Plataforma robótica para aprendizaje activo multidisciplinar

Álvaro Rodríguez del Nozal
Departamento de ingeniería
Universidad Loyola Andalucía
Sevilla, Spain
arodriguez@uloyola.es

Fabio Gómez-Estern
Departamento de ingeniería
Universidad Loyola Andalucía
Sevilla, Spain
fgestern@uloyola.es

Daniel Gutierrez
Departamento de ingeniería
Universidad Loyola Andalucía
Sevilla, Spain
dgutierrez@uloyola.es

***Abstract*—Las profesiones relacionadas con la Automática requieren conocimientos y competencias muy diversas, entre las que se incluyen materias tecnológicas, como la informática, robótica, sensores, aparamenta industrial, etc., y aspectos más abstractos, como modelado matemático, algoritmos de optimización o aprendizaje automático. En este trabajo se emplea un robot industrial, un dispositivo empotrado IoT con cámara web, y un servidor ejecutando algoritmos MATLAB para el desarrollo de prácticas de laboratorio destinadas a desarrollar competencias múltiples, en las materias mencionadas.**

***Index Terms*—Industria 4.0, Internet of Things, Educación en Automática, Robótica, Visión por computador, impresión aditiva.**

I. Introducción

La formación en el área de Automática es compleja y diversa. Por una parte, requiere un conocimiento especializado de distintas tecnologías: robots, sensores, actuadores, informática industrial, etc. Por otra parte, requiere una preparación abstracta (matemáticas, modelado, análisis de sistemas) y algorítmica (programación, optimización, aprendizaje automático). En un plan de estudios de grado, los estudiantes asisten a sesiones prácticas de laboratorio de cada una de estas materias específicas, pero en el nivel de máster este enfoque no es suficiente, al requerir habilidades de resolución de problemas más realistas, de naturaleza multidisciplinar. En la práctica, un ingeniero debe ser integrador de equipos, conceptos y procedimientos provenientes de distintas áreas. En este trabajo se presenta un equipo de prácticas de desarrollo propio y un conjunto de actividades con el fin de diseñar experiencias prácticas complejas en las que se desarrollen competencias de todas las materias mencionadas, de una forma integrada. La plataforma se ha desarrollado empleando elementos existentes en un típico laboratorio de ingeniería, con mínima inversión adicional. El resultado es un almacén automático robotizado en el que se programan las trayectorias del manipulador tanto a bajo nivel (lenguaje propio del robot) como en un entorno de desarrollo de algoritmos de métodos cuantitativos.

En la literatura podemos encontrar multitud de plataformas docentes de robots que van desde la construcción de pequeños robots seguidores de lineas hasta el estudio de microcontroladores que componen a éstos [1]- [4]. En primer lugar consideramos la programación de robots. Esta es una competencia típica a desarrollar en un segundo o tercer curso de un grado de la rama industrial, como el de ingeniería electromecánica o ingeniería de las tecnologías industriales. Existen multiples entornos en los que desarrollar estas capacidades, pero siempre es preferible que se trate de un sistema de programación offline, con capacidad de simulación y otras funcionalidades que proporcionen feedback al alumno: simulación 3D, interacción con piezas y elementos en el espacio de trabajo, interacción con otros robots, etc. Uno de los entornos más satisfactorios en este sentido es el RobotStudio de ABB [5], generalmente facilitado en condiciones especiales para los centros formativos. Este es el entorno que emplearemos, para programar robots manipuladores concretamente con el lenguaje RAPID.

Otra habilidad tecnológica que es preciso desarrollar en los futuros ingenieros y que es parte fundamental del paradigma de la Industria 4.0 es la programación de aplicaciones distribuidas mediante dispositivos IoT como Raspbery Pi [6] y Arduino [7]. La idea de trabajar con estos dispositivos va más allá del concepto de dominio de la programación y los microcontroladores propios de las titulaciones TIC. El movimiento Industria 4.0 está relacionado con el movimiento maker, y la posibilidad de desarrollar soluciones ad hoc [8] en el seno de cualquier sistema de producción sin necesidad de incurrir en altos costes ni de recurrir a intermediarios. Y una de ls plataformas en las que esto es posible, con una curva de aprendizaje rápida es el entorno Raspberry Pi con el lenguaje de programación Phyton [9].

Una gran oportunidad para el desarrollo de aplicaciones con capacidad de reaccionar en entornos industriales sin gran coste de instalación es la visión por computador. Hasta hace pocos años, esta tecnología ha sido del interés de una comunidad minoritaria muy especializada, existiendo una creencia general de que es preciso gozar de grandes conocimientos para poder explotarla en situaciones reales. Sin embargo, en la parte final de este trabajo se pondrá en evidencia que esto no es así. Con el mismo lenguaje de programación Phyton y empleando la plataforma Raspberry Pi, los alumnos desarrollarán un algoritmo de visión a través

de la cámara incorporada para determinar la información que el sistema necesita para actuar sobre su entorno. Más concretamente, se tratará de determinar a partir de una imagen capturada el estado completo del almacén automático. Mediante algoritmos tan sencillos como la identificación de los colores predominantes en una zona concreta, y mediante la adecuada transformación de coordenadas del sistema de referencia plano de la imagen al tridimensional del almacén, podremos identificar automáticamente qué tipo de pieza hay ubicada en cada posible hueco del almacén, evitando tener que cargar en memoria de programa esta información al inicio de la operación. Las bibliotecas emoleadas para el análisis de imagen son OpenCV.

Un sistema capaz de reaccionar y resolver problemas de producción en tiempo real debe estar dotado de una capacidad de inteligencia computacional efectiva. Por ejemplo, para toma de decisiones complejas como es la determinación de la trayectoria más corta del robot a la hora de vaciar el almacén, es necesario trabajar con matrices y con algoritmos de optimización. En nuestro caso, y por el valor didáctico al conectar con las competencias adquiridas en otras materias del grado de ingeniería, emplearemos MATLAB.En este entorno, será necesario desarrollar un algoritmo de toma de decisiones para el vaciado secuencial del almacén, de manera que se invierta el mínimo tiempo posible. Esto implica el uso de funciones de optimización entera parea el cálculo dela ruta mínima.

Finalmente, los componentes descritos deben operar con un conjunto de piezas construidas mediante impresión 3D por los estudiantes, y deben coordinarse mediante rutinas de comunicaciones basadas en sockets TCP/IP. La plataforma de aprendizaje consiste, por tanto, en un sistema de control y decisión distribuido con tres aplicaciones ejecutándose de manera coordinada: Robot IRB120 y su sistema operativo Robotware, Raspberry Pi y MATLAB.

II. Objetivos

El objetivo de este artículo es introducir una plataforma de desarrollo para prácticas multidisciplinares empleando un robot de ABB modelo IRB120, Raspberry Pi y MATLAB. En dicha plataforma desarrollaremos múltiples competencias profesionales. En concreto, las competencias a desarrollar son las siguientes:

- Programación de robots.
- Prototipado y fabricación aditiva.
- Introducción a algoritmos e inteligencia artificial.
- Programación en MATLAB.
- Automatización industrial.

Dichas competencias se desarrollan mediante un conjunto de actividades, algunas de las cuales tienen carácter competitivo.

El principal propósito es construir un almacén automático a pequeña escala. Por ejemplo, es habitual encontrar este tipo de sistemas en la industria farmacéutica y biotecnología. La plataforma construida se empleará en las asignaturas de Informática, Mecatrónica, Automatización y Robótica Industrial y Automatización de Sistemas de Producción para desarrollar diferentes aptitudes de los alumnos. Los medios empleados para el desarrollo de la actividad son lo siguientes:

- Impresora 3D de bajo coste con filamento ABS.
- Robot industrial ABB modelo IRB120 y su controlador correspondiente.
- Entorno de programación y simulación RobotStudio v6.01 o superior.
- MatLab R2017b (aunque bastaría con casi cualquier otra versión).
- Estantería de miniaturas.

Es importante resaltar el bajo coste de todos los recursos empleados a excepción del brazo robótico con el cual ya se contaba de partida.

III. Desarrollo de la plataforma

El prototipo de almacén automático a desarrollar se trata de una estantería de miniaturas con varias baldas. El robot, gracias a una garra diseñada y acoplada en su extremo, accederá a una serie de discos y los situará en las diferentes posiciones accesibles en cada balda. Es importante resaltar la posibilidad de apilar varios discos en una misma posición de la estantería. Tanto los discos como la garra han sido diseñados mediante fabricación aditiva. Un esquema de la estructura del prototipo puede verse en la Figura 3. En las siguiente secciones se detallan uno a uno los componentes constituyentes del almacén automático.

A. Estantería

Para la estructura del almacén se ha optado por adquirir una estantería de miniaturas de metacrilato de poco coste. Las estantería cuenta con cuatro baldas. Hemos decidido dividir cada balda en cuatro posiciones de tal manera que existen 16 posiciones posibles para situar los diferentes discos a almacenar en la estantería. Además, el diseño de los discos permite su apilación uno sobre otro. Hemos decidido fijar 4 como el número máximo de discos apilados en una posición de la estantería. Tenga en cuenta que todas estas posiciones pueden variar según la estantería utilizada y la alcanzabilidad del brazo robótico. Una foto de la estantería puede observarse en la Figura 1.

B. Herramienta

Con el fin de acoplar diferentes manipuladores al extremo del brazo robótico se ha diseñado mediante fabricación aditiva un cabezal adaptable (Figura 2). Este cabezal permite acoplar de forma robusta la garra diseñada gracias a un sistema de muelles que impiden el movimiento de la herramienta una vez introducida. Para poder facilitar el sistema lo máximo posible, se ha optado por diseñar una herramienta con forma de transpaleta de tal modo que pueda tomar los discos a través de hendiduras en la parte inferior de éstos y elevarlos como si de un pallet se tratara. Como se puede observar en la figura,

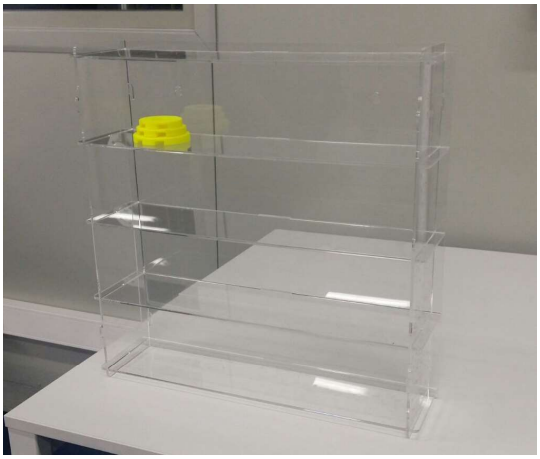

Fig. 1. Estantería utilizada para el montaje de almacén automático.

los brazos de la transpaleta se han orientado 45^o respecto a la normal de la base, que coincide a su vez con el extremo del brazo articulado. Así el acceso por parte del robot se facilita pudiendo cubrir un amplio rango de funcionamiento.

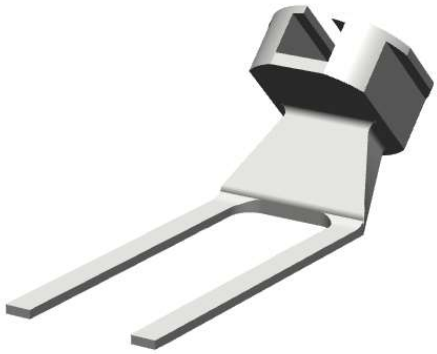

Fig. 2. Herramienta diseñada para el desarrollo de la práctica.

C. Discos

Con el fin de generar una plataforma versátil y escalable, se ha decidido optar por discos para su almacenamiento. Los discos ofrecen la posibilidad de apilarlos uno encima de otro permitiéndonos tener un sistema de almacenamiento con multitud de posibilidades. Con el fin de distinguir unos discos de otros, se ha optado por diseñarlos con diferentes colores. Todos los discos han sido diseñados mediante un programa CAD y han sido imprimidos mediante fabricación aditiva con filamento de diferentes colores. Además, para facilitar el apilamiento inicial de los discos por parte del usuario, se han introducido unas hendiduras en la parte inferior de los discos y unos salientes en la parte superior de los mismos de tal manera que una vez apilados encajan perfectamente uno encima del otro. En la Figura 3 se puede ver un conjunto en el entrono Robotstudio de todos los componentes que constituyen el sistema.

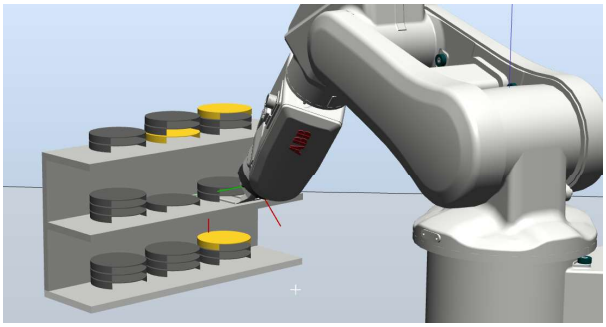

Fig. 3. Almacén automático en entrono de desarrollo y simulación RobotStudio.

IV. Entorno de programación

Uno de los conocimientos principales que el alumno debe adquirir durante la realización de la práctica es la familiarización con el entorno de programación y visualización RobotStudio desarrollado por la empresa multinacional de electricidad y robótica ABB; así como el lenguaje de programación específico de robots RAPID. Además, dada la amplia experiencia de los alumnos de la Universidad en el lenguaje de programación MatLab, se propone comunicar el simulador de RobotStudio con MatLab para poder generar programas y explotar las características conjuntas de ambos softwares.

A. RobotStudio

RobotStudio es un software de simulación y programación fuera de línea diseñado por ABB. Con este software podemos crear, programar y simular células y estaciones de robots industriales ABB. Es un simulador comercial potente que nos permite crear multitud de estaciones, importar geometrías 3D, programar y simular tanto la cinemática como la dinámica de las estaciones (gracias a su módulo de simulación física).

Los pasos generales para llevar a cabo la programación del robot son:

- Insertar el robot en el espacio de trabajo y definir su correspondiente sistema de control.
- Definir las posiciones del robot o targets. Dichas posiciones vienen definidas por la posición del extremo del robot en el espacio cartesiano tridimensional así como la orientación de la misma y la configuración del robot para alcanzarla.
- Definición de trayectorias entre las diferentes posiciones. En esta definición se pueden configurar varios parámetros según la naturaleza del movimiento que se quiere llevar a cabo.
- Generación del programa RAPID a partir de las simulaciones de trayectorias llevadas a cabo.
- Familiarización con las instrucciones generadas.
- Simulación y depuración.
- Carga en el robot real y ejecución manual.

En la Figura 4 se puede observar un ejemplo de programa y su correspondiente código RAPID.

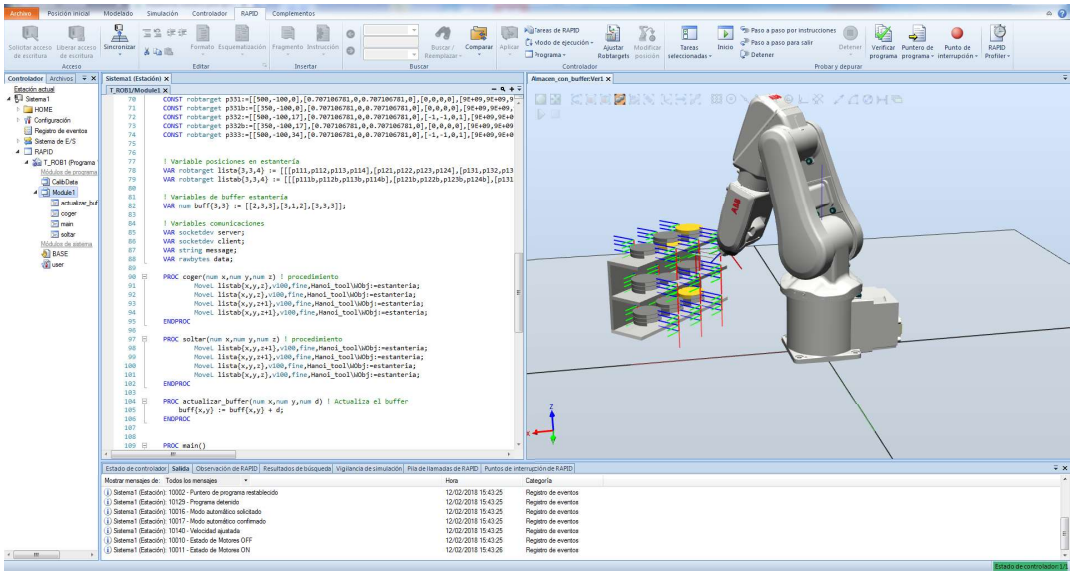

Fig. 4. Entorno de programación y simulación RobotStudio. A la izquierda se puede observar el código RAPID generado mientras que a la derecha el entrono de simulación con geometrías importadas.

B. MatLab

Dado que los alumnos de la Universidad tienen amplios conocimientos del lenguaje de programación MatLab, se decide introducir una interfaz de comunicación entre dicho programa y RobotStudio. De esta manera, se pueden generar problemas prácticos que no precisan que el alumno tenga unos profundos conocimientos de programación en RAPID ni del entorno RobotStudio según la asignatura en la que desee emplearse.

MatLab nos permite generar e implementar algoritmos de control que, gracias a la comunicación con el robot, defina los movimientos que debe llevar a cabo para un correcta gestión del almacén automático.

C. Raspberry Pi

Raspberry Pi es una computadora de bajo coste desarrollada en Reino Unido por la Fundación Raspberry Pi. El objetivo principal de la plataforma es estimular la enseñanza de ciencias de la computación en los centros de enseñanza. Nuestra plataforma contará con una Rapberry Pi conectada a una cámara de tal manera que podrá tomar imágenes de las diferentes posiciones en las que se sitúan los discos identificando de tal forma si existe un disco en esa posición y en caso afirmativo el color del mismo. Para ello nos apoyaremos en la biblioteca OpenCV. OpenCV es una biblioteca libre de visión artifical originalmente desarrollada por Intel. La computadora tomará imágenes del almacén y transferirá dicha información a MATLAB, que será el software encargado de procesarla y enviar las órdenes de movimiento al robot. La comunicación entre ambos dispositivos se realizará a través de TCP/IP.

V. Torres de Hanoi

Con el fin de realizar una sesión introductoria a todos los conceptos detallados en las secciones previas, se plantea el problema de las torres de Hanoi. El objetivo de esta práctica es familiarizarse con el robot articulado ABB IRB 120 con el cual se cuenta en el laboratorio de la Universidad Loyola Andalucía. Para ello, se proponen una serie de ejercicios a realizar mediante el software RobotStudio y su posterior implementación en el robot real. La práctica esta diseñada para realizarse en grupos de tres personas en dos horas y es necesario entregar el código RAPID generado por RobotStudio al finalizar la misma. Al tratarse de una práctica introductoria de familiarización con el entrono de desarrollo su evaluación se basará únicamente en la asistencia.

El trabajo a realizar consiste en el desarrollo de un programa para el robot apoyándose en el software RobotStudio que permita hacer movimientos de varios discos en una base resolviendo el problema de las torres de Hanoi. Para ello haremos uso de los discos y la herramienta definida en la Sección III.

Las torres de Hanoi es un rompecabezas matemático introducido en 1883 por el matemático francés Édouard Lucas. Consiste en un número de discos perforados de radio creciente que se apilan insertándose en uno de los tres postes fijados a un tablero. El objetivo es trasladar las torres desde un poste extremo al poste contrario teniendo en cuenta que no es posible apilar un disco de mayor radio sobre uno de menor. Es un juego de fácil comprensión y muy utilizado en la introducción a la teoría de algoritmos.

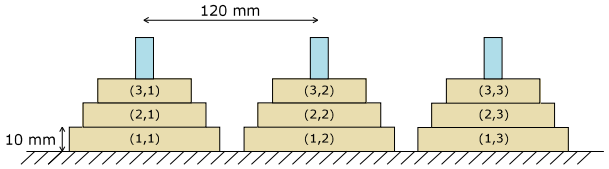

Fig. 5. Esquema aportado junto al guión de la práctica. Las coordenadas de la posición $(1, 1)$ son también proporcionadas.

El alumno contará con la posición inferior de uno de los postes y, a partir de él, debe ser capaz de obtener el resto de posiciones mediante desplazamientos relativos (observe la Figura 5). De igual forma el alumno debe ser capaz de:

- Insertar la herramienta de trabajo en el entorno de RobotStudio y acoplarla al extremo del robot.
- Definir las posiciones para que el robot pueda realizar, a partir de ellas, los movimientos necesarios para resolver el juego.
- Definir la trayectoria y asegurarse de introducir unas configuraciones correctas del brazo robótico.
- Simular el movimiento apoyándose en el simulador de fenómenos físicos que se dispone en RobotStudio a partir de la versión 6.01.
- Generar el código en lenguaje robot (RAPID) e implementarlo en el robot real.

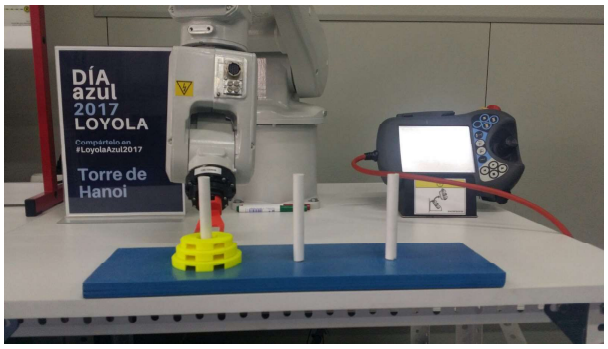

Fig. 6. Montaje de las torres de Hanoi.

Como se puede observar, esta práctica permite al alumno introducirse al entorno de programación de RobotStudio, así como implementar un algoritmo para la resolución de un rompecabezas matemático como es las torres de Hanoi. Ya que se trata de una sesión introductoria, se decide realizar el juego utilizando únicamente tres discos aunque se considera como trabajo adicional el desarrollo de un algoritmo genérico para la resolución del problema para un número indeterminado de discos.

VI. Almacén automático

Una vez introducido el entorno de desarrollo RobotStudio y que el alumno se haya familiarizado con el robot, en esta práctica se plantea el siguiente problema de optimización:

Problema: Desplazar un número determinado de discos de unas posiciones a otras en el mínimo tiempo posible. Tenga en cuenta que al estar los discos apilados uno sobre otro es posible querer mover un disco cuyo movimiento no puede realizarse hasta apartar los discos que descansan sobre él.

Esta práctica se introduce con un alto factor competitivo ya que cada grupo de alumnos debe desarrollar un algoritmo que permita resolver el problema en un tiempo inferior al del resto de grupos. La práctica se divide en varias etapas:

1) El alumno deberá medir los tiempos que el robot utiliza para moverse de una posición a otra e incluirlos como datos en la resolución de su problema.
2) El alumno deberá generar una función en MatLab que a partir de el número inicial de discos en cada posición de la estantería, las posiciones de los discos de partida y las posición de destino de los mismos, devuelva los movimientos que debe realizar el robot. Para el desarrollo de esta función se propone un algoritmo iterativo que, a partir de una solución factible, vaya mejorando hasta alcanzar una solución cercana o igual a la óptima. Tenga en cuenta que esta función es versátil y debe adaptarse a cualquier parámetro de entrada.
3) Finalmente, se facilita al alumno un programa en Robot Studio y una función en MatLab que permite la sincronización entre ambos elementos de software. Inicialmente la estrategia será probada en el entorno de simulación de RobotStudio y durante una última clase de esta práctica cada grupo de alumnos presentará su algoritmo de optimización teniendo lugar las pruebas finales y elección del mejor grupo.

Ahondemos algo más en la segunda etapa de la práctica concerniente al desarrollo del algoritmo óptimo. Para ello, introducimos las siguientes variables: Sea $n_{i,j}$ el número de discos en cada posición de la estantería dada por la fila $i \leq i_{max}$ y la columna $j \leq j_{max}$ donde $i_{max} \times j_{max}$ son las dimensiones de la estantería que en nuestro caso se limita a 4×4. Sea N el número máximo de discos que se pueden apilar en cada uno de los estantes. El tiempo que tarda el robot de transportar un elemento desde una posición de la estantería a otra viene dado por la siguiente expresión:

$$t = t_c + t_d + t_m d,$$

dónde t_c y t_d son el tiempo de carga y descarga de un disco por el robot, respectivamente; t_m es el tiempo que tarda el robot en desplazar una pieza un metro y d es la distancia entre los estantes. Con todo esto se pueden plantear distintas estrategias para la resolución del problema planteado:

- Una de las soluciones más evidentes es la de plantear por separado cada movimiento de piezas. Es decir, el movimiento de cada disco de una posición de origen a una posición de destino independientemente. Entonces, para cada movimiento, será necesario primero ver si la posición de destino del disco está en las condiciones adecuadas, es decir, si existe el número necesario de

discos apilados bajo su posición y no existen discos apilados en la posición o encima de ella. El siguiente paso sería retirar todos los discos que estén encima de la pieza de origen para poder tomarla. Una vez cogida la pieza, faltaría el último movimiento para situarla en la posición de destino. El movimiento de todas las piezas hasta llegar al último movimiento pueden realizarse de muchas maneras siendo la más inmediata buscar una posición libre diferente de la de origen y destino y depositar ahí el disco.

- La estrategia anterior podría mejorarse si los movimientos intermedios se realizan con un criterio de minimizar el tiempo en el que el robot se encuentra en movimiento, es decir, intentando depositar los discos en posiciones cercanas a la posición de dónde se retiran.
- Del mismo modo si los movimientos de los discos intermedios se considera como un problema único y no plantearlo cada vez que se retira uno, el tiempo total de movimiento puede disminuir aún más.
- Otra posibilidad es considerar más de un movimiento de discos dado por los argumentos de la función al mismo tiempo. Así, cuando realicemos un movimiento, los discos que no se utilicen no ocuparán posiciones que dificulten los posteriores movimientos.

La función a desarrollar por los alumnos tendrá la siguiente forma:

$$[mov] = algorithm(n_{i,j}, t_c, t_d, t_m, Origen, Destino),$$

dónde $Origen$ y $Destino$ son matrices de tres columnas donde se almacena la celda de la estantería a la que acceder (valor en eje x y en eje y) y altura del disco dentro del apilamiento en una misma posición. Tendrán tantas filas como movimientos se deseen realizar. Por otro lado, mov es una matriz que define los movimientos a realizar por el robot que tiene tantas filas como movimientos haya que realizar y en cuyas columnas se almacena un uno o un cero según sea un movimiento de coger o soltar un disco y la posición de la estantería en la cual hay que realizar la operación. Tenga en cuenta que en este caso no es necesario indicar la posición del disco dentro de la pila ya que el programa solo permite coger un disco de la última posición posible. Un ejemplo de lo expuesto anterior sería:

- $Origen = [1, 1, 2]$,
- $Destino = [1, 3, 1]$,
- Resultado: $mov = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 3 & 1 \end{bmatrix}$.

En este simple ejemplo se especifica que se ha de tomar el disco apilado en el segundo nivel de la primera balda y primera columna de la estantería y depositarlo en el primer nivel de la tercera balda, primera columna de la estantería. Como resultado el programa devuelve dos filas que indican que hay que coger un disco (0) de la posición $(1, 1)$ y soltarlo (1) en la posición $(3, 1)$. El programa de RobotStudio tienen un buffer en el que almacena el número de discos apilados en cada posición de la estantería y gestiona de forma autónoma la altura que debe tener en cuenta el robot para acceder a coger o soltar un disco de la estantería.

Se proporciona al alumno funciones de MatLab que a partir del vector mov traduce las instrucciones a movimientos del robot, comunicándose con RobotStudio y llevando a cabo los movimientos en el entorno simulado.

A. *Análisis del código desarrollado*

En esta sección se pretende describir el código desarrollado para la aplicación propuesta del almacén automático. Los programas de RobotStudio y MATLAB están estructurados de la siguiente manera:

1) Se definen en RobotStudio las variables *RobotTarget* que nos indicarán las posiciones alcanzables para el robot así como las configuraciones necesarias para alcanzarlas. Almacenamos las posiciones de la estantería en dos matrices de tres dimensiones (posición x, posición y y nivel del disco en la posición (x, y)). También se define el buffer que almacenará el número de discos apilados en cada posición de la estantería.

```
! Variable posiciones en estanteria
  VAR robtarget lista{3,3,4} := ... ;
  VAR robtarget listab{3,3,4} := ... ;

! Variables de buffer estanteria
  VAR num buff{3,3} :=
      [[2,3,3],[3,1,2],[3,3,3]];
```

2) A continuación inicializamos las comunicaciones entre RobotStudio y MATLAB. Para ello es necesario definir la dirección IP del cliente y del servidor. En el siguiente ejemplo se puede observar una configuración para trabajar con el simulador de RobotStudio y MATLAB en el mismo ordenador.

```
! Variables comunicaciones
  VAR socketdev server;
  VAR socketdev client;
  VAR string message;
  VAR rawbytes data;

! Crear comm
  SocketCreate server;
  SocketBind server,"127.0.0.1",55000;
  SocketListen server;
  SocketAccept server,client;
```

en el caso de MATLAB el código es aún más sencillo

```
% Se establece la conexion
  tc=tcpip('127.0.0.1',55000);
  tc.ByteOrder = 'littleEndian';
  fopen(tc);
```

3) Una vez abierta la comunicación entre ambos programas simplemente bastará con definir la longitud de datos a enviar y el contenido. Por ejemplo, para enviar desde MATLAB tres datos enteros utilizaremos la siguiente orden:

```
  fwrite(tc,[1 2 3],'int32')
  fread(tc,2);
```

mientras que desde RobotStudio tomaremos cada una de las componentes del vector especificando en qué variable se almacenará y que tipo de dato se espera. De esta manera almacenamos en las variables x, y y z los valores enteros recibidos. Tenga en cuenta que las variables han sido definidas previamente.

```
! Recibir mensaje del cliente
  SocketReceive client,\RawData:=data;
  UnpackRawBytes data,1,x,\IntX:=DINT;
  UnpackRawBytes data,5,y,\IntX:=DINT;
  UnpackRawBytes data,9,z,\IntX:=DINT;
```

4) Una vez finalizadas las comunicaciones entre ambos programas, será necesario indicarlo utilizando los siguientes comandos. En MATLAB:

```
fclose(tc);
```

mientras que en RobotStudio:

```
SocketClose client;
```

VII. Conclusiones y trabajo futuro

Este artículo ha introducido una plataforma robótica para aprendizaje activo multidisciplinar de bajo coste. Se han definido los diferentes componentes de la plataforma así como dos ejercicios prácticos introducidos en varias asignaturas del grado de ingeniería electromecánica. Las competencias adquiridas durante el desarrollo de la práctica son las siguientes:

- CAD y fabricación aditiva de piezas.
- Lenguaje RAPID.
- Entorno de desarrollo MatLab y RobotStudio.
- Simulación, depuración y ajuste de parámetros.
- Algoritmos de optimización.

Como trabajo futuro se considerará su implementación en el segundo cuatrimestre de un proyecto multidisciplinar en la asignatura de Automatización de Sistemas de Producción. La finalidad de este proyecto será abordar una serie de conceptos que componen a la Industria 4.0. Así mismo, se cuenta con el dispositivo Raspberry Pi y se ha comenzado con el desarrollo del reconocimiento visual de las piezas, sin embargo aún se encuentra en una etapa de desarrollo.

Agradecimientos

Proyecto parcialmente financiado por beca TEC2016-80242-P y DPI2016-75924-C2-2-R de AEI/FEDER.

References

[1] J. Ramón, A. Figueres, A. Oller y J. De la Rosa, Plataforma Docente de robots móviles, cooperantes y autónomos, Addison Wesley, Massachusetts, Proc. TAEE 98, Publicaciones UPM, 1998.

[2] J. Ramón, A. Figueres, A. Oller y J. De la Rosa, Laboratorio docente de robots móviles, cooperantes y autónomos, Addison Wesley, Massachusetts, Actas TAEE 2000, UAB, Septiembre 2000.

[3] G. González de Rivera, S. López-Buedo, I. González, C. Venegas, J. Gaarrido y E. Boemo, Plataforma hardware para la enseñanza de robótica en la titulación de ingeniería informática, E.T.S. Ingeniería Informática, Universidad Autónoma de Madrid, Actas TAEE 2002.

[4] Vera, Francisco Javier and Mendoza, Julio Pastor and Domeque, Esther Samper, "Feel like a Cyborg II": Demonstrative system of robot function, Actas TAEE 2012

[5] RobotStudio reference: http://new.abb.com/products/robotics/es/robotstudio

[6] Raspberry Pi Foundation: https://www.raspberrypi.org/

[7] Arduino: https://www.arduino.cc/

[8] Perkins, Charles E and others, Ad hoc networking, Addison-wesley Reading, 2001.

[9] Python Software Foundation: https://www.python.org/