remote laboratory for the students is also presented.

Keywords: EJsS, Node.js, Raspberry Pi.

Referencias

[1] Dobot. https://www.dobot.cc. Accessed: 2021-06-15.

[2] Mindstorms. https://www.lego.com/es-es/themes/mindstorms. Accessed: 2021-06-15.

[3] Moway. http://moway-robot.com. Accessed: 2021-06-15.

[4] Nao. https://aliverobots.com/robot-nao/. Accessed: 2021-06-15.

[5] Pepper. https://aliverobots.com/robot-pepper/. Accessed: 2021-06-15.

[6] Pydobot. https://github.com/luismesas/pydobot. Accessed: 2021-06-15.

[7] Repositorio del software desarrollado. https://gitlab.com/jcsombria/jj-aa-2021. Accessed: 2021-06-15.

[8] Scorbot. https://intelitek.com/scorbot-er-4u-educational-robot/. Accessed: 2021-06-15.

[9] UNILabs. https://unilabs.dia.uned.es/. Accessed: 2021-06-15.

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

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

https://doi.org/10.17979/spudc.9788497498043.232

[12] J. Bermudez-Ortega, E. Besada-Portas, L. de la Torre, J.A. Lopez-Orozco, and J.M. de la Cruz. Lightweight node.js ejss-based web server for remote control laboratories. In *IFAC Symposium on Advances in Control Education*, 2016.

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

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

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

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

[17] L. de la Torre, J. Sanchez, and S. Dormido. What remote labs can do for you. *Physics Today*, 69.

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

[19] E. Fabregas, G. Farias, S. Dormido-Canto, M. Guinaldo, J. Sánchez, and S. Dormido-Bencomo. Platform for teaching mobile robotics. *Journal of Intelligent Robotic Systems*, 81, 2016.

[20] E.K. Faulconer and A.B. Gruss. 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), 2018.

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

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

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

[24] R. Jiménez, O. Avies Sanchez, and M. Mauledeox. Remote lab for robotics applications. *International Journal of Online and Biomedical Engineering*, 14(1), 2018.

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

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

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

[28] Mu. Merdan, W. Lepuschitz, G. Koppensteiner, R. Balogh, and D. Obdržálek, editors. *Robotics in Education*. Springer, 2020.

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

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

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

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

https://doi.org/10.17979/spudc.9788497498043.232