Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Modelado, simulación y desarrollo de una práctica docente con el robot ABB IRB 120

Autor: Rodríguez López, Miguel

Tutor: Fraile Marinero, Juan Carlos

Departamento de Ingeniería de Sistemas y Automática

Valladolid, junio 2021

Resumen

Simulación, diseño y desarrollo de un prototipo para prácticas académicas con el robot IRB 120 de ABB, incorporando elementos mecánico-eléctricos que lo dotarán de un carácter aleatorio. Todos los componentes que forman la aplicación se agrupan en un tablero móvil que colocaremos en el área de trabajo del robot cuando queramos utilizarlo y, una vez haya finalizado el programa, lo podremos retirar.

La práctica desarrollada ha sido simulada en RobotStudio para comprobar y optimizar su funcionamiento, creando una estación virtual utilizando componentes inteligentes y señales digitales de entrada/salida del robot IRB 120. La aplicación está compuesta por un tablero con una serie de orificios en cada cual se encuentra un microinterruptor, dos torres de almacenamiento fabricadas mediante impresión 3D que contendrán bolas, y dos actuadores neumáticos con sus respectivas electroválvulas para controlarlos. Los actuadores empujarán las bolas desde las torres hacia los orificios del tablero, activando su sensor correspondiente. El robot recibirá la información de la posición y recogerá la bola para volver a introducirla en una de las torres.

Palabras clave (Keywords)

Simulación entornos robóticos, RobotStudio, RAPID, práctica docente

Abstract

Simulation, design and development of a prototype for academic practices with the ABB IRB 120 robot, incorporating mechanical-electrical elements that will give it a random character. All the components that form the application are grouped in a mobile board that we will place on the work area of the robot when we want to use it and, once the program is finished, we will be able to remove to use it in other tasks.

The practice developed has been simulated in RobotStudio to test and optimize it's operation, creating a virtual station using intelligent components and the digital signals of the robot IRB 120. The application consists of a board with a series of holes in each one is a micro switch, two storage towers made by 3D printing that will contain balls, and two pneumatic actuators with their respective solenoid valves to control them. The actuators will push the balls from the tower into the holes of the board activating their corresponding sensor. The robot will receive the position information and collect the ball to re-insert it into one of the towers.

Índice

Agradecimientos

Quiero comenzar este trabajo agradeciendo el apoyo incondicional que he recibido por parte de mi familia y amigos, en especial de mi padre y mi madre por todas las facilidades y oportunidades que me han brindado durante mi etapa en la Escuela de Ingenierías Industriales de la Universidad.

También quiero agradecer toda la ayuda y atención prestada por mi tutor Juan Carlos Fraile Marinero y todos los compañeros del ITAP y el laboratorio de robótica de la escuela.

1. Introducción y objetivos

Introducción al estado del arte

La utilización de robots similares al IRB 120 suelen emplearse en ámbitos industriales como, por ejemplo, cadenas de montaje de automóviles, producción en masa de cualquier tipo de producto farmacéutico o de alimentación, etc., ya que mejoran ampliamente el tiempo y la calidad del trabajo en las grandes fábricas, simplificando también tareas peligrosas para el ser humano y logrando una automatización de procesos. No obstante, su uso en el ámbito académico es una buena forma de introducir a los estudiantes en el mundo de la robótica industrial y en el aprendizaje de sus características mecánicas y de programación.

A continuación, se muestran varios ejemplos de proyectos relacionados con este trabajo en cuanto a la utilización del robot IRB 120 con fines académicos que podrían interpolarse al ámbito industrial.

Actividades de manipulación y soldadura con el robot IRB 120

En [1] encontraremos la ejecución de actividades de manipulación y soldadura de objetos con la ayuda del brazo robot IRB 120 en el centro integrado de formación profesional “Ciudad de Béjar” (Salamanca) como ejercicio académico, mostrado en la figura 1.

Figura 1. Actividades de manipulación y soldadura con el robot IRB 120

Ejercicios de manipulación y ensamblado de piezas con el robot IRB 120

Otro ejemplo de la misma institución se muestra en [2], en el que podemos ver un vídeo de la ejecución de un robot IRB 120 en actividades de manipulación y adhesión de piezas (ver figura 2) en un ámbito puramente académico, de nuevo en el centro integrado de formación profesional "Ciudad de Béjar" de Salamanca.

Figura 2. Actividades de manipulación y ensamblado de piezas con el IRB120

Proyecto de automatización y robótica industrial

Este trabajo [3] muestra un proceso automatizado de producción de coches en el que se realiza una simulación en RobotStudio y la construcción de una maqueta física con la que trabajará el robot IRB 120. Consiste en el desarrollo de un sistema que trabaja con el robot y un autómata programable de forma conjunta en el IES Sierra de Carrascoy. En la figura 3 se muestra un modelo desarrollado en RobotStudio para la simulación de la práctica descrita.

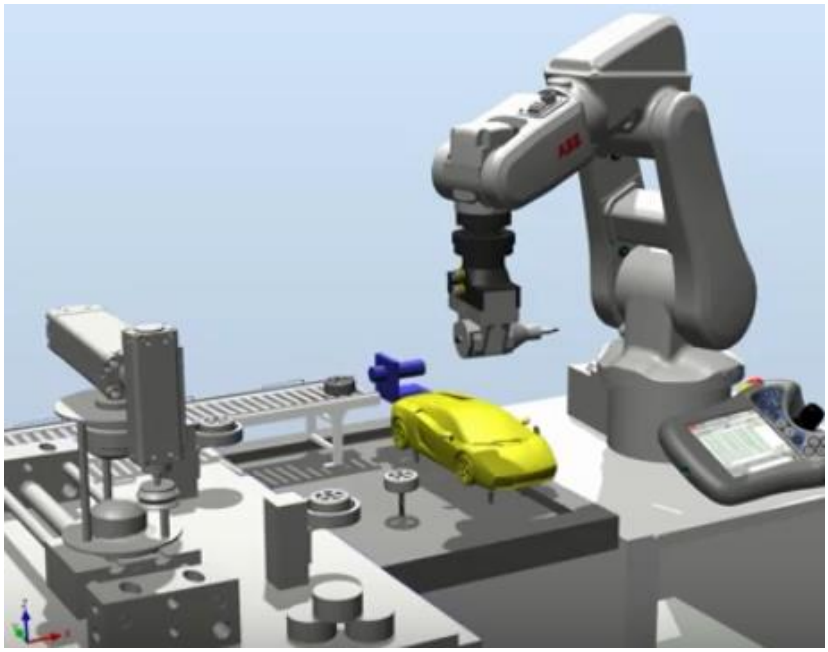


Figura 3. Actividad de simulación robótica y automatización en RobotStudio

Control de cinta transportadora mediante IRC5 y robot IRB 120

El documento [4] desarrollado como TFG en la EPS Alcalá (Madrid), se muestra en la figura 4 el desarrollo de una aplicación que combina visión artificial y programación robótica para trabajar en conjunto con una plataforma en una cinta transportadora.

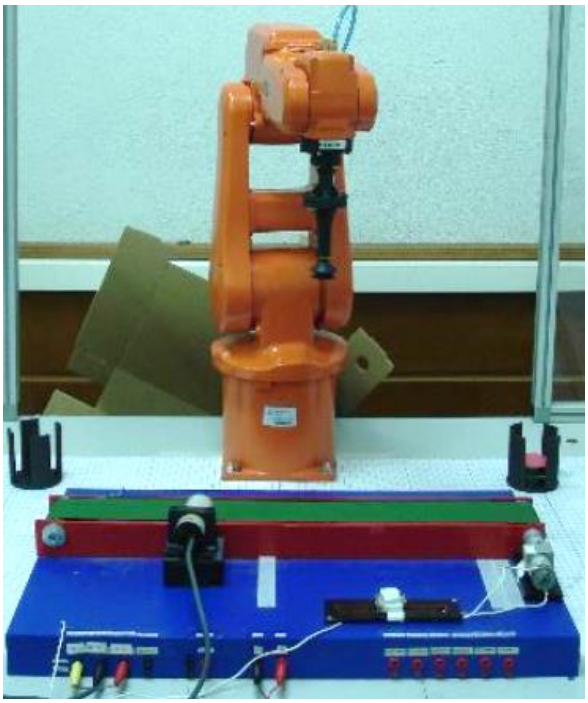


Figura 4. Actividad con IRB 120 y cinta transportadora

Objetivos del proyecto

El objetivo principal del trabajo es el diseño, fabricación y desarrollo de una aplicación para prácticas académicas que sirva como método de aprendizaje para la programación del robot IRB 120, añadiendo distintos componentes electrónicos, mecánicos y eléctricos que hagan de la práctica un trabajo completo, incluyendo varios campos de ingeniería y la doten de un carácter aleatorio. Todos los componentes que formarán la práctica se agruparán en un tablero móvil que colocaremos sobre el área de trabajo del robot cuando queramos utilizarla y, una vez haya finalizado el programa, podremos retirar para usar el robot en otras tareas. En los bordes del tablero dispondremos de las conexiones eléctricas y neumáticas necesarias para unir los mecanismos que formarán parte de la maqueta.

En lados opuestos del tablero móvil, posicionaremos dos torres de almacenamiento de bolas de madera, que contendrán un máximo de tres bolas cada una. Para controlar que no se saturen las capacidades de dichas torres, en la parte superior de cada una dispondremos de un sensor que nos permitirá conocer si la torre tiene completa su capacidad de almacenamiento. En la parte inferior, tendremos otro sensor que nos permitirá determinar si existe al menos una bola almacenada en la torre. Dichos sensores estarán conectados a las entradas digitales del robot IRB 120.

Las bolas almacenadas en cada torre se lanzarán hacia el tablero central mediante unos actuadores neumáticos accionados aleatoria y automáticamente, siempre y cuando el sensor ubicado en el fondo de cada torre esté activo, indicando que al menos una bola se encuentra en posición de salida. Los actuadores neumáticos estarán conectados a las salidas digitales del robot.

Dispondremos de un tablero fijo en la zona central de la maqueta, al que caerán las bolas procedentes de las torres de almacenaje. Este tablero dispone de nueve orificios distribuidos por su centro en forma de matriz 3x3. Cada uno de ellos contendrá un sensor tipo microinterruptor (iguales que los ubicados en las torres), que nos indicarán si una bola ha caído dentro del orificio correspondiente al sensor que esté activo. Estos sensores estarán conectados a las entradas digitales del robot, al cual será transmitida la información procedente de ellos. Entonces, el robot cogerá la bola donde se encuentre activo el sensor y la colocará aleatoriamente en una de las torres, siempre y cuando ésta tenga espacio libre para almacenar.

Inicialmente, el usuario introducirá las seis bolas en las torres, tres en cada una. A partir de ahí comenzará el juego, donde se comprobará mediante los sensores que ambas torres tienen al menos una bola en su interior. Uno de los actuadores neumáticos, elegido de forma aleatoria por el programa del IRB 120, empujará una bola, que caerá sobre el tablero central. Cuando dicha bola caiga en uno de los nueve orificios distribuidos por el tablero fijo, se activará el sensor de contacto correspondiente, que enviará la información al robot sobre la posición en la que se encuentra la bola. Tras ello, el brazo robot se dirigirá a las coordenadas de ese orificio, recogerá la bola y la llevará hacia una de las torres que disponga de espacio libre. Una vez que el robot haya recogido la bola del orificio, se activará nuevamente de forma aleatoria una de las electroválvulas que controla el accionamiento de los actuadores neumáticos, empujando de nuevo otra bola desde una de las torres hacia el centro del tablero, repitiéndose el mismo proceso de forma continua hasta que el usuario decida finalizar el juego.

Para alcanzar los objetivos propuestos en este TFG, se han seguido los siguientes pasos:

- Brainstorming junto al tutor para generar ideas que acompañen al objetivo principal del proyecto de forma real.
- Simulación mediante RobotStudio de la práctica.
- Búsqueda de información y componentes necesarios para desarrollar la aplicación.
- Montaje del tablero que serán la base de nuestro trabajo.
- Diseño y fabricación mediante impresión 3D de las torres de almacenamiento y sujeción de los actuadores neumáticos.
- Posicionamiento y conexionado eléctrico de los sensores en cada uno de los orificios del tablero central y en ambas torres de almacenaje.
- Comprobación de la correcta activación de todos los sensores mediante Arduino.
- Conexión y comprobación del funcionamiento de la parte neumática de la práctica mediante Arduino, controlando las electroválvulas correspondientes a cada actuador junto con un módulo de relés.
- Instalación y conexión del cableado eléctrico de los sensores y las electroválvulas con el panel de E/S del robot.
- Instalación y conexión de la parte neumática de los actuadores y electroválvulas a la toma de aire comprimido del laboratorio de la Escuela de Ingenierías Industriales.
- Puesta en marcha y validación de la práctica completa junto al robot IRB 120.

2. Modelado y simulación de la práctica docente

2.1. Simulación en RobotStudio

Introducción y objetivos

Comenzaremos realizando una simulación del proyecto a desarrollar en el programa RobotStudio [5], donde crearemos de forma virtual un modelo similar al que posteriormente realizaremos de forma física, junto con la programación y descripción de todas las entradas y salidas digitales que necesitaremos, y el código de programación que contendrá los movimientos e instrucciones a realizar por el robot.

Introducción a RobotStudio

La programación offline es la mejor forma de maximizar el retorno de la inversión para los sistemas de robot. El software de simulación y programación offline de ABB, permite que la programación del robot se realice desde un PC en cualquier lugar sin interrumpir el trabajo.

RobotStudio se basa en ABB VirtualController, una copia exacta del software real que ejecuta sus robots en producción. Esto permite realizar simulaciones muy realistas, utilizando programas de robot reales y archivos de configuración idénticos a los utilizados en el taller.

Objetivos de la simulación

El objetivo de esta simulación, aparte de crear una primera aproximación al proyecto real, consiste en aprender a utilizar el software RobotStudio de forma amplia y entretenida, teniendo previamente unos primeros conceptos básicos de programación en esta herramienta.

En este programa crearemos nuestra estación de prácticas académicas de forma virtual, aprenderemos a programar las E/S digitales que formarán parte del trabajo, realizaremos una introducción al uso de SmartComponents (componentes inteligentes) de la aplicación, mostrando y explicando la programación propia que hemos utilizado en este proyecto, y comentaremos el código realizado para el funcionamiento del programa.

Creación de la estación virtual

Creación de los componentes que forman la estación virtual

En primer lugar, en la pestaña *"Modelado"* de RobotStudio, creamos los componentes que forman nuestra estación y fijaremos sus posiciones donde posteriormente colocaremos el tablero de forma real a la hora de trabajar con el robot físico.

En la pestaña *"Posición inicial"*, importamos desde la biblioteca ABB que contiene todo el catálogo de robots propios de la empresa, el IRB 120 que utilizaremos en nuestra estación como se muestra en la figura 5 siguiente.

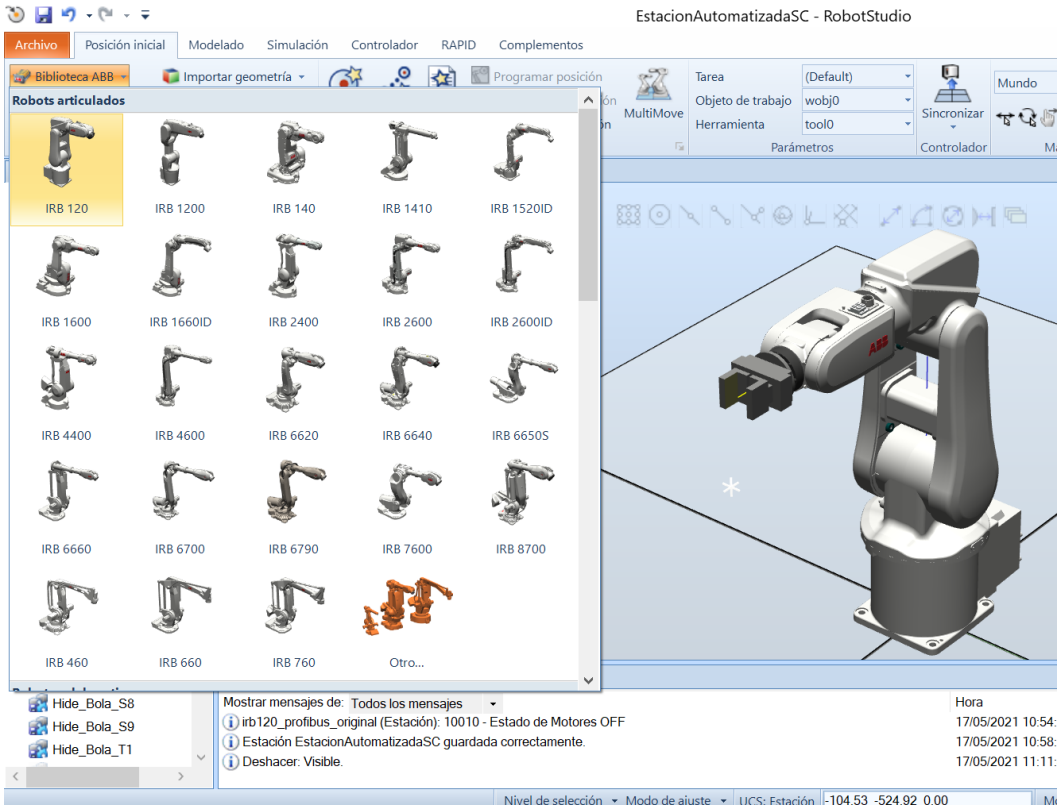


Figura 5. Importación del IRB 120 desde la biblioteca ABB

Tras ello, crearemos el tablero móvil como se muestra en la figura 6, una estructura rectangular de 800x600x10 mm, que colocaremos delante del robot en una posición centrada respecto a la base de coordenadas del robot *"wobj0"* que tendremos por defecto tras importar éste desde el programa. Podremos configurar sus características físicas y de diseño para dar una imagen más aproximada a la realidad.

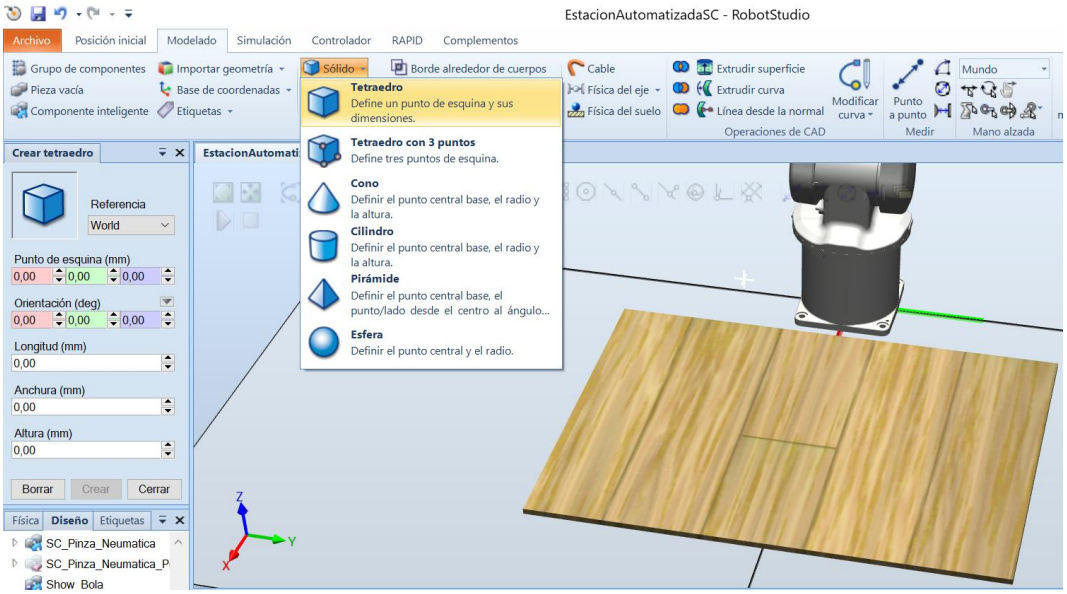


Figura 6. Creación del tablero móvil virtual

Para formar el tablero fijo, tendremos que crear un sólido rectangular con las medidas específicas del proyecto y nueve cilindros ubicados en las posiciones donde se encontrarán los orificios del tablero. El programa RobotStudio nos permite "restar" los elementos creados desde la pestaña *"modelado"* para formar un único componente. En nuestro caso, restaremos cada uno de los cilindros de 30 mm de diámetro al rectángulo que formará el tablero fijo como vemos en la figura 7.

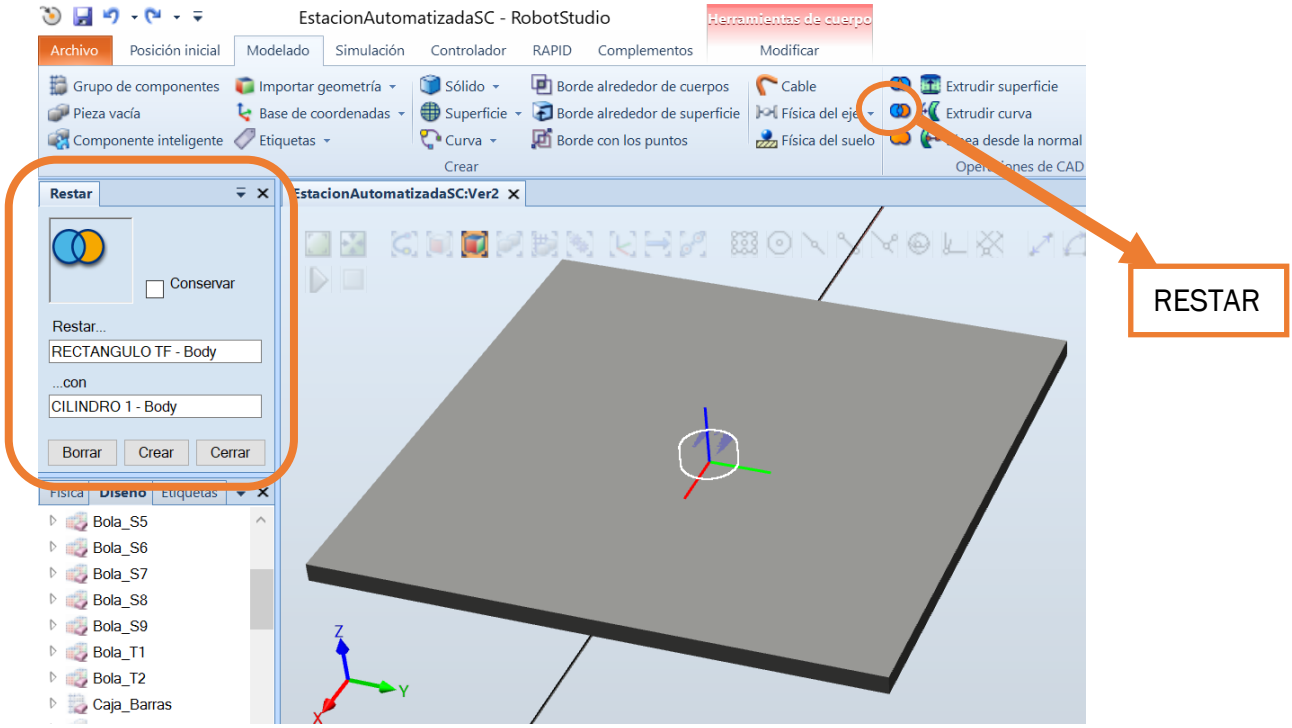


Figura 7. Herramienta "restar" elementos

El resultado final tras aplicar la resta a los nueve cilindros del tablero y después de haber modificado la posición y configuración de diseño es el mostrado en la figura 8 a continuación.

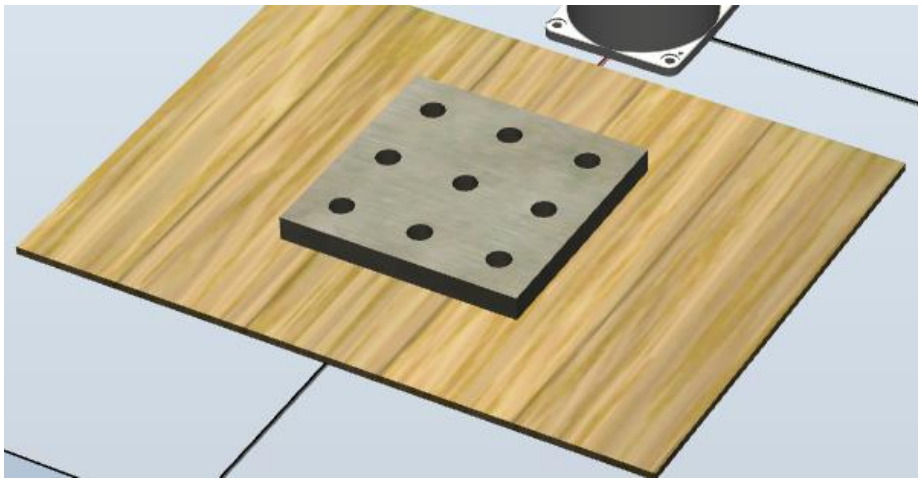


Figura 8. Creación del tablero fijo virtual

De la misma forma, crearemos las dos torres de almacenamiento de bolas mediante el posicionamiento de un cilindro externo y otro interno de menor diámetro, y utilizando la herramienta “restar” para crear el elemento final como apreciamos en la figura 9.

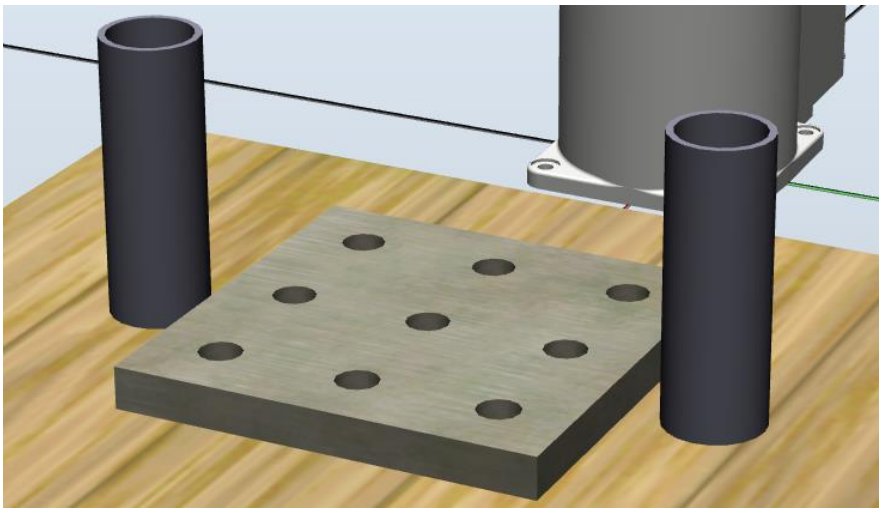


Figura 9. Creación de las torres de almacenamiento virtuales

La estación virtual final se muestra a continuación en la figura 10, a partir de la cual comenzaremos a trabajar en el software de simulación.

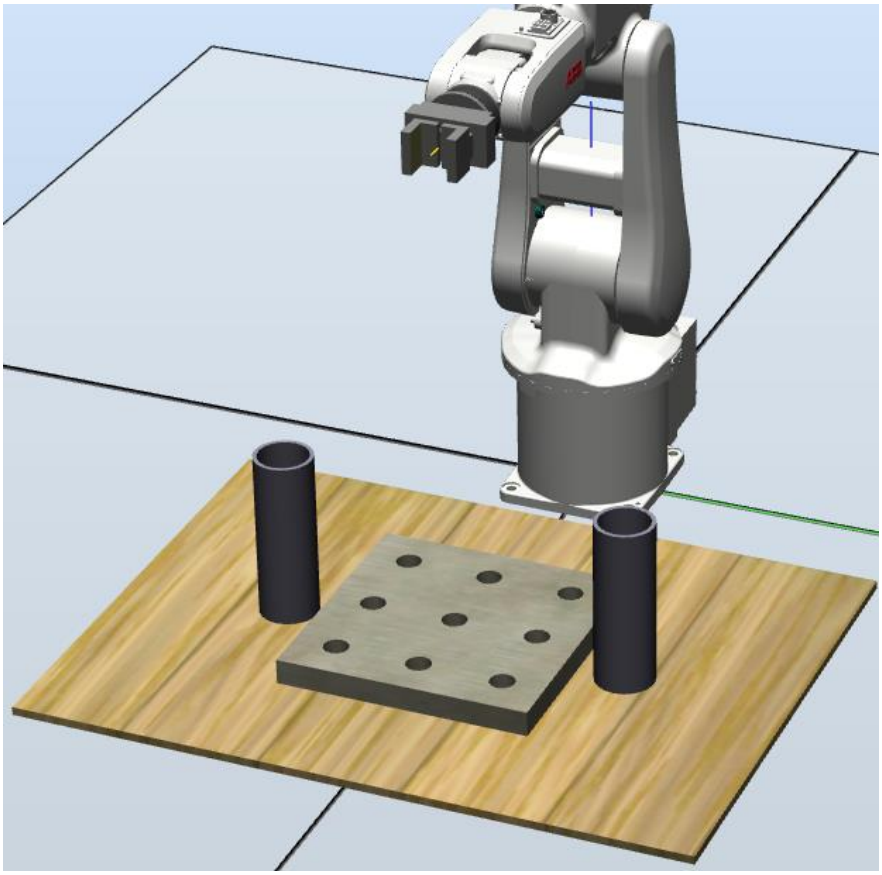


Figura 10. Estación virtual

Creación de los WorkObjects de la estación virtual

Debemos crear un sistema de referencia para cada componente de nuestra estación con el objetivo de facilitar el diseño de las posiciones y trayectorias que seguirá el robot en el apartado de programación. Los ejes de todos los sistemas de referencia que se utilizan en nuestro programa vendrán dados por el siguiente SR base, como indica la figura 11.

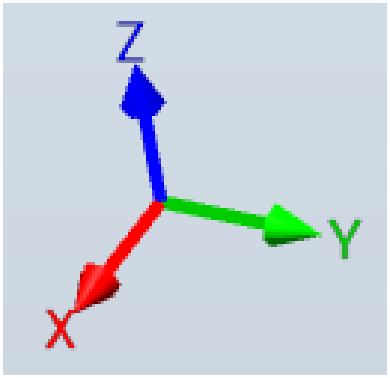


Figura 11. Sistema de referencia base

Un WorkObject (Objeto de Trabajo) es un sistema de referencia que asignaremos a un componente de la estación, al cual irán ligados todos los puntos que creemos en ese componente para facilitar su modificación en el caso de que fuera necesario.

En nuestro programa crearemos dos objetos de trabajo, uno correspondiente al tablero fijo, y otro que servirá de referencia para ambas torres, a parte del wobj0 que aparece por defecto en la base del robot IRB 120 como sistema de referencia general. Para ello, nos dirigimos a la pestaña "Posición inicial" del programa RobotStudio y en la ventana "Trayectorias y puntos" creamos los objetos de trabajo WO_TableroFijo y WO_Torres.

La siguiente tabla recoge las posiciones exactas (en mm) en las que irán ubicadas los objetos de trabajo que vamos a crear, referidas al wobj0 por defecto y con los ejes del anterior sistema de referencia base X, Y, Z.

Tabla de coordenadas de los objetos de trabajo del tablero y las torres respecto al sistema de referencia base 'wobj0'

Objeto de trabajo	X (mm)	Y (mm)	Z (mm)
WO_TableroFijo	320	-100	30
WO_Torres	420	-205	210

El objeto de trabajo del tablero fijo lo situaremos en la primera posición en la que irá ubicado el sensor como muestra la figura 12.

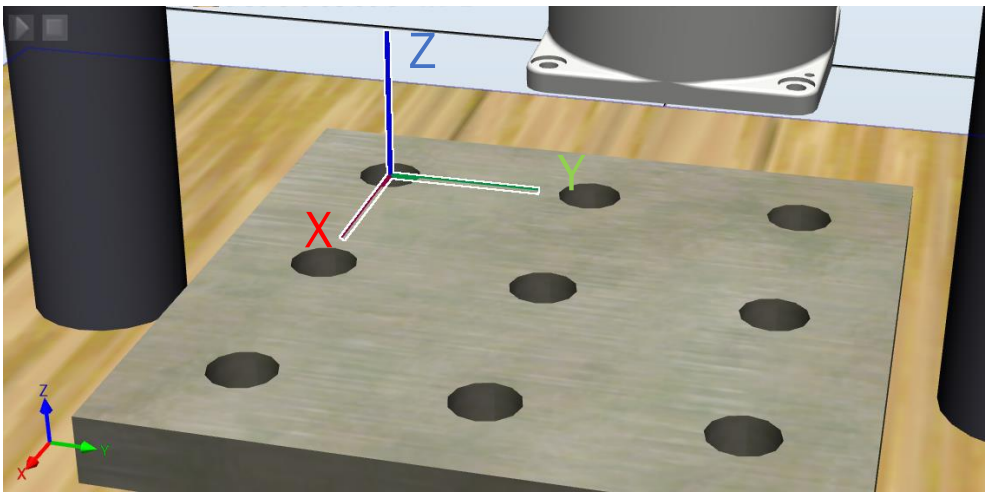


Figura 12. Ubicación del objeto de trabajo "WO_TableroFijo"

El sistema de referencia asignado a las torres de almacenamiento está ubicado en la parte superior de la torre mostrada en la zona izquierda de la figura 13 siguiente, a la cual denominaremos en adelante torre 1.

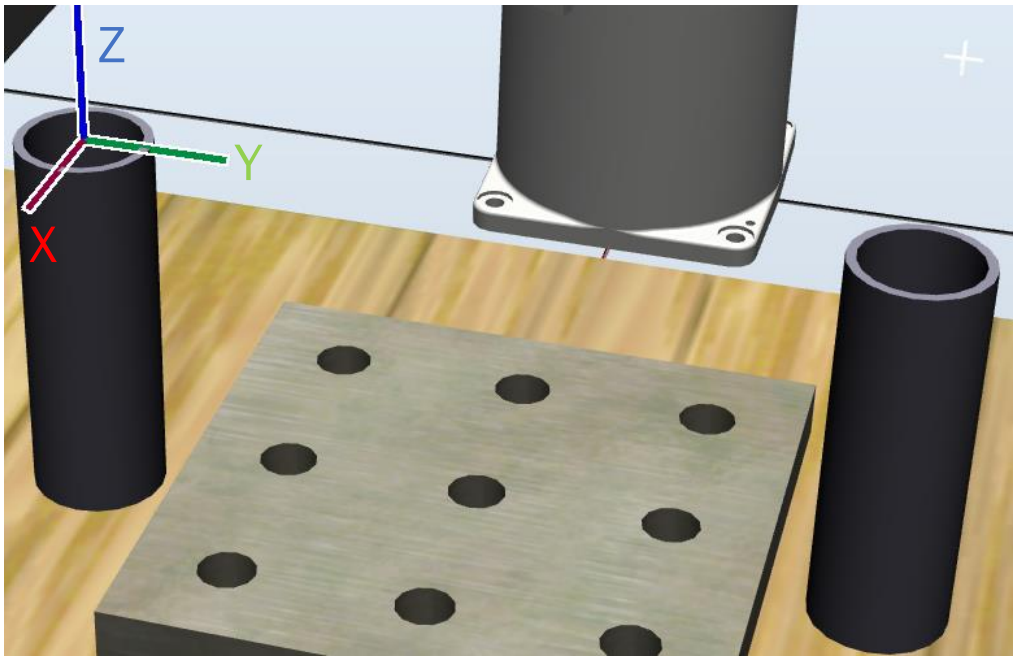


Figura 13. Ubicación del objeto de trabajo "WO_Torres"

Los objetos de trabajo creados hasta ahora podremos verlos en la pestaña "Trayectorias y puntos" como se muestra en la figura 14

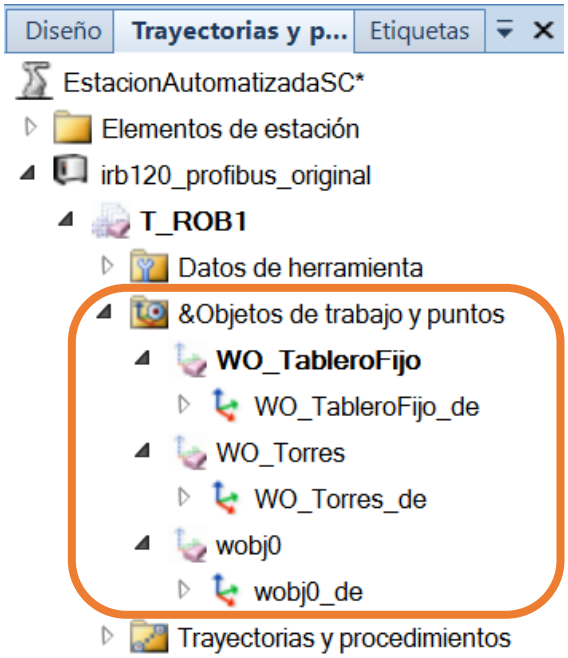


Figura 14. Creación de los objetos de trabajo

A continuación, pasaremos a crear los puntos ubicados dentro de cada sistema de referencia, los cuales aparecerán girados 180º respecto al eje Y (verde) debido a que la herramienta del robot tiene su propio sistema de referencia (TCP), y así conseguiremos que el robot se dirija a cada punto con una orientación apropiada.

En la figura 15, se muestra el sistema de referencia de la pinza neumática "Pinza20170721" que utilizaremos como herramienta en la simulación del programa.

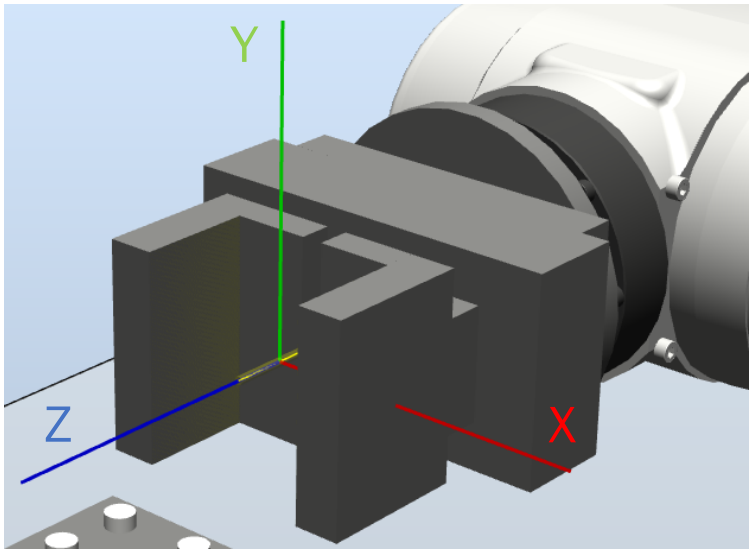


Figura 15. SR de la pinza neumática (TCP)

Las posiciones de los sensores de la estación donde deberá dirigirse el IRB 120 en sus trayectorias se muestran en la figura 16. Crearemos otros puntos a una altura superior en cada sistema de referencia para que el robot pueda aproximarse de forma rápida y ágil, y otros donde pueda ir más lenta y precisamente a las posiciones exactas de los sensores ubicados en los orificios del tablero y de las torres.

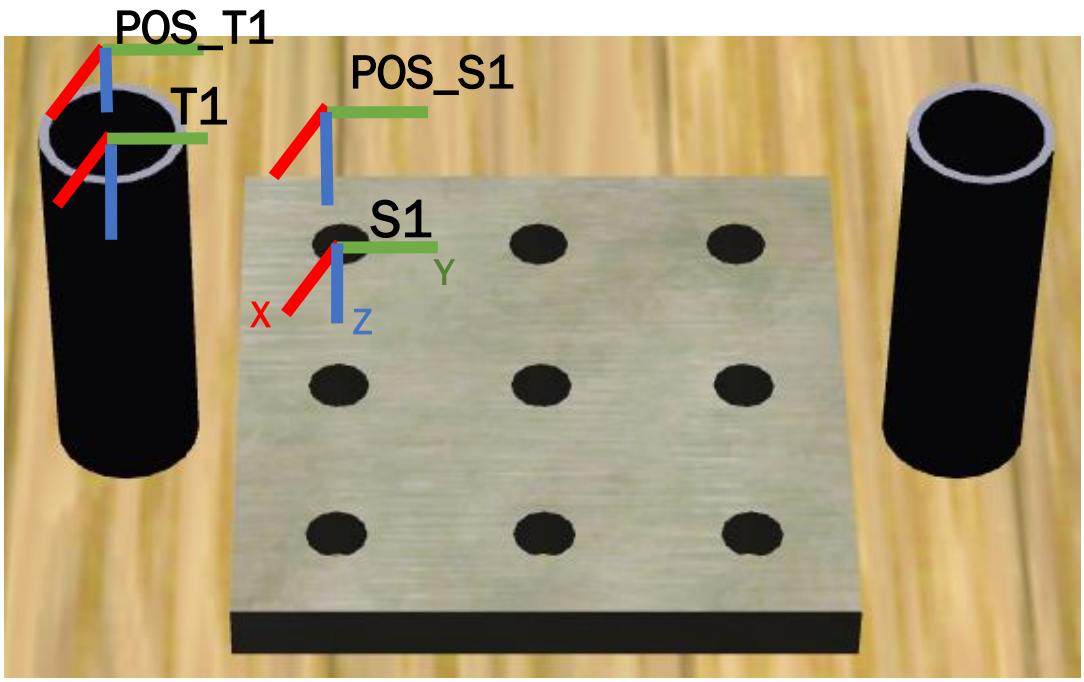


Figura 16. Posiciones de los sensores en la estación

A partir del objeto de trabajo del tablero fijo (WO_TableroFijo), crearemos los puntos necesarios en las posiciones donde se alojan los sensores para posteriormente programar las trayectorias del robot hacia esos puntos. En la figura 17, se muestran los puntos pertenecientes al sistema de referencia del tablero fijo en los nueve orificios de éste.

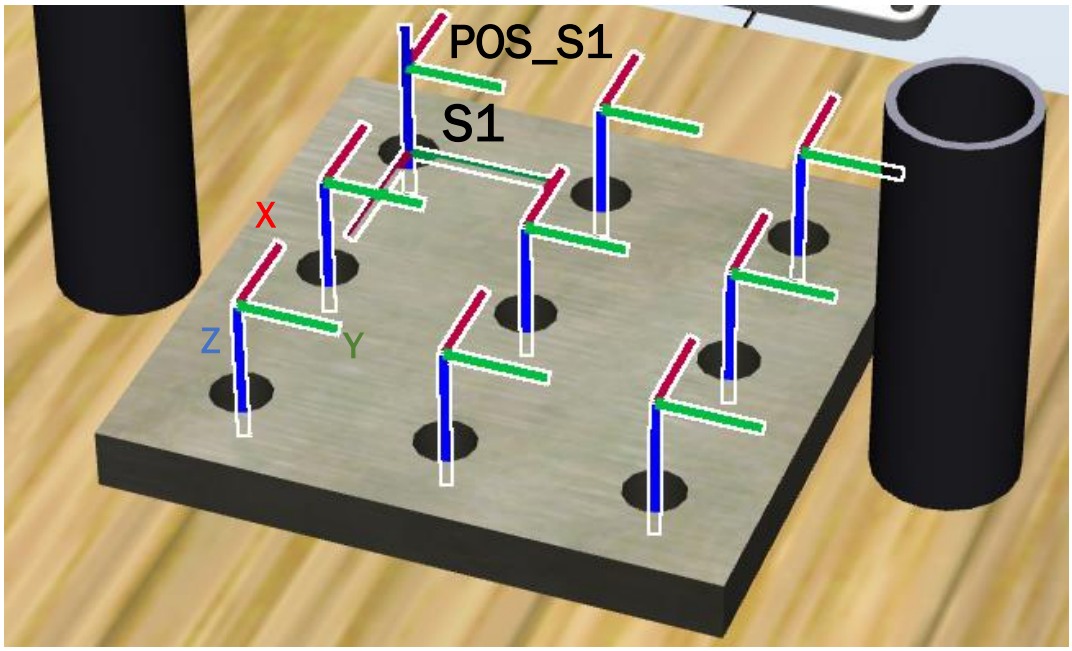


Figura 17. Puntos en el objeto de trabajo del tablero fijo (WO_TableroFijo)

Tabla de coordenadas de los puntos del tablero referidos al WO_TableroFijo

Puntos del tablero	X (mm)	Y (mm)	Z (mm)
S1	0	0	25
S2	0	100	25
S3	0	200	25
S4	100	0	25
S5	100	100	25
S6	100	200	25
S7	200	0	25
S8	200	100	25
S9	200	200	25
Pos_S1	0	0	50
Pos_S2	0	100	50
Pos_S3	0	200	50
Pos_S4	100	0	50
Pos_S5	100	100	50
Pos_S6	100	200	50
Pos_S7	200	0	50
Pos_S8	200	100	50
Pos_S9	200	200	50

De igual forma, crearemos los puntos necesarios para programar las trayectorias del robot hacia las torres de almacenamiento, esta vez con el sistema de referencia de las torres (WO_Torres) como vemos en la figura 18.

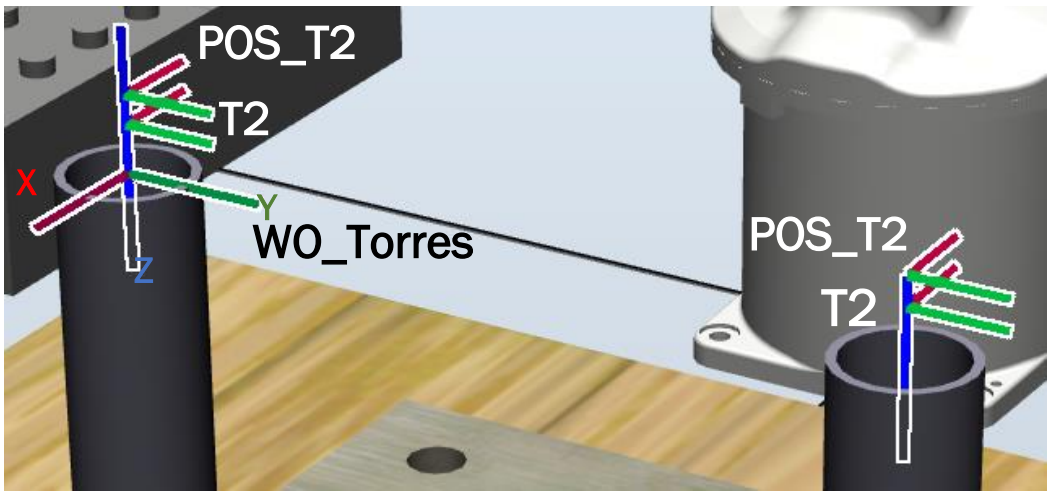


Figura 18. Puntos en el objeto de trabajo de las torres (WO_Torres)

Tabla de coordenadas de los puntos de las torres referidos al WO_Torres

Puntos de las torres	X (mm)	Y (mm)	Z (mm)
T1	0	0	25
T2	0	410	25
Pos_T1	0	0	40
Pos_T2	0	410	40

Descripción de E/S digitales

A la hora de simular señales de E/S digitales, podemos crear eventos que establezcan los valores de las señales cuando se cumplan unas determinadas condiciones de disparo, o bien podemos establecer los valores de las señales manualmente.

El sistema de E/S maneja las señales intercambiadas con el controlador IRC5 del robot. La ventana "Sistema de E/S" del programa RobotStudio se utiliza para ver y establecer las señales configuradas para activar y desactivar las unidades de entradas y salidas.

El sistema de E/S de un controlador se compone de buses de E/S, unidades de E/S y señales de E/S. Estas señales permiten la comunicación entre el controlador y los equipos externos o el cambio de variables dentro de un programa del robot.

Las señales de entrada notifican algo al controlador. En nuestro caso, las entradas digitales que manejaremos en la parte real corresponderán a los sensores del tablero y las torres. En cambio, en este apartado de simulación, todas nuestras señales digitales serán salidas para conseguir que el programa funcione de forma automatizada sin necesidad de activar manualmente los sensores del tablero, como ocurriría si los configurásemos como entradas digitales. El controlador usa las señales de salida para notificar que se ha cumplido una condición determinada. Por ejemplo, una vez que el robot ha finalizado una secuencia, es posible establecer una señal de salida.

Las señales virtuales son señales que no están configuradas para pertenecer a ninguna unidad de E/S física. Se encuentran dentro de la memoria del controlador, y se utilizan para establecer variables y almacenar los cambios en un programa del robot.

Simulación de E/S digitales

Para simular las E/S de nuestro trabajo en RobotStudio, primero debemos configurar las señales en la pestaña *"Controlador"* del programa.

En nuestro caso, tenemos asignada una unidad llamada "board10" que contiene las señales de las que dispone el robot físico del laboratorio de la escuela. Esta unidad contiene todos los buses necesarios para el funcionamiento del robot. La tarjeta de señales dispone de 16 entradas y 16 salidas para configurar. En el siguiente punto se muestra un cuadro resumen con las E/S digitales que necesitaremos para el funcionamiento de nuestro programa.

La distribución de los sensores de la estación para configurar las salidas digitales se muestra en la siguiente figura 19.

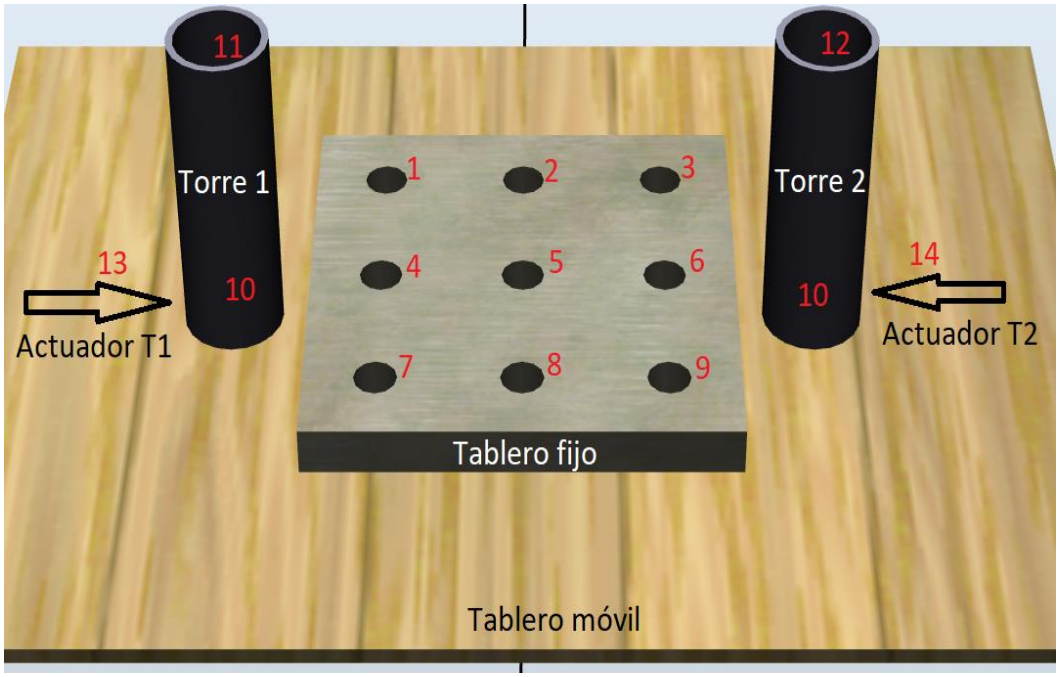


Figura 19. Componentes y sensores de la estación. Vista usuario

Los sensores inferiores de las torres en la simulación siempre van a estar activos porque dichas torres siempre tendrán bolas en su interior, por ello utilizamos únicamente una salida digital (do10) conjunta para ambos sensores, la cual estará activa en todo momento. En la práctica real los sensores inferiores de las torres nos ayudarán a comprobar que existen bolas almacenadas antes de comenzar el ensayo.

Los sensores superiores de las torres ocuparán dos salidas digitales distintas (do11 y do12), que estarán activas hasta que se lance una bola desde una de las torres (de forma virtual). Entonces, se desactivará la salida correspondiente a la torre de la que haya salido la bola. A través de las salidas do13 y do14, las cuales nos indican qué actuador neumático se ha activado, podremos conocer desde qué torre se ha lanzado la bola, y comprobar de forma visual que el sensor superior de la torre correspondiente se desactivará.

Cuando una bola lanzada desde cualquier torre caiga en uno de los nueve orificios del tablero fijo, se activará la correspondiente salida digital (do1 - do9) que representa a cada uno de dichos sensores. El robot se dirigirá hacia la posición del sensor del cual se haya activado su salida digital, cogerá la bola de forma virtual, y la depositará en la torre.

La siguiente tabla muestra las conexiones entre señales de salida digitales en RobotStudio y los componentes que forman la estación.

Tabla E/S digitales de la estación simulada

Name	Type of Signal	Assigned to Unit	Signal Identification Label
do1	Digital Output	board10	Sensor 1 del tablero fijo
do2	Digital Output	board10	Sensor 2 del tablero fijo
do3	Digital Output	board10	Sensor 3 del tablero fijo
do4	Digital Output	board10	Sensor 4 del tablero fijo
do5	Digital Output	board10	Sensor 5 del tablero fijo
do6	Digital Output	board10	Sensor 6 del tablero fijo
do7	Digital Output	board10	Sensor 7 del tablero fijo
do8	Digital Output	board10	Sensor 8 del tablero fijo

do9	Digital Output	board10	Sensor 9 del tablero fijo
do10	Digital Output	board10	Sensores inferiores de las torres
do11	Digital Output	board10	Sensor superior de la torre 1
do12	Digital Output	board10	Sensor superior de la torre 2
do13	Digital Output	board10	Actuador neumático de la torre 1
do14	Digital Output	board10	Actuador neumático de la torre 2
do15	Digital Output	board10	Abrir pinza neumática
do16	Digital Output	board10	Cerrar pinza neumática

Programación de SmartComponents

En las referencias [6] y [7] incluidas en la bibliografía al final de este trabajo, podemos ver unos videotutoriales sobre programación de SmartComponents en RobotStudio, que explicaremos aplicados al trabajo que hemos desarrollado para lograr la simulación completa de nuestra práctica docente.

En la pestaña *"Simulación"* de RobotStudio, podremos configurar la lógica de la estación para crear componentes inteligentes que nos ayuden a perfeccionar el diseño y la imagen de nuestro proyecto a la hora de simular el programa. Podemos ver unos ejemplos en los puntos [5] y [6] que contienen enlaces a video tutoriales sobre la programación de SmartComponents.

En primer lugar, crearemos en la pestaña *"Modelado"* una esfera en cada una de las posiciones de los sensores, que representarán a las bolas que irán cayendo en cada uno de los orificios tras ser lanzadas desde las torres de almacenamiento. También crearemos una esfera en la posición del TCP (Tool Center Point) de la herramienta, en nuestro caso, la pinza neumática del robot.

La siguiente figura 20, representa en nuestro modelo lo indicado anteriormente.

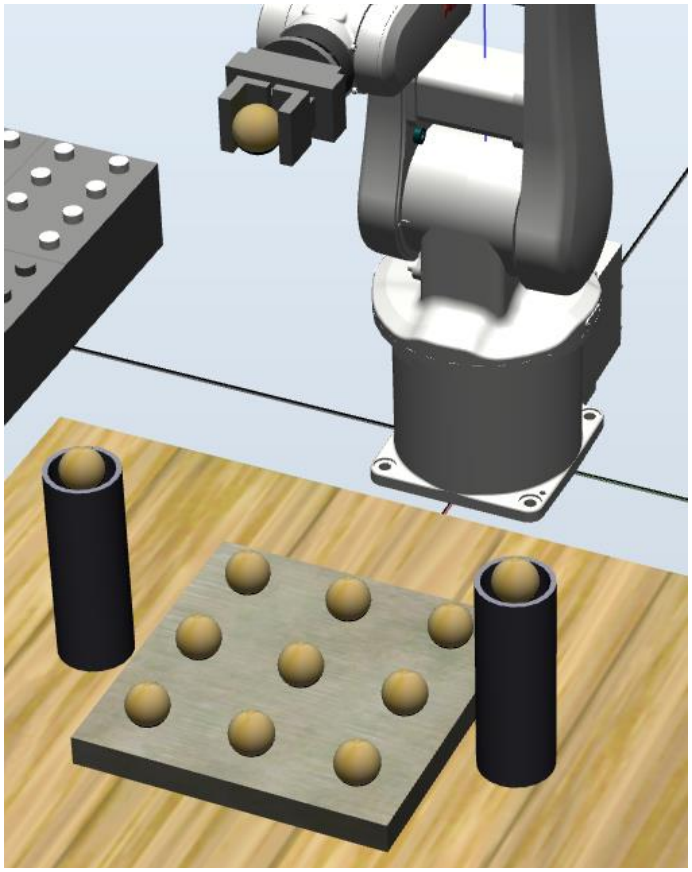


Figura 20. Representación de las bolas de madera en la estación

Tras este paso, nos dirigimos a la pestaña de *"Simulación"*, donde abriremos la lógica de la estación y comenzaremos a programar los componentes inteligentes de nuestro proyecto.

En la pestaña *"Componer"* dentro de la lógica de estación, añadiremos once componentes llamados *"Collision Sensor"*, que representarán a los sensores superiores de las torres y del tablero, como muestra la figura 21.

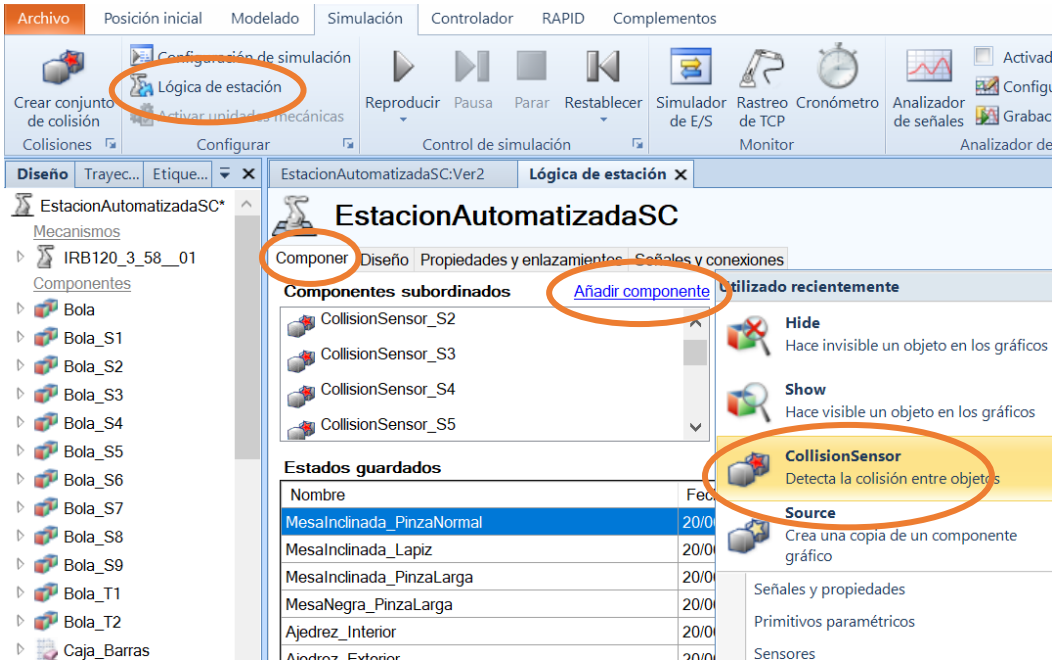


Figura 21. Lógica de estación. Creación CollisionSensor

Estos componentes permiten detectar al programa de simulación la colisión entre objetos. En nuestro caso, detectará cuando la bola ubicada en el TCP de la herramienta que hemos creado anteriormente colisiona con una de las bolas localizadas en las posiciones de los sensores de la estación.

A continuación, se muestra un ejemplo de la configuración de uno de los componentes *"Collision Sensor"* correspondiente al sensor S1 (posición del sensor 1) del tablero fijo.

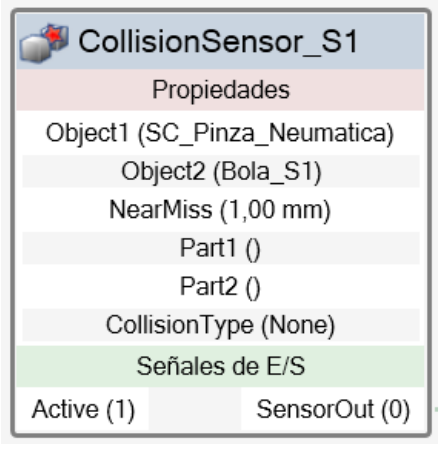


Figura 22. Lógica de estación. Configuración CollisionSensor del sensor S1

En la figura 22 anterior podemos observar cómo hemos configurado las propiedades del componente para que, al colisionar el SmartComponent "Pinza Neumática" con la esfera ubicada en la posición 1 del tablero fijo (bola

correspondiente al sensor S1), se activa el sensor de colisión (Collision Sensor) pasando de valor 0 a 1. La salida "SensorOut" del componente de colisión, la uniremos en el diseño de la lógica de estación a otro componente que crearemos y explicaremos en el siguiente apartado.

Configuramos el resto de los componentes de la misma forma que antes, obteniendo el siguiente resultado en el diseño (ver figura 23).

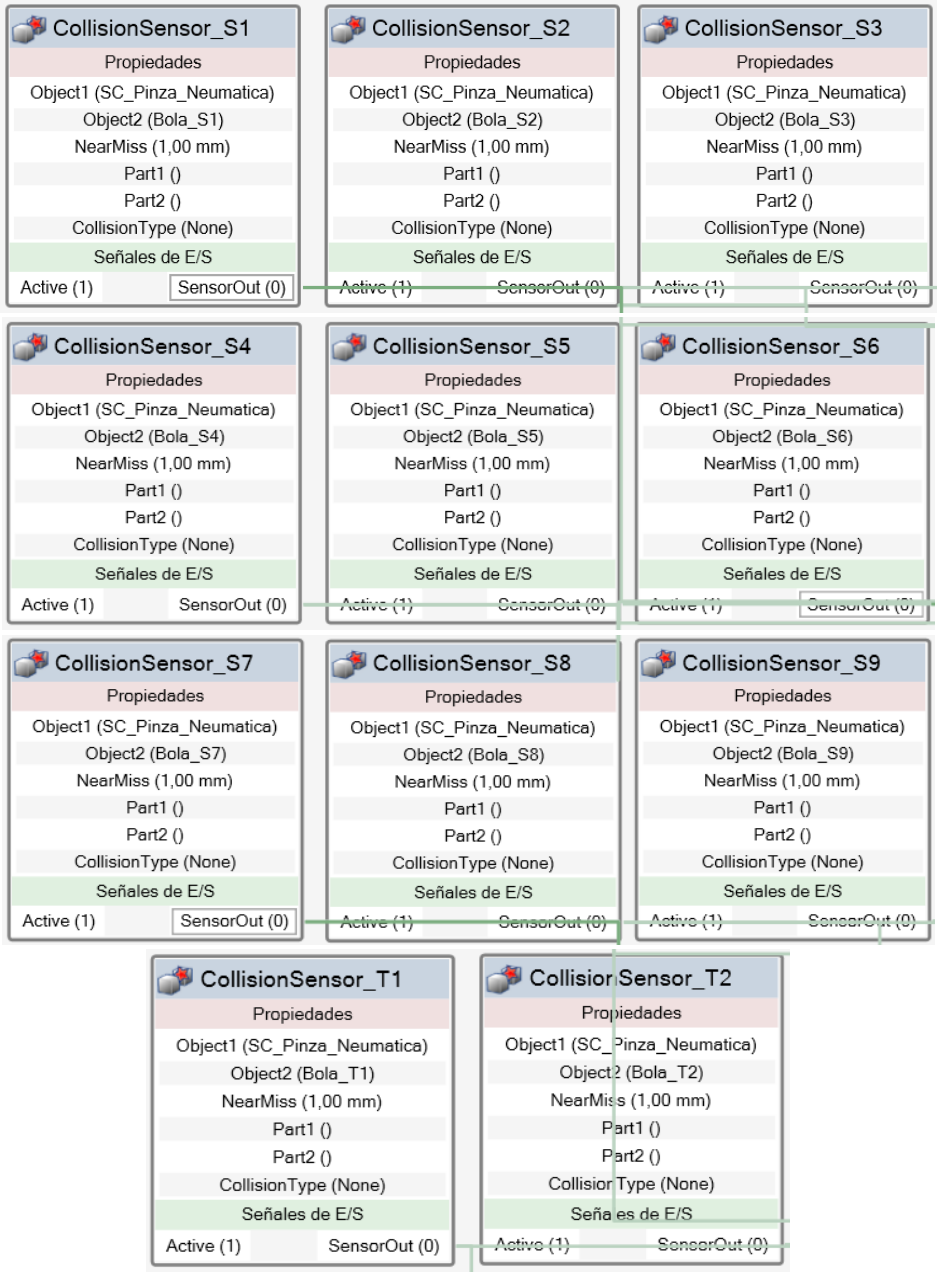


Figura 23. Lógica de estación. Componentes CollisionSensor de la estación

En la estación virtual existe un componente inteligente llamado "botonera", que nos indica la activación y desactivación de las entradas y salidas digitales

que componen nuestro trabajo de forma animada. Los botones de color rojo indican la activación de las salidas digitales correspondientes.

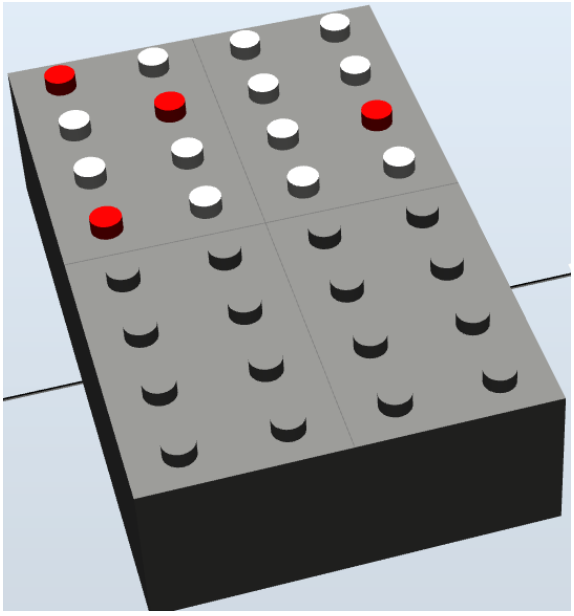


Figura 24. Componente inteligente botonera

Este SmartComponent (ver figura 24) representa a la botonera real mostrada en la figura 25 que se encuentra presente en el laboratorio de la escuela, en la cual irá conectado el cableado eléctrico de las conexiones de las entradas y salidas digitales, tanto del robot como de nuestra estación.

Figura 25. Botonera E/S digitales

En la lógica de estación de nuestro programa en RobotStudio, uniremos las E/S digitales de la botonera virtual con las E/S del IRB 120 y del SmartComponent de la pinza neumática, como se muestra en la figura 26.

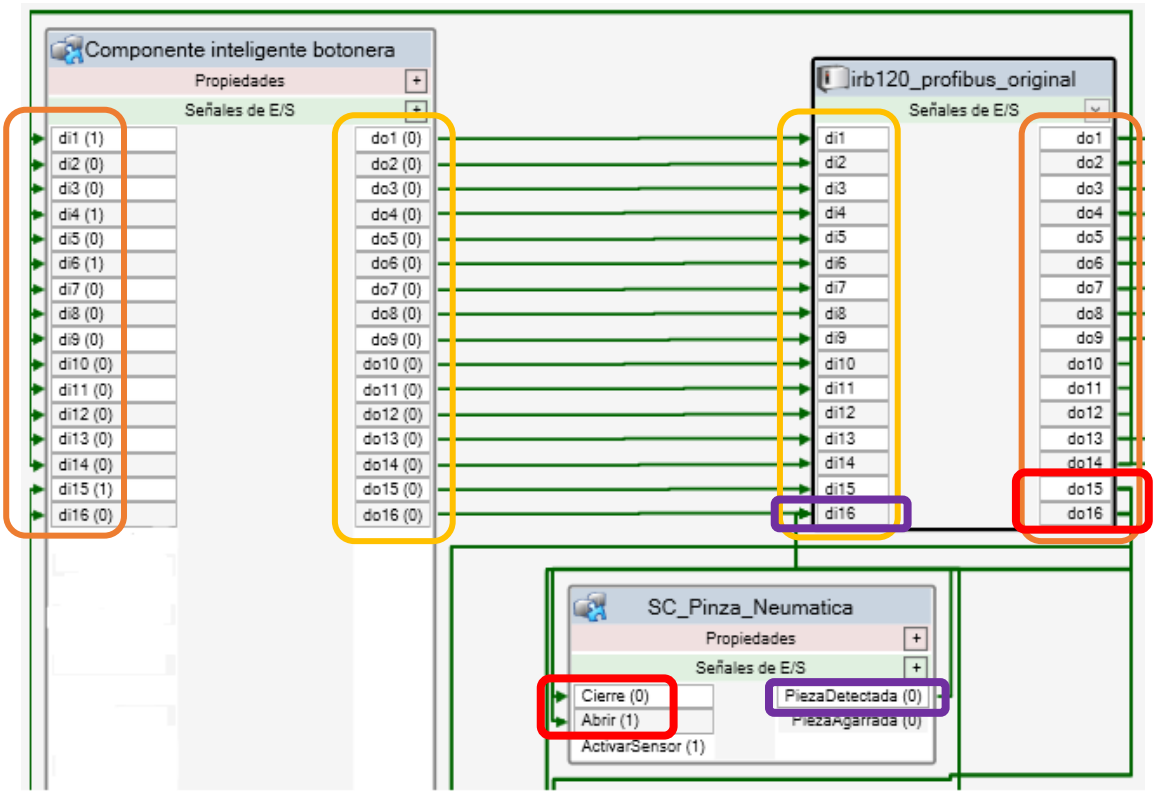


Figura 26. Lógica de estación. Conexión E/S botonera – IRB 120

De esta forma, lograremos conectar las E/S virtuales del robot y la pinza neumática a la botonera virtual para obtener una simulación animada donde podremos observar la activación o desactivación de las señales digitales en tiempo real.

En las siguientes tablas se representa el conexionado de la lógica de estación de las E/S digitales descritas anteriormente para conseguir el funcionamiento deseado.

Tabla lógica de estación. Conexiones E/S botonera – IRB 120

Entradas botonera	Salidas IRB 120
di1	do1
di2	do2
di3	do3
di4	do4
di5	do5
di6	do6
di7	do7
di8	do8
di9	do9
di10	do10
di11	do11
di12	do12
di13	do13
di14	do14
di15	do15
di16	do16

Tabla lógica de estación. Conexiones E/S IRB 120 – botonera

Salidas botonera	Entradas IRB 120
do1	di1
do2	di2
do3	di3
do4	di4
do5	di5
do6	di6
do7	di7
do8	di8
do9	di9
do10	di10
do11	di11
do12	di12
do13	di13
do14	di14
do15	di15
do16	di16

Tabla lógica de estación. Conexiones pinza neumática – IRB 120

Pinza neumática	IRB 120
Cierre	do15
Apertura	do16
Pieza detectada	di16

Simulación de la aparición y desaparición de bolas

Para lograr un diseño de la simulación más aproximado a la realidad, simularemos la aparición y desaparición de bolas que saldrán de las torres lanzadas por los actuadores neumáticos hacia los orificios del tablero fijo.

En primer lugar, crearemos las bolas virtuales en las posiciones deseadas, como ya describimos con anterioridad en el punto 2.1.4. (Programación de SmartComponents) de este documento.

Tras ello, nos dirigiremos a la pestaña de lógica de estación del programa RobotStudio y añadiremos dos nuevos componentes inteligentes llamados “Show” y “Hide” (ver figura 27).

Figura 27. Lógica de estación. Creación Show y Hide

Estos SmartComponents nos permitirán hacer visible o invisible un objeto creado previamente en los gráficos, en nuestro caso, las bolas de madera en cada una de las posiciones descritas anteriormente.

Para configurar las características de la simulación y obtener un buen resultado de la animación, nos dirigiremos a la pestaña de diseño dentro de la lógica de estación. En ella podremos ver los componentes inteligentes “Show” y “Hide” creados en el paso anterior como se muestra en la siguiente figura 28.

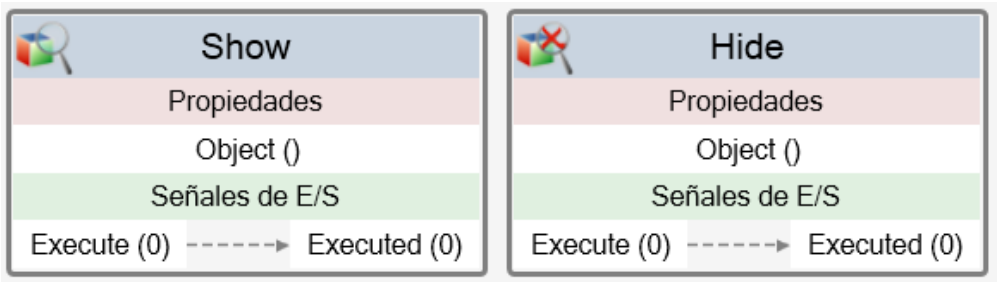


Figura 28. Lógica de estación. Componentes Show y Hide

Configuración de los SmartComponents "Show" correspondientes a los sensores del tablero fijo S1 – S9

Configuraremos cada uno de los componentes "Show" para la aparición de las bolas en las diferentes posiciones del tablero fijo, uniendo las salidas digitales del robot (do1 – do9) que corresponden a los sensores del tablero con la entrada "Execute" del componente "Show" que describe a cada bola en su posición del tablero.

Por ejemplo, para la aparición de la bola virtual de la posición S1 del tablero, primero tenemos que asignar dicha bola a las propiedades del componente "Show", como se muestra en la figura 29.

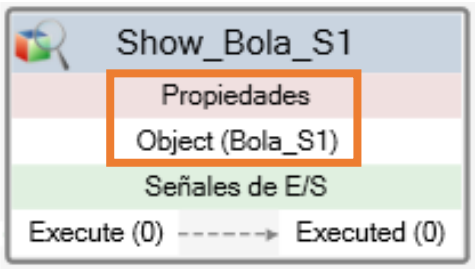


Figura 29. Lógica de estación. Configuración de propiedades bloque "Show"

Tras ello, unimos la salida digital del IRB 120 correspondiente a la posición S1 de la bola en el tablero, en este caso do1, con la señal de E/S "Execute" del componente "Show" como vemos en la figura 30. Así conseguiremos que, al iniciar la simulación del programa, si la aleatoriedad del juego hace que la bola expulsada desde una de las torres de almacenamiento caiga en la posición del sensor S1 del tablero, la bola virtual aparezca en dicha posición.

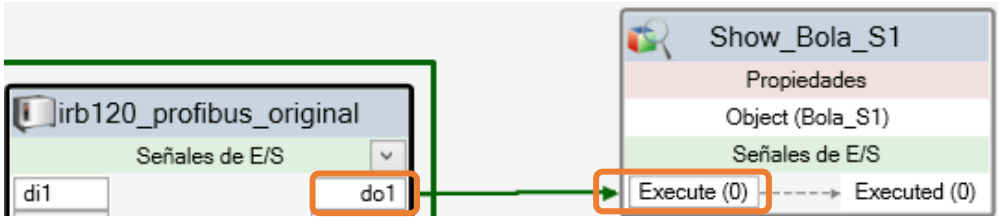


Figura 30. Lógica de estación. Configuración E/S IRB 120 para la aparición de la bola en la posición S1 del tablero

Realizaremos el mismo procedimiento para todas y cada una de las bolas del tablero fijo, configurando como se ha descrito en el paso anterior los componentes "Show" para la lograr la aparición de las bolas en las correspondientes posiciones cuando dé comienzo la simulación (ver figura 31).

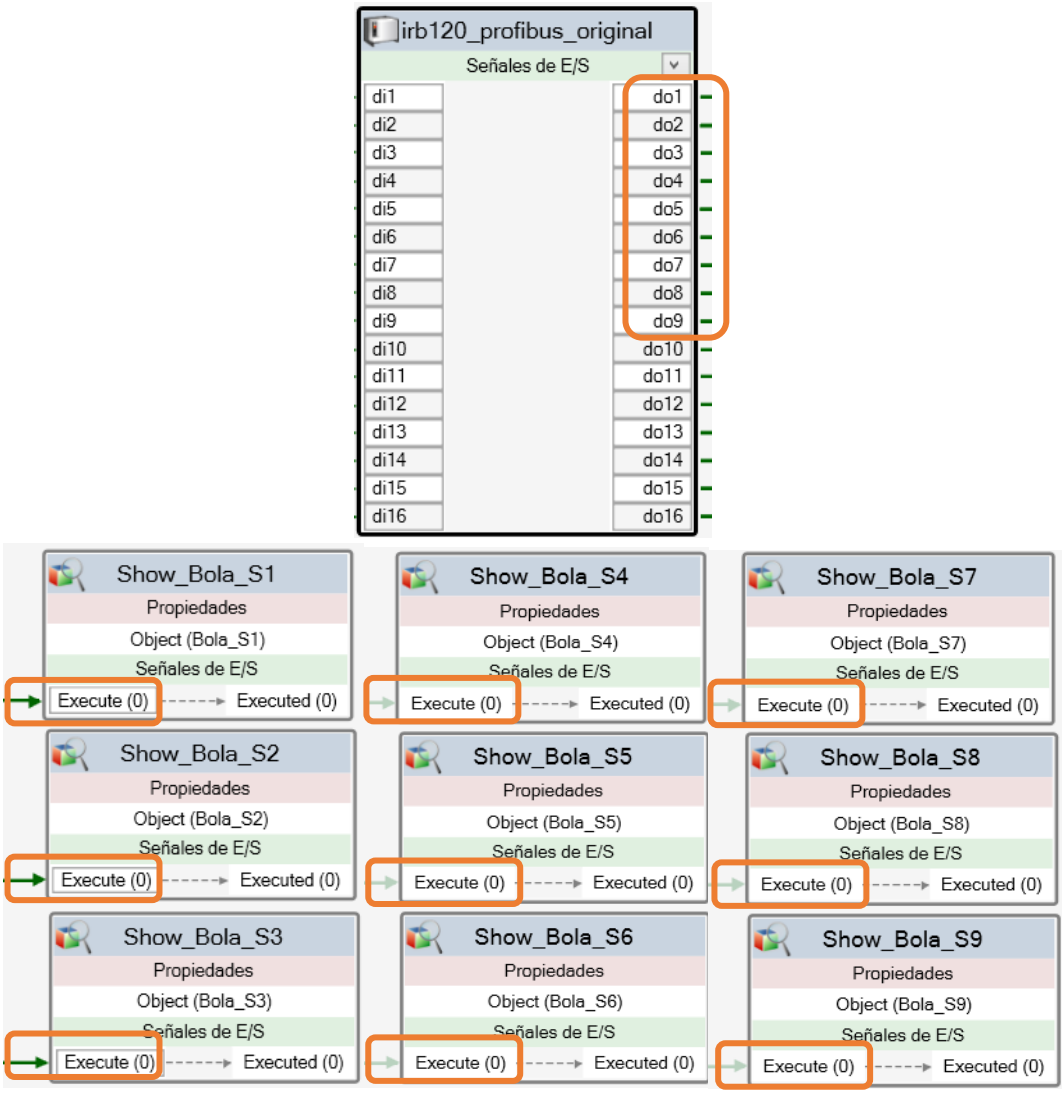


Figura 31. Lógica de estación. Configuración E/S IRB 120 para la aparición de bolas en las posiciones de los sensores del tablero

En la siguiente tabla se muestra las uniones realizadas para conseguir la correcta configuración de la estación.

Tabla de configuración de los SmartComponent "Show" correspondiente a los sensores del tablero fijo S1 – S9

Señales E/S IRB 120	SmartComponent "Show_Bola"
do1	Execute S1
do2	Execute S2
do3	Execute S3
do4	Execute S4
do5	Execute S5
do6	Execute S6
do7	Execute S7
do8	Execute S8
do9	Execute S9

Configuración del SmartComponent "Show" correspondiente a la herramienta del robot IRB 120

Para conseguir que aparezca la bola creada en la posición del TCP de la herramienta pinza neumática, configuraremos el componente "Show" uniendo las salidas "SensorOut" de todos los bloques CollisionSensor correspondientes a los sensores del tablero (S1 – S9) creados con anterioridad a la entrada "Execute" del bloque "Show" de la bola ubicada en la posición de la herramienta, como se muestra en la figura 32.

En este caso conectamos la ejecución que muestra la bola a los bloques CollisionSensor debido a que, cuando la herramienta detecte la existencia de un objeto (en este caso la bola), deberá aparecer en el TCP de la pinza neumática del robot. Esta es la función de los bloques de colisión. Cuando se detecte la colisión de una bola ubicada en uno de los sensores del tablero con el TCP de la pinza neumática, dicha bola desaparecerá de su posición y se mostrará la bola en la herramienta, la cual seguirá la trayectoria descrita por el robot desde que recoge esa bola hasta que la deposita en una de las torres de almacenaje.

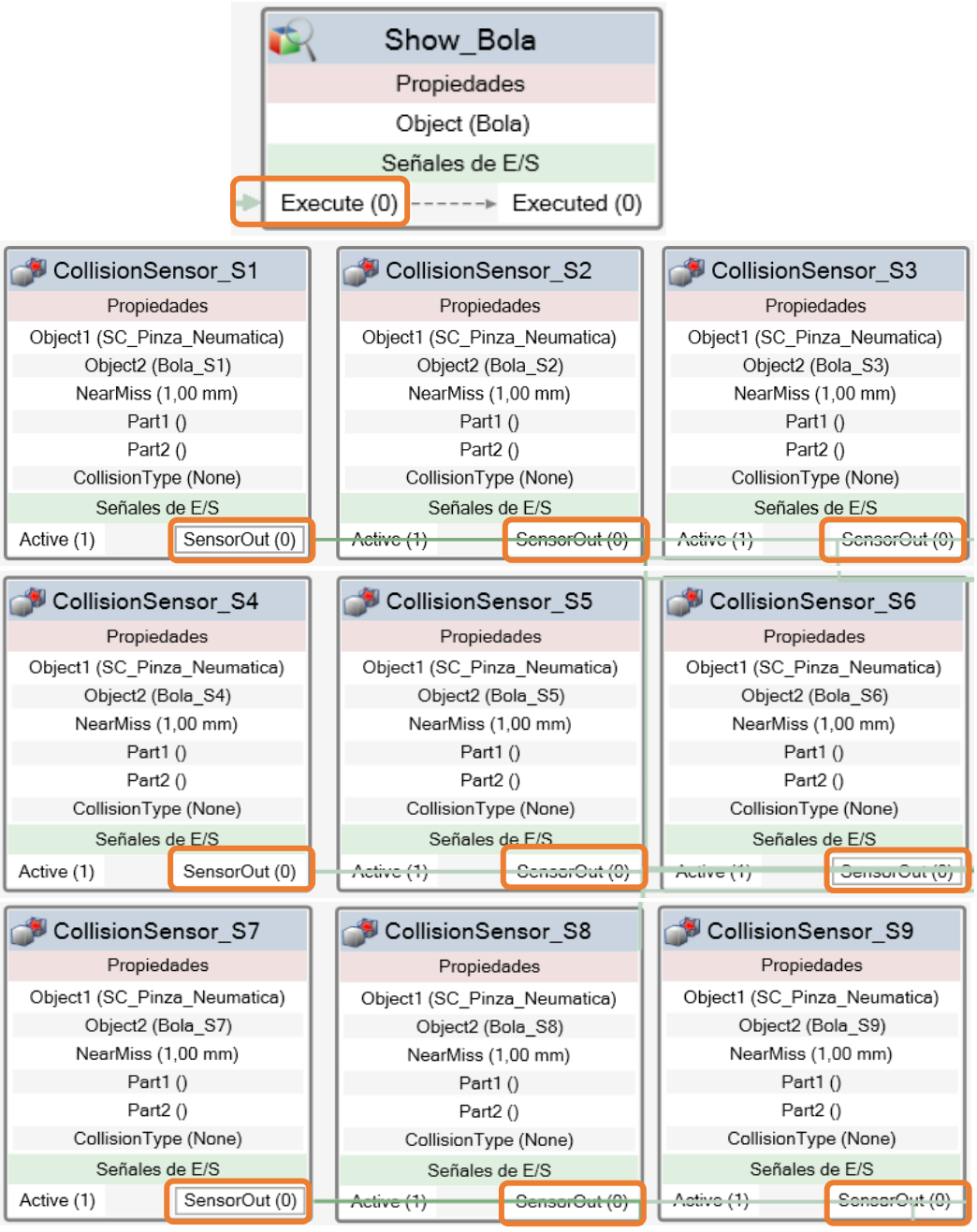


Figura 32. Lógica de estación. Configuración para la aparición de la bola en la posición del TCP de la herramienta pinza neumática

En la siguiente tabla se muestra las uniones realizadas para conseguir la correcta configuración de la estación.

Tabla de configuración del SmartComponent "Show" correspondiente a la herramienta del robot IRB 120

CollisionSensor	SmartComponent "Show_Bola"
SensorOut S1	Execute
SensorOut S2	Execute
SensorOut S3	Execute
SensorOut S4	Execute
SensorOut S5	Execute
SensorOut S6	Execute
SensorOut S7	Execute
SensorOut S8	Execute
SensorOut S9	Execute

Configuración de los SmartComponents "Show" correspondientes a los sensores de las torres de almacenamiento

Para mostrar la bola cuando el robot la deposite en una de las torres al finalizar la trayectoria de recogida desde las posiciones del tablero, añadiremos otros dos componentes inteligentes tipo "Show" que harán aparecer la bola virtual en la parte superior de las torres, indicando así la capacidad completa de la torre correspondiente, es decir, la existencia de tres bolas en su interior.

Para ello, seguimos el mismo procedimiento que en el apartado anterior, uniendo las entradas de los bloques "Show" correspondientes a las torres con las salidas "SensorOut" de los bloques CollisionSensor que representan a los sensores de dichas torres (ver figura 33).

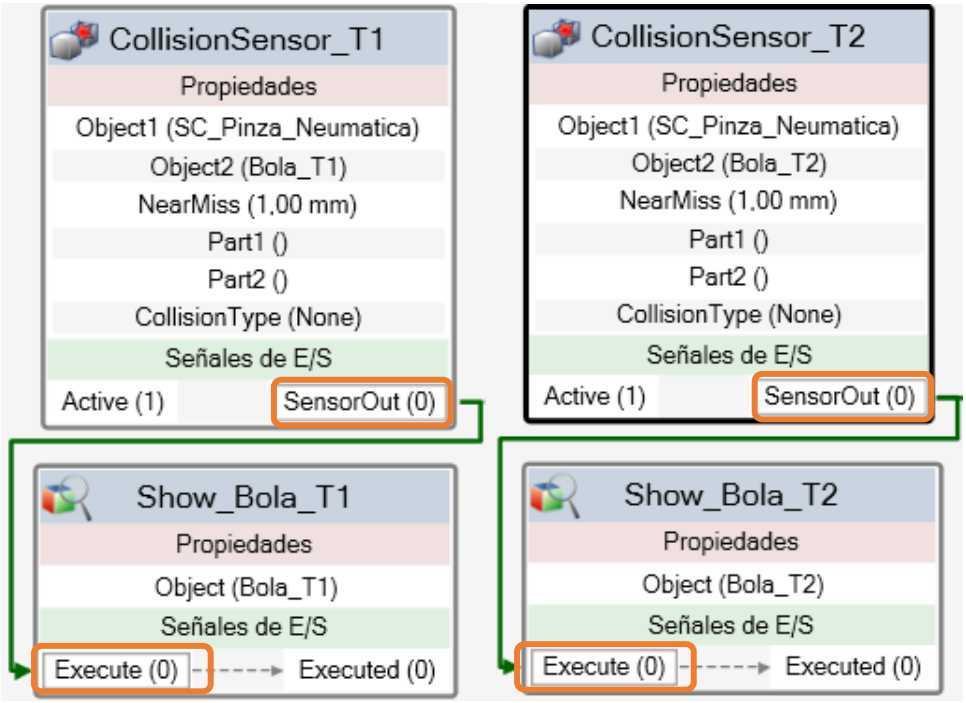


Figura 33. Lógica de estación. Configuración de la aparición de las bolas en las posiciones superiores de las torres de almacenamiento

En la siguiente tabla se muestra las uniones realizadas para conseguir la correcta configuración de la estación.

Tabla de configuración de los SmartComponents "Show" correspondientes a las torres de almacenamiento

CollisionSensor	SmartComponent "Show_Bola"
SensorOut T1	Execute T1
SensorOut T2	Execute T2

Configuración de los SmartComponents "Hide" correspondientes a los sensores del tablero fijo S1 – S9

Al igual que con el componente inteligente "Show", configuraremos cada uno de los bloques "Hide" para la desaparición de las bolas en las diferentes posiciones del tablero fijo, esta vez, uniendo las salidas de los bloques CollisionSensor con la entrada "Execute" del componente "Hide" que describe a cada bola en su posición del tablero.

Por ejemplo, para la desaparición de la bola virtual de la posición S1 del tablero, primero tenemos que asignar dicha bola a las propiedades del componente "Hide" como se muestra en la siguiente figura 34.

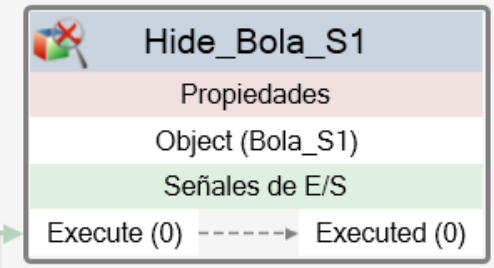


Figura 34. Lógica de estación. Configuración de propiedades bloques "Hide"

Tras ello, pasamos a unir las salidas de los bloques de colisión correspondientes a las posiciones S1 – S9 de las bolas en el tablero con las señales de E/S "Execute" de los componentes "Hide" correspondientes a cada sensor del tablero (ver figura 35).

Así conseguiremos que, por ejemplo, si la aleatoriedad del juego hace que la bola expulsada desde una de las torres de almacenamiento caiga en la posición del sensor S1 del tablero, la bola virtual aparezca en dicha posición (paso descrito por los bloques "Show" anteriormente explicados), y cuando la herramienta del robot la recoja, esta bola desaparezca al mismo tiempo que aparecerá en el TCP de la pinza neumática para acompañar la trayectoria del robot.

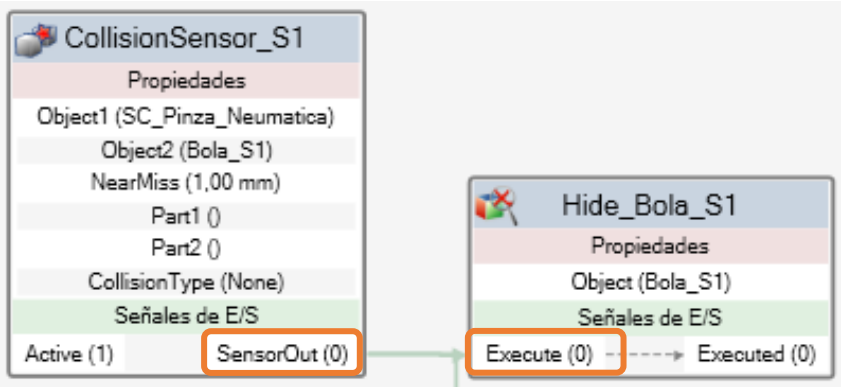


Figura 35. Lógica de estación. Configuración para la desaparición de la bola en la posición S1 del tablero

Realizaremos las conexiones de los bloques para todos los sensores del tablero (ver figura 36) de la misma forma que en la figura 35 anterior.

Figura 36. Lógica de estación. Configuración para la desaparición de la bola en la posición S1 del tablero

En la siguiente tabla se muestra las uniones realizadas para conseguir la correcta configuración de la estación.

Table de configuración de los SmartComponents "Hide" correspondientes a los sensores del tablero fijo S1 – S9

CollisionSensor	SmartComponent "Hide_Bola"
S1	Execute S1
S2	Execute S2
S3	Execute S3
S4	Execute S4
S5	Execute S5
S6	Execute S6
S7	Execute S7
S8	Execute S8
S9	Execute S9

Configuración del SmartComponent "Hide" correspondiente a la herramienta del robot IRB 120

Para conseguir que desaparezca la bola creada en la posición del TCP de la herramienta pinza neumática, configuraremos el componente "Hide" uniendo las salidas "SensorOut" de los bloques CollisionSensor correspondientes a los sensores de las torres de almacenamiento T1 – T2 a la entrada "Execute" del bloque "Hide" de la bola ubicada en la posición de la herramienta, como vemos en la figura 37.

Cuando se detecte la colisión de una bola ubicada en uno de los sensores del tablero con el TCP de la pinza neumática, el robot recogerá la bola y se dirigirá hacia una de las dos torres para depositarla en su interior. Cuando llegue a la parte superior de la torre (zona de descarga de bolas), el sensor de colisión de las torres detectará la presencia de la herramienta del robot. Entonces, la bola virtual ubicada en el TCP de la herramienta desaparecerá, y se mostrará la bola localizada en la parte superior de la torre de almacenamiento que corresponda, haciendo visible que esa torre está completamente llena, como se ha descrito anteriormente en el apartado de configuración de los SmartComponents "Show" para las torres.

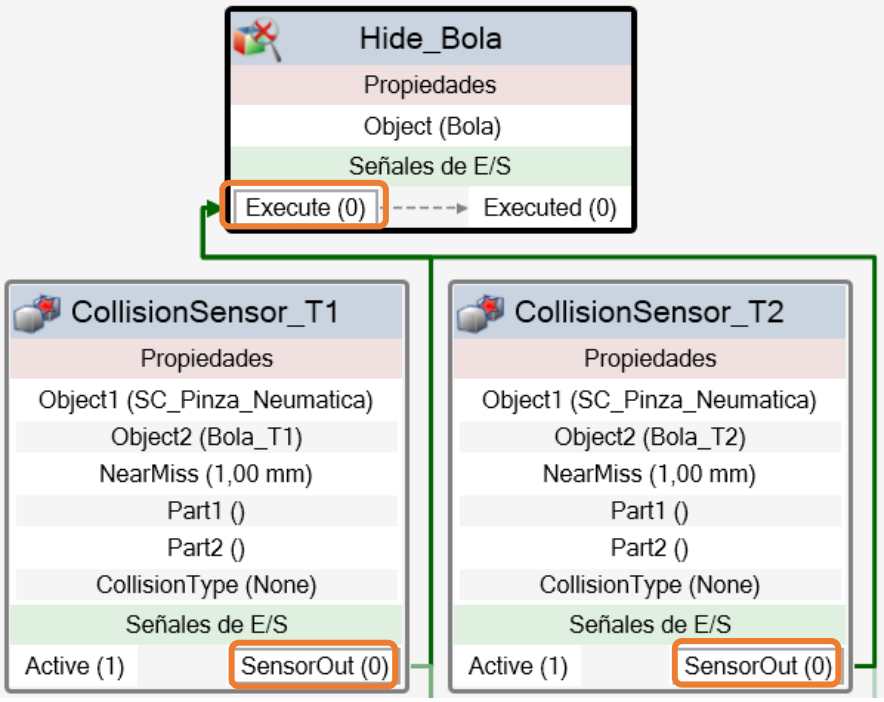


Figura 37. Lógica de estación. Configuración para la desaparición de la bola en la posición del TCP de la herramienta pinza neumática

En la siguiente tabla se muestra las uniones realizadas para conseguir la correcta configuración de la estación.

Tabla de configuración de los SmartComponents "Hide" correspondientes a la herramienta del robot IRB 120

CollisionSensor	SmartComponent "Hide_Bola"
SensorOut T1	Execute
SensorOut T2	Execute

Configuración de los SmartComponents "Hide" correspondientes a los sensores de las torres de almacenamiento

Para que la bola desaparezca cuando el robot se encuentre en posición de descarga encima de una de las torres al finalizar la trayectoria de recogida desde las posiciones del tablero, añadiremos otros dos componentes inteligentes tipo "Hide" que conectaremos a las salidas digitales (do13 y do14) del robot IRB 120. Escogemos estas salidas del robot porque son las encargadas de la activación de las electroválvulas, las cuales harán que los actuadores neumáticos de cada torre lancen una bola hacia el tablero, repitiendo el ciclo del programa. Cuando se active una de estas salidas do13 o

do14 del robot, el actuador correspondiente lanzará una bola virtualmente, que desaparecerá de la cima de la torre y aparecerá de forma aleatoria en uno de los nueve orificios del tablero fijo.

Para ello, unimos las entradas "Execute" de los bloques "Hide" correspondientes a las torres de almacenamiento con las salidas digitales do13 y do14 del bloque que representa al brazo robot IRB 120 (ver figura 38).

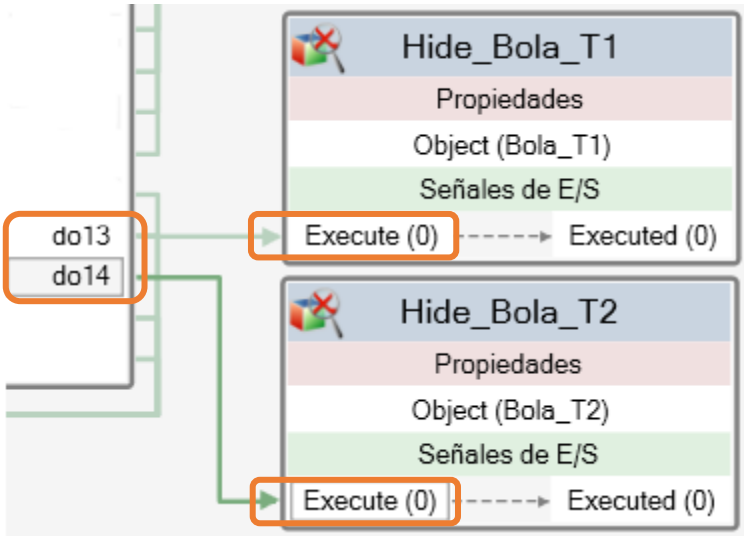


Figura 38. Lógica de estación. Configuración para la desaparición de las bolas en las posiciones superiores de las torres de almacenamiento

En la siguiente tabla se muestra las uniones realizadas para conseguir la correcta configuración de la estación.

Tabla de configuración de los SmartComponents "Hide" correspondientes a los sensores de las torres de almacenamiento

Señales E/S IRB 120	SmartComponent "Hide_Bola"
do13	Execute T1
do14	Execute T2

2.2. Programa RAPID de la simulación

Descripción y objetivos

El objetivo principal de esta simulación es la comprobación de que el proyecto realizado puede llevarse a cabo de forma física y trabajar con él como actividad de prácticas académicas en la EII. Este programa realizará de forma automática todos los movimientos del robot IRB 120 descritos en sus trayectorias hacia los diferentes puntos de la estación mediante la activación y desactivación, también automática y conjunta, de todas las E/S digitales que representan a los componentes electrónicos, eléctricos y neumáticos que más tarde se implementarán en la práctica real junto con el robot.

Mediante el programa RAPID de simulación en RobotStudio, desarrollaremos una secuencia de instrucciones que controlarán el robot y que dividiremos en tres partes.

- Rutina principal (main), donde se inicia la ejecución del programa
- Subrutinas que servirán para dividir el programa en partes más pequeñas con el fin de obtener un programa modular
- Datos del programa que definirán posiciones, valores numéricos y sistemas de coordenadas

Sincronización y descripción de elementos de la estación con RAPID

En primer lugar, sincronizaremos nuestra estación creada en RobotStudio con el programa de RAPID para importar las coordenadas de los componentes, objetos de trabajo y posiciones de los puntos de la estación. Desde la pestaña de “Posición inicial” podremos realizar lo expuesto previamente (ver figura 39).

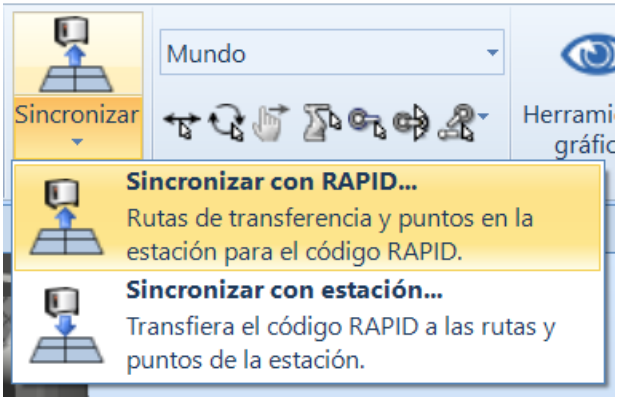


Figura 39. Sincronización con RAPID de la estación

Una vez completado este paso, nos dirigimos a la pestaña “RAPID” para comenzar a realizar el código de nuestro programa, que estará compuesto de dos módulos principales (CalibData y EstaciónAutomatizadaSC).

Al sincronizar la estación, nos aparece en el módulo principal del programa de RAPID todos los puntos y componentes que forman la estación, con sus respectivos sistemas de referencia (objetos de trabajo de cada punto). A continuación, se muestran estos datos importados desde la estación al código del programa.

- Módulo CalibData: Representa los objetos de trabajo creados y la herramienta acoplada al robot.

```
PERS tooldata Pinza20170721:=[TRUE,[[0,0,67],[0.707
TASK PERS wobjdata WO_TableroFijo:=[FALSE,TRUE,"",[
TASK PERS wobjdata WO_Torres:=[FALSE,TRUE,"",[[420,
```

- Módulo EstaciónAutomatizadaSC

- Desde el wobj0, creamos la posición inicial de referencia del robot en la que estará ubicado al arrancar el programa (ver figura 40, izquierda), y una posición base (ver figura 40, derecha) desde la que el robot trabajará y se dirigirá más fácilmente hacia los puntos del tablero.

```
!Posicion inicial del robot
CONST robtarget Pos_Inicial:=[[374.55,0,630.67],[0.710481,0.00000

!Posicion base para ejecutar desplazamientos hacia los objetivos
CONST robtarget Pos_Base:=[[433.10135365,0,288.094692696],[0.0012
```

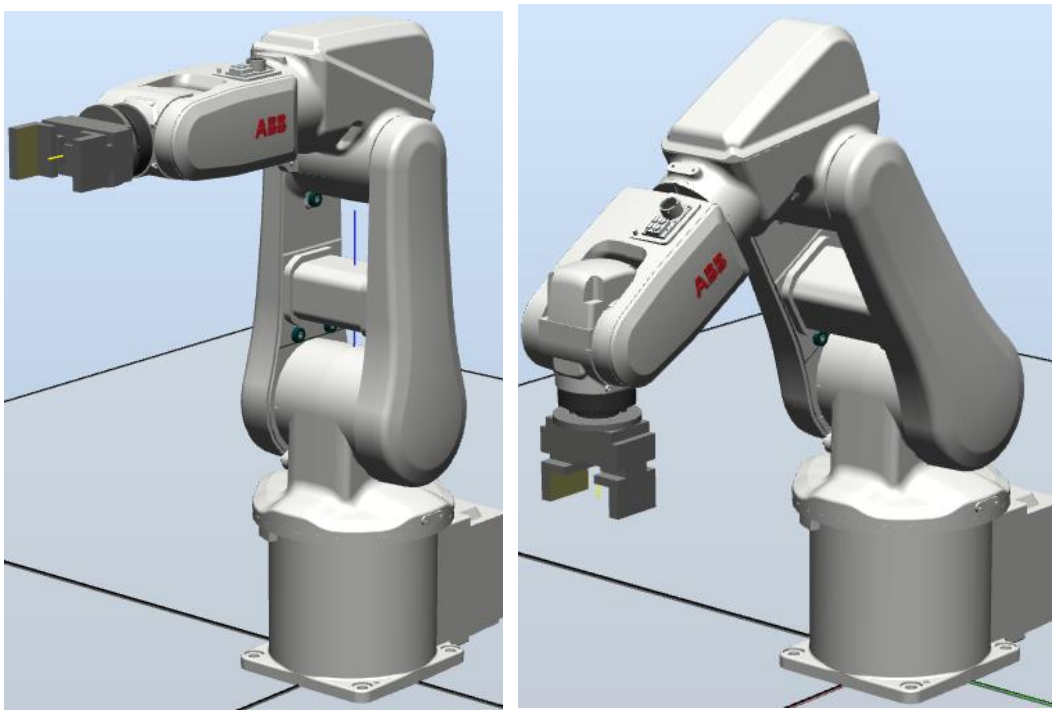


Figura 40. Posición inicial y posición base de trabajo del IRB 120

- Desde el WO_TableroFijo y el WO_Torres, observamos las distintas posiciones creadas anteriormente en los orificios del tablero y en las partes superiores de las torres de almacenaje. En cada uno de ellos tenemos dos posiciones:

En la primera (ver figura 41), ubicada unos centímetros por encima del objeto de trabajo, el robot se dirigirá de forma rápida para agilizar el juego y hacerlo más entretenido. Estas posiciones se denominarán “Pos_SX”, donde la X representa al número correspondiente a cada sensor del tablero, y “Pos_TY”, donde la Y representa al número correspondiente a cada torre.

```
!Posiciones de los puntos exactos en tablero fijo y torres
CONST robtarget S1:=[[0,0,25],[0,0,1,0],[-1,0,-2,0],[9E+09,
CONST robtarget S2:=[[0,100,25],[0,0,1,0],[0,0,-2,0],[9E+09
CONST robtarget S3:=[[0,200,25],[0,0,1,0],[0,-1,-1,0],[9E+0
CONST robtarget S4:=[[100,0,25],[0,0,1,0],[-1,0,-2,0],[9E+0
CONST robtarget S5:=[[100,100,25],[0,0,1,0],[0,0,-2,0],[9E+
CONST robtarget S6:=[[100,200,25],[0,0,1,0],[0,-1,-1,0],[9E
CONST robtarget S7:=[[200,0,25],[0,0,1,0],[-1,0,-2,0],[9E+0
CONST robtarget S8:=[[200,100,25],[0,0,1,0],[0,0,-2,0],[9E+
CONST robtarget S9:=[[200,200,25],[0,0,1,0],[0,0,0,0],[9E+0
CONST robtarget T1:=[[0,0,25],[0,0,1,0],[-1,0,-2,0],[9E+09,
CONST robtarget T2:=[[0,410,25],[0,0,1,0],[0,0,0,0],[9E+09,
```

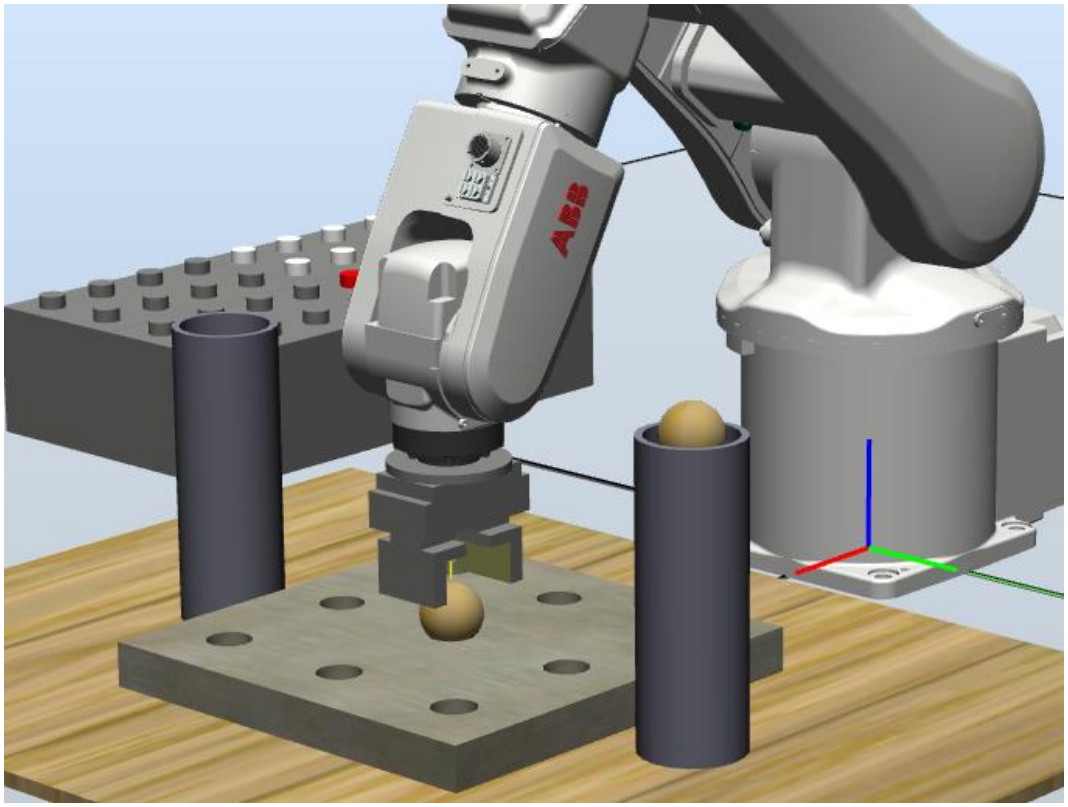


Figura 41. IRB 120 en posición "Pos_S5"

La segunda posición se localiza unos milímetros por encima de la base de cada objeto (ver figura 42), donde el robot se dirigirá más lentamente y de forma precisa después de llegar al primer punto para recoger la bola desde uno de los sensores del tablero o descargarla en la torre correspondiente sin ocasionar ningún problema. Estas posiciones se denominarán "SX", donde la X representa al número correspondiente a cada sensor del tablero, y "TY", donde la Y representa al número correspondiente a cada torre.

```
!Posiciones cercanas a los puntos en tablero fijo y torres
CONST robtarget Pos_S1:=[[0,0,50],[0,0,1,0],[-1,0,-2,0],[9E
CONST robtarget Pos_S2:=[[0,100,50],[0,0,1,0],[0,0,-2,0],[9
CONST robtarget Pos_S3:=[[0,200,50],[0,0,1,0],[0,-1,-1,0],[
CONST robtarget Pos_S4:=[[100,0,50],[0,0,1,0],[-1,0,-2,0],[
CONST robtarget Pos_S5:=[[100,100,50],[0,0,1,0],[0,0,-2,0],
CONST robtarget Pos_S6:=[[100,200,50],[0,0,1,0],[0,-1,-1,0]
CONST robtarget Pos_S7:=[[200,0,50],[0,0,1,0],[-1,0,-2,0],[
CONST robtarget Pos_S8:=[[200,100,50],[0,0,1,0],[0,0,-2,0],
CONST robtarget Pos_S9:=[[200,200,50],[0,0,1,0],[0,-1,-1,0]
CONST robtarget Pos_T1:=[[0,0,40],[0,0,1,0],[-1,0,-2,0],[9E
CONST robtarget Pos_T2:=[[0,410,40],[0,0,1,0],[0,-1,-1,0],[
```

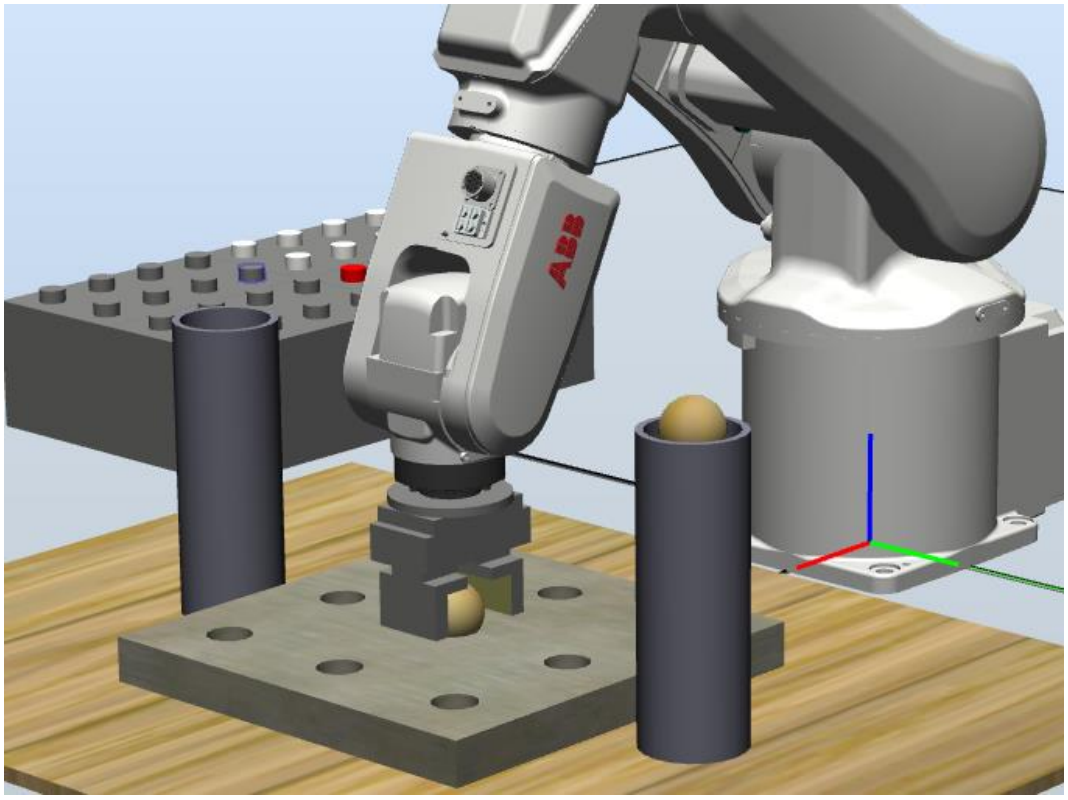


Figura 42. IRB 120 en posición "S5"

Programación y descripción de las funciones del programa

A continuación, se muestra el conjunto de funciones utilizadas para el correcto funcionamiento del programa, explicando el objetivo de cada una de ellas y una breve descripción de su funcionamiento.

Funciones "AbrirPinza ()" y "CerrarPinza ()"

Para que la herramienta que acoplamos al IRB 120 en nuestra simulación recoja las bolas del tablero y las descargue en las torres de almacenamiento, necesitaremos las funciones que permiten la apertura y cierre del componente inteligente que compone la pinza neumática. Mediante la programación de las E/S digitales que ya se describieron en apartados anteriores, conseguiremos programar estos procedimientos para lograr el objetivo descrito.

```
!Abrir pinza neumática
PROC AbrirPinza()
    Set do15; !Abrir y desattacher
    WaitTime 2;
    Reset do15;
ENDPROC

!Cerrar pinza neumática
PROC CerrarPinza()
    Set do16;  !Activar los sensores de attacher
    WaitTime 2;
    Reset do16;
ENDPROC
```

Las salidas digitales do15 y do16 estarán ligadas a la apertura y cierre de la pinza, respectivamente. En ambas funciones, activaremos la salida digital correspondiente mediante la sentencia "Set", introduciremos un tiempo de espera de 2 segundos para visualizar en pantalla la activación de dicha salida digital y tras ello, la desactivaremos con la sentencia "Reset".

Función "aleatorio ()"

Esta función nos permitirá obtener un número aleatorio entre 0 y 1. Debemos declarar una variable denominada "semilla" al principio del programa, que utilizaremos para inicializar un generador de números pseudoaleatorios.

```
!Número aleaotorio, semilla
LOCAL PERS num seed:=13197;

!Obtiene valor aleatorio entre 0 y 1
FUNC num aleatorio()
    seed:=(171*seed) MOD 30269;
    RETURN seed/30269;
ENDFUNC
```

Dependiendo del valor actual del número semilla, la función aleatoria nos devolverá un nuevo valor comprendido entre 0 y 1 que posteriormente utilizaremos para la activación aleatoria de los actuadores neumáticos de las torres, y también para designar la trayectoria del robot sobre hacia qué torre debe dirigirse tras recoger una bola del tablero fijo.

Funciones "Path_SX ()"

Este conjunto de procedimientos describirá la trayectoria del robot hacia las posiciones de los sensores del tablero a la hora de recoger una bola. La X representa el número correspondiente a cada posición del tablero, desde el sensor 1 al 9.

Para explicar el funcionamiento del conjunto, tomaremos como ejemplo la trayectoria descrita por el robot hacia la posición del sensor 1 del tablero, representada por el procedimiento Path_S1 que se muestra a continuación.

```
!Trayectorias hacia los sensores del tablero fijo
PROC Path_S1()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S1,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S1,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do1,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S1,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC
```

Al comienzo del programa, el robot se dirigirá desde la posición inicial a la posición base con un movimiento lineal descrito por la sentencia "MoveJ". Si el robot ya ha iniciado el programa previamente, partirá desde la posición base de forma directa. Después, se dirigirá hacia la primera posición del sensor correspondiente en el tablero (en este caso el sensor 1) de forma lineal, y llamaremos al procedimiento encargado de abrir la pinza cuando concluya la trayectoria hacia dicho punto. Tras la apertura de la pinza, el robot se desplazará lentamente hacia la posición exacta del sensor 1 para recoger la bola con cuidado, y conjuntamente desactivará la salida digital correspondiente (en este caso do1, que representa al sensor 1) mediante la sentencia "MoveJDO". Introduciremos una espera temporal para que el robot se posicione de forma correcta para recoger la bola virtual, y cerraremos la pinza llamando al procedimiento correspondiente. Tras otra espera de 2 segundos, el robot se volverá a la primera posición por encima del orificio del tablero mediante un movimiento lineal. Posteriormente, llevará la bola recogida hacia una de las dos torres de forma aleatoria como explicaremos más adelante.

Utilizaríamos el mismo procedimiento para describir la trayectoria del robot hacia los diferentes puntos del tablero fijo con las distintas funciones Path_SX.

Funciones "Path_TY ()"

En este apartado describiremos los procedimientos que ejecutan las trayectorias del robot hacia ambas torres de almacenamiento de bolas. La Y representa el número correspondiente a cada posición encima de las torres, del 1 al 2.

Al igual que antes, explicaremos el funcionamiento del conjunto tomando como ejemplo la trayectoria descrita por el robot hacia la torre 1 (ver figura 43), representada por el procedimiento Path_T1 que mostramos a continuación.

```
!Trayectorias hacia las torres de almacenamiento
PROC Path_T1()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_T1,v100,z0,Pinza20170721\WObj:=WO_Torres;
    MoveJ T1,v20,fine,Pinza20170721\WObj:=WO_Torres;
    WaitTime 2;
    AbrirPinza;
    bolasT1:=bolasT1+1;
    WaitTime 2;
    CerrarPinza;
    MoveJ Pos_T1,v100,z0,Pinza20170721\WObj:=WO_Torres;
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
ENDPROC
```

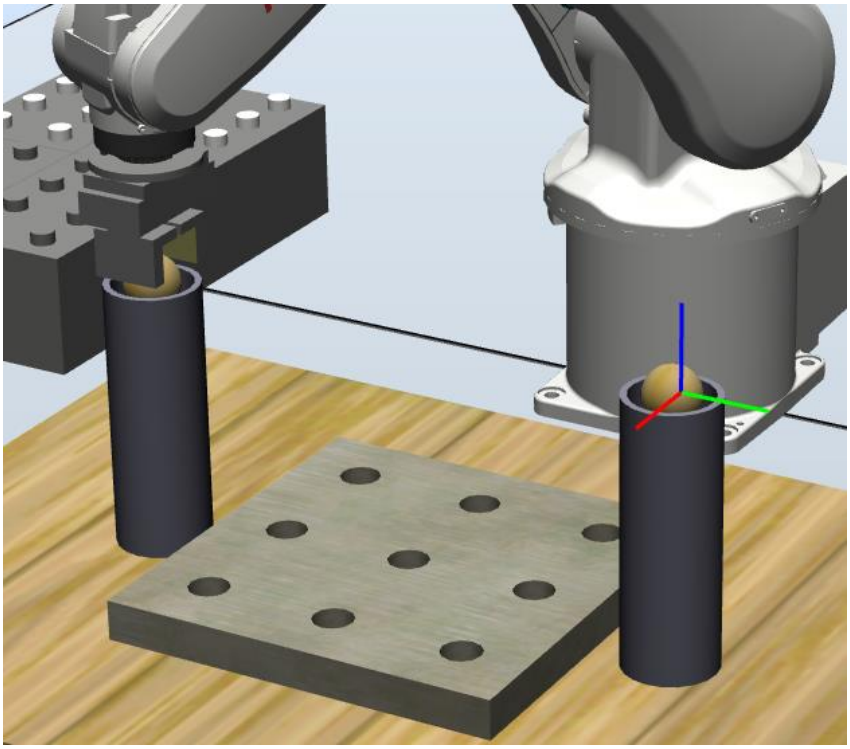


Figura 43. IRB 120 en posición de descarga en la torre 1

Tras la ejecución de la trayectoria del robot para recoger una bola del tablero, éste se dirigirá a la posición base de trabajo de forma lineal. A continuación, se desplazará hasta la primera posición en la parte superior de una de las torres (en este caso la torre 1) tras haberse ejecutado en el main del programa la función aleatoria que designa en cuál de las dos torres debe depositarse la bola recogida. El robot se posicionará más lentamente en el punto de descarga más cercano a la torre mediante un movimiento lineal, esperará 2 segundos y se activará el procedimiento de apertura de la pinza neumática, que hará que la bola caiga dentro de la torre de almacenamiento. Tras ello, un contador que debemos declarar al principio del programa inicializado a 3 (número de bolas que contiene cada torre al comienzo del programa) llamado "BolasTY", actualizará su valor incrementando en 1 el número de bolas que contiene la torre en la que se acabe de ejecutar esta trayectoria. Por último, se cerrará la pinza neumática, el robot subirá lentamente hasta la primera posición de la parte superior de la torre y se dirigirá a la posición base de trabajo hasta que caiga una nueva bola en uno de los orificios del tablero, donde comenzará de nuevo la ejecución del juego.

Los contadores de bolas en el interior de cada torre descritos anteriormente son los siguientes, y deberán ir declarados al inicio del programa.

```
VAR num bolasT1:=3;
VAR num bolasT2:=3;
```

Programación y descripción del main principal del programa

La función main sirve como punto de partida para la ejecución del programa RAPID, la cual controla su ejecución dirigiendo las llamadas a otras funciones y procedimientos.

Previamente, tenemos que declarar las variables que utilizaremos en el procedimiento main para su ejecución. En nuestro caso son las siguientes.

```
VAR num rand;
VAR num RandomActuadores;
VAR num RandomTablero;
VAR num RandomTorres;
```

La variable rand, la utilizamos para llamar a la función "aleatorio ()" dentro de nuestro main, descrita en apartados anteriores.

Las variables “RandomActuadores”, “RandomTablero” y “RandomTorres” almacenarán distintos valores aleatorios dados por la función que genera estos números. Dichos valores se renovarán con cada ejecución del programa.

- La variable “RandomActuadores” guarda un valor generado por la función “aleatorio ()” multiplicado por dos, para activar automáticamente una de las dos electroválvulas, las cuales accionarán sus correspondientes actuadores de forma virtual.
- La variable “RandomTablero” guarda un valor generado por la función “aleatorio ()” multiplicado por nueve, que determinará en qué orificio del tablero deberá aparecer la bola virtual activándose su correspondiente salida digital.
- La variable “RandomTorres” guarda un valor generado por la función “aleatorio ()” multiplicado por dos, que nos indicará la trayectoria hacia una de las dos torres que deberá seguir el robot para descargar la bola en su interior.

Al iniciar el programa, se ejecuta la instrucción de movimiento lineal que situará al robot en la posición inicial (ver figura 44) en la que deberíamos encontrarlo en el laboratorio de la escuela. La configuración de los eslabones del IRB 120 en la posición inicial se muestra en la siguiente figura.

```
MoveJ Pos_Inicial,v300,fine,tool0;
```

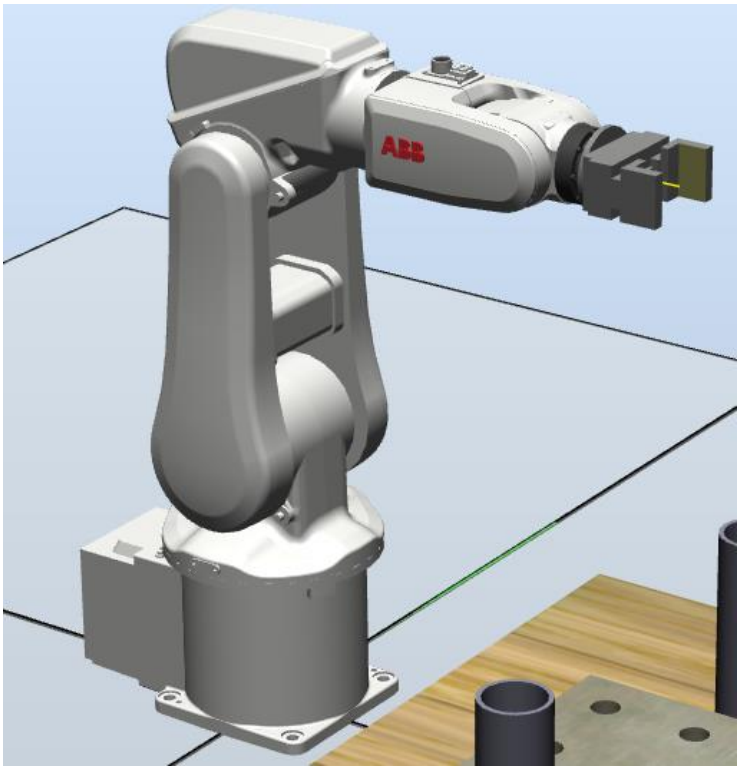


Figura 44. Simulación IRB 120 en la posición inicial

Como vemos en la figura anterior, las articulaciones 2 y 3 del robot forman un ángulo de 90º, de esta forma conseguiremos arrancar desde una posición de reposo la ejecución del programa.

La anterior instrucción sólo se ejecutará al comienzo del programa. Cuando se haya completado el primer ciclo completo de movimientos, es decir, cuando el robot haya depositado una bola en la torre correspondiente tras recogerla del tablero, se dirigirá a la posición base de trabajo desde la cual comenzará a ejecutar las nuevas instrucciones de movimiento hacia los distintos puntos de la estación hasta que finalicemos el programa.

Para lograr lo anteriormente descrito, nuestro programa tendrá una etiqueta llamada "inicio" tras la instrucción de movimiento hacia la posición inicial del robot en el principio del main. A esta etiqueta se dirigirá el puntero del programa mediante un control de flujo cuando se haya ejecutado el ciclo completo del programa, llamando a la instrucción "GOTO inicio" que encontraremos al final del main.

```
inicio:
-
-
-
GOTO inicio;
```

Con la siguiente parte del código activaremos y desactivaremos al comienzo de la ejecución del programa los sensores superiores e inferiores de las torres de almacenamiento, dependiendo de si estas tienen almacenadas en su interior la máxima capacidad de bolas, alguna bola o ninguna.

```
IF bolasT1>0 THEN
    Set do10;
    IF bolasT1=3 THEN
        Set do11;
    ELSE
        Reset do11;
    ENDIF
ELSE
    Reset do10;
    Reset do13;
ENDIF
```

```
IF bolasT2>0 THEN
    Set do10;
    IF bolasT2=3 THEN
        Set do12;
    ELSE
        Reset do12;
    ENDIF
ELSE
    Reset do10;
    Reset do14;
ENDIF
```

El procedimiento será el mismo para ambas torres. Mediante un control de flujo con la intrucción IF, determinaremos si una torre tiene bolas en su interior, activándose la salida digital de los sensores inferiores de las torres (do10).

En caso de no tener ninguna bola, reseteríamos las salidas digitales do10 y do13/14, correspondientes a los sensores inferiores de las torres (do10) y a los actuadores neumáticos de cada una (do13 o do14) para que no puedan activarse las electroválvulas que los accionan hasta que no haya bolas en el interior de una torre. Si existe al menos una bola almacenada, comprobaremos mediante otra instrucción IF si la capacidad de dicha torre esta completa, es decir, si contiene tres bolas en su interior. En ese caso, se activará la salida digital que representa a los sensores superiores de cada torre (do11 o do12), indicándonos de forma visual en la simulación que las torres se encuentran a máxima capacidad. Si no se cumple la condición anterior, se desactivará la salida digital do11 o do12 indicando que exiten una o dos bolas en el interior de la torre de almacenaje.

A continuación, ejecutaremos la primera llamada a la función de generación de números aleatorios que activará una de las dos electroválvulas para el consiguiente accionamiento del actuador neumático correspondiente.

```
rand:=aleatorio();
RandomActuadores:=rand*2;
```

El valor generado por la función “aleatorio ()”, se almacenará en la variable “rand” declarada al inicio del programa. Para conseguir un valor entre 0 y 2, multiplicaremos el valor almacenado en dicha variable por 2 y lo guardaremos en la variable “RandomActuadores”. En el siguiente fragmento de código describiremos la activación y desactivación de las salidas digitales correspondientes a los actuadores, dependiendo del valor pseudoaleatorio almacenado en la variable “RandomActuadores”.

```
IF (RandomActuadores>=0 AND RandomActuadores<1) AND bolasT1>0 THEN
    Set do13;
    bolasT1:=bolasT1-1;
    Reset do11;
    WaitTime 1;
    Reset do13;
ELSEIF (RandomActuadores>=1 AND RandomActuadores<=2) AND bolasT2>0 THEN
    Set do14;
    bolasT2:=bolasT2-1;
    Reset do12;
    WaitTime 1;
    Reset do14;
ENDIF
```

Mediante un nuevo control de flujo IF, determinaremos el accionamiento de uno u otro actuador neumático.

Si el número almacenado en la variable "RandomActuadores" está comprendido en el intervalo entre 0 y 1, y se cumple la condición que indica que existe al menos una bola almacenada en la torre 1, se activará la salida digital do13 correspondiente al actuador localizado en dicha torre, el cual lanzará la bola hacia el tablero fijo. Tras ello, se actualizará la variable "bolasT1" indicando que ahora hay una bola menos en esta torre. Si ésta contenía tres bolas, desactivaremos la salida digital do11 correspondiente al sensor superior y resetearemos la salida do13 para que el actuador vuelva a su posición de reposo.

Por el contrario, si el número almacenado en la variable "RandomActuadores" está comprendido entre 1 y 2 y se cumple la condición que indica que existe al menos una bola en la torre 2, se ejecutará la activación y desactivación de las salidas digitales correspondientes al actuador que ésta contiene y al sensor superior en caso de que la torre contuviese tres bolas en su interior, de la misma forma que se ha descrito para la primera torre. También se actualizará el nuevo valor "bolasT2" para saber que en la torre 2 existe actualmente una bola menos.

Para describir las trayectorias que debe seguir el robot dependiendo del orificio en el que haya caído la bola tras ser lanzada desde una de las torres de almacenamiento, se ejecutará la siguiente parte del código.

```
IF (RandomActuadores>=0 AND RandomActuadores<1) OR (RandomActuadores>=1 AND RandomActuadores<=2) THEN
    rand:=aleatorio();
    RandomTablero:=rand*9;
    !Para obtener valor aleatorio entre 0 y 9
    RandomTorres:=rand*2;
    !Para obtener valor aleatorio entre 0 y 2
```

En primer lugar, comprobaremos si se ha accionado correctamente uno de los dos actuadores neumáticos mediante la instrucción IF que contendrá el resto del fragmento de código. Tras ello, volveremos a llamar a la función generadora de números pseudoaleatorios almacenando el nuevo valor en la variable "rand". Para indicar en qué orificio del tablero caerá la bola, activándose la salida digital correspondiente a cada sensor, guardaremos el valor de "rand" multiplicado por nueve en la variable "RandomTablero". De la misma forma, almacenaremos el valor de "rand" multiplicado por dos en la variable "RandomTorres" para indicar hacia qué torre debe dirigirse el robot para descargar la bola recogida previamente desde el orificio activo del tablero.

En el siguiente fragmento de código se explica paso por paso la ejecución que seguirá el programa para describir las trayectorias del IRB 120 en la simulación.

```
IF RandomTablero>=0 AND RandomTablero<1 THEN
    Set do1;
    Path_S1;
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
        IF bolasT1<3 THEN
            Path_T1;
        ELSEIF bolasT2<3 THEN
            Path_T2;
        ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
        IF bolasT2<3 THEN
            Path_T2;
        ELSEIF bolasT1<3 THEN
            Path_T1;
        ENDIF
    ENDIF
```

Si el número almacenado en la variable "RandomTablero" se encuentra en el intervalo entre 0 y 1, se activará la salida digital do1 que indica que la bola ha caído (de forma virtual) en la posición 1 del tablero fijo. El robot ejecutará la trayectoria "Path_S1" descrita en apartados anteriores hacia la posición del sensor 1 del tablero, recogerá la bola y esperará al siguiente paso. Si el número almacenado en la variable "RandomTorres" está comprendido entre 0 y 1 y la torre 1 no está a capacidad completa (tiene menos de tres bolas en su interior), el robot se dirigirá a dicha torre ejecutando el procedimiento "Path_T1", explicado en apartados anteriores, descargando la bola recogida desde la posición 1 del tablero en la zona de almacenamiento de la torre 1. En caso de que esta torre contenga tres bolas en su zona de carga, el robot seguirá la trayectoria descrita por el procedimiento "Path_T2", depositando la bola recogida en la torre 2 de almacenaje.

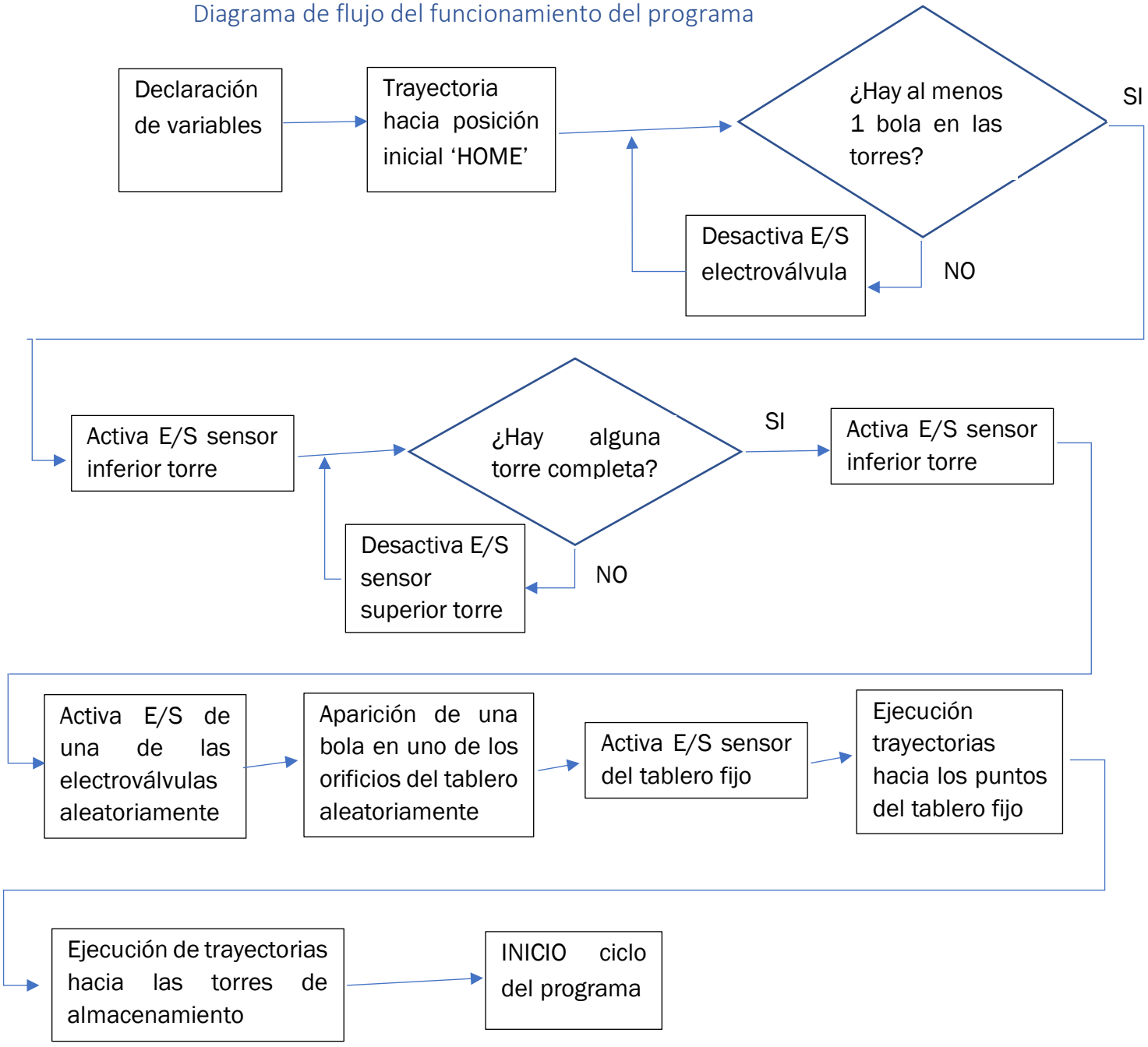
De la misma forma, si el número guardado en la variable “RandomTorres” está comprendido en el intervalo entre 1 y 2 y la torre 2 contiene menos de tres bolas, se ejecutará el procedimiento “Path_T2”. Si esta condición no se cumple, se descargará la bola en la torre 1 siguiendo las instrucciones del “Path_T1” si esta torre no se encuentra a su máxima capacidad.

Seguiremos el mismo procedimiento en todos los casos dependiendo de si el valor de la variable “RandomTablero” está comprendido en el intervalo entre 0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8 u 8-9. El siguiente fragmento de código muestra como ejemplo la parte en que el valor de “RandomTablero” está comprendido entre 6 y 7.

```
ELSEIF RandomTablero>=6 AND RandomTablero<7 THEN
    Set do7;
    Path_S7;
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
        IF bolasT1<3 THEN
            Path_T1;
        ELSEIF bolasT2<3 THEN
            Path_T2;
        ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
        IF bolasT2<3 THEN
            Path_T2;
        ELSEIF bolasT1<3 THEN
            Path_T1;
        ENDIF
    ENDIF
```

2.3. Estación de trabajo desarrollada para simular la práctica docente

Diagrama de flujo del funcionamiento del programa



Estación desarrollada en la simulación

A continuación, podemos ver la estación completa en la figura 45 creada en RobotStudio para realizar la simulación de la práctica docente, compuesta por el robot RIB 120, la botonera de señales digitales y la maqueta que fabricaremos formada por los tableros y torres de almacenamiento. También se muestran dos bolas virtuales (SmartComponents), una en la posición del sensor 8 del tablero fijo y otra en la parte superior de la torre 1 de almacenamiento, indicando que esta se encuentra a capacidad completa (contiene tres bolas).

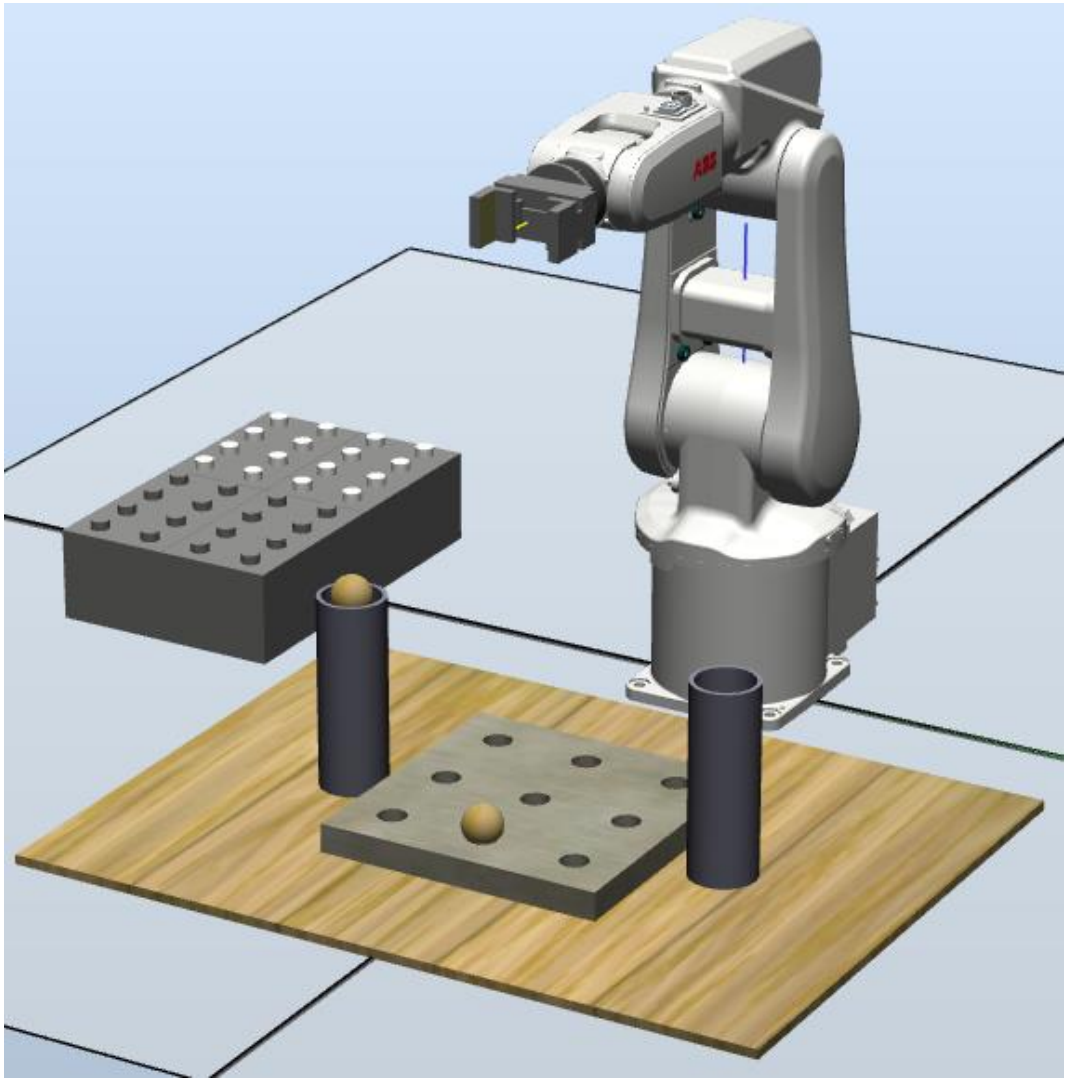


Figura 45. Estación simulada en RobotStudio

3. Diseño, montaje y validación de la práctica docente

3.1. Diseño y montaje de la práctica

Introducción y objetivos

El desarrollo de este proyecto se ha realizado utilizando el robot ABB IRB 120 (ver figura 46), que se encuentra disponible en el laboratorio de robótica de la Escuela de Ingenierías Industriales de la Universidad de Valladolid. Comenzamos este trabajo durante el periodo de prácticas en empresa en la recopilación de ideas e información para el diseño y desarrollo de un Setup de prácticas para robots manipuladores.

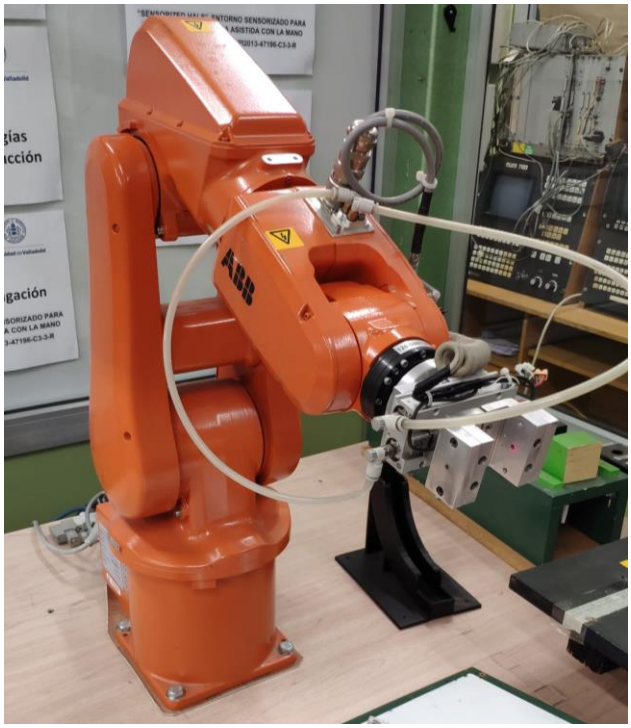


Figura 46. Robot IRB 120 de ABB disponible en la EII

En un principio se propusieron varias ideas que podríamos desarrollar de forma física para trabajar con el IRB 120 en el laboratorio de la escuela realizando prácticas académicas. La idea principal era fabricar un tablero con torres que incluyesen una parte neumática y otra parte electrónica y eléctrica, que funcionasen en conjunto con el robot IRB 120. Se crearon varios diseños del tablero, los cuales se muestran a continuación.

Descripción de los modelos previos

Primer diseño del prototipo

Inicialmente propusimos una mesa con siete orificios (ver figuras 47 y 48) en los que caerían las bolas procedentes de las torres de almacenamiento. En caso de no activarse ninguno de los sensores contenidos en los orificios del tablero en menos de ***X*** segundos, otro empujador neumático situado en la parte inferior del tablero fijo haría que éste se inclinase un pequeño ángulo para que la bola se introdujese en un orificio llamado "sumidero", situado en uno de los lados del tablero fijo. El robot recogería la bola de este sumidero y la colocaría sobre aquella torre que dispusiese de espacio libre para almacenar.

Los elementos que componen los diseños mostrados en las figuras 47 y 48 son los siguientes:

- Sensores

1.- Conectores de las conexiones del robot hacia la mesa
2.- Conectores de la parte neumática
3.- Conectores del cableado eléctrico
4.- Actuadores neumáticos
5.- Torres de almacenamiento
6.- Actuador neumático en la parte inferior del tablero fijo
7.- Tablero fijo
8.- Sumidero
9.- Orificios
10.- Tablero móvil

A continuación, se muestran los dos primeros diseños que se crearon.

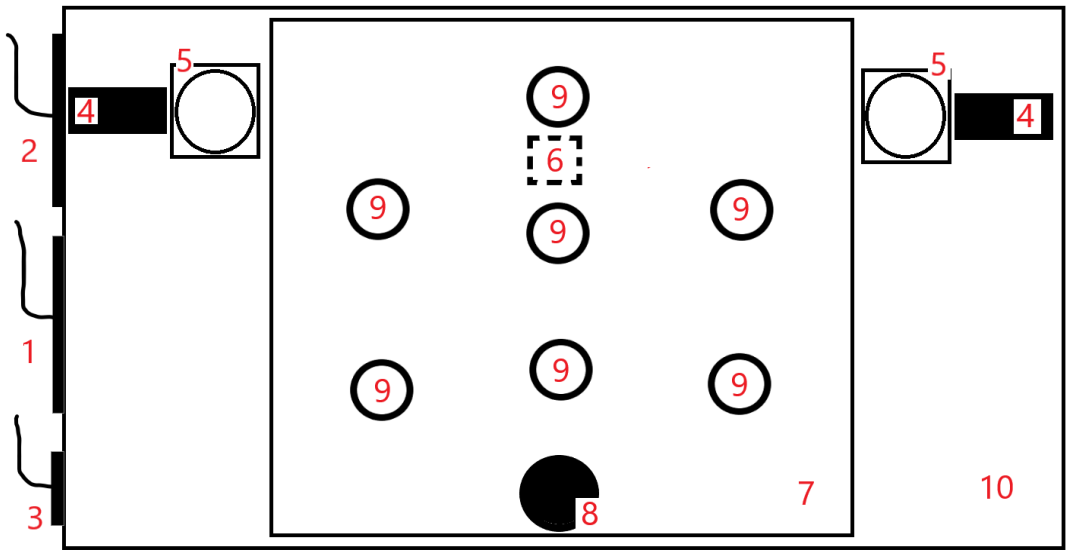


Figura 47. Diseño 1 de la maqueta de prácticas

En este primer diseño mostrado en la figura 47, el tablero fijo tiene forma cuadrada, con los orificios distribuidos por su centro en forma hexagonal. En uno de los laterales se encuentra el "sumidero", que hubiésemos utilizado para recoger las bolas que en un principio no cayesen de forma directa en uno de los otros orificios, ayudándonos del empujador para dirigir las bolas perdidas hacia él. Las torres de almacenamiento junto a los actuadores neumáticos se encontrarían en lados opuestos del tablero fijo, las cuales podríamos desplazar de forma lineal a lo largo del lateral del tablero para dar mayor aleatoriedad al juego al empujar las bolas consiguiendo distintas trayectorias.

Segundo diseño del prototipo

En el segundo diseño (ver figura 48) el tablero fijo tiene una forma más compleja, situando el "sumidero" en un lateral de éste formando una esquina para que, al activarse el empujador ubicado en el lado opuesto a dicho sumidero, las bolas cayesen en él en el caso de que no hubiesen caído en ningún orificio. Las torres de almacenamiento se encuentran en el mismo lado del tablero fijo, y los orificios distribuidos por el centro también en forma hexagonal.

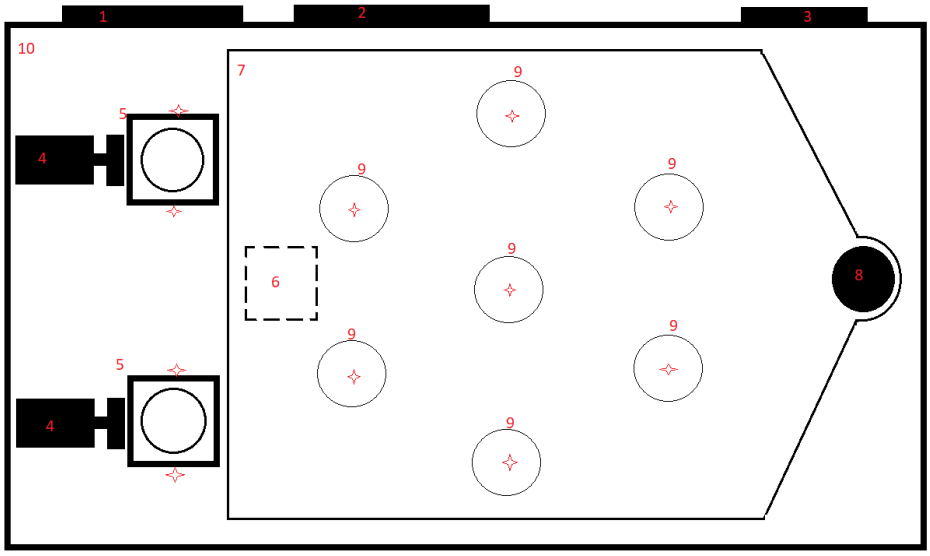


Figura 48. Diseño 2 de la maqueta de prácticas

Diseño final del prototipo

En el diseño final del prototipo mostrado en la figura 49, el tablero fijo se ha desarrollado en forma cuadrada de dimensiones 37x37 cm, con nueve orificios distribuidos por el centro formando una matriz de 3x3. Se ha prescindido del uso del "sumidero" y del empujador ubicado bajo el tablero, debido a la complejidad de manejar ambos componentes para que las bolas cayesen de forma segura. Las torres de almacenaje están ubicadas en lados opuestos del tablero fijo, en posiciones opuestas para maximizar la aleatoriedad del juego.

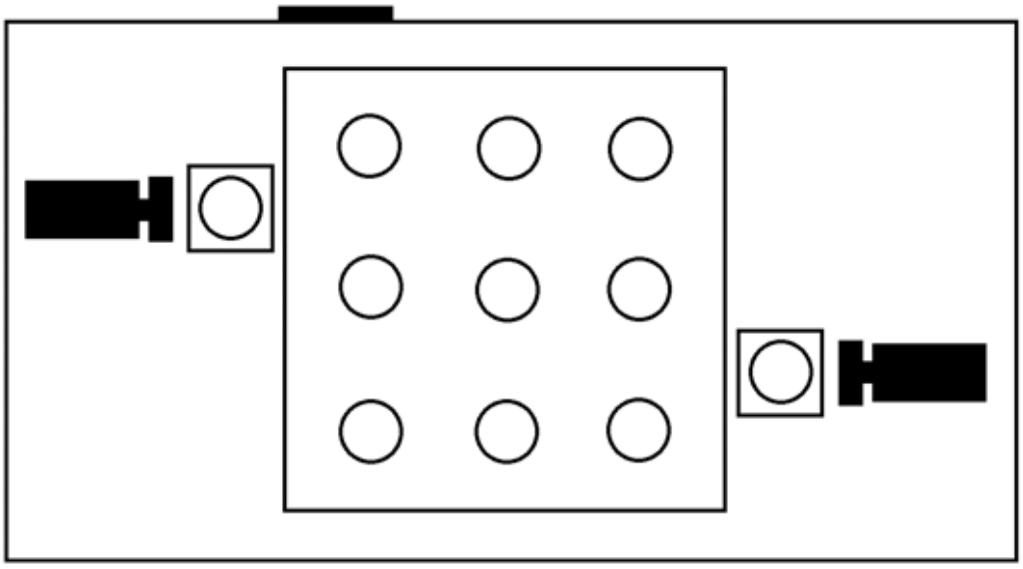


Figura 49. Diseño final del prototipo

Componentes necesarios para el desarrollo del prototipo

Tras obtener la idea principal del prototipo que queríamos desarrollar, necesitamos obtener los componentes que formarán la práctica para que funcione de forma correcta como teníamos pensado. A lo largo del periodo de prácticas en empresa, determinamos y adquirimos los elementos necesarios para la construcción de la maqueta, buscando los componentes electrónicos que planteamos en diferentes empresas de Valladolid y mediante compra online.

En estas tiendas especializadas en electrónica industrial hemos adquirido lo siguiente:

- Microinterruptor sensible con palanca de 55 mm, 16 A/250 VAC (ver figura 50), que irán colocados en las torres de almacenaje y en los distintos orificios del tablero fijo.
    - https://www.dieltron.com/rl6-oem-microrruptor-rleil-rl6-ultra-sensible-palanca-55mm-13588.html

Figura 50. Microinterruptor RLEIL RL6

- Dos cilindros neumáticos de simple efecto y vástago simple con sus respectivas electroválvulas de 3 vías de acción directa para su control (ver figura 51).
    - Actuador neumático: Product Data sheet (smc-static-resources-prd.s3.eu-central-1.amazonaws.com)
    - Electroválvula: Product Data sheet (smc-static-resources-prd.s3.eu-central-1.amazonaws.com)

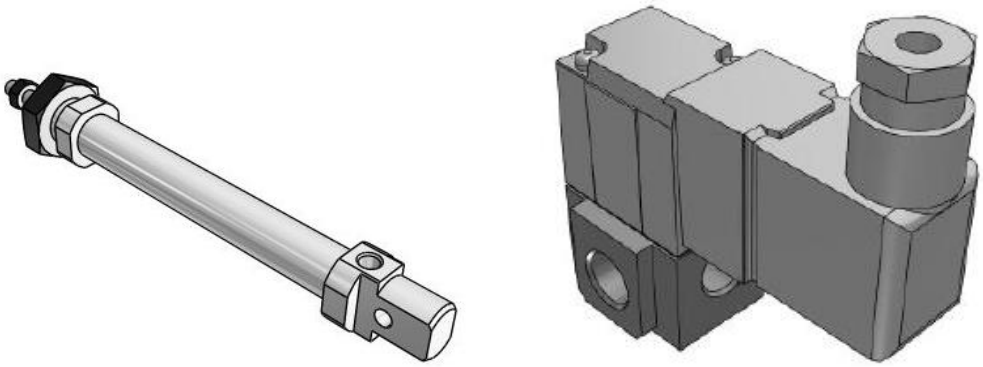


Figura 51. Actuador neumático y electroválvula

- Cableado eléctrico para las conexiones de los sensores y electroválvulas.

- Necesitaremos dividir la salida de aire comprimido disponible en el laboratorio mediante una serie de conectores instantáneos que podremos encontrar en el este enlace:
    - https://www.smc.eu/es-es/productos/conexiones-instantaneas~20947~nav#s&select_4063634[]=Air

  En la siguiente figura 52, se muestra la toma de aire comprimido disponible en la zona de trabajo del laboratorio de la EII, donde tendremos que colocar un conector en forma de T para dividir una salida de aire comprimido hacia el brazo robot y otra hacia nuestra estación de trabajo.

Figura 52. Toma de aire comprimido de la EII

- Para realizar las conexiones de toda la parte neumática utilizaremos tubos neumáticos de 8mm de diámetro externo, ya que este tamaño es el empleado en el laboratorio de la Escuela de Ingenierías Industriales. En el siguiente enlace se muestra un tubo de poliuretano de uso general del catálogo de SMC, capaz de aguantar una presión máxima de 0.8MPa a 20ºC, y que tiene el diámetro externo de 8mm que necesitamos.
    - https://www.smc.eu/es-es/productos/tu-tubo-de-poliuretano-tamano-sistema-metrico~49694~cfg#&select_4129182[]=Air/Inert%20gas

- También utilizaremos una llave de paso (ver figura 53) para abrir y cerrar el aire comprimido dirigido hacia nuestra maqueta (cuando queramos utilizar nuestra aplicación) desde la toma de aire disponible en el laboratorio, ya que el aire destinado hacia la parte neumática del IRB 120 siempre va a estar abierto para que éste se utilice en las distintas aplicaciones que se encuentran en la zona de trabajo del robot dentro de la EII.

Figura 53. Racord neumático

    - http://www.gav.it/producto/serie-uni/?lang=es (UNI, A2)

    - http://www.gav.it/producto/43a/?lang=es (43, A2)

- Materiales de soldadura
- Bolas macizas de madera de haya de 33 mm de diámetro (figura 54).
    - https://www.amazon.es/Gerileo-Lote-Bolas-futbol%C3%ADn-Madera/dp/B0848Q4VXN/ref=mp_s_a_1_4?dchild=1&keywords=bolas+futbolin&qid=1600795357&sr=8-4

Figura 54. Bolas de madera de haya

Las bolas que utilizaremos en el juego debían ser de madera o un material duro similar para no deformarse con los golpes, y cierto peso suficiente para que los sensores puedan detectar la presencia de dichas bolas en su contacto. Sus dimensiones debían ser de entre 3-4 cm de diámetro para que las pinzas del IRB 120 puedan cogerlas cómodamente, ya que la apertura máxima de éstas es de 4 cm.

- Kit de Arduino Uno y módulo de relés (ver figuras 55 y 56) para comprobar que los sensores y actuadores están correctamente conectados al conector eléctrico del tablero, antes de trabajar con el controlador del robot, para verificar que no haya fallos de conexiones mal cableadas. Necesitaremos el módulo de relés para convertir la tensión de 5V que maneja Arduino a los 24VDC que necesitamos para el control de las electroválvulas.

    - https://www.velleman.eu/downloads/29/vma501_a4v02.pdf
    - https://www.velleman.eu/products/view/?id=435594

Figura 55. Kit Arduino Uno VMA 501

- https://www.profetolocka.com.ar/2015/05/09/modulo-de-4-reles-para-arduino/

Figura 56. Módulo de relés

- Conectores macho-hembra de 25 pines, mostrados en la figura 57, que unirán las conexiones de la parte eléctrica que instalaremos en los marcos del tablero móvil para poder conectar y desconectar la estación al robot cuando queramos utilizarla. En la siguiente página web podemos ver información sobre este tipo de conectores, sus distintas formas y funcionalidades:

- https://www.monografias.com/trabajos17/conectores/conectores.shtml#conect

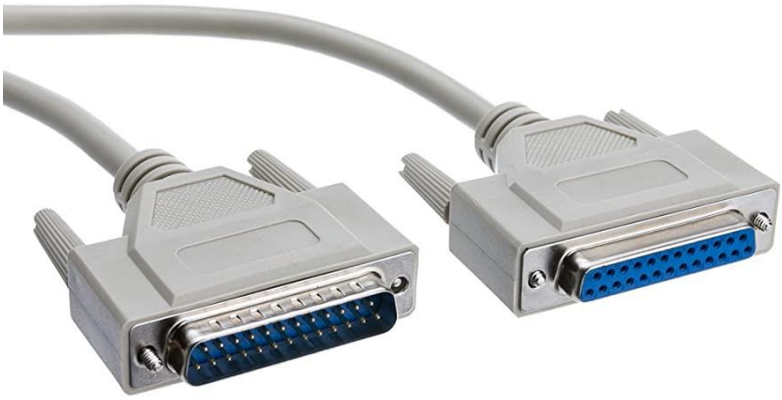


Figura 57. Conectores db25

En las pinzas del robot hemos colocado unas “almohadillas” (ver figura 58) con un material similar a una esponja para que al recoger las bolas de los orificios en los que se detecte su presencia mediante los sensores, el robot las recoja y agarre con facilidad, y evitar así el deslizamiento.

Figura 58. Almohadillas en la pinza del robot IRB 120

Hemos utilizado un aglomerado de madera para construir los tableros en los que desarrollaremos el juego, gracias a la ayuda de uno de los talleres de la Escuela de Ingenierías Industriales de la universidad que nos proporcionó

el material con las dimensiones específicas que planificamos. A continuación, se muestra el tablero fijo en la figura 59, de dimensiones 370x370x10mm con sus nueve orificios distribuidos en forma de matriz 3x3 a una distancia entre cada uno de ellos de 100mm.

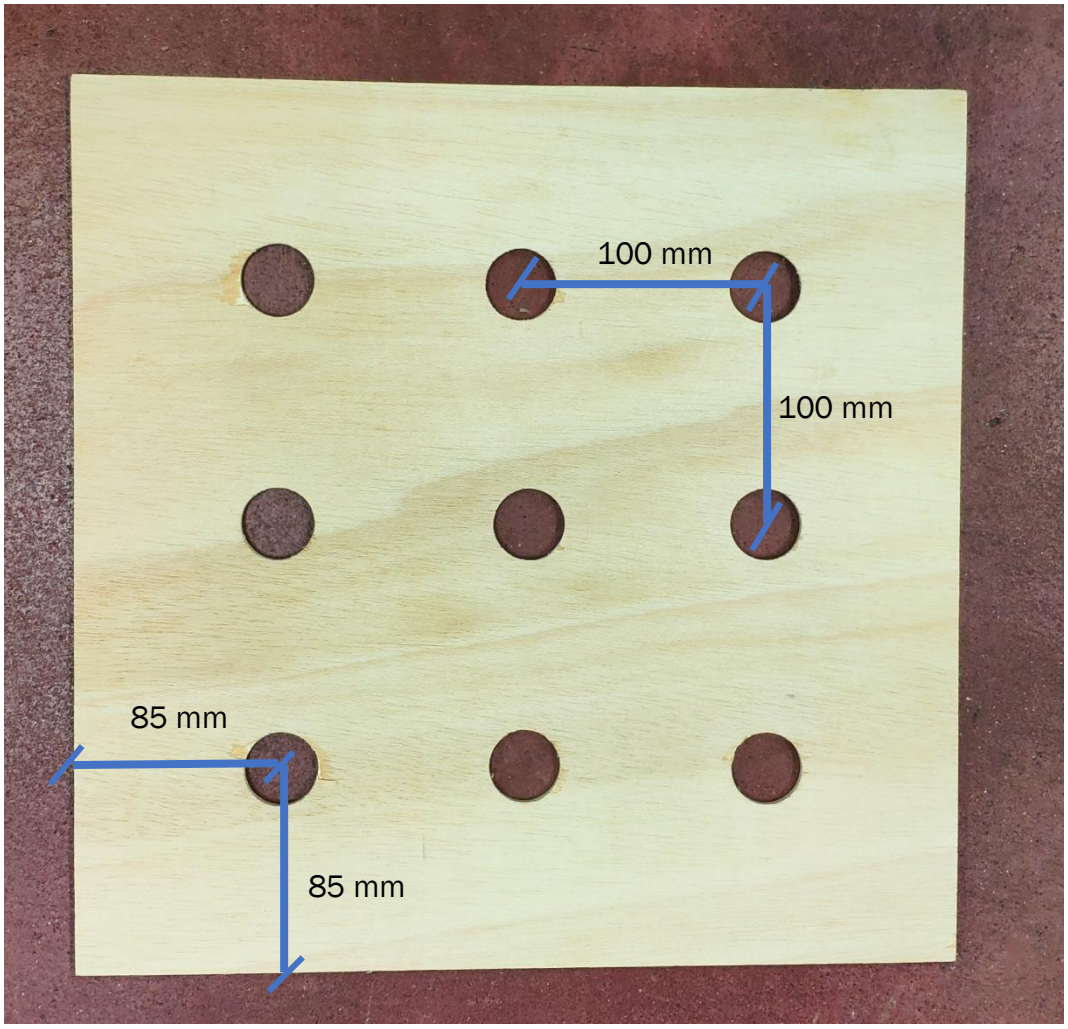


Figura 59. Ubicación de los orificios en el tablero fijo

La construcción de las torres de almacenamiento se ha realizado mediante impresión 3D gracias a la ayuda de uno de los miembros del ITAP – Instituto de las Tecnologías Avanzadas de la Producción.

Deseamos que cada una de las torres pueda almacenar un máximo de tres bolas. Se han diseñado dos soportes específicos, uno para los sensores que irán alojados en las torres, y otro para colocar el actuador neumático que empujará las bolas hacia el tablero móvil. En la parte superior del cilindro en el que se almacenará las bolas, tendremos un elemento que hará de embudo para que el posicionamiento del robot a la hora de colocar las bolas en su interior sea más sencillo.

En las siguientes figuras se muestran los prototipos de las piezas, realizados utilizando el programa de diseño mecánico Catia.

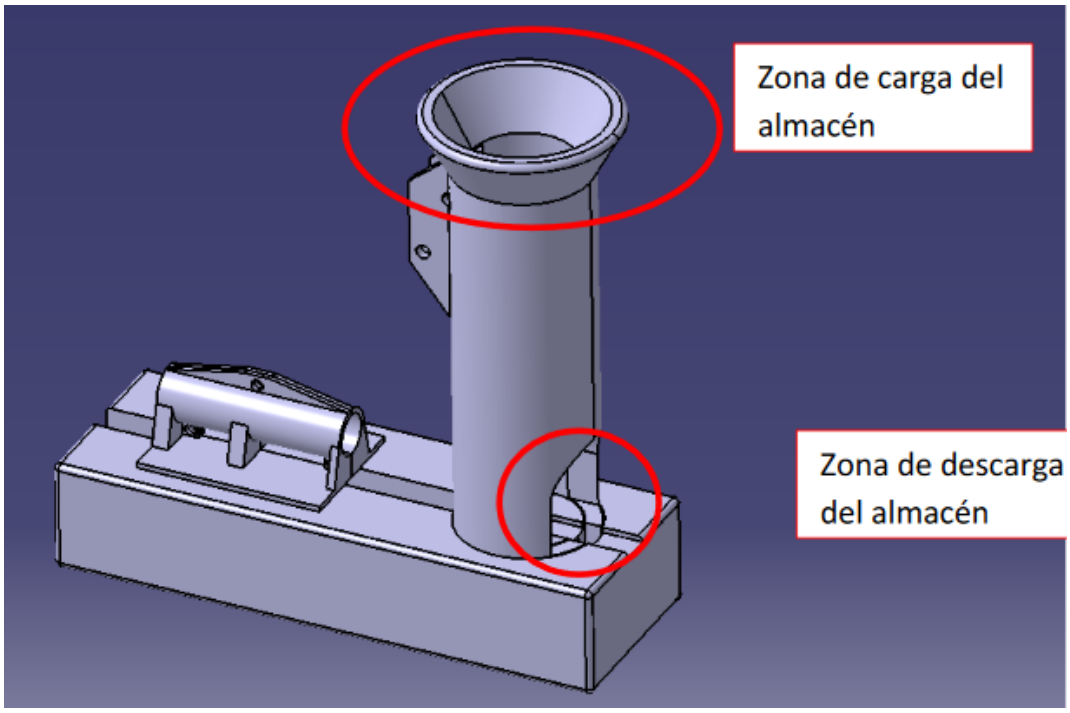


Figura 60. Almacén – Zona de carga y descarga de bolas

En la figura 60 anterior, la parte superior de la torre tiene una forma semicónica con un diámetro de circunferencia máxima de 75 mm para facilitar la descargar de bolas y asegurar que caigan dentro de las torres. En la parte inferior de la torre se encuentra la zona de descarga, de 40mm de anchura y 35mm de altura.

Las torres tienen una altura desde la base del soporte de 195 mm y un diámetro exterior de 50 mm para almacenar las bolas de madera. Las bases rectangulares que sujetan las torres verticales tienen una longitud de 200 mm, una anchura de 70 mm y una altura de 40 mm para ajustarse a los bordes del tablero fijo.

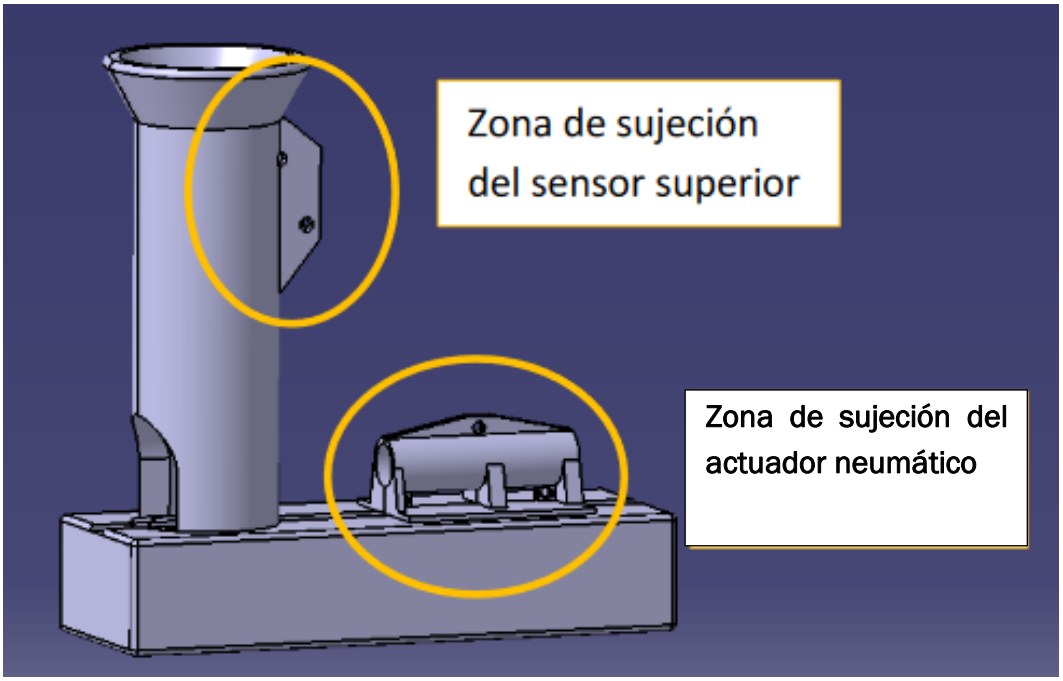


Figura 61. Almacén – Zona de sujeción del sensor superior y del actuador

La sujeción del actuador neumático que podemos ver en la figura 61 tiene una longitud de 70mm.

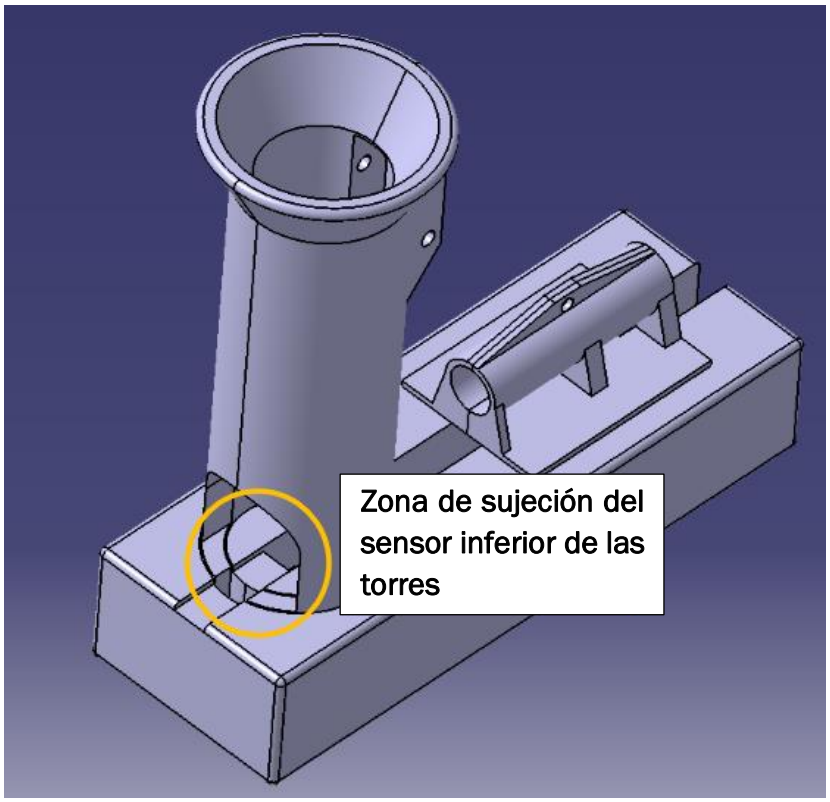


Figura 62. Almacén – Elemento de sujeción del sensor inferior

En los extremos de la parte inferior de las torres de almacenaje, se han diseñado unos pequeños orificios (ver figura 62) para fijarlas mediante tornillos al suelo del tablero móvil y conseguir así una mayor estabilidad, evitando que se descoloquen de su posición.

Comprobación de las conexiones mediante Arduino

En este apartado vamos a realizar la comprobación del correcto funcionamiento y conexión de los sensores y electroválvulas que componen nuestra estación de trabajo. Para ello, realizaremos un pequeño código en la plataforma Arduino apoyándonos en el pack de programación básica que adquirimos, que contiene componentes electrónicos que nos ayudarán a realizar las conexiones y comprobaciones que necesitamos antes de conectar la maqueta al robot IRB 120 disponible en el laboratorio de la escuela.

Conexiones eléctricas

Las conexiones eléctricas están formadas por todo el cableado que une los sensores que componen nuestro proyecto y las electroválvulas encargadas de controlar la parte neumática. Comenzaremos realizando la comprobación del correcto funcionamiento de los sensores programando un pequeño código de Arduino y conectando un sensor a la placa base de la que dispone nuestro kit de Arduino junto con un indicador LED que nos permitirá comprobar la activación y desactivación del sensor.

En primer lugar, conectamos el sensor a la placa mediante los cables, resistencias y la luz LED disponibles en el kit de Arduino de la forma que se muestra en las figuras 63 y 64.

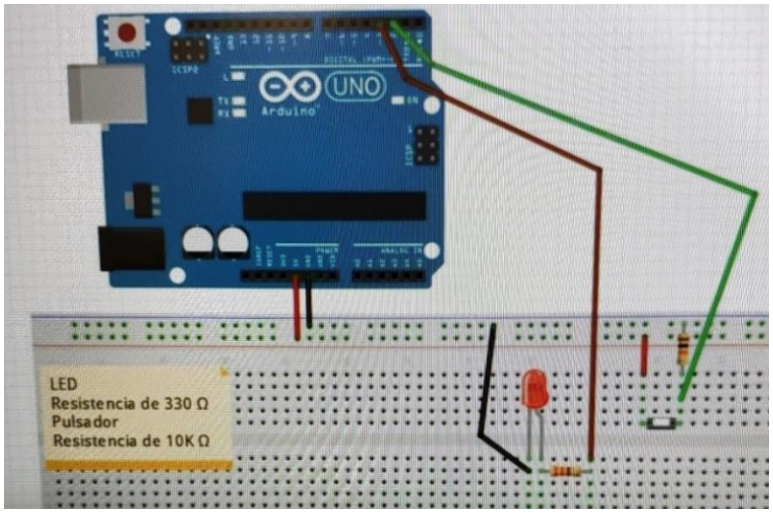


Figura 63. Conexión Arduino

En lugar del pulsador mostrado en la figura anterior, colocaremos el sensor tipo microinterruptor que queremos comprobar.

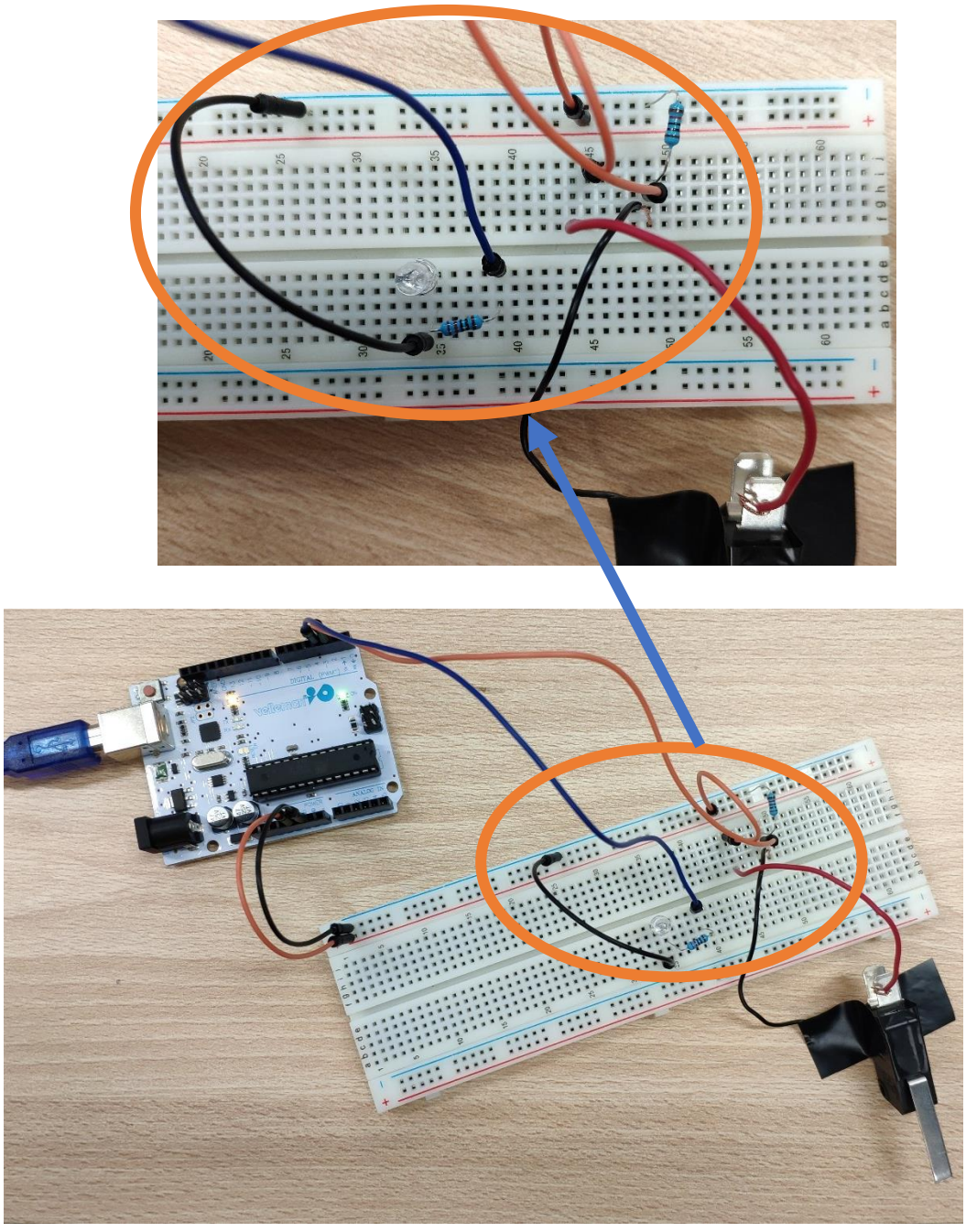


Figura 64. Conexión Arduino – Sensor

Tras ello, realizaremos el programa de Arduino mostrado en la figura 65 para comprobar la activación del sensor.

Comprobacion_SENSOR Arduino 1.8.15 (Windows Store 1.8.49.0)

Archivo Editar Programa Herramientas Ayuda

Comprobacion_SENSOR §

```
int SENSOR = 2;
int LED = 3;
int ESTADO = LOW; //LED inicialmente apagado

void setup(){
  pinMode(SENSOR, INPUT); // pin 2 como entrada
  pinMode(LED, OUTPUT);   // pin 3 como salida
}

void loop(){
  while(digitalRead(SENSOR)== ESTADO){
    // mientras el sensor este a nivel bajo, no hace nada
  }
  ESTADO = digitalRead(LED); // guardamos en la variable ESTADO el valor del LED
  digitalWrite(LED, !ESTADO); // cambiamos el valor del LED
}
```

Figura 65. Código del programa Arduino – comprobación sensores

Comenzaremos declarando las variables necesarias y las asignamos a los pines que ocuparán en la placa de Arduino una vez realizadas las conexiones. El sensor ocupará el pin 2 como señal de entrada y la luz LED el pin 3 como señal salida que nos indicará la activación/desactivación del dicho sensor cuando pulsemos su interruptor.

Para comprobar la correcta activación de las electroválvulas que controlarán los actuadores neumáticos, necesitaremos un generador de tensión que nos proporcione 24V (ver figura 66), que es el máximo voltaje que soportan las electroválvulas, un módulo de relés (ver figura 67) para transformar la tensión de Arduino de 5V a los 24VDC que necesitan las electroválvulas y un nuevo programa para comprobar el funcionamiento junto a las conexiones con el kit de Arduino.

Figura 66. Generador de tensión 24VDC

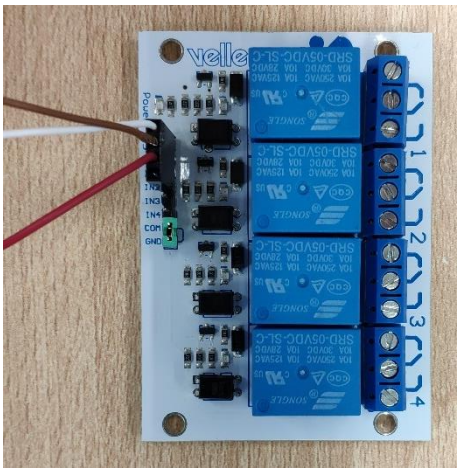


Figura 67. Módulo de 4 relés

La figura 68 muestra la conexión completa para la comprobación del funcionamiento de una electroválvula con Arduino.

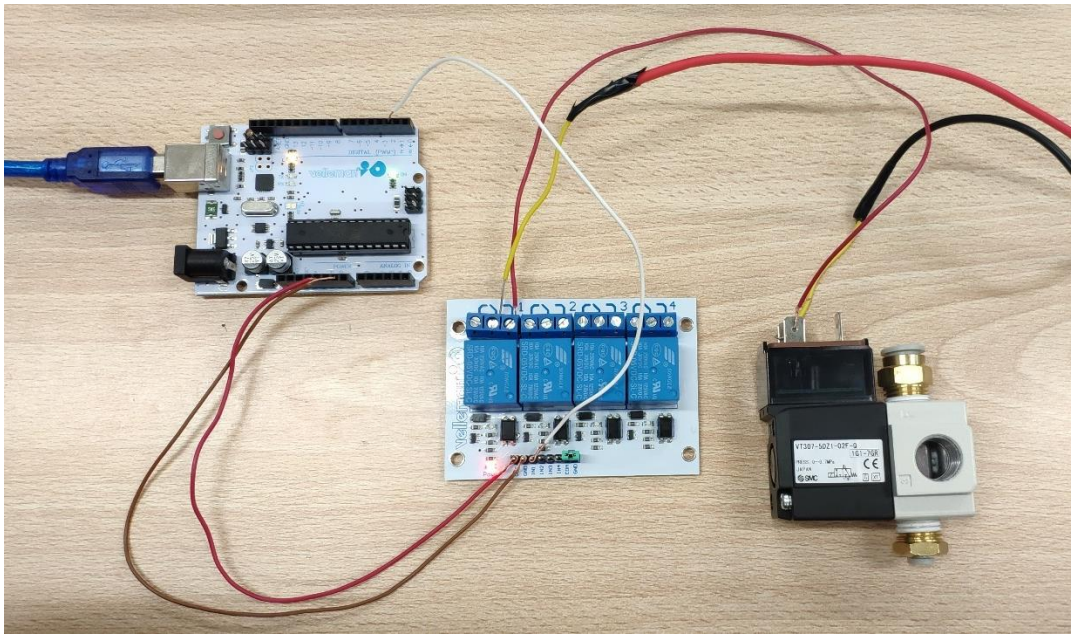


Figura 68. Conexión Arduino – Electroválvula

En la figura 69 se muestra el pequeño programa de Arduino que hemos utilizado para realizar la comprobación del funcionamiento de las electroválvulas.

Figura 69. Código programa Arduino – comprobación electroválvulas

En este caso, declaramos la variable que representa al módulo de relés ocupando el pin 2 de salida de la placa de Arduino. La electroválvula se activará durante 5 segundos cuando se enciende el módulo de relés y existe una tensión aproximada de 24VDC suficiente para generar la tensión que necesita el solenoide interno de la electroválvula para su activación, y permanecerá apagada durante otros 5 segundos cuando se desactiven los relés.

Conexiones eléctricas de la práctica desarrollada con el robot IRB 120

En las figuras 70 y 71, mostraremos como se ha guiado el cableado eléctrico y neumático de los sensores y electroválvulas de la práctica desarrollada.

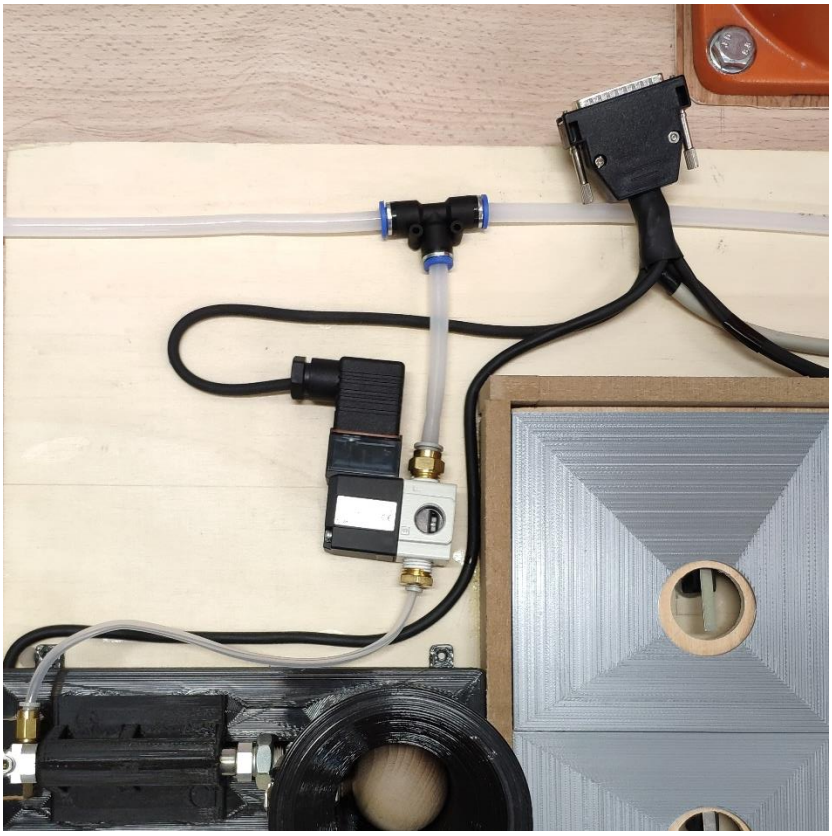


Figura 70. Cableado eléctrico y neumático en la práctica real (parte izquierda)

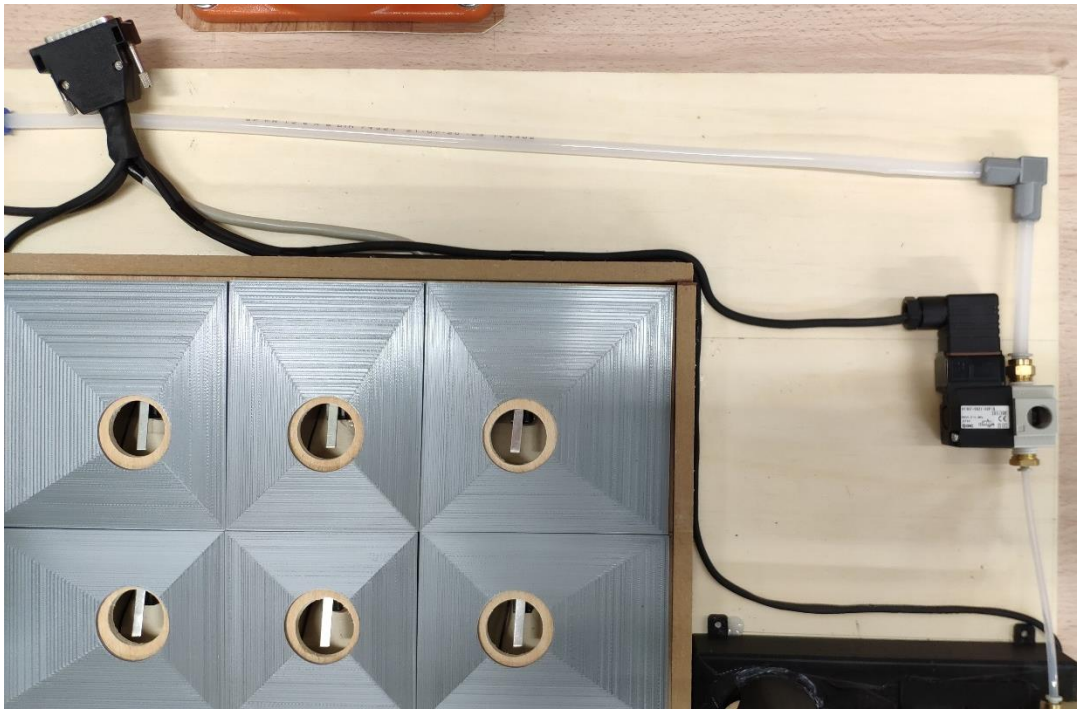


Figura 71. Cableado eléctrico y neumático en la práctica real (parte derecha)

El cableado eléctrico une los sensores del tablero y de las torres hacia el conector db25 macho de la forma más ordenada posible para optimizar el diseño del prototipo.

En la siguiente tabla mostramos las conexiones de las señales digitales de la práctica desarrollada con las E/S digitales del robot IRB 120.

Tabla de conexiones de E/S digitales de la botonera a la estación real

Name	Type of Signal	Assigned to Unit	Signal Identification Label
di1	Digital Input	board10	Sensor 1 tablero fijo
di2	Digital Input	board10	Sensor 2 tablero fijo
di3	Digital Input	board10	Sensor 3 tablero fijo
di4	Digital Input	board10	Sensor 4 tablero fijo
di5	Digital Input	board10	Sensor 5 tablero fijo
di6	Digital Input	board10	Sensor 6 tablero fijo
di7	Digital Input	board10	Sensor 7 tablero fijo
di8	Digital Input	board10	Sensor 8 tablero fijo
di10	Digital Input	board10	Sensor 9 tablero fijo
di11	Digital Input	board10	Sensor inferior torre 1
di12	Digital Input	board10	Sensor superior torre 1
di13	Digital Input	board10	Sensor inferior torre 2
di14	Digital Input	board10	Sensor superior torre 2
do1	Digital Output	board10	Control electroválvula torre 1
do2	Digital Output	board10	Control electroválvula torre 2
do15	Digital Output	board10	Abrir pinza neumática
do16	Digital Output	board10	Cerrar pinza neumática

Nuevamente, hemos guiado los cables de forma ordenada y no surjan problemas de conexión. Soldaremos todo el cableado eléctrico de los sensores que provienen de las torres y del tablero a los pines del conector db25 macho, siguiendo el diseño mostrado en la figura 72 a continuación, donde también incluiremos la parte eléctrica de las electroválvulas.

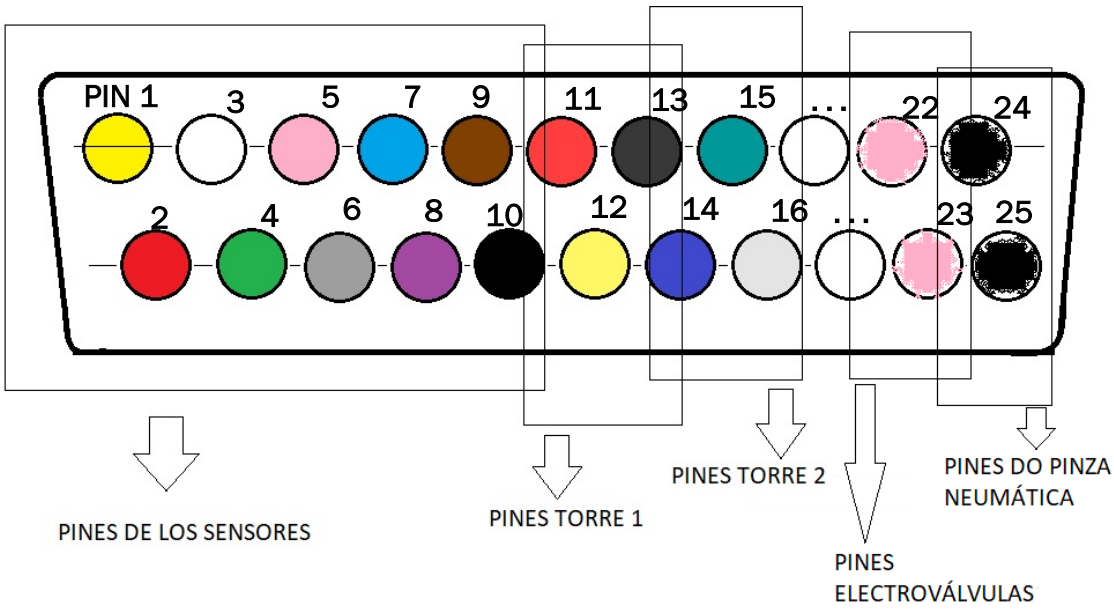


Figura 72. Esquema pines conector db25

Tabla guía de las conexiones eléctricas en los pines del conector db25

PIN	Descripción	Entrada digital	Salida digital
1	Sensor 1 tablero	di1	-
2	Sensor 2 tablero	di2	-
3	Sensor 3 tablero	di3	-
4	Sensor 4 tablero	di4	-
5	Sensor 5 tablero	di5	-
6	Sensor 6 tablero	di6	-
7	Sensor 7 tablero	di7	-
8	Sensor 8 tablero	di8	-

9	Sensor 9 tablero	di10	-
10	Neutro sensores tablero	Neutro	-
11	Sensor inferior torre 1	di12	-
12	Sensor superior torre 1	di13	-
13	Neutro sensores torre 1	Neutro	-
14	Sensor inferior torre 2	di14	-
15	Sensor superior torre 2	di16	-
16	Neutro sensores torre 2	Neutro	-
17	-	-	-
18	-	-	-
19	-	-	-
20	Electroválvula 1	-	do1
21	Electroválvula 1	-	24VDC
22	Electroválvula 2	-	do2
23	Electroválvula 2	-	24VDC
24	DO pinza neumática	-	do15
25	DO pinza neumática	-	do16

El conector representado en la figura 72 enviará los datos al IRB 120 a través de la controladora del robot, para que éste detecte la activación o desactivación de las E/S digitales que correspondan a cada pin mediante la botonera de E/S, disponible en el laboratorio, a la que irá conectada la otra toma del conector db25.

A continuación, se muestra en la figura 73 un esquema con las conexiones realizadas entre la estación, el conector del cableado eléctrico db25 y los pines de E/S digitales de la botonera del robot IRB 120, tomando como ejemplo el sensor de la posición 1 del tablero fijo. El resto de los sensores,

que irán conectados de la misma forma, pero en los distintos pines del conector db25 macho y en las E/S de la botonera, tal y como se han especificado en las tablas anteriores.

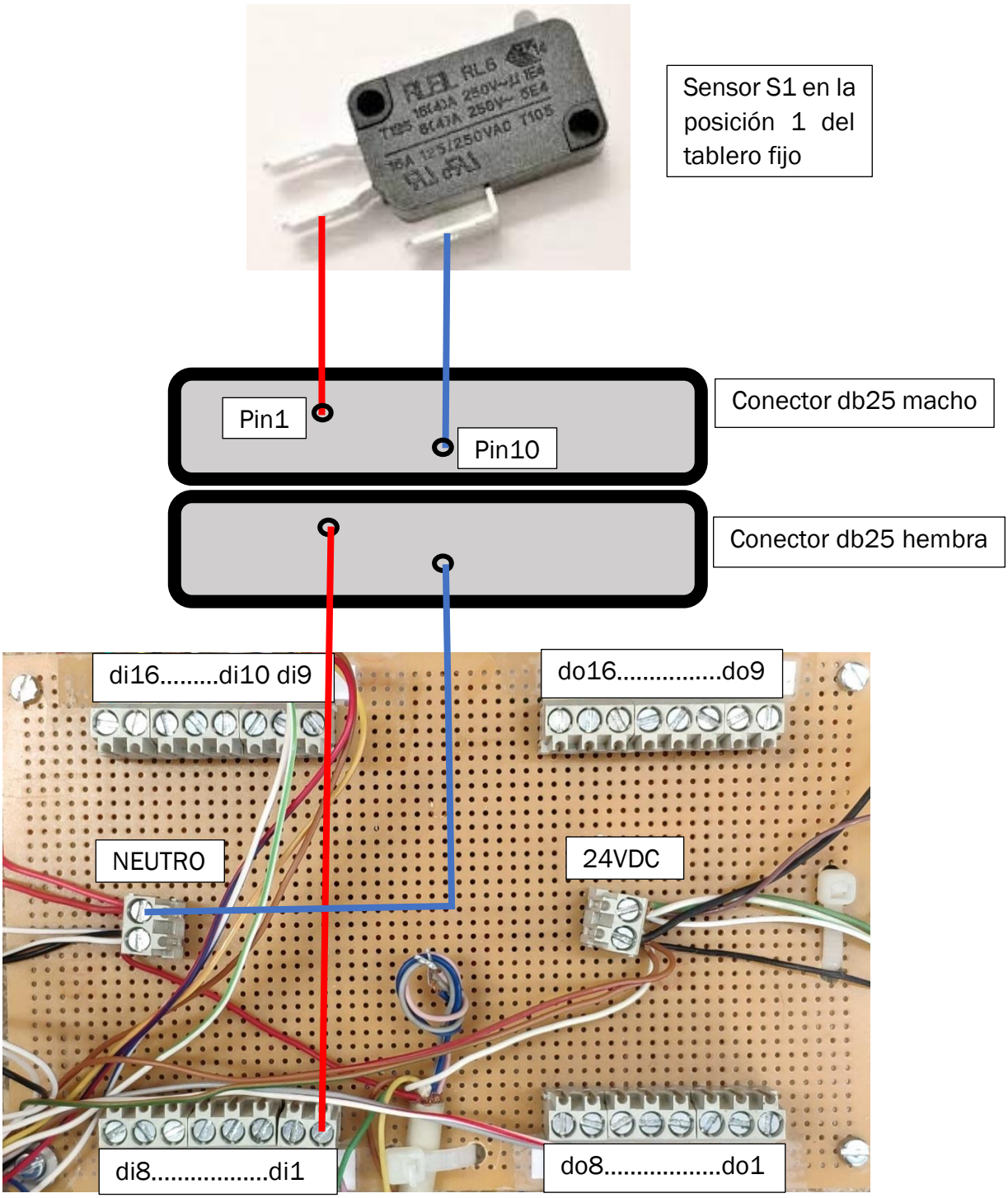


Figura 73. Esquema conexión E digitales sensores – conector db25 – botonera E/S del robot IRB 120

Para la conexión de las electroválvulas al conector eléctrico y la botonera de E/S del robot (ver figura 74), el procedimiento será igual que el visto en el anterior esquema, pero conectando dos de los pines de las salidas digitales de la botonera en uno de los pines de la electroválvula, en lugar de las entradas digitales asignadas a los sensores y, además, la conexión del otro pin de la electroválvula al pin de 24V de la botonera para que ésta se active al llegarla la señal correspondiente y dicha tensión.

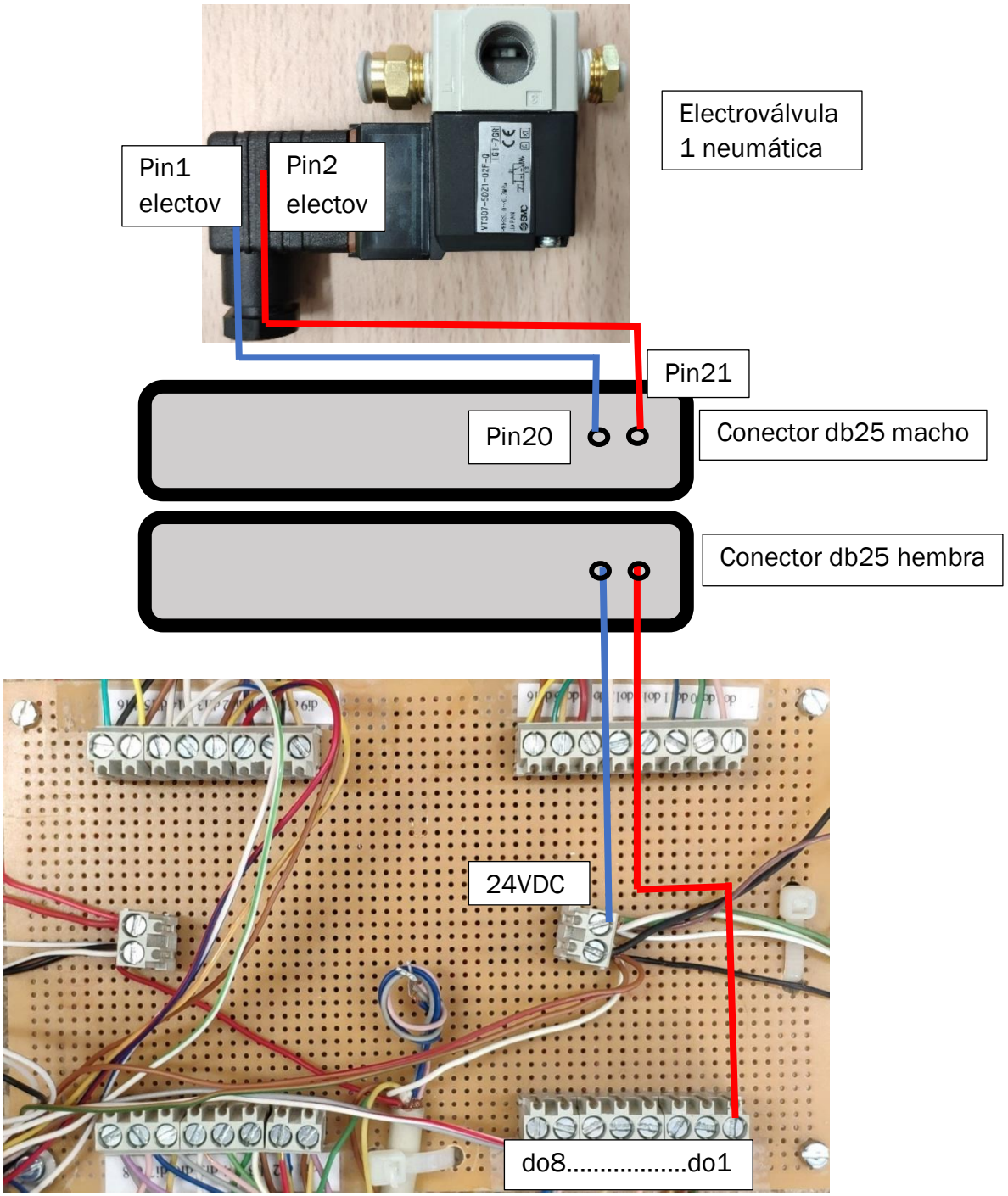


Figura 74. Esquema conexión S digitales electroválvulas – conector db25 – botonera E/S del robot IRB 120

Conexiones neumáticas de la práctica desarrollada con el robot IRB 120

Las conexiones neumáticas estarán formadas por las electroválvulas y los actuadores junto con el cableado neumático que conectará estos componentes. Este cableado especial para trabajar con aire comprimido irá conectado a los pines neumáticos habilitados en la electroválvula. En la siguiente figura 75 se muestran estos pines neumáticos, de dos diámetros distintos: un tubo de 8 mm y otro de 4 mm de diámetro.

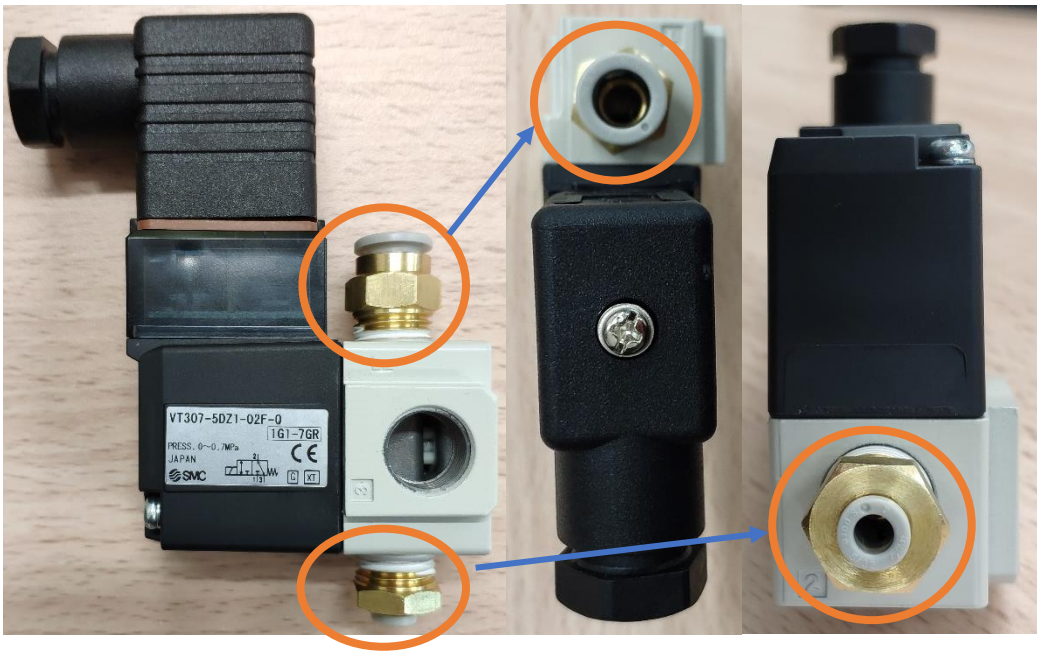


Figura 75. Pines neumáticos de la electroválvula

Como podía verse en las figuras 70 y 71 anteriormente descritas en la parte eléctrica, guiaremos el cableado neumático mediante racores mostrados en la figura 76.

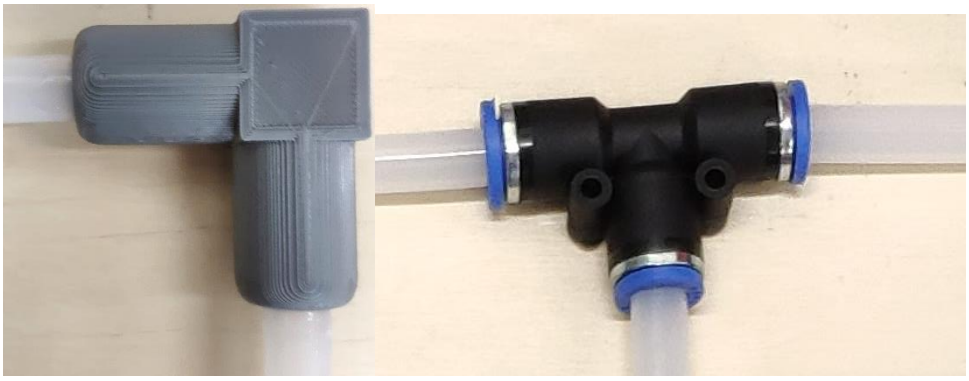


Figura 76. Racores neumáticos

Tras conectar todo el cableado a sus componentes correctamente, comprobaremos el funcionamiento de la parte neumática abriendo la llave de paso de aire comprimido (ver figura 77). Configuraremos el regulador de aire del laboratorio para alcanzar una presión de salida algo inferior a 0'1 MPa, suficiente para el correcto funcionamiento de la apertura y cierre de la pinza neumática, y para que los actuadores neumáticos lancen las bolas con la fuerza suficiente y necesaria para que sean expulsadas de las torres de almacenaje y no se salgan del tablero fijo.

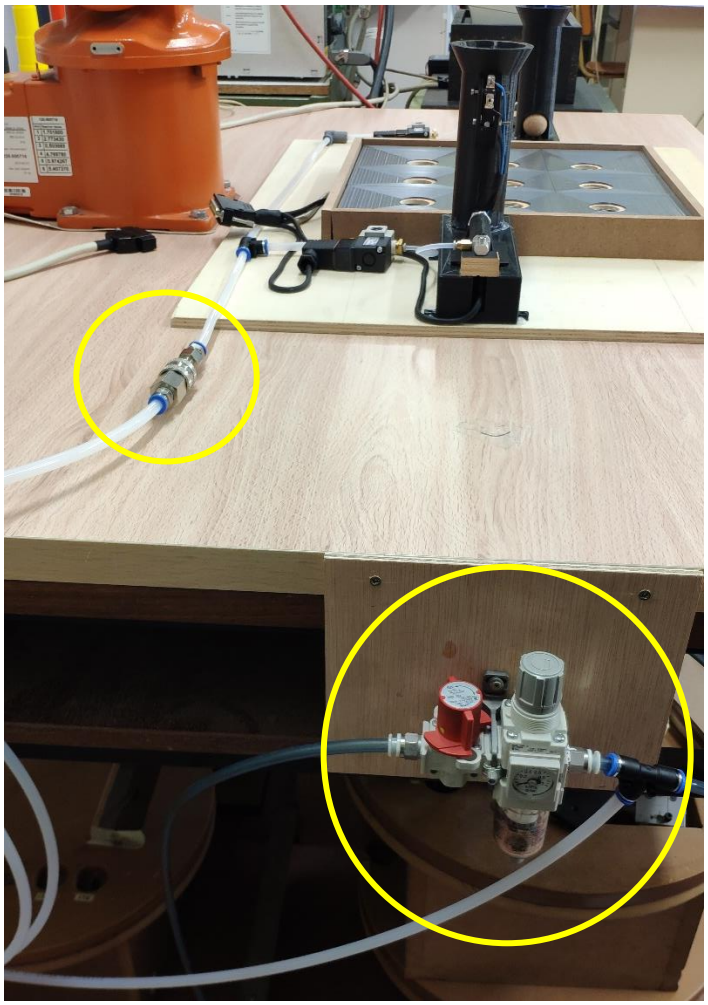


Figura 77. Conexión de la parte neumática con el regulador de aire

Montaje mecánico de la estación

Una vez reunidos todos los materiales necesarios para el desarrollo del prototipo de prácticas, comenzamos a construir la maqueta con la que trabajaremos junto con el IRB 120.

Montaje de los tableros fijo y móvil

El taller de diseño de la Escuela de Ingenierías Industriales nos proporcionó los tableros de madera para realizar la base de nuestro proyecto.

Las dimensiones de los elementos que componen la estación son las siguientes:

- Tablero móvil: 800 x 600 x 10 mm
- Tablero fijo: 370 x 370 x 10 mm
- Orificios tablero fijo: Nueve orificios de 30mm de diámetro en forma de matriz 3x3, con una distancia entre ellos hasta sus centros equivalente de 100 mm y con un margen hasta el borde del tablero de 85mm (ver figura 56).
- Laterales del tablero fijo: Cuatro rectángulos de 370x50x10mm
- Soportes del tablero fijo: Cuatro rectángulos de 23x40x10mm

Los soportes del tablero fijo sostendrán éste a una altura de 23 mm para que cuando caiga una bola en uno de los orificios, haga contacto con el sensor perfectamente y la señal de entrada llegue al robot sin ningún problema.

Por último, para dar mayor aleatoriedad al juego y asegurar que las bolas lanzadas desde las torres caigan en alguno de los nueve orificios disponibles en el tablero, se han fabricado unas placas mediante impresión 3D, centradas en cada uno de los orificios que encajarán en el tablero fijo formando una especie de puzle de nueve piezas (ver figura 78). Estos elementos tendrán forma cuadrada o rectangular, una altura máxima de 5 mm en su extremo, que irá disminuyendo hacia el centro de cada placa, donde se encontrará un orificio de 40 mm de diámetro, suficiente para que las bolas caigan y la pinza neumática del robot pueda recogerlas sin ninguna dificultad.

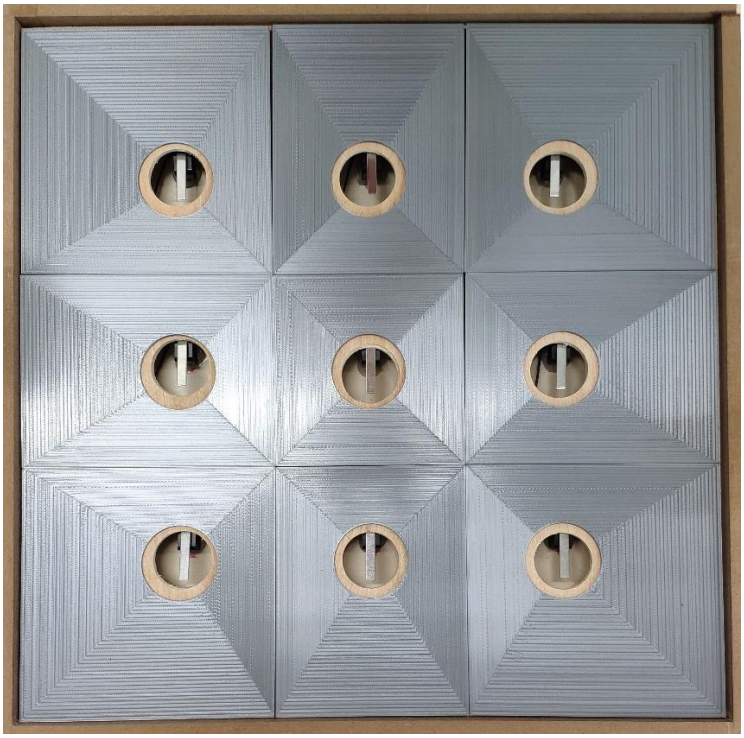


Figura 78. Placas de impresión 3D montadas sobre el tablero fijo

Montaje y conexión de los sensores en la base del tablero

En primer lugar, necesitamos cortar las patillas metálicas de los sensores, ajustándolas para que encajen perfectamente en los orificios del tablero fijo. Tras ello, comenzamos a colocar los sensores en la base del tablero móvil, comprobando que se ajustan perfectamente a las posiciones de los orificios del tablero fijo que irá colocado justamente encima de ellos.

Una vez aseguradas la ubicación de cada sensor, conectamos todos los terminales neutros (ver figura 79) de éstos entre sí, colocando los cables de forma limpia para que no estorben, soldándolos a cada terminal para que en el conector eléctrico situado en el borde del tablero móvil tengamos únicamente un cable que determine el neutro de los nueve sensores del tablero como se muestra en la figura 80.

Figura 79. Terminal neutro del sensor

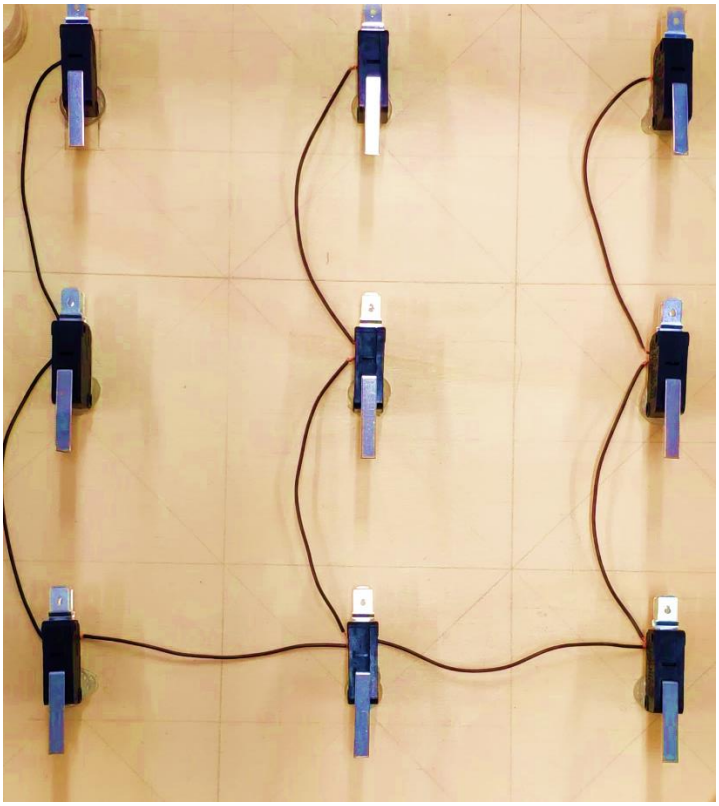


Figura 80. Conexión de los terminales neutros de los sensores del tablero

Para fijar los sensores al tablero móvil utilizamos una pistola de silicona, asegurándonos de que todos se sostengan perfectamente, soportando el peso de las bolas de madera sin balancearse.

A continuación, soldamos los terminales NA (Normalmente Abierto) de cada sensor con cables de distintos colores para diferenciar cada uno de los finales de carrera, guiando dicho cableado por el tablero de forma ordenada para unirlos todos en un cable multifilar que reúna

el conjunto de la forma más limpia posible, como se muestra en la figura 81.

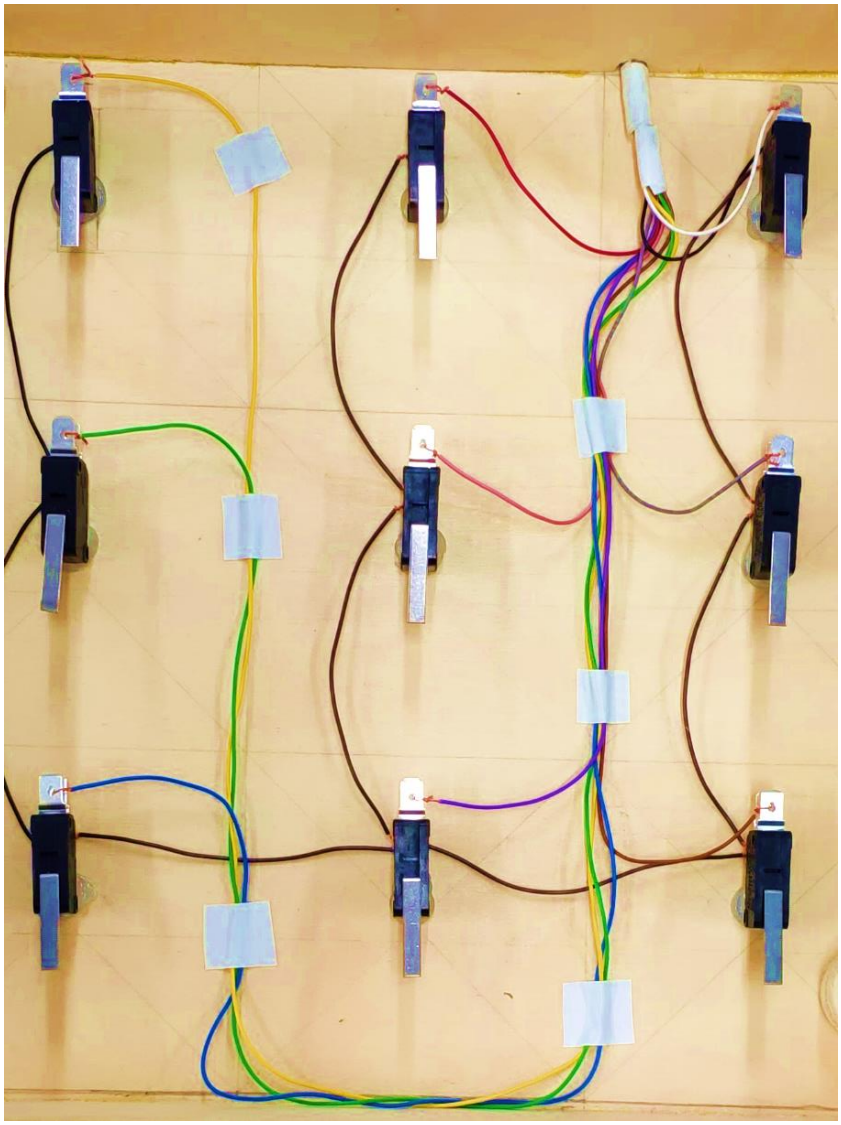


Figura 81. Conexión eléctrica de los sensores del tablero

Necesitamos conectar los terminales NA de los sensores para que al activarse uno de ellos, la tensión de 24 VDC que genera pueda ser detectada por la controladora del IRB 120 sin ocasionar problemas, ya que si conectásemos los terminales NC (Normalmente Cerrado), llegarían continuamente 24 VDC al controlador IRC5 del robot, y este se saturaría dando fallo en el sistema y concluyendo el programa.

Por último, soldaremos todo el cableado a los pines del conector macho del cable multifilar. Este conector es de tipo db25.

Montaje y conexión de los sensores en las torres de almacenamiento

Para posicionar los sensores inferiores y superiores de las torres en los compartimentos especialmente fabricados para su sujeción, comenzaremos soldando el cableado eléctrico en los terminales de los sensores y guiando dicho cableado de forma que quede lo más ordenado y limpio posible dentro de nuestra maqueta. Protegeremos el cableado que proviene de cada torre con un tubo termoretráctil del diámetro adecuado, que someteremos a una fuente de calor para que este material se contraiga y proporcione una sujeción adecuada de los cables. Dirigiremos el cableado de cada torre hacia el conector eléctrico db25 de nuestra maqueta y soldaremos cada hilo al pin correspondiente de dicho conector.

Fijaremos los sensores inferiores en las torres con un pequeño trozo de cinta, y los superiores con unos tornillos pasantes con tuercas alojados en los orificios disponibles que diseñamos a la hora de fabricar las torres para conseguir la máxima sujeción posible. En la siguiente figura 82 se muestra el montaje final de los sensores en las torres de almacenamiento.

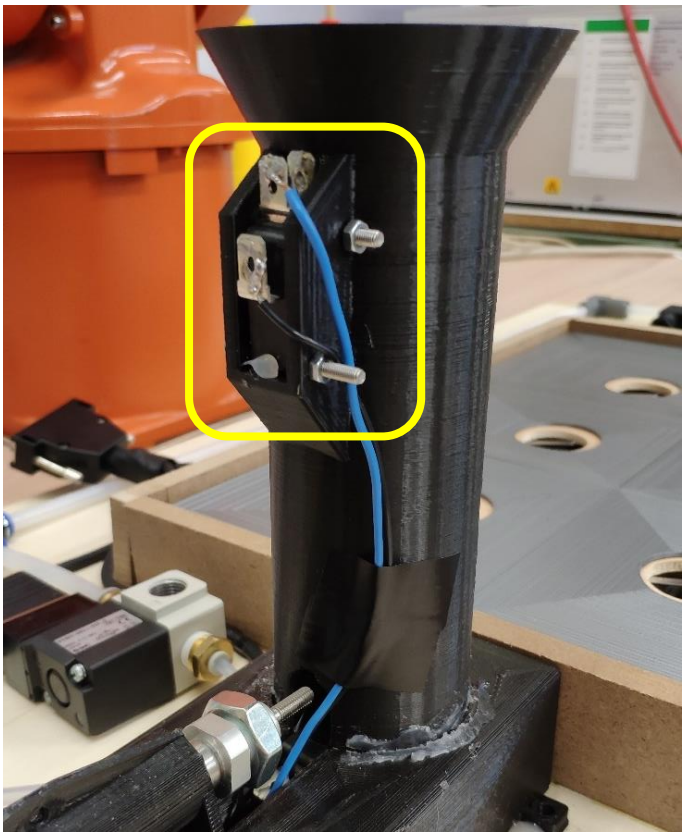


Figura 82. Fijación de los sensores en las torres de almacenamiento

Montaje y conexión de la parte neumática

En primer lugar, fijaremos los actuadores neumáticos a sus respectivos soportes fabricados previamente con impresión 3D, y los ubicaremos en las bases de las torres de almacenaje. Nos ayudaremos de un pequeño bloque de madera para fijar los actuadores con ayuda de tornillos, tuercas y pegamento de fijación rápida como se muestra en la figura 83.

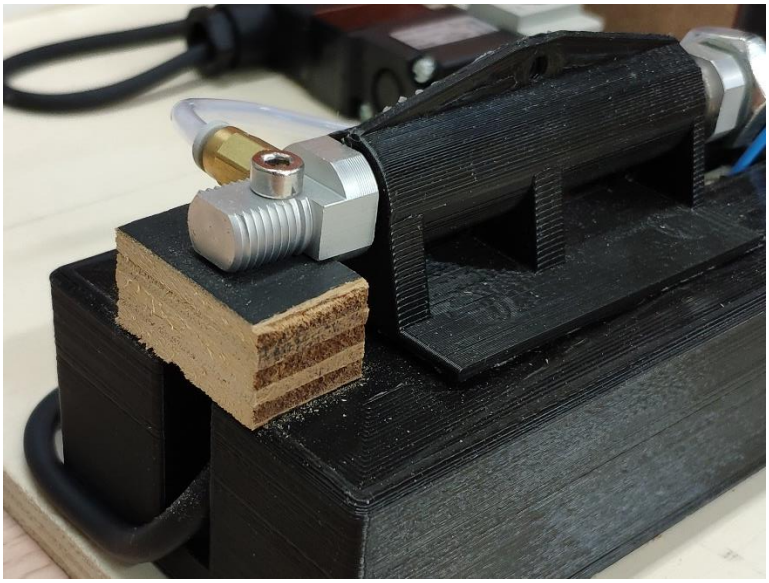


Figura 83. Fijación de los actuadores neumáticos a la base de las torres

Conectaremos el cableado neumático a los racores de los actuadores y las electroválvulas, previamente fijadas en el tablero móvil, para formar la estructura de forma ordenada y lo más parecido posible al diseño del esquema eléctrico y neumático creado previamente.

A continuación, se muestra la conexión y fijación de la parte neumática de nuestra estación en las figuras 84 y 85, en las que vemos los actuadores fijados en las torres y las electroválvulas de ambos lados del tablero.

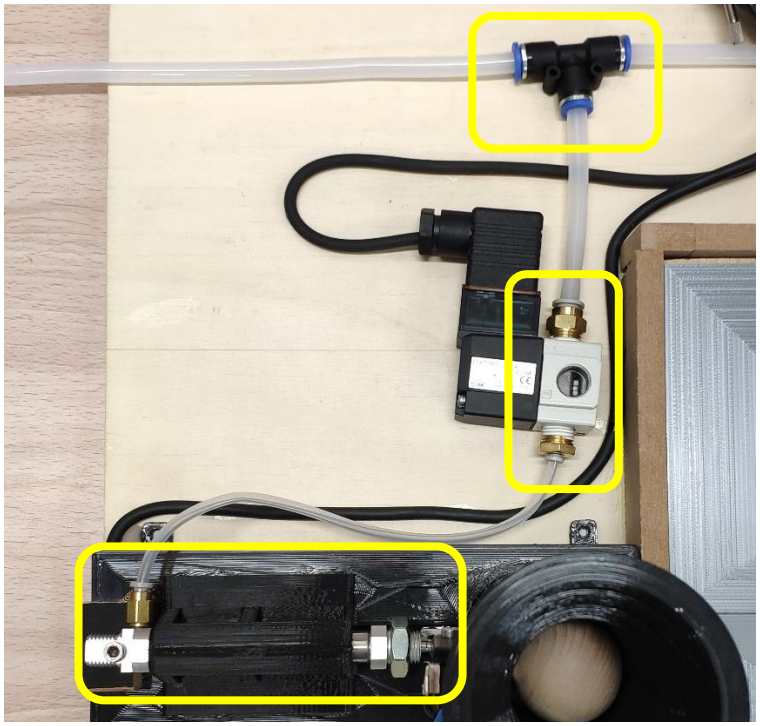


Figura 84. Conexión neumática parte izquierda de la estación

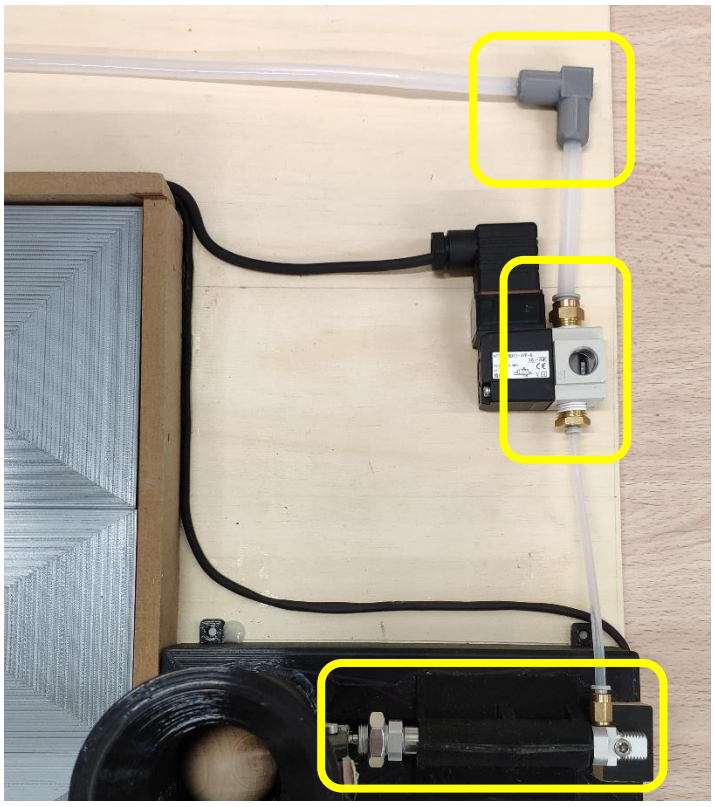


Figura 85. Conexión neumática parte derecha de la estación

La siguiente figura 86 muestra la conexión de la parte neumática de la estación con la toma reguladora de aire comprimido disponible en el laboratorio de la escuela.

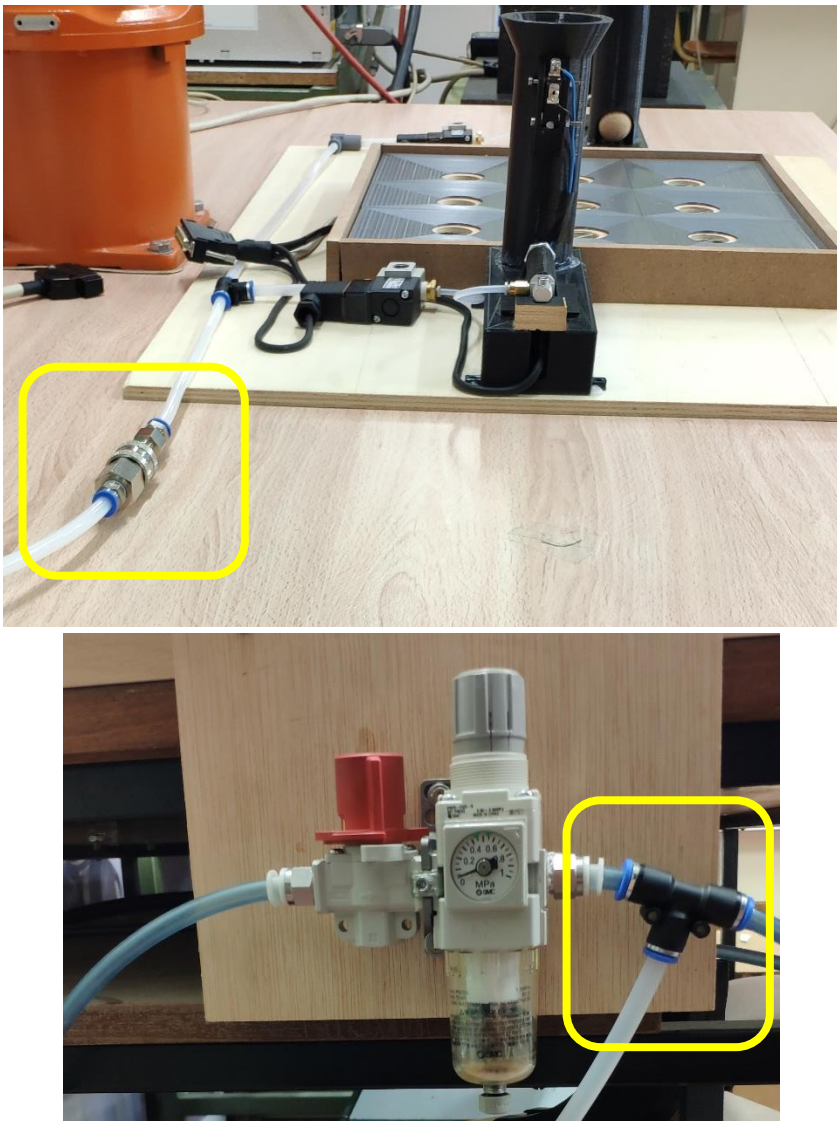


Figura 86. Conexión neumática hacia el regulador de aire comprimido

Diseño y montaje final de la estación

A continuación, en las figuras 87, 88 y 89 se muestra la práctica final construida, ubicada en la mesa de trabajo del robot IRB 120 en el laboratorio de robótica de la Escuela de Ingenierías Industriales de la Universidad de Valladolid.

En las figuras mencionadas podemos observar todos los componentes necesarios para realizar la práctica docente: el robot IRB 120, la controladora IRC5 del robot, la botonera de señales de entrada y salida digitales, la maqueta fabricada, la toma de aire comprimido y el cableado neumático y eléctrico que componen este trabajo.

Figura 87. Estación final

Figura 88. Estación final

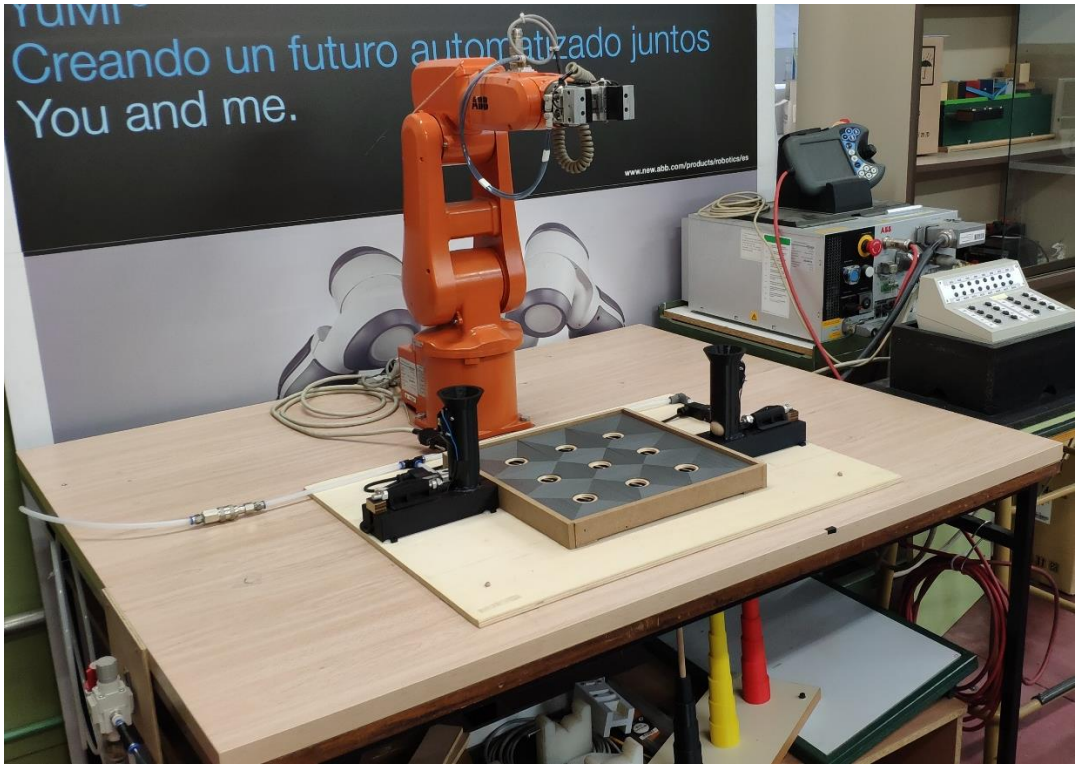


Figura 89. Estación final

3.2. Práctica académica final

Descripción del funcionamiento

Para poder utilizar la maqueta de prácticas académicas que hemos fabricado, comenzaremos colocando la maqueta en los puntos habilitados de la mesa de trabajo del robot IRB 120. Seguidamente conectaremos los conectores eléctricos y neumáticos, abriremos la llave que permite la circulación de aire hasta la toma del. Con la FlexPendant del robot ABB, cargaremos el programa RAPID desarrollado para el correcto funcionamiento de la práctica real. Finalmente, arrancamos el robot IRB 120.

Comenzaremos introduciendo tres bolas en cada torre de almacenamiento. Tras ello, arrancaremos el programa RAPID desde la FlexPendant posicionando el puntero en el main del programa.

Programa RAPID para la práctica desarrollada

Para adaptar el programa de simulación a la estación real, haremos unos ajustes en algunas partes del código RAPID. A continuación, mostraremos y explicaremos con detalle los cambios efectuados en el programa de la estación real.

Las variables, objetos de trabajo y las posiciones del robot y de los elementos de la estación serán los mismos para ambos programas.

Tras ejecutar la función de generación de números pseudoaleatorios por primera vez para determinar cuál de los dos actuadores de las torres deberá accionarse, programamos la siguiente parte de código con algunas variaciones respecto al programa de simulación.

En primer lugar, comprobamos que no hay ninguna bola en los orificios del tablero fijo antes de comenzar el programa mediante la siguiente sentencia, que comprueba que las entradas digitales de los sensores del tablero están desactivadas. Si se cumple dicha condición, continua el siguiente paso. Si por el contrario alguno de los orificios estuviese ocupado por una bola, el IRB 120 se dirige a la posición del sensor activo y ejecuta las trayectorias pertinentes para dirigir la bola hacia una de las torres, y después, continúa con los pasos que se describen en este apartado.

```
IF (di1=0 AND di2=0 AND di3=0 AND di4=0 AND di5=0 AND di6=0 AND di7=0 AND di8=0 AND di10=0) THEN
```

Como ejemplo para explicar el siguiente fragmento de código, tomaremos el accionamiento del actuador situado en la torre 1. Para el segundo actuador neumático se seguirán los mismos pasos, únicamente variará el valor de la variable "RandomActuadores" y la señal digital del sensor inferior de dicha torre.

```
IF (RandomActuadores>=0 AND RandomActuadores<1) AND di11=1 THEN
    REPETIR_1:
    !Si el actuador no ha lanzado correctamente la bola desde la torre 1 de almacenamiento
    !vuelve a intentar lanzarla hasta que caiga en uno de los orificios del tablero
    Set do1;
    WaitTime 1;
    Reset do1;
    WaitTime 5;
    IF (di1=0 AND di2=0 AND di3=0 AND di4=0 AND di5=0 AND di6=0 AND di7=0 AND di8=0 AND di10=0) THEN
        GOTO REPETIR_1;
    ENDIF
```

Incluiremos un ciclo de ejecución dado por la sentencia "GOTO REPETIR_1", que volverá a activar el accionamiento del actuador neumático de la torre 1 hasta que éste consiga lanzar la bola hacia el tablero fijo en caso de que dicha bola no haya podido salir de la torre. Para ello programamos la condición de que, si ninguna de las entradas digitales correspondientes a los sensores localizados en el tablero fijo se ha activado en un tiempo menor a 5 segundos, volverá a activarse la electroválvula que accionará el actuador correspondiente. De la misma forma ocurrirá con la torre 2 y su actuador.

```
ELSEIF (RandomActuadores>=1 AND RandomActuadores<=2) AND di13=1 THEN
    REPETIR_2:
    !Si el actuador no ha lanzado correctamente la bola desde la torre 2 de almacenamiento
    !vuelve a intentar lanzarla hasta que caiga en uno de los orificios del tablero
    Set do2;
    WaitTime 1;
    Reset do2;
    WaitTime 5;
    IF (di1=0 AND di2=0 AND di3=0 AND di4=0 AND di5=0 AND di6=0 AND di7=0 AND di8=0 AND di10=0) THEN
        GOTO REPETIR_2;
    ENDIF
ENDIF
```

Modificaremos la parte del código de activación de los sensores del tablero para corregir que, una vez la bola haya caído en uno de los orificios activando el correspondiente sensor y el robot se dirija hacia él para recogerla, si éste no coge la bola correctamente a la primera, volverá a intentarlo hasta que lo consiga y se deposite la bola en la torre de almacenamiento que corresponda.

Lograremos este objetivo mediante el ciclo de ejecución descrito por la sentencia "GOTO CORREGIR" que introduciremos en la modificación de este fragmento del código, junto con la condición que dicta que si alguna de las entradas digitales correspondientes a los sensores del tablero permanece

activa tras haberse ejecutado la trayectoria de recogida de una bola por parte del robot, vuelva a ejecutarse dicha trayectoria hasta que se consiga recoger la bola de la posición correspondiente al sensor activo. Tomaremos como ejemplo el fragmento de código que representa a la activación del sensor 1 de la primera posición del tablero fijo. Para el resto de las posiciones, el procedimiento será similar variando únicamente la activación de las entradas digitales correspondientes a cada sensor del tablero y la ejecución de la trayectoria del robot hacia dicha posición.

```
IF (RandomActuadores>=0 AND RandomActuadores<1) OR (RandomActuadores>=1 AND RandomActuadores<=2) THEN
    rand:=aleatorio();
    RandomTorres:=rand*2;

    CORREGIR:
    !Si el robot no ha cogido la bola a la primera vuelve al punto donde
    !se encuentre esta antes de que el sistema lance una bola de nuevo

    IF di1=1 THEN
        Path_S1;

        IF di1=1 THEN
            GOTO CORREGIR;
        ENDIF

        IF (RandomTorres>=0 AND RandomTorres<1) THEN
            IF di12=0 THEN
                Path_T1;
            ELSEIF di14=0 THEN
                Path_T2;
            ENDIF
        ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
            IF di14=0 THEN
                Path_T2;
            ELSEIF di12=0 THEN
                Path_T1;
            ENDIF
        ENDIF
```

Puesta en marcha conjunta de la estación con el IRB 120

La siguiente secuencia de imágenes muestra un ejemplo del funcionamiento de la práctica con el robot IRB 120.

En primer lugar, uno de los actuadores lanza una bola hacia el centro del tablero. En este caso (ver figura 90), la bola cae en la posición 1 del tablero fijo, activando su correspondiente sensor.

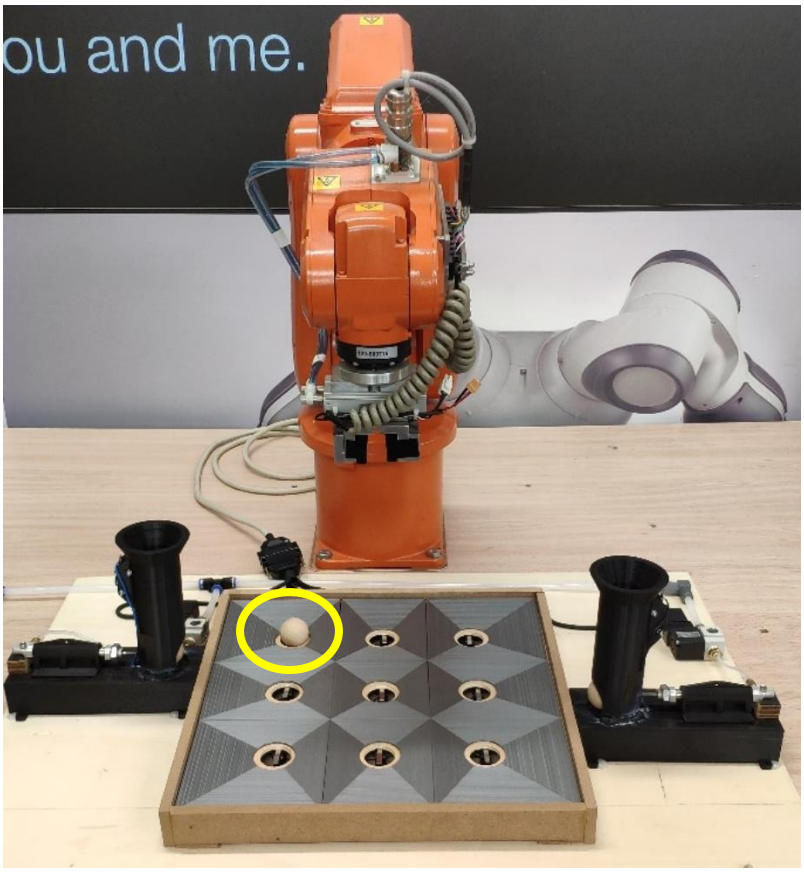


Figura 90. Bola en posición S1 del tablero

En la figura 91, vemos como el robot IRB 120 ejecuta la trayectoria hacia la posición en la que se encuentra dicha bola.

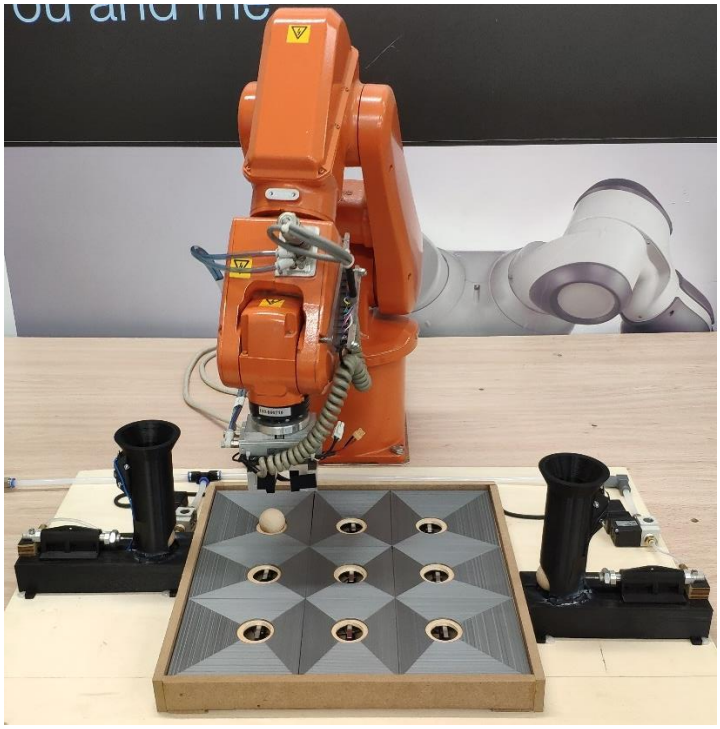


Figura 91. Trayectoria hacia la posición S1 del tablero

La figura 92 muestra la apertura de la pinza del IRB 120 activando la correspondiente salida digital una vez ha llegado a la posición donde se encuentra la bola.

Figura 92. Apertura de pinza del robot IRB 120 en la posición S1 del tablero

El robot ejecuta la acción para cerrar la pinza y sujetar la bola para desplazarla hacia una de las torres de almacenamiento, como vemos en la figura 93.

Figura 93. Cierre de pinza del robot IRB 120

En la figura 94 podemos ver la trayectoria del robot, una vez recogida la bola de la posición S1 del tablero fijo, dirigiéndose hacia una de las torres que disponga de espacio libre para almacenar.

Figura 94. Trayectoria hacia una de las torres de almacenamiento

En la siguiente figura 95 podemos ver el robot IRB 120 finalizar la trayectoria hacia una de las torres (en este caso la torre 2) para proceder a depositar la bola en su interior, tras comprobar mediante las entradas digitales correspondientes de la torre que ésta tiene espacio disponible.

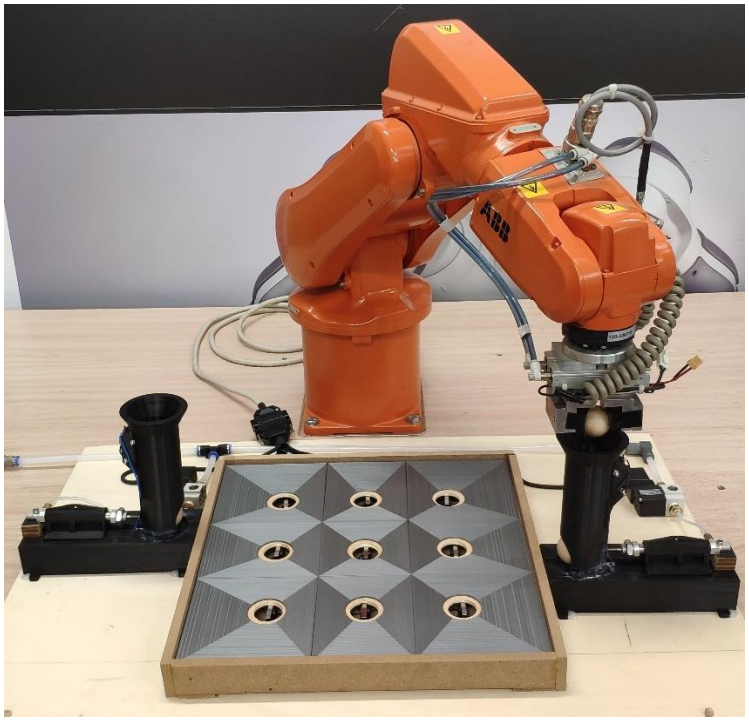


Figura 95. IRB 120 en posición la T2 para descargar la bola

Por último, vemos en la figura 96 la apertura de la pinza del robot para conseguir que la bola caiga en el interior de la torre 2 de almacenamiento. Tras ello, el IRB 120 se dirigirá a la posición base para continuar con la ejecución de la práctica cuando uno de los actuadores lance una nueva bola.

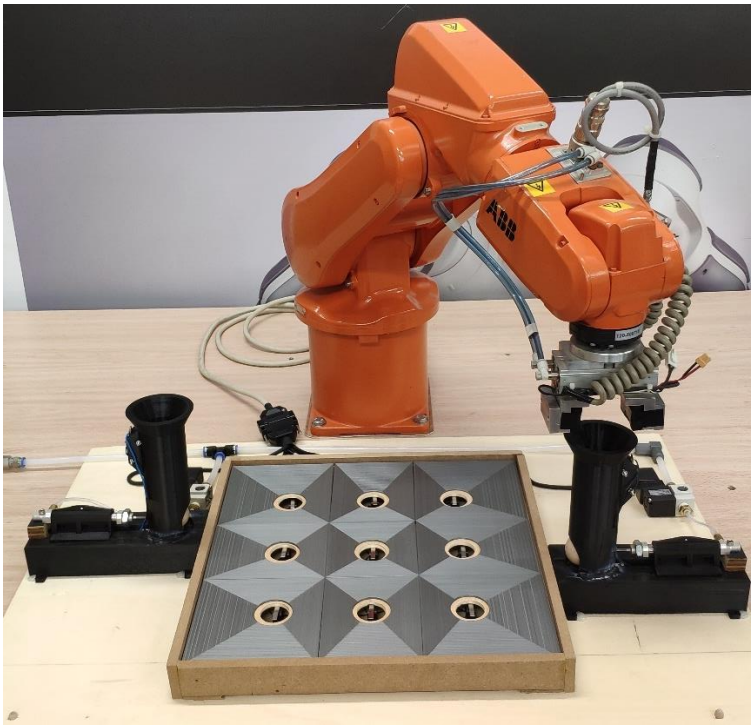


Figura 96. Apertura de pinza del robot IRB 120 en la posición T2 de las torres

4. Conclusiones

En el presente Trabajo Final de Grado se ha realizado el modelado, simulación, diseño y fabricación de una aplicación para prácticas docentes junto con el robot ABB IRB 120 disponible en la Escuela de Ingenierías Industriales de la Universidad de Valladolid.

La simulación de la práctica docente utilizando RobotStudio, nos ha permitido realizar una muy buena aproximación a la práctica real que queríamos desarrollar con el robot ABB IRB120. En primer lugar, se creó una estación de trabajo virtual con RobotStudio, en la que se realizó toda la programación del robot en el lenguaje RAPID, así como su interacción con el entorno mediante señales digitales del robot. Esta simulación nos permitió tener un muy buen conocimiento de cómo se iba a comportar la práctica docente real que después construimos. Además, el trabajo desarrollado en esta simulación permitirá el aprendizaje de la herramienta de programación RobotStudio para futuros estudiantes de robótica, en el ámbito de los componentes inteligentes de RobotStudio.

Con toda la información obtenida de la simulación, se realizó el montaje físico de la práctica académica mediante la utilización de actuadores neumáticos y sensores eléctricos digitales. Un aspecto importante fue la integración de esta práctica con el controlador del robot IRB120 mediante el cableado correspondiente. Una vez elaborado el prototipo, el código de programa creado en la etapa de simulación se utilizó, con pequeños ajustes, para controlar el robot IRB120 en la práctica académica construida.

Los resultados obtenidos, tanto en la simulación, como en la práctica real construida, han sido muy satisfactorios, logrando los objetivos planteados al comienzo del trabajo.

Personalmente, el presente proyecto me ha servido para ampliar y mejorar mis conocimientos en programación robótica con RobotStudio, realizar labores más técnicas de ingeniería en cuanto al diseño y montaje de la práctica docente y, por último, refrescar y aprender más sobre conexiones eléctricas, electrónicas y neumáticas de señales digitales.

5. Bibliografía

[1] Martín Castillo, J. C. (2016). Actividades con el robot IRB 120 de ABB. Centro integrado de FP "Ciudad de Béjar" (Salamanca).

https://www.youtube.com/watch?v=GvFE_QbSq3o

El anterior enlace fue visitado por última vez el 15/06/2021.

[2] Martín Castillo, J. C. (2017). Ejercicio de manipulación y pegado de piezas con el robot IRB 120 de ABB. Centro integrado de FP "Ciudad de Béjar" (Salamanca).

https://www.youtube.com/watch?v=U5IjvFBxfdY

El anterior enlace fue visitado por última vez el 15/06/2021.

[3] Martínez, M. y Nicolás, M. (2018). Proyecto final de automatización y robótica industrial con el robot IRB 120. IES Sierra de Carrascoy.

https://www.youtube.com/watch?v=76itEjphvKI

El anterior enlace fue visitado por última vez el 15/06/2021.

[4] Senén Estremera, A. (2015). Diseño de una estación de trabajo para el robot IRB 120. Control de cinta transportadora mediante IRC5. Escuela politécnica superior de la Universidad de Alcalá.

https://ebuah.uah.es/dspace/bitstream/handle/10017/26937/TFG-Sen%C3%A9n-Estremera.pdf?sequence=1&isAllowed=y

El anterior enlace fue visitado por última vez el 15/06/2021.

[5] https://new.abb.com/products/robotics/es/robotstudio

[6] Cañete López, A. (2018). Pick and place con objetos inteligentes.

https://www.youtube.com/watch?v=UNDgMr2cYs4

El anterior enlace fue visitado por última vez el 15/06/2021.

[7] Tapia, A. (2015). Tutorial con RobotStudio de la simulación de componentes inteligentes.

https://www.youtube.com/watch?v=gSQcvkKY9HU

El anterior enlace fue visitado por última vez el 15/06/2021.

6. Anexos

6.1. Código del programa RAPID "Estación Automatizada"

```
MODULE CalibData
        PERS tooldata Pinza20170721:=[TRUE,[[0,0,67],[0.707106781,0,0,0.707106781]],[1,[0,0,67],[1,0,0,0],0,0,0]];
        TASK PERS wobjdata WO_TableroFijo:=[FALSE,TRUE,"",[[320,-100,40],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
        TASK PERS wobjdata WO_Torres:=[FALSE,TRUE,"",[[420,-205,210],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
ENDMODULE

MODULE Modulo_EstacionAutomatizadaSC
   !Posicion inicial del robot
   CONST robtarget Pos_Inicial:=[[374.55,0,630.67],[0.710481,0.000001647,0.703716,0.000003144],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   !Posicion base para ejecutar desplazamientos hacia los objetivos
   CONST robtarget Pos_Base:=[[433.10135365,0,288.094692696],[0.001269304,-0.707105642,-0.707105642,0.001269304],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   !Posiciones de los puntos exactos en tablero fijo y torres
   CONST robtarget S1:=[[0,0,25],[0,0,1,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S2:=[[0,100,25],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S3:=[[0,200,25],[0,0,1,0],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S4:=[[100,0,25],[0,0,1,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S5:=[[100,100,25],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S6:=[[100,200,25],[0,0,1,0],[0,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S7:=[[200,0,25],[0,0,1,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S8:=[[200,100,25],[0,0,1,0],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget S9:=[[200,200,25],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
   CONST robtarget T1:=[[0,0,25],[0,0,1,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget
T2:=[[0,410,25],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   !Posiciones cercanas a los puntos en tablero fijo y torres
   CONST robtarget Pos_S1:=[[0,0,50],[0,0,1,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S2:=[[0,100,50],[0,0,1,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S3:=[[0,200,50],[0,0,1,0],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S4:=[[100,0,50],[0,0,1,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S5:=[[100,100,50],[0,0,1,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S6:=[[100,200,50],[0,0,1,0],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S7:=[[200,0,50],[0,0,1,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S8:=[[200,100,50],[0,0,1,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_S9:=[[200,200,50],[0,0,1,0],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_T1:=[[0,0,40],[0,0,1,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget Pos_T2:=[[0,410,40],[0,0,1,0],[0,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   !Número aleaotorio, semilla
   LOCAL PERS num seed:=2667;

   VAR num bolasT1:=3;
   VAR num bolasT2:=3;

   PROC main()
      VAR num rand;
      VAR num RandomActuadores;
      VAR num RandomTablero;
      VAR num RandomTorres;
      VAR pos inicio;

      IF bolasT1>0 THEN
         Set do10;
         IF bolasT1=3 THEN
            Set do11;
         ELSE
            Reset do11;
```

```
        ENDIF
    ELSE
        Reset do10;
        Reset do13;
    ENDIF

    IF bolasT2>0 THEN
        Set do10;
        IF bolasT2=3 THEN
            Set do12;
        ELSE
            Reset do12;
        ENDIF
    ELSE
        Reset do10;
        Reset do14;
    ENDIF

    rand:=aleatorio();
    RandomActuadores:=rand*2;

    IF (RandomActuadores>=0 AND RandomActuadores<1) AND bolasT1>0
THEN
        Set do13;
        bolasT1:=bolasT1-1;
        Reset do11;
        WaitTime 1;
        Reset do13;
    ELSEIF (RandomActuadores>=1 AND RandomActuadores<=2) AND
bolasT2>0 THEN
        Set do14;
        bolasT2:=bolasT2-1;
        Reset do12;
        WaitTime 1;
        Reset do14;
    ENDIF

    IF (RandomActuadores>=0 AND RandomActuadores<1) OR
(RandomActuadores>=1 AND RandomActuadores<=2) THEN
        rand:=aleatorio();
        RandomTablero:=rand*9;
        !Para obtener valor aleatorio entre 0 y 9
        RandomTorres:=rand*2;
        !Para obtener valor aleatorio entre 0 y 2

        IF RandomTablero>=0 AND RandomTablero<1 THEN
            Set do1;
            Path_S1;
```

```
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF bolasT1<3 THEN
        Path_T1;
      ELSEIF bolasT2<3 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF bolasT2<3 THEN
        Path_T2;
      ELSEIF bolasT1<3 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF RandomTablero>=1 AND RandomTablero<2 THEN
    Set do2;
    Path_S2;
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF bolasT1<3 THEN
        Path_T1;
      ELSEIF bolasT2<3 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF bolasT2<3 THEN
        Path_T2;
      ELSEIF bolasT1<3 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF RandomTablero>=2 AND RandomTablero<3 THEN
    Set do3;
    Path_S3;
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF bolasT1<3 THEN
        Path_T1;
      ELSEIF bolasT2<3 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF bolasT2<3 THEN
        Path_T2;
      ELSEIF bolasT1<3 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF RandomTablero>=3 AND RandomTablero<4 THEN
    Set do4;
    Path_S4;
```

```
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF bolasT1<3 THEN
        Path_T1;
      ELSEIF bolasT2<3 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF bolasT2<3 THEN
        Path_T2;
      ELSEIF bolasT1<3 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF RandomTablero>=4 AND RandomTablero<5 THEN
    Set do5;
    Path_S5;
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF bolasT1<3 THEN
        Path_T1;
      ELSEIF bolasT2<3 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF bolasT2<3 THEN
        Path_T2;
      ELSEIF bolasT1<3 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF RandomTablero>=5 AND RandomTablero<6 THEN
    Set do6;
    Path_S6;
    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF bolasT1<3 THEN
        Path_T1;
      ELSEIF bolasT2<3 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF bolasT2<3 THEN
        Path_T2;
      ELSEIF bolasT1<3 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF RandomTablero>=6 AND RandomTablero<7 THEN
    Set do7;
    Path_S7;
```

```
        IF (RandomTorres>=0 AND RandomTorres<1) THEN
          IF bolasT1<3 THEN
            Path_T1;
          ELSEIF bolasT2<3 THEN
            Path_T2;
          ENDIF
        ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
          IF bolasT2<3 THEN
            Path_T2;
          ELSEIF bolasT1<3 THEN
            Path_T1;
          ENDIF
        ENDIF
      ELSEIF RandomTablero>=7 AND RandomTablero<8 THEN
        Set do8;
        Path_S8;
        IF (RandomTorres>=0 AND RandomTorres<1) THEN
          IF bolasT1<3 THEN
            Path_T1;
          ELSEIF bolasT2<3 THEN
            Path_T2;
          ENDIF
        ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
          IF bolasT2<3 THEN
            Path_T2;
          ELSEIF bolasT1<3 THEN
            Path_T1;
          ENDIF
        ENDIF
      ELSEIF RandomTablero>=8 AND RandomTablero<=9 THEN
        Set do9;
        Path_S9;
        IF (RandomTorres>=0 AND RandomTorres<1) THEN
          IF bolasT1<3 THEN
            Path_T1;
          ELSEIF bolasT2<3 THEN
            Path_T2;
          ENDIF
        ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
          IF bolasT2<3 THEN
            Path_T2;
          ELSEIF bolasT1<3 THEN
            Path_T1;
          ENDIF
        ENDIF
      ENDIF
    ENDIF
```

```
ENDPROC

!Obtiene valor aleatorio entre 0 y 1
FUNC num aleatorio()
   seed:=(171*seed) MOD 30269;
   RETURN seed/30269;
ENDFUNC

!Abrir pinza neumática
PROC AbrirPinza()
   Set do15; !Abrir y desattacher
   WaitTime 2;
   Reset do15;
ENDPROC

!Cerrar pinza neumática
PROC CerrarPinza()
   Set do16;  !Activar los sensores de attacher
   WaitTime 2;
   Reset do16;
ENDPROC

!Trayectorias hacia los sensores del tablero fijo
PROC Path_S1()
   MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
   MoveL Pos_S1,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
   AbrirPinza;
   MoveJDO S1,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do1,0;
   WaitTime 2;
   CerrarPinza;
   WaitTime 2;
   MoveJ Pos_S1,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S2()
   MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
   MoveL Pos_S2,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
   AbrirPinza;
   MoveJDO S2,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do2,0;
   WaitTime 2;
   CerrarPinza;
   WaitTime 2;
   MoveJ Pos_S2,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S3()
   MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
```

```
    MoveL Pos_S3,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S3,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do3,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S3,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S4()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_S4,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S4,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do4,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S4,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S5()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_S5,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S5,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do5,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S5,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S6()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_S6,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S6,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do6,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S6,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S7()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_S7,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S7,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do7,0;
    WaitTime 2;
```

```
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S7,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S8()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_S8,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S8,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do8,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S8,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

PROC Path_S9()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_S9,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJDO S9,v20,fine,Pinza20170721\WObj:=WO_TableroFijo,do9,0;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S9,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
ENDPROC

!Trayectorias hacia las torres de almacenamiento
PROC Path_T1()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_T1,v100,z0,Pinza20170721\WObj:=WO_Torres;
    MoveJ T1,v20,fine,Pinza20170721\WObj:=WO_Torres;
    WaitTime 2;
    AbrirPinza;
    bolasT1:=bolasT1+1;
    WaitTime 2;
    CerrarPinza;
    MoveJ Pos_T1,v100,z0,Pinza20170721\WObj:=WO_Torres;
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
ENDPROC

PROC Path_T2()
    MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveL Pos_T2,v100,z0,Pinza20170721\WObj:=WO_Torres;
    MoveJ T2,v20,fine,Pinza20170721\WObj:=WO_Torres;
    WaitTime 2;
    AbrirPinza;
    bolasT2:=bolasT2+1;
```

```
        WaitTime 2;
        CerrarPinza;
        MoveJ Pos_T2,v100,z0,Pinza20170721\WObj:=WO_Torres;
        MoveL Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    ENDPROC

ENDMODULE
```

6.2. Código del programa RAPID "Estación Real"

```
MODULE CalibData
        PERS tooldata Pinza20170721:=[TRUE,[[0,0,67],[0.707106781,0,0,0.707106781]],[1,[0,0,67],[1,0,0,0],0,0,0]];
        TASK PERS wobjdata WO_TableroFijo:=[FALSE,TRUE,"",[[320,-100,40],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
        TASK PERS wobjdata WO_Torres:=[FALSE,TRUE,"",[[420,-205,210],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
ENDMODULE

MODULE Modulo_EstacionReal
    !Posicion inicial del robot
    CONST robtarget Pos_Inicial:=[[374.55,0,630.67],[0.710481,0.000001647,0.703716,0.000003144],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !Posicion base para ejecutar desplazamientos hacia los objetivos
    CONST robtarget Pos_Base:=[[421.076616387,0,339.998197536],[0.001269304,-0.707105642,-0.707105642,0.001269304],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !Posiciones de los puntos exactos en tablero fijo y torres
    CONST robtarget S1:=[[0,0,30],[0,0.707106781,0.707106781,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget S2:=[[0,102.5,30],[0,0.707106781,0.707106781,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget S3:=[[-0.00000001,205,30],[0,0.707106781,0.707106781,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget S4:=[[102.5,0,30],[0,0.707106781,0.707106781,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget S5:=[[102.5,102.5,30],[0,0.707106781,0.707106781,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget S6:=[[102.5,205,30],[0,0.707106781,0.707106781,0],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget
S7:=[[205,0.00000001,30],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget S8:=[[205,102.5,30],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
S9:=[[205.00000001,204.99999999,30],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget T1:=[[0,0,40],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget T2:=[[100,480,40],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !Posiciones cercanas a los puntos en tablero fijo y torres
    CONST robtarget Pos_S1:=[[0,0,50],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_S2:=[[0,102.5,50],[0,0.707106781,0.707106781,0],[-
1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_S3:=[[-
0.00000001,205,50],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_S4:=[[102.5,0,50],[0,0.707106781,0.707106781,0],[-
1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Pos_S5:=[[102.5,102.5,50],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_S6:=[[102.5,205,50],[0,0.707106781,0.707106781,0],[-
1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Pos_S7:=[[205,0.00000001,50],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_S8:=[[205,102.5,50],[0,0.707106781,0.707106781,0],[-
1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Pos_S9:=[[205.00000001,204.99999999,50],[0,0.707106781,0.707106781,0],[-
1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_T1:=[[0,0,60],[0,0.707106781,0.707106781,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Pos_T2:=[[100,480,60],[0,0.707106781,0.707106781,0],[-
1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !Número aleaotorio, semilla
    LOCAL PERS num seed:=14780;

    PROC main()
        VAR num rand;
        VAR num RandomActuadores;
        VAR num RandomTablero;
```

```
    VAR num RandomTorres;
    VAR num contador:=0;

    MoveJ Pos_Inicial,v300,fine,tool0;

    INICIO:

    rand:=aleatorio();
    RandomActuadores:=rand*2;
    !Para obtener valor aleatorio entre 0 y 2

    IF (di1=0 AND di2=0 AND di3=0 AND di4=0 AND di5=0 AND di6=0
AND di7=0 AND di8=0 AND di10=0) THEN
      IF (RandomActuadores>=0 AND RandomActuadores<1) AND di11=1
THEN
        REPETIR_1:
        !Si el actuador no ha lanzado correctamente la bola desde la torre 1 de
almacenamiento
        !vuelve a intentar lanzarla hasta que caiga en uno de los orificios del
tablero
        Set do1;
        WaitTime 1;
        Reset do1;
        WaitTime 6;
        IF (di1=0 AND di2=0 AND di3=0 AND di4=0 AND di5=0 AND di6=0
AND di7=0 AND di8=0 AND di10=0) THEN
          GOTO REPETIR_1;
        ENDIF
      ELSEIF (RandomActuadores>=1 AND RandomActuadores<=2) AND
di13=1 THEN
        REPETIR_2:
        !Si el actuador no ha lanzado correctamente la bola desde la torre 2 de
almacenamiento
        !vuelve a intentar lanzarla hasta que caiga en uno de los orificios del
tablero
        Set do2;
        WaitTime 1;
        Reset do2;
        WaitTime 6;
        IF (di1=0 AND di2=0 AND di3=0 AND di4=0 AND di5=0 AND di6=0
AND di7=0 AND di8=0 AND di10=0) THEN
          GOTO REPETIR_2;
        ENDIF
      ENDIF
    ENDIF

    IF (RandomActuadores>=0 AND RandomActuadores<1) OR
(RandomActuadores>=1 AND RandomActuadores<=2) THEN
```

```
rand:=aleatorio();
RandomTorres:=rand*2;

CORREGIR:
!Si el robot no ha cogido la bola a la primera vuelve al punto donde
!se encuentre esta antes de que el sistema lance una bola de nuevo

IF di1=1 THEN
   Path_S1;

   IF di1=1 THEN
      GOTO CORREGIR;
   ENDIF

   IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF di12=0 THEN
         Path_T1;
      ELSEIF di14=0 THEN
         Path_T2;
      ENDIF
   ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF di14=0 THEN
         Path_T2;
      ELSEIF di12=0 THEN
         Path_T1;
      ENDIF
   ENDIF
ELSEIF di2=1 THEN
   Path_S2;

   IF di2=1 THEN
      GOTO CORREGIR;
   ENDIF

   IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF di12=0 THEN
         Path_T1;
      ELSEIF di14=0 THEN
         Path_T2;
      ENDIF
   ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF di14=0 THEN
         Path_T2;
      ELSEIF di12=0 THEN
         Path_T1;
      ENDIF
   ENDIF
ELSEIF di3=1 THEN
```

```
    Path_S3;

    IF di3=1 THEN
      GOTO CORREGIR;
    ENDIF

    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF di12=0 THEN
        Path_T1;
      ELSEIF di14=0 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF di14=0 THEN
        Path_T2;
      ELSEIF di12=0 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF di4=1 THEN
    Path_S4;

    IF di4=1 THEN
      GOTO CORREGIR;
    ENDIF

    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF di12=0 THEN
        Path_T1;
      ELSEIF di14=0 THEN
        Path_T2;
      ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
      IF di14=0 THEN
        Path_T2;
      ELSEIF di12=0 THEN
        Path_T1;
      ENDIF
    ENDIF
  ELSEIF di5=1 THEN
    Path_S5;

    IF di5=1 THEN
      GOTO CORREGIR;
    ENDIF

    IF (RandomTorres>=0 AND RandomTorres<1) THEN
      IF di12=0 THEN
```

```
            Path_T1;
        ELSEIF di14=0 THEN
            Path_T2;
        ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
        IF di14=0 THEN
            Path_T2;
        ELSEIF di12=0 THEN
            Path_T1;
        ENDIF
    ENDIF
ELSEIF di6=1 THEN
    Path_S6;

    IF di6=1 THEN
        GOTO CORREGIR;
    ENDIF

    IF (RandomTorres>=0 AND RandomTorres<1) THEN
        IF di12=0 THEN
            Path_T1;
        ELSEIF di14=0 THEN
            Path_T2;
        ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
        IF di14=0 THEN
            Path_T2;
        ELSEIF di12=0 THEN
            Path_T1;
        ENDIF
    ENDIF
ELSEIF di7=1 THEN
    Path_S7;

    IF di7=1 THEN
        GOTO CORREGIR;
    ENDIF

    IF (RandomTorres>=0 AND RandomTorres<1) THEN
        IF di12=0 THEN
            Path_T1;
        ELSEIF di14=0 THEN
            Path_T2;
        ENDIF
    ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
        IF di14=0 THEN
            Path_T2;
        ELSEIF di12=0 THEN
```

```
            Path_T1;
          ENDIF
        ENDIF
      ELSEIF di8=1 THEN
        Path_S8;

        IF di8=1 THEN
          GOTO CORREGIR;
        ENDIF

        IF (RandomTorres>=0 AND RandomTorres<1) THEN
          IF di12=0 THEN
            Path_T1;
          ELSEIF di14=0 THEN
            Path_T2;
          ENDIF
        ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
          IF di14=0 THEN
            Path_T2;
          ELSEIF di12=0 THEN
            Path_T1;
          ENDIF
        ENDIF
      ELSEIF di10=1 THEN
        Path_S9;

        IF di10=1 THEN
          GOTO CORREGIR;
        ENDIF

        IF (RandomTorres>=0 AND RandomTorres<1) THEN
          IF di12=0 THEN
            Path_T1;
          ELSEIF di14=0 THEN
            Path_T2;
          ENDIF
        ELSEIF (RandomTorres>=1 AND RandomTorres<=2) THEN
          IF di14=0 THEN
            Path_T2;
          ELSEIF di12=0 THEN
            Path_T1;
          ENDIF
        ENDIF
      ENDIF
    ENDIF

    GOTO INICIO;
```

```
ENDPROC

!Obtiene valor aleatorio entre 0 y 1
FUNC num aleatorio()
   seed:=(171*seed) MOD 30269;
   RETURN seed/30269;
ENDFUNC

!Abrir pinza neumática
PROC AbrirPinza()
   Set do15;
   !Abrir y desattacher
   WaitTime 2;
   Reset do15;
ENDPROC

!Cerrar pinza neumática
PROC CerrarPinza()
   Set do16;
   !Activar los sensores de attacher
   WaitTime 2;
   Reset do16;
ENDPROC

!Trayectorias hacia los sensores del tablero fijo
PROC Path_S1()
   MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
   MoveJ Pos_S1,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
   AbrirPinza;
   MoveJ S1,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
   WaitTime 2;
   CerrarPinza;
   WaitTime 2;
   MoveJ Pos_S1,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
   WaitTime 1;
ENDPROC

PROC Path_S2()
   MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
   MoveJ Pos_S2,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
   AbrirPinza;
   MoveJ S2,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
   WaitTime 2;
   CerrarPinza;
   WaitTime 2;
   MoveJ Pos_S2,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
   WaitTime 1;
ENDPROC
```

```
PROC Path_S3()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S3,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S3,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S3,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC

PROC Path_S4()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S4,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S4,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S4,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC

PROC Path_S5()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S5,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S5,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S5,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC

PROC Path_S6()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S6,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S6,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S6,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC
```

```
PROC Path_S7()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S7,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S7,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S7,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC

PROC Path_S8()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S8,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S8,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S8,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC

PROC Path_S9()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_S9,v100,z0,Pinza20170721\WObj:=WO_TableroFijo;
    AbrirPinza;
    MoveJ S9,v10,fine,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 2;
    CerrarPinza;
    WaitTime 2;
    MoveJ Pos_S9,v10,z0,Pinza20170721\WObj:=WO_TableroFijo;
    WaitTime 1;
ENDPROC

!Trayectorias hacia las torres de almacenamiento
PROC Path_T1()
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    MoveJ Pos_T1,v100,z0,Pinza20170721\WObj:=WO_Torres;
    MoveJ T1,v10,fine,Pinza20170721\WObj:=WO_Torres;
    WaitTime 2;
    AbrirPinza;
    WaitTime 2;
    CerrarPinza;
    MoveJ Pos_T1,v10,z0,Pinza20170721\WObj:=WO_Torres;
    MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
```

```
    ENDPROC

    PROC Path_T2()
        MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
        MoveJ Pos_T2,v100,z0,Pinza20170721\WObj:=WO_Torres;
        MoveJ T2,v10,fine,Pinza20170721\WObj:=WO_Torres;
        WaitTime 2;
        AbrirPinza;
        WaitTime 2;
        CerrarPinza;
        MoveJ Pos_T2,v10,z0,Pinza20170721\WObj:=WO_Torres;
        MoveJ Pos_Base,v100,z0,Pinza20170721\WObj:=wobj0;
    ENDPROC

ENDMODULE
```