Universidad Zaragoza
1542

Trabajo Fin de Grado

Generación de escenarios y programación de brazos robóticos con RobotStudio y RAPID (ABB)

Autor

Daniel Vicente Moya

Directores

Rosario Aragüés Muñoz

Raúl Igual Catalán

Escuela Universitaria Politécnica de Teruel

2016

Generación de escenarios y programación de brazos robóticos con RobotStudio y RAPID (ABB)

Resumen

Este Trabajo Fin de Grado se enmarca en el aprendizaje y utilización del programa ***RobotStudio*** para la simulación de procesos industriales con presencia de elementos robóticos. En una primera fase se centra en la recopilación de la información necesaria para manejar el programa ***RobotStudio***, haciendo uso para ello de video-tutoriales y del propio manual del programa. Así mismo, se recopila información de diversos robots y herramientas industriales mediante el estudio de manuales y de los respectivos productos.

Los conocimientos adquiridos en esta primera etapa se afianzan después mediante la realización de tres escenarios virtuales de diversos procesos y tareas. Estos escenarios abarcan el uso del programa desde un nivel de conocimientos básico hasta un nivel avanzado del mismo. Con todo ello, la dificultad en el uso de las herramientas del programa se va incrementando con cada escenario.

De este modo, para el primer escenario se aplican modelos sencillos y tareas de desplazamiento de objetos. En el segundo escenario el incremento de la complejidad radica en la implementación de tareas paralelas sobre un mismo controlador, así como en la creación de componentes complejos en el escenario.

Para finalizar, se realiza una réplica de un escenario real utilizando la herramienta de simulación. En concreto, se implementa el escenario virtual del Laboratorio de Tecnologías Industriales de la Escuela Universitaria Politécnica de Teruel para el que se desarrollan dos programas a modo de ejemplo de uso. Para poder llevar esto a cabo es necesario recrear los diferentes elementos del escenario real en modelos virtuales dentro del programa. Los programas de prueba realizados permitieron comprobar si los parámetros del escenario real se habían ajustado correctamente en la simulación, concluyendo que el escenario simulado está completo y compuesto de elementos plenamente funcionales.

Índice

1. Introducción

1.1. Objetivos

El objetivo de este proyecto es aprender el uso avanzado del programa ***RobotStudio*** y de sus herramientas con el fin de realizar una réplica simulada del Laboratorio de Tecnologías Industriales de la Escuela Universitaria Politécnica de Teruel. Esta replica contará con los componentes dispuestos en el área asignada del laboratorio. Mediante el desarrollo de tres escenarios se obtendrán los conocimientos necesarios para desarrollar el trabajo. La complejidad de estos escenarios irá aumentando así como las herramientas y funciones aplicadas.

- Escenario1: Conceptos básicos y pasos en la creación de escenarios.
- Escenario2: Gestión de tareas, control y creación de sensores y herramientas.
- Escenario3: Implementación de cintas, comunicación entre controladores y gestión de interrupciones del sistema.

Como resultado del TFG se proporcionarán los escenarios, librerías de objetos y programas, junto con una documentación detallada paso a paso de su desarrollo. De esta manera, se facilitará el desarrollo posterior de aplicaciones en ***RobotStudio*** para aquellos que estuviesen interesados.

1.2. Estructura proyecto

A continuación pasamos a describir brevemente la estructura de la memoria de este Trabajo Fin de Grado (TFG).

1) Contexto:

El primer apartado se enumera los materiales y elementos del laboratorio utilizados para la realización del proyecto. Asimismo, se proporciona una breve referencia al programa utilizado y las fuentes de información que sirvieron de referencia.

2) Descripción de escenarios:

Este apartado recoge toda la información referente a la creación de escenarios. Sirve como guía de implementación, desde los apartados básicos a otros más avanzados.

3) Réplica del laboratorio:

Recoge la creación del escenario y las librerías necesarias para el funcionamiento de un entorno simulado similar al que podamos encontrar en el laboratorio real.

4) Conclusiones y trabajo futuro:

Compendio de ideas y listado de dificultades encontradas en el trabajo, así como líneas de trabajo derivadas que se podrían seguir desarrollando en un futuro.

Se proporciona además una serie de Anexos donde se incluyen las especificaciones técnicas de los brazos robóticos y de los elementos del entorno del Laboratorio de Tecnologías Industriales, así como información detallada de los escenarios y los programas (Anexo 1 al Anexo 5). Estos Anexos se encuentran en el CD/DVD que acompaña a esta memoria. Este mismo CD/DVD contiene asimismo el texto completo de la memoria, y los archivos de los escenarios, objetos, librerías y programas desarrollados.

2. Contexto

En este apartado describimos todos los elementos utilizados en este TFG.

2.1. Antecedentes

En los últimos años, la robótica industrial ha alcanzado un gran protagonismo en los sistemas de producción automatizados.

Una de las primeras fases a la hora de implantar una solución que involucre el uso de robots industriales, consiste en el modelado del entorno, de los robots disponibles y de las interacciones entre los mismos mediante un programa de simulación. De esta manera, se pueden verificar, validar y optimizar tanto la disposición de los elementos del entorno, como los programas que ejecutarán los robots.

Esto es de especial importancia en entornos didácticos, donde el objetivo es formar a estudiantes en la programación de robots industriales. El poder contar con un simulador, permite que los usuarios adquieran una adecuada formación antes de enfrentarse a la programación de los robots reales. En este TFG, se facilitarán estas tareas mediante la realización del modelado en un entorno virtual del Laboratorio de Tecnologías Industriales de la Escuela Universitaria Politécnica de Teruel.

2.2. Software utilizado

Como herramienta de desarrollo se ha utilizado el software de simulación ***RobotStudio 6.01***, proporcionado por **ABB**[1]. Este programa permite gestionar, simular y optimizar sistemas robóticos antes de su implantación en el sistema real. El programa utiliza ***VirtualControler***, que es el software utilizado para el control robótico de los sistemas **ABB**. Esto permite que los programas desarrollados se puedan cargar directamente en el controlador del robot. El lenguaje de programación desarrollado por **ABB** y que usa para el control robótico, es el lenguaje ***RAPID***. Este es un lenguaje textual de alto nivel que se estructura en un programa principal y una serie de módulos del sistema.

Con ***RobotStudio*** se han generado todos los escenarios, programas ***RAPID*** y librerías, que permitieron el entrenamiento y posterior desarrollo del TFG. Los escenarios están descritos en profundidad en el apartado 3, y la réplica del laboratorio, junto con las librerías, en el apartado 4. Para mayor referencia sobre los programas ***RAPID*** de cada escenario se dispone de ellos en el Anexo 5. Cualquier interesado puede hacer uso de los recursos generados para este TFG con los que podrá generar nuevos elementos más avanzados con ***RobotStudio***, como se sugiere en el apartado 5.3 sobre las líneas de trabajo futuro.

2.3. Laboratorio

Las instalaciones utilizadas para el desarrollo del trabajo, son las disponibles en el Laboratorio de Tecnologías Industriales. El laboratorio está alojado en el sótano de la Escuela Universitaria Politécnica de Teruel. Solo interesa del mismo la mitad disponible para las aplicaciones robóticas. La Ilustración 1 muestra una de las jaulas disponibles en el laboratorio.

Ilustración 1 Jaula de un robot del Laboratorio de Tecnologías Industriales

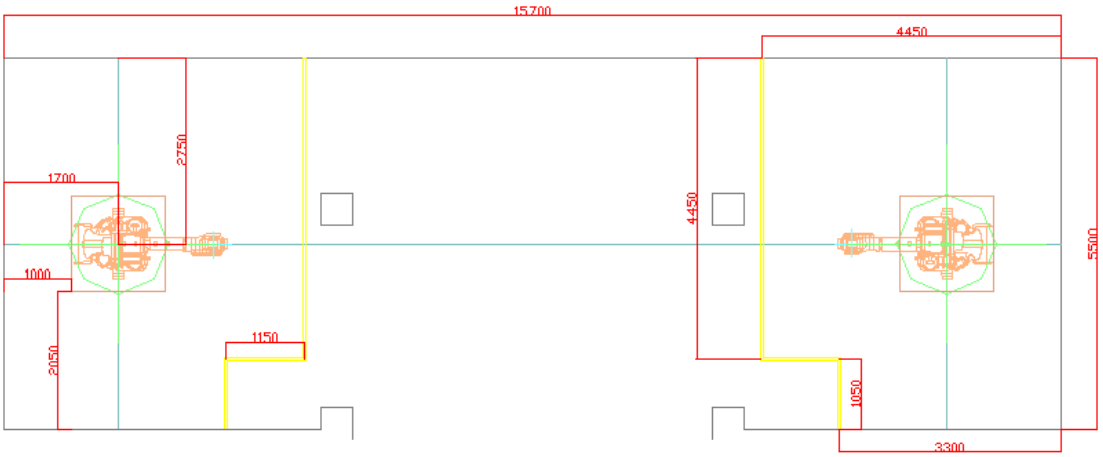

Ilustración 2 Plano jaulas laboratorio

Como se observa en el plano (Ilustración 2) las jaulas están dispuestas en los extremos de la sala, una enfrente de la otra. Ambas jaulas son simétricas con los robots dispuestos en el centro.

Los robots son dos **ABB IRB 6400 2.4-150** [13], modelos de 6 ejes (Ilustración 3) diseñados para la fabricación de sistemas de automoción. Estos modelos disponen de capacidad máxima de manipulación de 150kg y un alcance máximo de 2.4m desde el centro de sus muñecas. Tienen un peso de 2050kg y están pensados para ser montados en el suelo. Las especificaciones técnicas del mismo pueden encontrarse en el Anexo 1.1.

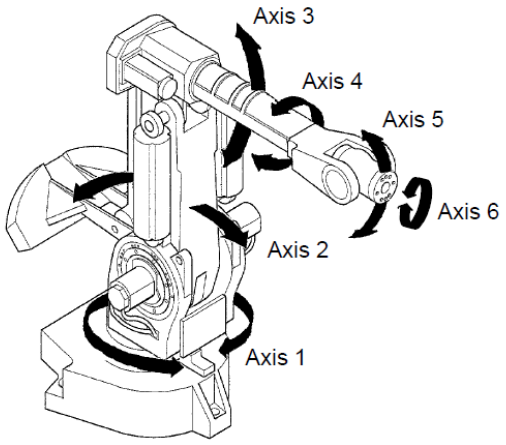

Ilustración 3 Robot 6 ejes

Los controladores hacen uso del sistema operativo ***BaseWare OS***, el cual gestiona todos los aspectos del robot ya sea el movimiento del manipulador, la comunicación con los periféricos o la ejecución del programa. El armario del controlador (Ilustración 4) tiene un peso de 240kg y un volumen de 950x850x540 [15]. Estos armarios están situados delante de sus respectivos manipuladores en el exterior de las jaulas.

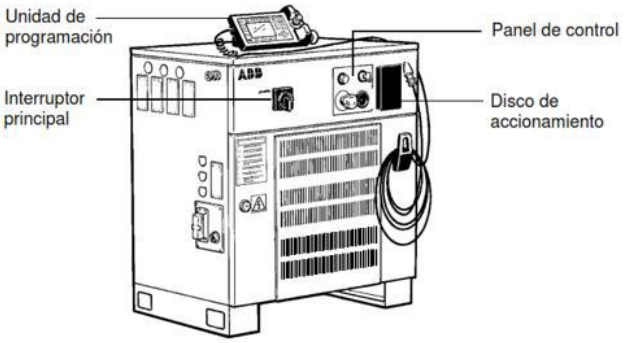

Ilustración 4 Armario controlador

En cuanto a la seguridad del área de trabajo se partirá del estudio realizado en el Trabajo Fin de Grado de Sergio Lorente [18]. En él se indica que, para evitar choques, el movimiento de los ejes de los robots estará limitado. El primer eje estará limitado físicamente mediante un dispositivo hardware, **anillo *Balluf***, y en los demás ejes se limitarán vía software.

- El eje 1 está limitado de 18^{o} a 79^{o}.
- El eje 2 está limitado de 30^{o} a 55^{o}
- El eje 3 está limitado de 20^{o} a 40^{o}

Las jaulas están equipadas con sensores para comprobar el estado de las puertas (Ilustración 5). De esta forma saltará la parada de emergencia si los finales de carrera detectan la apertura de la puerta de la jaula.

Ilustración 5 Sensor de apertura de puerta

Los robots están equipados con pinzas neumáticas paralelas de gran apertura **MHL2-16D2 (SMC)** [16] (Ilustración 6) a las que se les han añadido dos dedos que permiten la manipulación de objetos cúbicos.

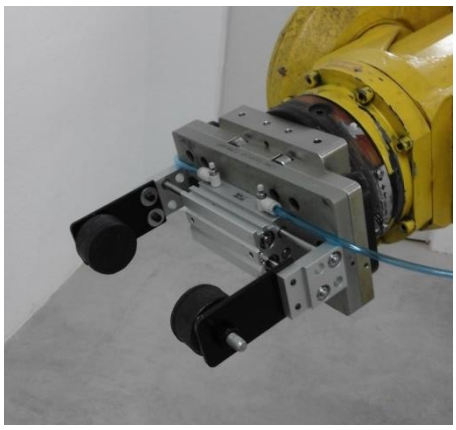

Ilustración 6 Herramienta MHL2-16D2

2.4. Aprendizaje

Como método para recabar información del trabajo se hizo uso de los video-tutoriales [9] dispuestos por ABB para aprender a manejar ***RobotStudio***. Como en ciertos aspectos eran básicos, se buscaron tutoriales alternativos que fueran más extensos en la plataforma "***Youtube***" [2 - 8]. Estos tutoriales recogen todos o casi todos los temas tratados más adelante en el desarrollo del trabajo. Esta plataforma es útil ya que muchos profesores de universidad u otros profesionales, que tienen conocimientos básicos del programa, incluyen tutoriales para ayudar a la gente que empieza a manejar este tipo de programas. Por otro lado, la larga duración de los video-tutoriales y el hecho de que cada uno se centra en la explicación de algún entorno o aplicación en concreto, hacen necesario disponer de documentación para poder centrar al usuario en temas específicos.

Una vez que el manejo del programa fue más ágil, los manuales [10 - 16] y la ayuda [17] del programa permitieron ampliar los conocimientos del mismo en las herramientas más complejas (por ejemplo: crear una herramienta, definir señales, etc). Con estos recursos se desarrolló una metodología para la correcta implementación de los escenarios del TFG. Este método se desglosa en una serie pasos que se desarrollan en el apartado siguiente junto con la documentación de la creación de escenarios.

3. Descripción de escenarios

A continuación, Vamos a proceder a la descripción de los diferentes escenarios simulados en este TFG.

3.1. Escenario 1

En este escenario se expone el montado de una pieza dividida en dos mitades y unida por un macho. El objetivo del escenario es aprender el manejo e implementación de herramientas básicas en ***RobotStudio***.

A modo de guía rápida, a continuación se muestra una lista de los elementos más importantes que se describen en este escenario: creación de una estación; adición de herramientas pre-definidas; creación de objetos mediante formas básicas y la función *Restar*; creación de sistemas robóticos; definición de Entradas/Salidas; configuración de herramientas para permitir el agarre de objetos; uso de conjuntos de colisión; programación basada en trayectorias; opciones de simulación del escenario; y compartición de archivos "*Pack & Go*".

3.1.1. Modelado de la estación

Se crea un estación vacía e introducimos el robot **IRB 140_6_81_C_03** en la posición (0,0,0), al que se le conecta la herramienta ***Schunk Gripper***. Esta herramienta ha sido descargada como librería a través de la pestaña de ***Complementos de Robot Studio***. Para cargar la herramienta y el robot al escenario los importamos a través de las **Bibliotecas ABB** e **Importar Bibliotecas**, respectivamente (Ilustraciones 7 y 8).

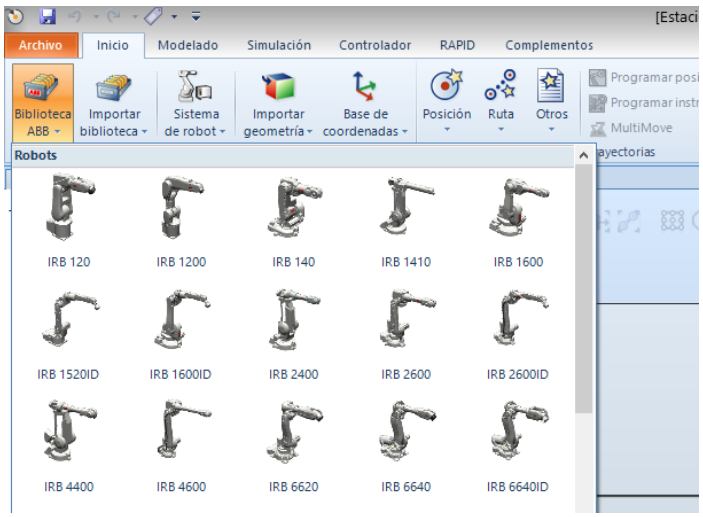

Ilustración 7 Selección biblioteca ABB

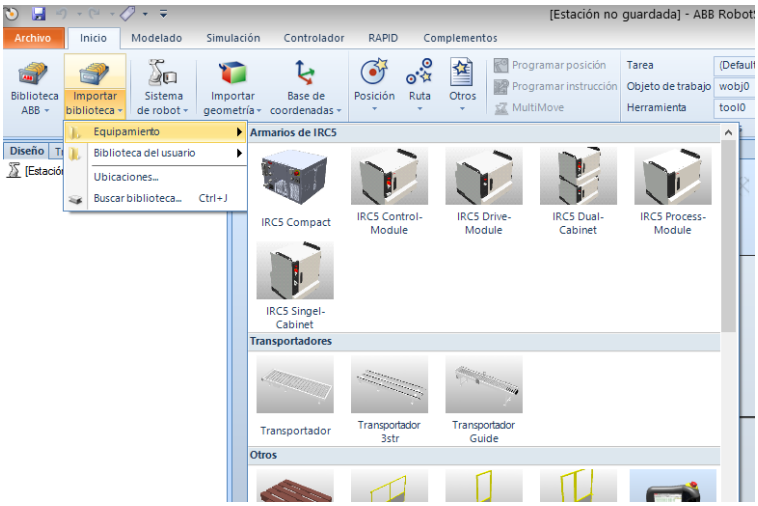

Ilustración 8 Selección biblioteca equipamiento

Para conectar la herramienta al robot, seleccionamos la herramienta y con 'clic derecho' en el componente, se selecciona la opción **Conectar a** (Ilustración 9). En esta elegimos al robot y luego pulsamos sobre actualizar posición. Como puntualización, esta opción no solo permite unir la herramienta a nuestro robot, sino que podemos conectar dos objetos según nuestras necesidades.

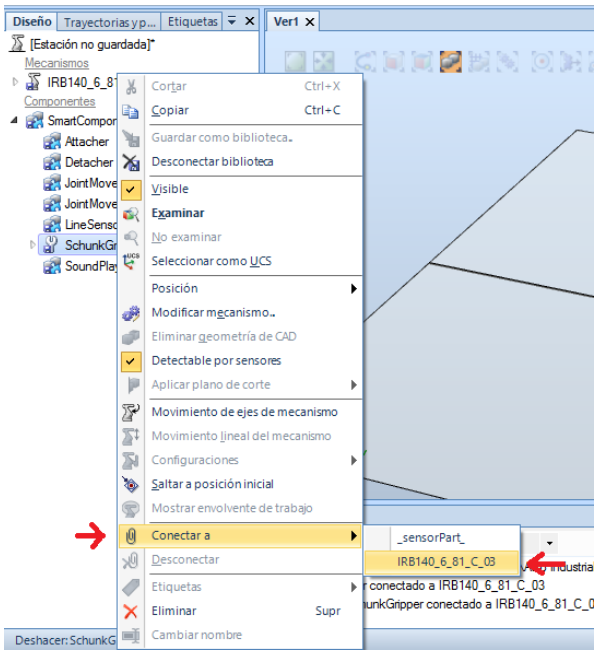

Ilustración 9 Conectar herramienta

Con las herramientas de modelado **Crear sólido** se crea la pirámide de 4 lados(con altura de 500mm y de centro a lado 250mm) y la pieza central (50x50x250mm) como formas geométricas sencillas. Con un cubo mayor que la pirámide se realiza la función **Restar (**Ilustración 10) para obtener las dos mitades, las cuales con el macho y la misma función **Restar** darán forma a la abertura de la unión. Para esto aplicamos la función resta de la pestaña de modelado y **restamos la pirámide con el cubo**. La Ilustración 11 permite visualizar el proceso de creación de la pieza.

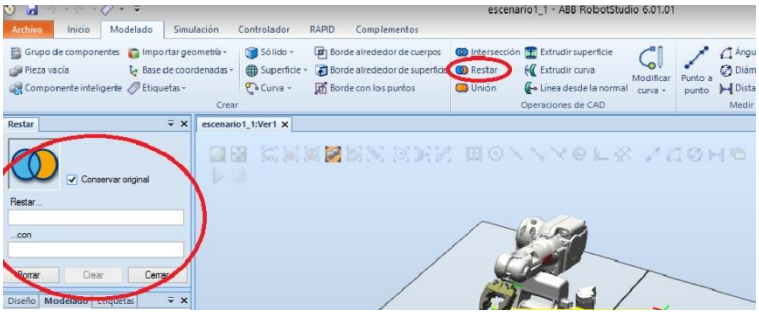

Ilustración 10 Herramienta Restar Sólidos

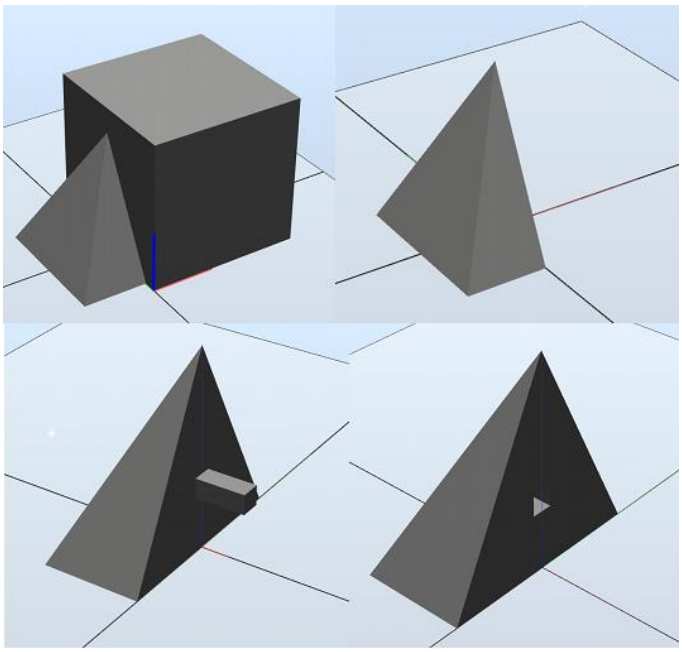

Ilustración 11 Secuencia en la creación de la pieza con la herramienta Resta

Como peculiaridad se destaca la necesidad de dejar una zona de agarre en el extremo de la mitad a unir para facilitar el trabajo del robot. La pieza es de 35x100x100mm y se colocará el centro de una de sus caras laterales en la punta de la pirámide (Ilustración 12).

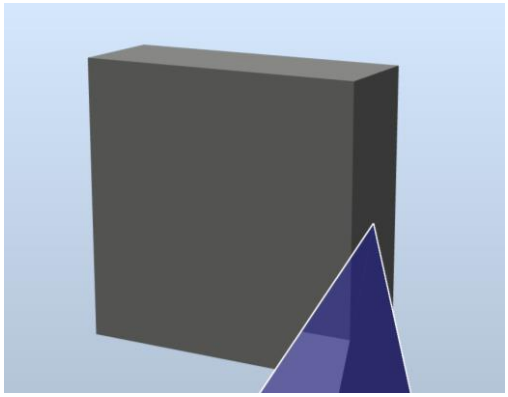

Ilustración 12 Detalle de la unión de la asidera

La distribución se dispone con el robot en el centro del escenario. Una mitad y la pieza central a los lados del robot, situados en las coordenadas del escenario (50,500,0) y (25,-775,0) respectivamente; y la mitad principal al frente del robot para el ensamblado en (600,0,0) (Ilustración 13).

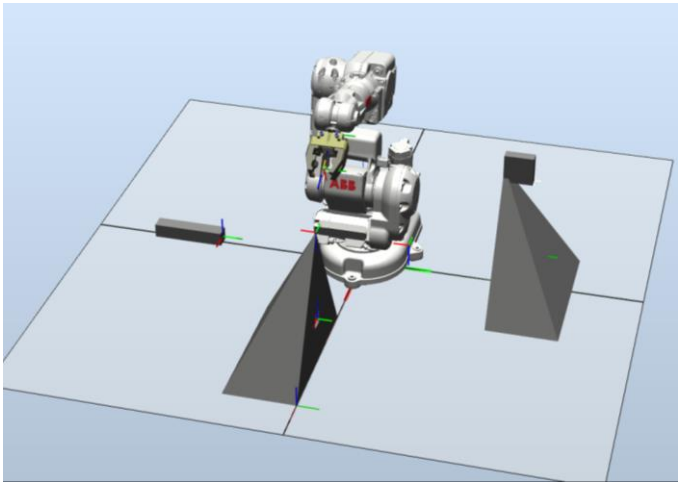

Ilustración 13 Escenario1

3.1.2. Simulación

Se crea el **Sistema Robot desde diseño**, en **Inicio**, con la versión ***RobotWare 5.61.05_5004*.** En la siguiente pantalla seleccionamos el robot implicado y pasamos a la generación del sistema compuesto por el robot y su controlador. Una vez finalizada le damos a **Finalizar**.

- Asignación de entradas y salidas

Con el sistema creado dentro de la pestaña **Controlador**, en el sistema deseado, seleccionamos **Configuración** (Ilustración 14) para crear los buses y unidades necesarias. Estos elementos se pueden crear en la respectiva pestaña usando el 'clic derecho' del ratón. Se necesita crear un tipo de unidad que sea virtual. De esta forma tanto el bus como la unidad nueva las crearemos como tipo "*Virtual*". Con el nuevo bus, llamado por comodidad "*Virtual1*", podemos crear la nueva unidad "*Comunicaciones*" para asignar a esta las nuevas señales. En esta estación únicamente hará falta crear una señal digital de salida para el control de la pinza "*actpinza*".

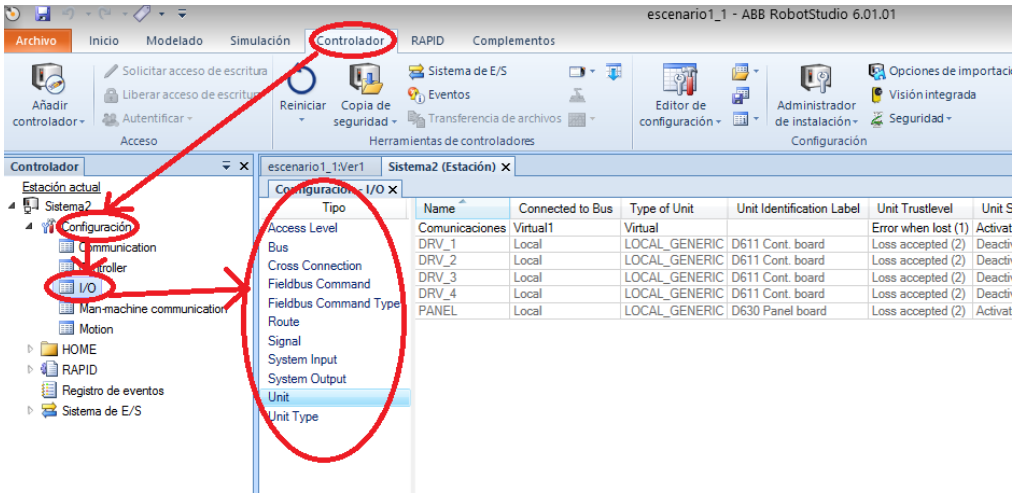

Ilustración 14 Entradas/Salidas del Controlador

Después de los cambios, con **Reiniciar** se hará un reinicio en caliente para que el sistema cargue los datos que hayamos agregado.

En la pestaña **Simulación** en el **Gestor de eventos** (Ilustración 15) creamos dos eventos para que la pinza agarre los objetos del escenario. Uno conectará los objetos y el otro los desconectará según la activación de la señal "*actpinza*".

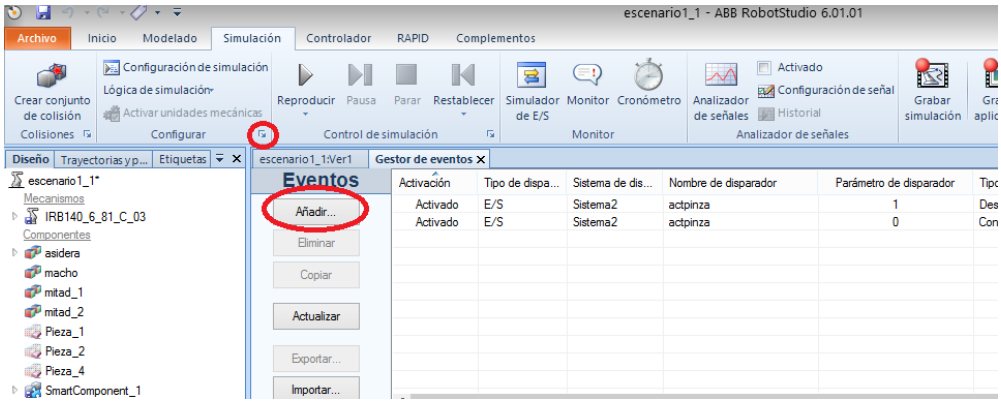

Ilustración 15 Pestaña de Eventos

Para ello con **Añadir evento** nos saltará la ventana de creación de eventos. En esta ventana se puede seleccionar si se dará el evento al activarse o desactivarse una señal o por simulación (este permite, como caso especial, elegir el tiempo de simulación en que disparará el evento); así como el tipo de disparo por el cambio de una señal, conexión de una señal o colisión. Puesto que se necesita una señal que cambie, se selecciona la opción de **cambio de señales E/S**. En la siguiente ventana se elige la señal que se ha creado para que dispare el evento y la condición de disparo ("***TRUE***" o "***FALSE***"). En la ventana siguiente, que es común para todos los eventos, se elige el evento a realizar. Se escogerá el de conectar o desconectar según la condición de la señal para abrir o cerrar la pinza. Para la siguiente ventana se selecciona que el objeto más próximo al **TCP** (*Tool Center Point*) se conecte al ***Schunk Gripper*** o que cualquier

objeto se desconecte del ***Schunk Gripper***, dependiendo del evento a crear. En el caso de conectar la opción **Conservar posición** servirá para que se mantenga conectado en la posición en que lo agarre la pinza.

Se ha utilizado este método de simular una pinza debido a su simplicidad, ya que facilita las simulaciones. Como inconveniente tiene que se conecta cualquier objeto aunque no esté en contacto con el **TCP**. Esto, por otra parte, no debería ser un problema si la tarea está bien implementada. Para evitarse esto se debería disponer en la pinza de un sensor, que en la lógica de la herramienta indique cuál es el objeto a conectar. Esto obligaría a colocar una función de espera en el programa ***RAPID*** para esperar a que termine de realizar la acción correctamente, "*WaitTime 0.5;*". De esta forma, hasta que no finalice el movimiento de cerrar la pinza, la pieza no se conectará. Este método se ha utilizado en el apartado 4.3.2, debido a que no se podía aplicar el método basado en eventos explicado anteriormente.

- Lógica de estación

Ahora se pasa a conectar las señales de los componentes de la estación con el controlador. Para ello se abre la opción **Lógica de estación** en la pestaña **Simulación**. A continuación, vamos a señales y conexiones para añadir las conexiones necesarias para nuestra estación (Ilustración 16).

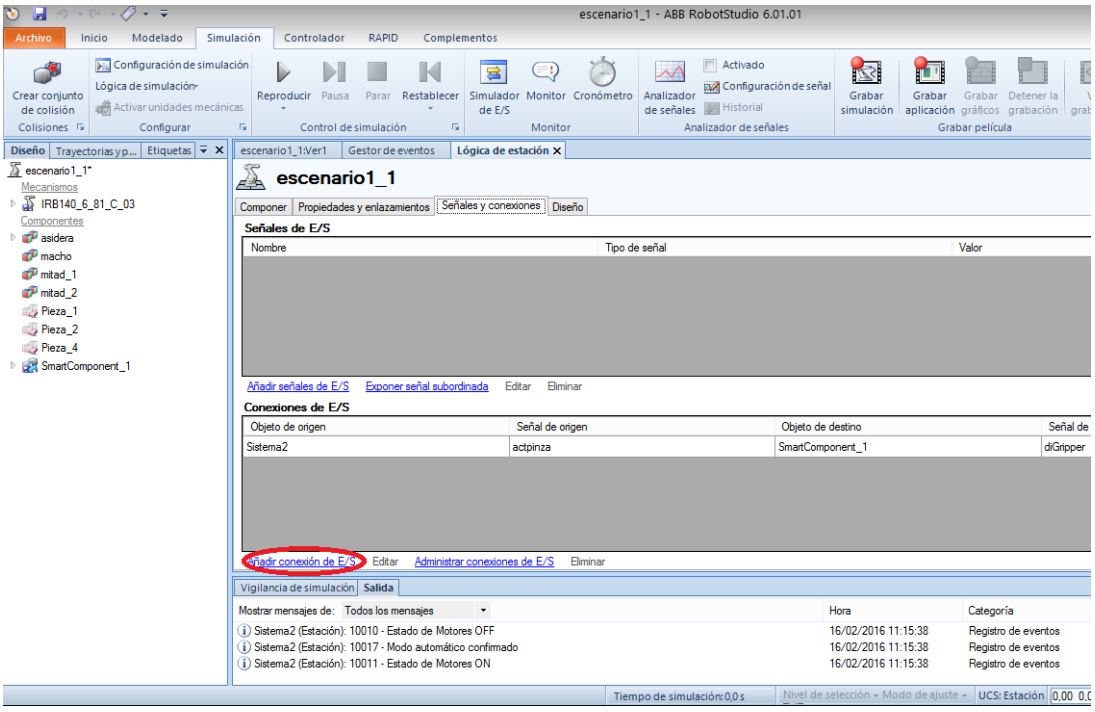

Ilustración 16 Conexiones E/S entre elementos del escenario

En la pestaña de **Diseño** podemos comprobar cómo queda la conexión de las señales y cambiarla si lo deseamos (Ilustración 17).

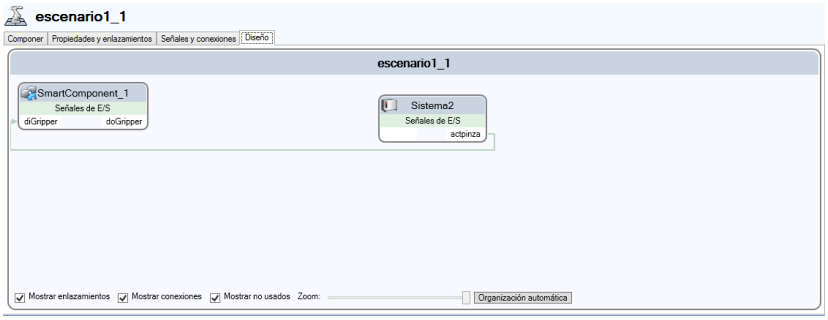

Ilustración 17 Lógica de Estación del Escenario1

- Creación conjunto de colisiones

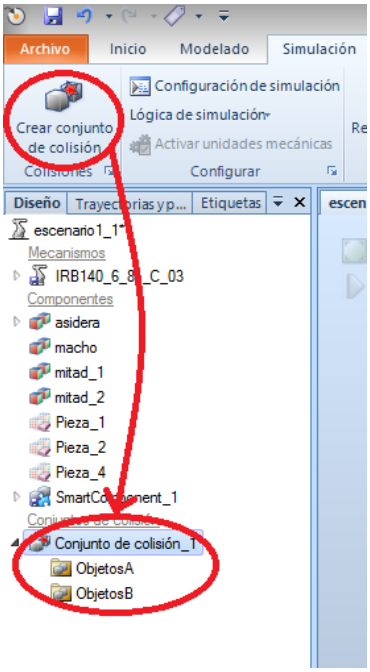

Ilustración 18 Creación Conjuntos de colisión

Como su nombre indica, los conjuntos de colisiones nos ayudarán a identificar las colisiones que se produzcan una vez estemos comprobando el programa del robot. Para crear el conjunto de este escenario seleccionamos **Crear conjunto de colisiones** (Ilustración 18). Al comprobar si el robot o la herramienta colisionan con algún objeto del escenario introduciremos a estos en el **grupo A** y los objetos que queremos comprobar en el **grupo B**. Haciendo 'clic derecho' sobre el conjunto de colisión que se ha creado y seleccionando **Modificar conjunto de colisión** (Ilustración 19) aparecerá una ventana en la que se puede configurar la distancia de casi-colisión, los colores con los que se representa la colisión y la casi-colisión, marcar la colisión, detectar colisiones con objetos invisibles, y la activación o desactivación del conjunto.

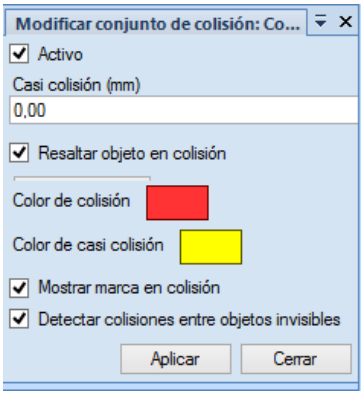

Ilustración 19 Ventana modificación de los Conjuntos de colisión

En cuanto a acceder a las opciones de la herramienta de simulación de colisiones, se utiliza la pestaña pequeña que está debajo de **Crear conjunto de colisiones** (Ilustración 20). En ella podremos cambiar las opciones de cuándo se realiza la detección de las colisiones y cómo se indican las mismas. Normalmente las detectaremos siempre, ya que los escenarios son sencillos de seguir si hay colisiones. Con que registre las colisiones en la ventana de salida será suficiente.

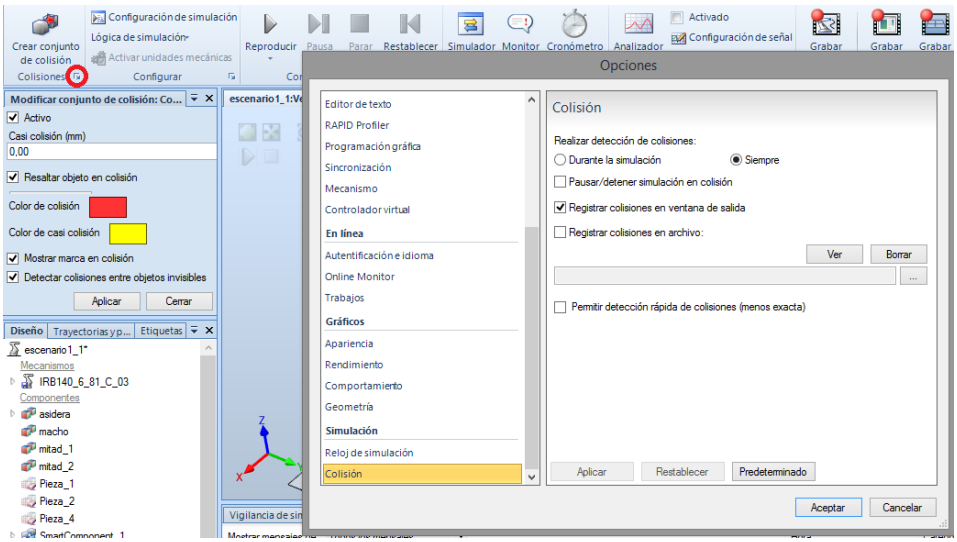

Ilustración 20 Ventana opciones de los Conjuntos de colisión

- Creación de trayectorias

Lo primero es crear un objeto de trabajo en la pantalla **Inicio** pestaña **Otros** en **Crear objeto de trabajo**. Seleccionando las piezas creamos para cada una un objeto de trabajo. Al lado de la pantalla de crear objetos tenemos la pantalla para crear los puntos seleccionando herramienta, objeto de trabajo y tarea (Ilustración 21). Los objetos de trabajo se pueden encontrar en la calibración del sistema una vez cargados.

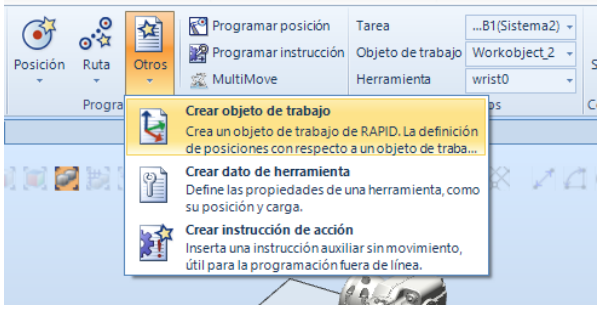

Ilustración 21 Creación Objetos de trabajo

Trayectoria coger pieza central (Path_10):

Se crea un punto en la posición de reposo, *target_10*; este punto lo usaré siempre al principio y al final de cada trayectoria para guardar un orden de donde empieza y de donde acaba el robot. El punto de agarre, *target_30 [0,-50,25]*, se situó a 50mm del lado más próximo al robot. El punto de aproximación, *target_60 [0,-50,125]*, se creó a 100mm sobre el *target_30*. Ambos puntos tienen una orientación para la incidencia perpendicular de la herramienta, como se puede observar en la trayectoria en amarillo en la Ilustración 22.

La secuencia queda: reposo → punto aproximación → punto agarre → señal cerrar pinza → punto aproximación → reposo.

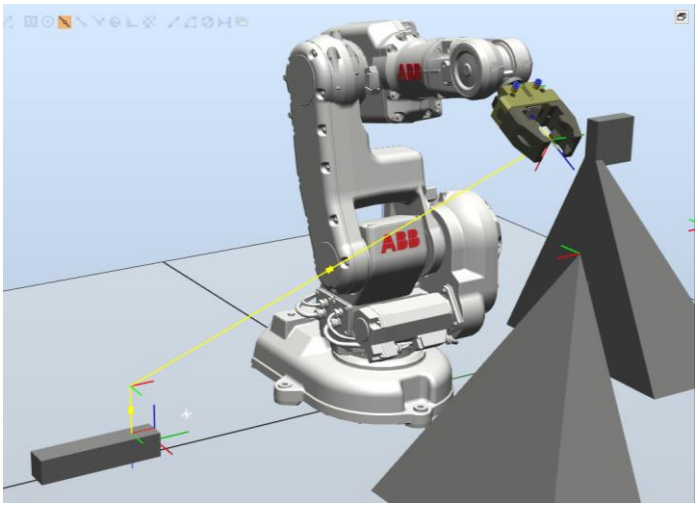

Ilustración 22 Trayectoria Path10

Trayectoria colocar pieza central (Path_20):

Se crea un punto auxiliar, *target_90 [-250,300,550]*, para evitar colisiones y facilitar el alcance al punto de aproximación, *target_80 [-250,300,191.667]*. Por último el punto donde la pieza está colocada estará en *target_70 [-250,75,191.667]*. Las orientaciones serán las mismas que las del anterior '*Path*' para que la pieza quede perpendicular al orificio donde se situará. Trayectoria en amarillo en la Ilustración 23.

La secuencia queda: reposo → punto auxiliar → punto aproximación → punto pieza colocada → señal abrir pinza → punto auxiliar → reposo.

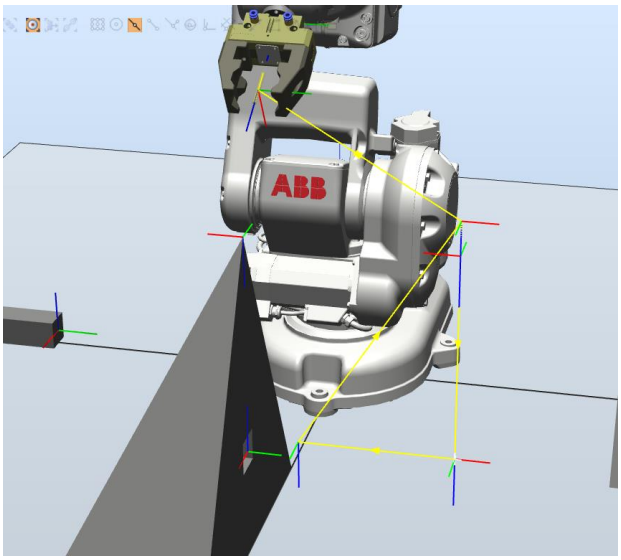

Ilustración 23 Trayectoria Path20

Trayectoria colocación pieza mitad (Path_30):

Se crea un punto de aproximación, *target_140 [0,-75,413.333]*, a la pieza mitad y otro de agarre, *target_130 [0,-75,308.333]*, sobre la asidera. Hace falta un punto auxiliar, *target_180 [550,-325,393.333]*, para la correcta colocación con el punto de aproximación, *target_120 [-250,300,500]*, de la pirámide sin que colisione con la pieza central. El punto donde soltará la pieza será el *target_150 [550,-575,308.333]*. Las orientaciones estarán dispuestas de forma que las caras con los orificios estén paralelas. Trayectoria en amarillo en la Ilustración 24.

La secuencia queda: reposo → punto aproximación → punto agarre → señal cerrar pinza → punto auxiliar → punto aproximación → punto saltar pieza → señal abrir pinza → reposo.

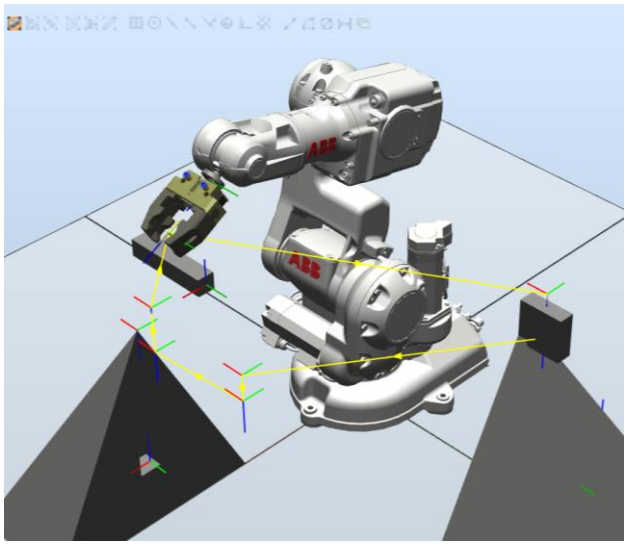

Ilustración 24 Trayectoria Path30

- Ficheros Rapid

Una vez tenemos las trayectorias y el escenario listos, en ***RAPID*** se tendrá que seleccionar **Sincronizar con estación** (Ilustración 25). De esta forma, se cargan automáticamente, trayectorias, calibraciones de la herramienta y los objetos de trabajo que hayamos creado previamente. Una vez terminado el programa RAPID (ver Anexo 5.1) lo cargamos en la estación mediante **Sincronizar con RAPID**.

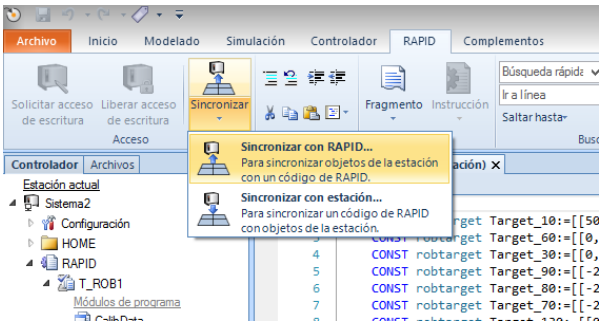

Ilustración 25 Pestaña sincronización del Controlador

3.1.3. Comprobación

Estando ya todo listo para la comprobación del funcionamiento, controlaremos la reproducción de la simulación con las opciones que aparecen en **Simulación** (Ilustración 26).

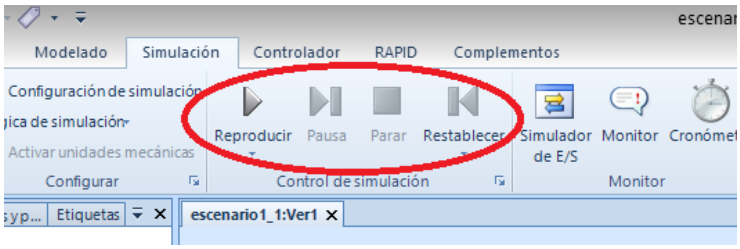

Ilustración 26 Pestañas control de simulación

Restablecer permite devolver la estación al estado inicial. La herramienta permite guardar el estado actual del escenario implementado, lo que hace posible almacenar varios estados de forma que podamos comprobar distintas situaciones por separado durante la implementación. Para guardar un estado, en **Restablecer** en la opción **Guardar estado actual** (Ilustración 27) le damos un nombre y descripción e incluimos los componentes deseados en la ventana de la Ilustración 28.

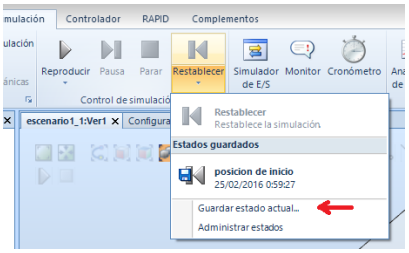

Ilustración 27 Pestaña guardado del estado de la estación

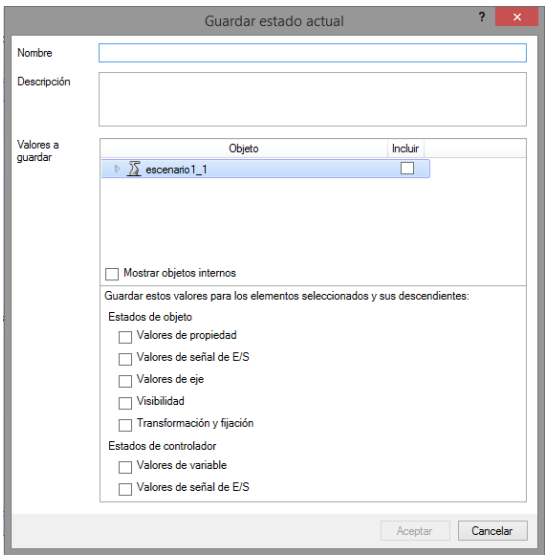

Ilustración 28 Ventana guardado del estado de la estación

En algunos escenarios es necesario comprobar las señales de la estación o necesitaremos activarlas manualmente. Para ello abrimos el simulador de señales con la pestaña de **Simulador de señales E/S** en **Simulación** (Ilustración 29). De esta forma solo necesitaríamos buscar la señal o señales deseadas entre los componentes del escenario. La ventana nos indicará el estado de las señales del componente elegido. Haciendo 'clic izquierdo' sobre ellas, se tendrá la opción de fijarlas a 1 o 0.

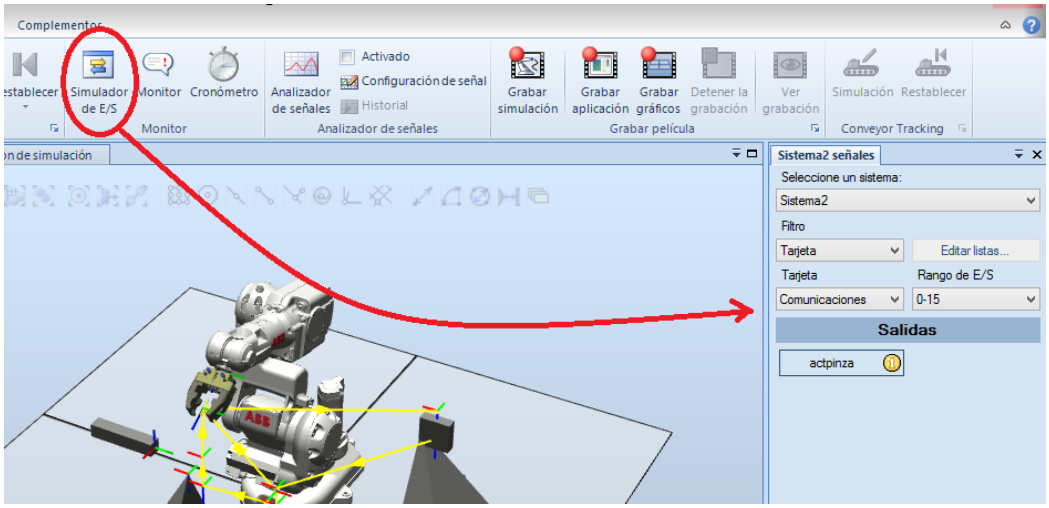

Ilustración 29 Pestaña Simulador de E/S

Como nota importante, si queremos que la simulación funcione en ciclo continuo o un solo ciclo se necesitará activar a través de la **Configuración de la simulación**. Esta ventana (Ilustración 30) también nos ofrece la opción de elegir el estado con el que iniciamos la simulación, así como la elección de otros escenarios abiertos que queramos simular.

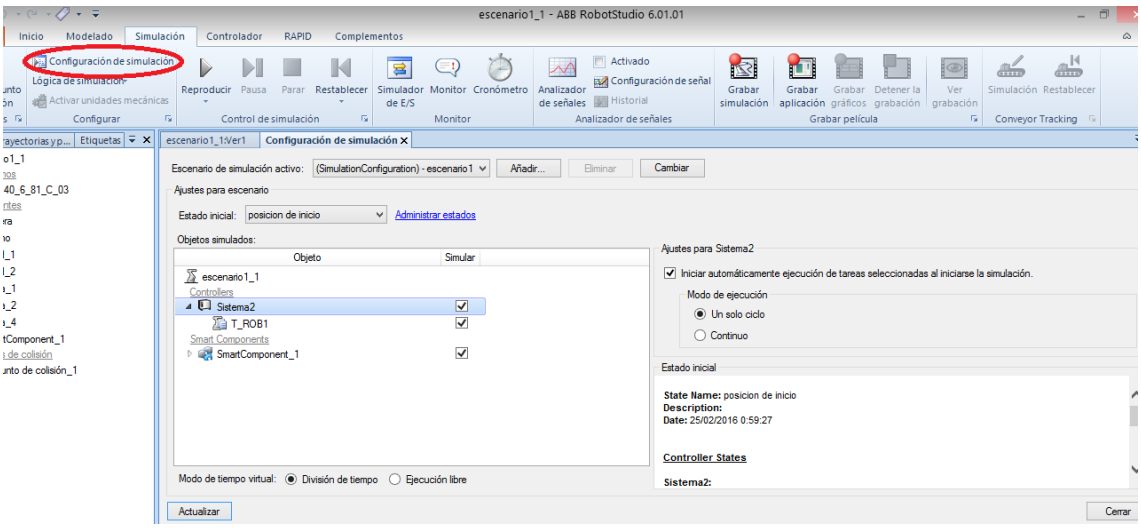

Ilustración 30 Ventana configuración de la simulación

Para finalizar le damos a **Reproducir** para iniciar la simulación. Se observa que la tarea se lleva a cabo correctamente y que no se producen colisiones. El simulador indicará que se producen choques cuando la pinza coge los objetos, pero esto no es inconveniente ya que la simulación recogerá casos como este debido a que es inevitable que la herramienta toque el cuerpo del objeto, esto se dará en los otros escenarios por motivos similares.

Los casos más probables de fallo se dan cuando no permitimos al controlador virtual cargarse correctamente durante la apertura del archivo, causando fallos en la ejecución. Al hacer una restauración del escenario que no es adecuada para el estado necesario o los objetos no están correctamente situados en ese estado pueden darse problemas de funcionamiento con el evento de conectar a la pinza. Estos y otros problemas derivados de ellos se pueden solucionar fácilmente asegurándonos de que los controladores virtuales están correctamente iniciados y guardando previamente el estado del escenario antes de la comprobación para que al restaurar tengamos siempre un estado inicial común entre comprobación y comprobación.

3.1.4. Pack and go

Ya finalizado el escenario, para guardar el conjunto del escenario con todas sus librerías y elementos en un mismo archivo compreso, se utilizará la herramienta ***Pack and Go*** de ***RobotStudio*** (Ilustración 31). En **Archivo**, en la pestaña **Compartir** están las opciones para comprimir y abrir los archivos de la estación.

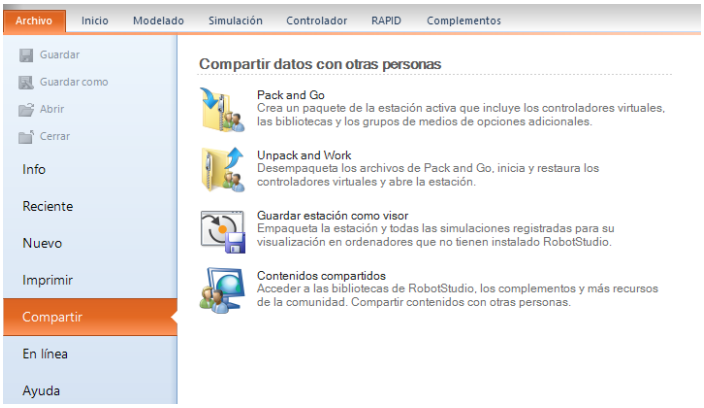

Ilustración 31 Pestaña herramienta Pack and Go

La opción ***Pack and go*** pedirá el destino para crear el archivo comprimido, de tipo "*.rspag*", con todo lo necesario del escenario. Mientras que la opción ***Unpack and Work*** pide el archivo compreso y el destino para descomprimirlo y poder trabajar con el archivo propio de ***RobotStudio***, de tipo estación *".rsstn*". Es importante tener en cuenta que el ordenador que abra este archivo tenga la misma versión de ***RobotStudio*** y ***RobotWare*** que el ordenador que lo comprimió para evitar problemas de carga y de ejecución.

Este procedimiento es idéntico para todos los escenarios de este trabajo u otros que se deseen crear.

3.2. Escenario 2

En este escenario se expone un proceso de tratado y transporte de dos piezas; una cilíndrica y otra cúbica. El escenario dispone de tres robots. El robot central transporta las piezas desde el palé de entrada hasta las mesas de tratado, donde los respectivos robots realizan una tarea de soldado específica para cada pieza. Cuando la tarea se completa el robot de transporte recoge las piezas y las lleva a las mesas de finalizado. El robot central irá equipado con una ventosa y los otros dos con una herramienta de soldado. Las mesas de tratado disponen de sensores para indicar la presencia de piezas para el tratado. Los tres robots son controlados mediante el mismo controlador que, a su vez, se encarga de monitorizar y activar los sensores.

Los objetivos son aprender a desarrollar un escenario con varios robots controlados por un único controlador, la implementación e intercalación de las distintas tareas, la implementación de sensores y la creación de herramientas simuladas.

A modo de guía rápida, en este escenario se explican los siguientes elementos de ***RobotStudio***: uso de varios robots en la estación, creación de herramientas personalizadas, definición de componentes inteligentes y de sensores, generación de trayectorias automáticas, y programación mediante el uso de variables persistentes.

3.2.1. Modelado estación

Se crea una estación vacía y se introduce el robot **IRB260_30_150__02** en la posición (0,0,0). Este robot se encarga del transporte de las piezas. Además se introducen dos robots **IRB1600ID_4_150__03**, en la posición (0,2200,0) y en (0,-2200,0). Estos dos robots se ocuparán del tratado de las piezas. A los robots **IRB1600** se les conecta la herramienta **PKI_500_di_M2001** de la biblioteca de componentes. Esta herramienta emulará un soldador de arco conectado al robot. El robot **IRB260** estará equipado con una herramienta de tipo ventosa. Para ello, crearemos una nueva herramienta simulada a través de la pestaña **Crear herramienta** de la pestaña **Modelado**, como se explica a continuación.

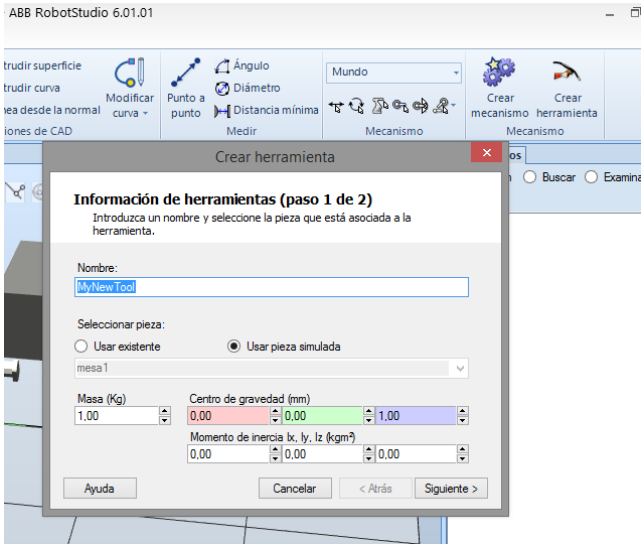

Ilustración 32 Ventana creación Herramienta simulada

En la ventana **Crear herramienta** introduciremos los datos de la herramienta (Ilustración 32). Como es un escenario de aprendizaje dejaremos los datos por defecto del paso uno, ya que no será necesario cambiarlos. En el segundo paso se crea el **TCP** en la posición (0,0,130) y se le llama "*ventosa*" para diferenciarlo. Con la flecha se introduce en la lista de **TCPs** (Ilustración 33). Al pulsar **Terminado** la herramienta se genera en el escenario.

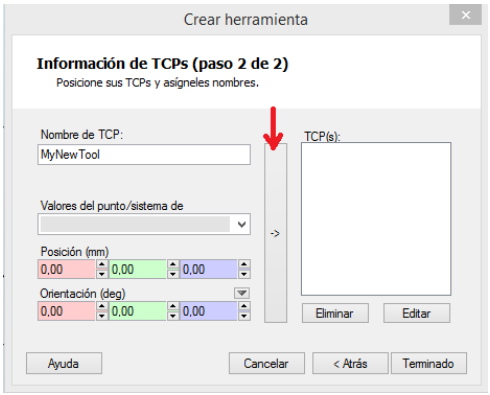

Ilustración 33 Ventana creación TCP Herramienta simulada

En cuanto a los elementos del escenario se crean con **Crear sólido** la pieza cubo (250x250x250mm), la pieza cilindro (diámetro de 200mm y altura de 300) las mesas 1 y 2 (500x500x500) y las mesas 3 y 4 (1000x500x500). El palé lo cargamos de la biblioteca de equipamiento.

Para los sensores habrá que crear un **Componente inteligente** y en la ventana seleccionar **Añadir componente**. El componente para los sensores será ***LineSensor*** en la pestaña de **Sensores** (Ilustración 34). De forma alternativa, se podrían haber añadido los sensores a través de la pantalla de lógica de estación directamente, sin estar asociados a un **Componente inteligente**.

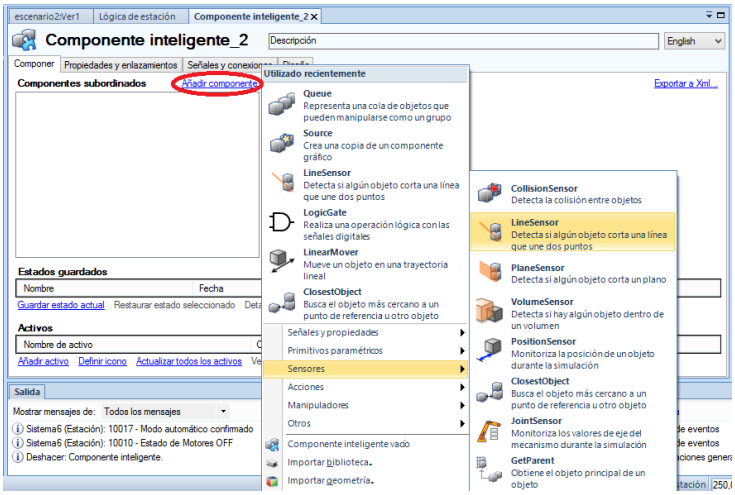

Ilustración 34 Pestaña de componentes para el Componente inteligente

Para configurar los sensore,s hacemos doble ‘clic izquierdo’ sobre el componente, abriendo se esta manera una ventana que nos permite modificar las propiedades del sensor (Ilustración 35). Se modifica para que se inicie en la posición (0,0,0) y termine en (500,0,0), configurando de esta manera los puntos de la línea de detección del sensor. El resto de datos no es necesario cambiarlos, pero la opción del radio del sensor es interesante para simular componentes reales. ***SensedPart*** saca el cuerpo que está detectando el sensor; esto es útil para aplicaciones lógicas que realizan una acción sobre un cuerpo, como es el caso de conectar un objeto a una herramienta. La detección de los cuerpos va con relación a la proximidad al punto de inicio. Esto hace que el más próximo aparece primero y en las conexiones lógicas será el dato que se transmita.

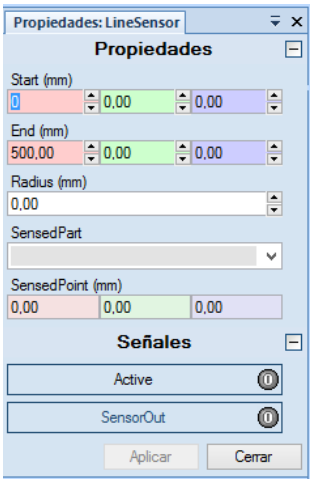

Ilustración 35 Ventana propiedades Sensor

Una vez colocados los robots la distribución de los elementos quedaría: *mesa 1* centrada en su base en (0,1200,0), *mesa 2* centrada en su base en (0,-1200,0), *mesa 3* centrada en su base en (-1500,350,0), *mesa 4* centrada en su base en (-1500,-350,0), palé centrado en su base aproximadamente en (1800,0,0), la pieza cilindro centrada en su base en aproximadamente (1200,-213,144) y la pieza cubo centrada en su base en aproximadamente (1272,162,144) (Ilustración 36). El cuerpo de palé no permite un mejor ajuste de su posición por lo que él mismo y las piezas de encima se han ajustado con poca precisión. En cuanto a los sensores, el *sensor 1* lo colocaremos detectando el centro de la *mesa 2*, el *sensor 2* detecta el centro de la *mesa 1*, el *sensor 3* detecta la entrada del cubo en el palé y el *sensor 4* la entrada de cilindros. Todos los sensores están a una altura de 50mm sobre la mesa o palé en que operen.

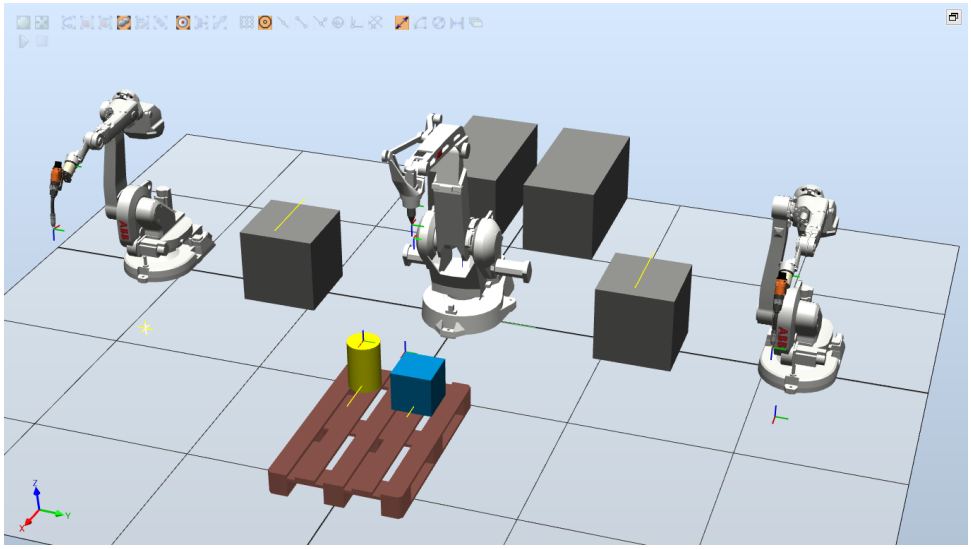

Ilustración 36 Escenario2

3.2.2. Simulación

Se crea el **Sistema Robot desde diseño**, con versión ***RobotWare 5.61.05_5004N***. En este escenario se seleccionan los tres robots para que los controle el mismo controlador. El controlador se encargará de controlar los sensores descritos en la sección anterior. Se usan además conjuntos de colisión para verificar el correcto funcionamiento del programa. Cada robot se implementa asociado a una tarea o '*task*' dentro del controlador. Cada tarea está sincronizada con otra de forma que se inician al finalizar la anterior.

Dado que la implementación de Entradas/Salidas, conjuntos de colisión y lógica de estación apenas difieren de lo mostrado en el escenario 1, estos se pueden encontrar con una información más completa en el Anexo 4.

- Creación de trayectorias

Se crea para cada robot un punto de reposo en la posición en la que es cargado al escenario, *target_10*. Como en el escenario anterior, se usará para marcar el punto de inicio y fin de la trayectoria. Como objeto de trabajo creamos uno en el cilindro y otro en el cubo.

Tarea transporte T_ROB1 (IRB260):

Este robot de transporte, está encargado de llevar el cilindro a la mesa 2 y a la mesa 4, y de llevar el cubo a las mesas 1 y 3. En las Ilustraciones 36 – 39 se muestran estas trayectorias en amarillo. Además, a continuación se explica en detalle la trayectoria para llevar el cilindro a la mesa 2. El resto de trayectorias, que son muy similares a la descrita anteriormente, se encuentran descritas en detalle en el Anexo 4.3.

Trayectoria llevar cilindro a mesa2 (Path_10):

Se crean dos puntos de aproximación, *target_110 [0,0,256]* para coger la pieza de entrada, y otro para dejarla en la mesa, *target_120 [-1186.146,-987.434,656]*. La ventosa cogerá el cilindro en el punto central de la base superior, *target_30 [-1186.146,-987.434,356]*. Este cilindro lo soltará en la mesa sobre el punto *target_20 [0,0,0]*. La orientación de los puntos será perpendicular a la superficie de la pieza con la misma orientación que la herramienta. Además, variará noventa grados de mesa a mesa para mantener la orientación de la pieza con el movimiento del robot.

La secuencia queda: reposo → punto aproximación → punto agarre → señal activar ventosa → punto aproximación → punto aproximación → punto soltar pieza → señal desactivar ventosa → punto aproximación → reposo.

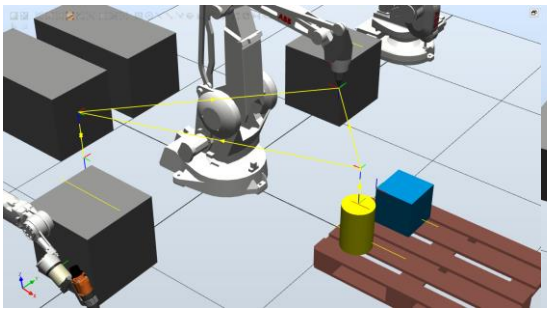

Ilustración 37 Path10 T_ROB1 (llevar cilindro a mesa4)

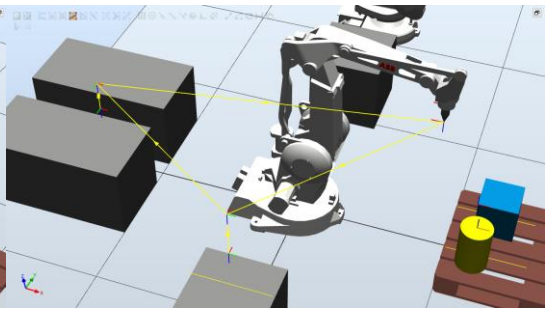

Ilustración 38 Path20 T_ROB1 (llevar cilindro a mesa4)

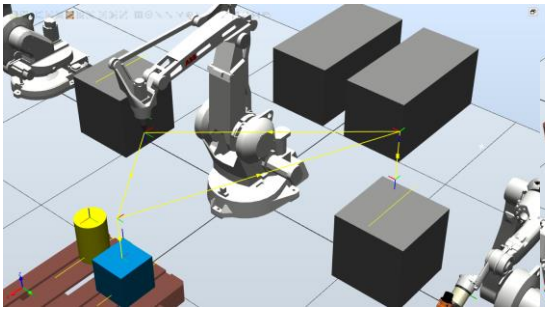

Ilustración 39 Path30 T_ROB1 (llevar cubo a mesa1)

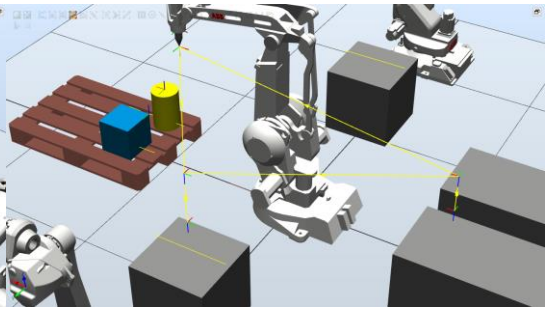

Ilustración 40 Path40 T_ROB1 (llevar cubo a mesa3)

Tarea soldado cilindro T_ROB2 (IRB1600):

Trayectoria soldado cilindro (Path_10):

Se crea un punto de aproximación a la superficie, *target_70 [-1086.146,-987.434,456]*. Para el movimiento circular se crean cuatro puntos; *target_20 [-1086.146,-987.434,356]*, *target_30 [-1186.146,-887.434,356]*, *target_40 [-1286.146,-987.434,356]*, *target_50 [-1186.146,-1087.434,356]* y t*arget_60 [-1086.146,-987.434,356]*. Para ello se hace uso de la creación de **Trayectorias automáticas** (Ilustración 42). En la ventana de trayectorias automáticas (Ilustración 41) se selecciona la aproximación **Circular** y se introducen los datos. Se puede seleccionar la superficie del modelo con el cuadro de **Superficie de referencia**, en lugar de introducir datos donde crear la trayectoria. Las orientaciones de todos los puntos tendrán que alterarse para lograr que la herramienta esté perpendicular a la superficie y no cambie de orientación entre punto y punto. Esto se puede visualizar en la trayectoria en amarillo en la Ilustración 43.

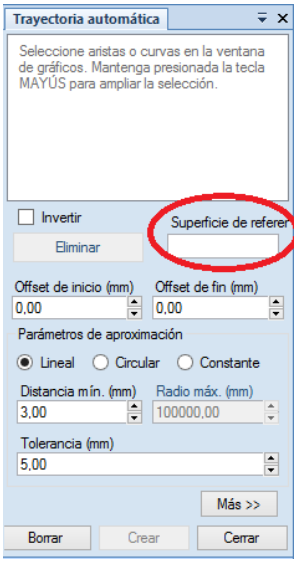

Ilustración 41 Ventana creación de Trayectorias automáticas

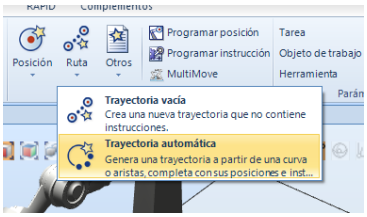

Ilustración 42 Pestaña Trayectorias automáticas

La secuencia queda: reposo → punto aproximación → movimiento circular → punto aproximación → reposo

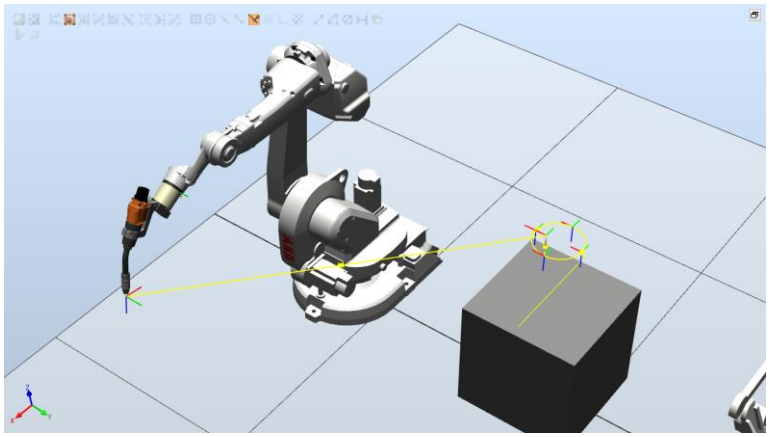

Ilustración 43 Path10 T_ROB2

Tarea soldado cubo T_ROB3 (IRB1600_2):

Trayectoria soldado cilindro (Path_10):

En este caso, se puede hacer como en la tarea anterior, con trayectorias automáticas, o creando cuatro puntos con trayectorias lineales. Los puntos que se crean son: t*arget_20 [-1021.588,-1161.499,355.991]*, *target_30 [-1021.588,-911.499,355.991]*, *target_40 [-1271.588,-911.499,355.991]*, *target_50 [-1271.588,-1161.499,355.991]*. Como punto de aproximación se tiene *target_60 [-1021.588,-1161.499,505.991]*. La orientación de los puntos seguirá el mismo patrón que en la tarea anterior. Esto puede verse en la trayectoria en amarillo en la Ilustración 44.

La secuencia queda: reposo → punto aproximación → puntos esquinas → punto aproximación → reposo.

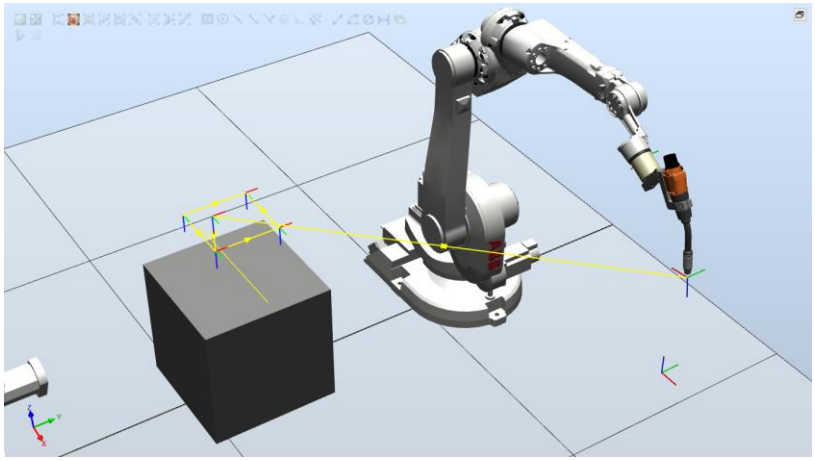

Ilustración 44 Path10 T_ROB3

- Ficheros Rapid

Para la programación de este escenario hay que tener en cuenta que los robots no choquen durante la realización de las tareas. Creando cuatro **variables persistentes**, "*tarea1*", "*tarea2*", "*mesa1libre*", "*mesa2libre*", indicaremos si las mesas están ocupadas y si los robots de tratado han terminado. De este modo, cuando se transporta una pieza a una mesa los robots de tratado no iniciarán la tarea hasta que no esté libre la mesa y no se recogerán las piezas hasta que no finalice la tarea. Los ficheros de cada tarea están recogidos en el Anexo 5.2.

3.2.3. Comprobación

Se cargan los ficheros y en **Configuración de simulación** seleccionamos ejecutar en modo continuo. Reproducimos la simulación y se comprueba que no hay errores ni colisiones en el escenario. La disposición del programa nos permitiría introducir continuamente piezas por el palé siguiendo así el funcionamiento normal del programa.

3.3. Escenario 3

En este escenario se expone un proceso de transporte, tratado y apilado de piezas. Un robot se ocupará de trasportar las piezas a la mesa de tratado y de ésta a la cinta transportadora, mientras otro robot se ocupa de tratar las piezas en la mesa. Una vez se dejan las piezas en la cinta, éstas son transportadas hasta el final de la cinta, donde se encuentra un tercer robot encargado de apilarlas en un palé. Los robots de transporte y apilado estarán equipados con ventosas y el de tratamiento con una herramienta de soldado. Tanto las mesas como la cinta están equipadas con sensores para la detección de piezas. Cada robot está controlado por un controlador propio. El controlador del robot de transporte se ocupa de la gestión del controlador de la mesa de entrada de piezas. El controlador del robot de tratado se encargará de la gestión del sensor de la mesa de tratado. Los sensores de la cinta son gestionados por el controlador del robot de apilado. Para verificar el estado de la tarea entre los robots de transporte y tratado, éstos tendrán sus controladores conectados entre sí a través de señales de Entrada / Salidas digitales. La estación dispondrá de una parada de emergencia conectada a los tres controladores.

El objetivo del escenario es aprender a implementar cintas transportadoras, la implementación de sistemas robot comunicados entre sí y la creación de interrupciones para la parada y posterior arranque del sistema.

A modo de guía rápida, en este escenario se explican los siguientes elementos de **RobotStudio**: sincronización mediante Entradas/Salidas, interrupciones, elementos asociados a componentes inteligentes (Source, Sink, LinearMover, Queue).

3.3.1. Modelado estación

Se crea una estación vacía. Se introduce el robot **IRB1410_5_144__01** en la posición (0,-800,200), otro robot **IRB1410_5_144__01** en la posición (5000,-800,200) y un tercer robot **IRB1520ID_4_150__02** en (0,800,0). Los **IRB1410** se colocan sobre una base de 500x500x200mm. Al robot **IRB1520** se le conecta la herramienta **PKI_500_di_M2001** de la biblioteca de componentes y para los robots **IRB1410** se necesita crear una herramienta simulada con las mismas características que la vetosa del escenario 2. Para el robot de transporte la llamaremos "*herramienta1*" y para el de apilado "*herramienta2*". La Ilustración 45 muestra las herramientas montadas.

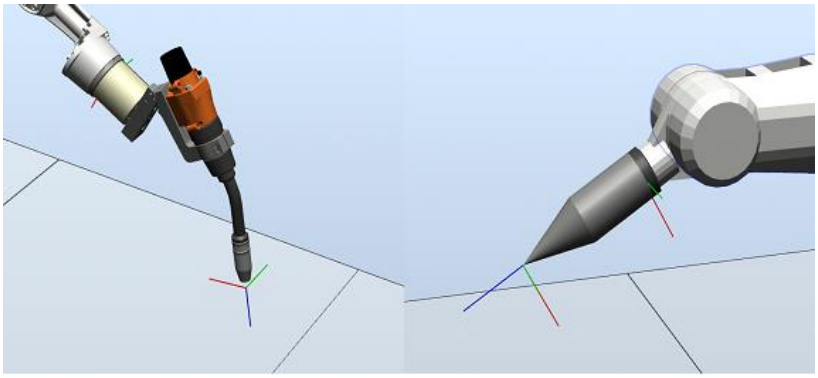

Ilustración 45 Herramienta PKI_500_di_M2001 (izquierda) y Herramienta simulada (derecha)

Con **Crear sólido** se implementan los elementos del escenario, la mesa 1 (1000x500x500mm), la mesa 2 (500x500x500mm), y la pieza a tratar (400x200x200mm). Para la cinta, el modelo se importa de la biblioteca de equipamiento, ***400_guide***, y lo mismo con el palé.

Como en el escenario anterior, se crean los sensores para la *mesa 1* y *mesa 2*. Se pueden introducir los dos sensores en el componente o crearse a parte en el escenario. Para crear más de una pieza para procesar, se necesita un componente inteligente en el que se introduce un componente ***Source*.** En ***Source*** seleccionaremos la pieza deseada para que cree copias. Este componente es preferible activarlo de forma manual en la simulación creando una señal en la estación. Para simular un cambio de palé, se eliminan las piezas mediante el componente ***Sink***. Se introducen en un componente inteligente cuatro sensores y cuatro componentes ***Sink***. Los sensores dispuestos en el palé identificarán los objetos que eliminará ***Sink*** con la señal de cambiar palé (Ilustración 46).

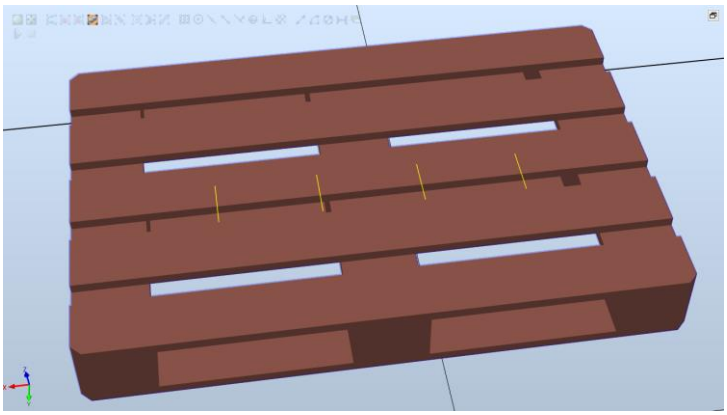

Ilustración 46 Palé con sensores función Sink

En el componente inteligente de la cinta introducimos tres sensores. El sensor 1 al principio de la cinta para detectar las piezas que entran, el sensor 2 al final de la cinta y el sensor 3 a lo largo de la cinta, para saber si hay piezas en ella (Ilustración 47). Se evitarán atascos en la cinta limitando en el programa las piezas en la cinta. Para mover los objetos hacen falta los componentes ***LinearMover*** y ***Queue***. En el ***LinearMover*** se indicará que mueva los objetos que están dentro de la lista ***Queue***, que serán los que detecte el sensor 1. Sólo se introducirán en la lista los objetos detectados por el sensor 1 y que el robot de transporte suelte, indicada por la señal "*piezasuelta*". De esta forma al simular se evita que la cinta arrastre al robot con la pieza. Saldrán del ***Queue*** las piezas detectadas por el sensor 2 y estas se detendrán para ser apiladas. Las conexiones del diseño de la lógica quedan según la Ilustración 48.

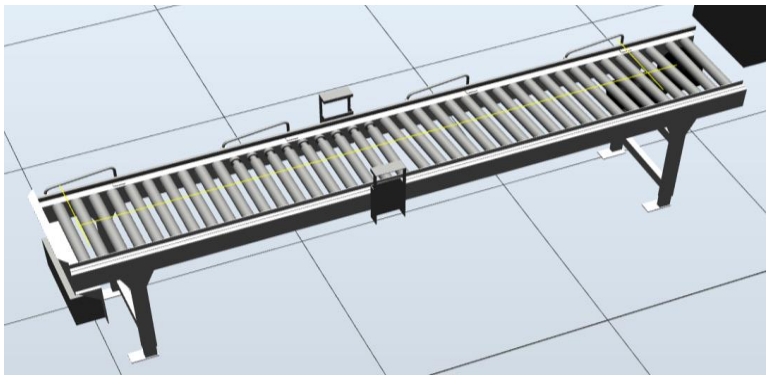

Ilustración 47 Cinta

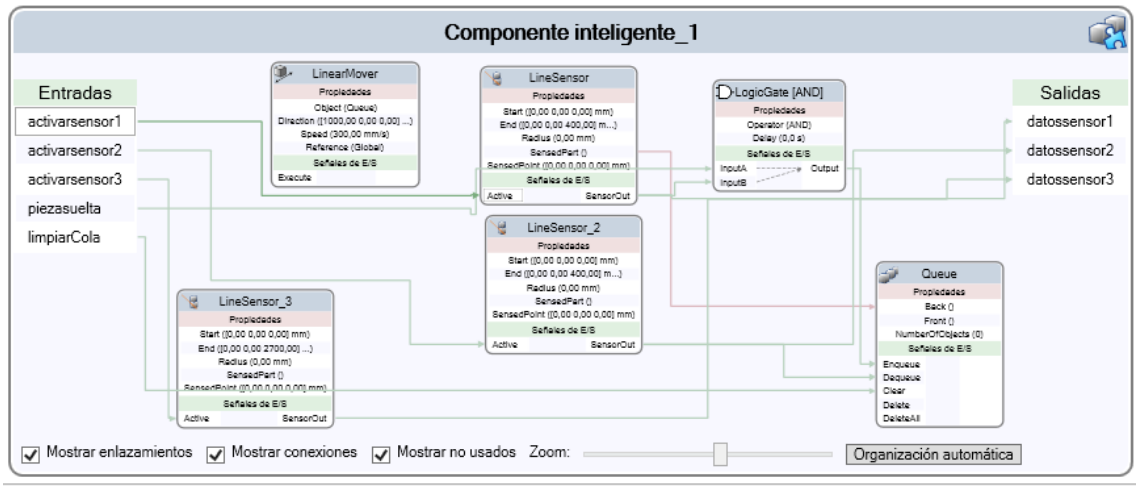

Ilustración 48 Lógica Cinta

Con los robots colocados, la disposición queda: el centro de la base de mesa2 en (0,0,0), el centro de la base de mesa1 en (0,-1600,0),el centro de la base de la pieza en (0,-1600,500), el centro de la base de palé aproximadamente en (5200,490,0) y la cinta aproximadamente en (1400,-600,0), (1400,-1000,0), (3900,-600,0) y (3900,-1000,0) para cada una de sus patas (ver Ilustración 49). Como cuando los objetos complejos son importados desde la librería de componentes no se pueden seleccionar correctamente sus aristas y centros, se ha intentado situar la cinta con la mayor precisión posible entre los dos robots.

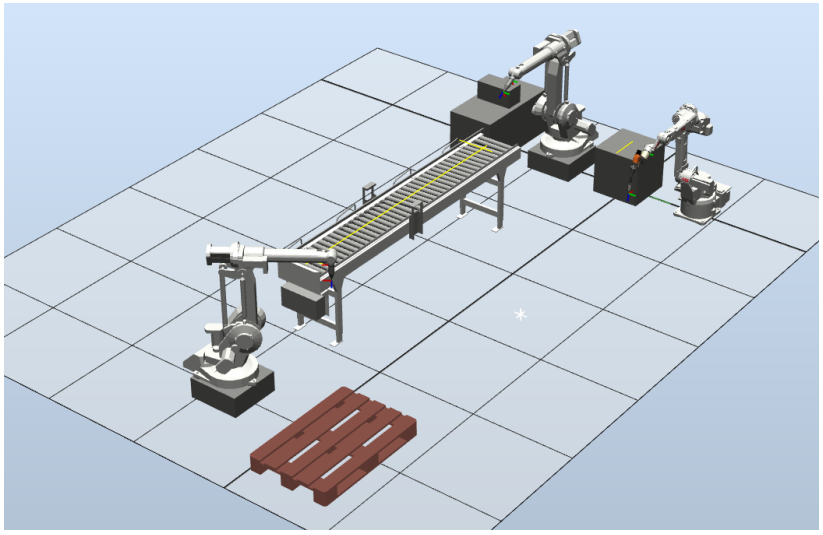

Ilustración 49 Escenaro3

3.3.2. Simulación

Se crea el **Sistema Robot desde diseño**, con versión ***RobotWare 5.61.05_5004N*** (pero uno para cada robot del escenario), de forma que el "*Sistema7*" comprende al robot **IRB1410_5_144__01** con "*herramienta1*", el "*Sistema8*" al robot **IRB1410_5_144__01_2** con "*herramienta2*" y el "*Sistema9*" el **IRB1520ID_4_150__02** con la herramienta **PKI_500_di_M2001**. Nótese que se han dejado los nombres por defecto de los sistemas al crearse los mismos, pero estos pueden ser modificados.

- Asignación de entradas y salidas

Para todos los sistemas se crea una unidad de comunicaciones para las señales, solo en el caso de los sistemas "*Sistema7*" y "*Sistema9*" se ha creado una unidad específica, "*RedControladores*", para la comunicación entre ambos. En ambos sistemas se requiere crear dos señales para la correcta ejecución de las tareas: "*EstadoTarea*" para indicar que el robot de tratado ha finalizado y "*MesaOcupada*" para indicar que no se puede realizar la tarea con un robot en la zona de trabajo. Todos los sistemas estarán conectados a la parada de emergencia y a la de reanudar, por lo que en su unidad comunicaciones se les crea las señales "*interrupcionparo*" y "*continuar*".

El "*Sistema7*", al encargarse de transportar las piezas, necesita conocer la presencia de piezas en las mesas. Se crean las salidas "*herramienta1*"para el control de la herramienta y la señal "*ActivarsensorMesa1*" para el sensor. Como toma de datos de los sensores se crean las entradas "*DastosSensorCinta*", "*DatosSensorMesa1*" y "*DatosSensorMesa2*".

Para el "*Sistema8*", que se encarga del apilado, como salidas se crea "*herramienta2*" para control de herramienta, "*PaleLleno*" para indicar que el palé está listo para cambiar y las señales de activación de sensores "*Actsensor1*", "*Actsensor2*" y "*Actsensor3*". Como entradas tenemos las toma de datos "*Datossensor2*" y "*cambiarpale*" para cuando se llene el palé

indicar el cambio del mismo y "*palenuevo*" que indica un cambio de pale por cualquier circunstancia.

El "*Sistema9*" se ocupará del tratamiento de piezas. Se crean la salida "*ActSensorMesa2*" para la activación del sensor y la entrada de toma de datos "*DatosSensorMesa2*".

Para las herramientas se crean eventos similares que los del escenario 1. Cada herramienta tendrá sus respectivos eventos de conectar y desconectar piezas.

- Lógica de estación

Se conectan las señales de los sensores con sus respectivos sistemas, y la señal "*EstadoTarea*" entre el "*Sistema7*" y el "*Sistema9*" (Ilustración 50). Todos tendrán las señales "*interrupcionparo*" y "*continuar*" conectadas a las señales "*Paro*" y "*Marcha*" creadas en la estación para facilitar la simulación de estas. La señal que indica el llenado del palé se conecta a la señal "*Pale_Lleno*".

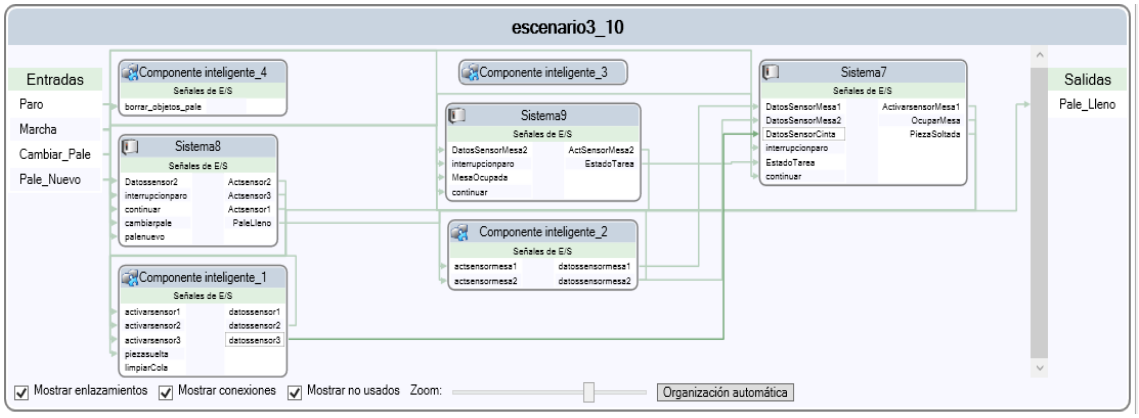

Ilustración 50 Lógica estación Escenario3

- Creación de conjuntos de colisiones

Siguiendo el procedimiento, para cada robot se crea un conjunto de colisiones con el robot y su herramienta en el **Grupo A**. Para el robot **IRB1410** el **Grupo B** incluirá las mesas, la cinta y el robot **IRB1520**. Para el robot **IRB1520** el **Grupo B** incluirá la mesa 2. Y para el robot **IRB1410_2** el **Grupo B** incluye el palé y la cinta. Para comprobar el choque de la pieza al ser transportada crearemos un conjunto con esta en el **Grupo A** y los posibles obstáculos como la cinta, las mesas y el palé en el **Grupo B**.

- Creación de trayectorias

Como en el escenario anterior se crea el *target_10* en el punto de reposo cuando son cargados al escenario los robots. Como objetos de trabajo tenemos que crear uno cuando la pieza está en la mesa 1 para el "*Sistema7*", cuando está la pieza en posición en la mesa 2 para el "*Sistema9*" y al llegar al final de la cinta para el "*Sistema8*".

Sistema7:

Trayectoria llevar pieza a la mesa 2 (Path_10):

Se crea para cada mesa un punto de aproximación *target_30 [100,1800,200]* y *target_50 [100,3400,200]*. Los puntos de coger y dejar pieza que se crean son *target_20 [100,1800,0]* y *target_40 [100,3400,0]*. Como punto auxiliar tendremos *target_80 [977.417557498, 2600, 460.117093341]*. Este punto auxiliar garantiza una trayectoria sin colisiones cuando la pieza está conectada a la herramienta. Trayectoria en amarillo en la Ilustración 51.

La secuencia queda: reposo → punto aproximación → punto agarre → señal activar herramienta → punto aproximación → punto auxiliar → punto aproximación → punto soltar pieza → señal desactivar herramienta → punto aproximación → reposo.

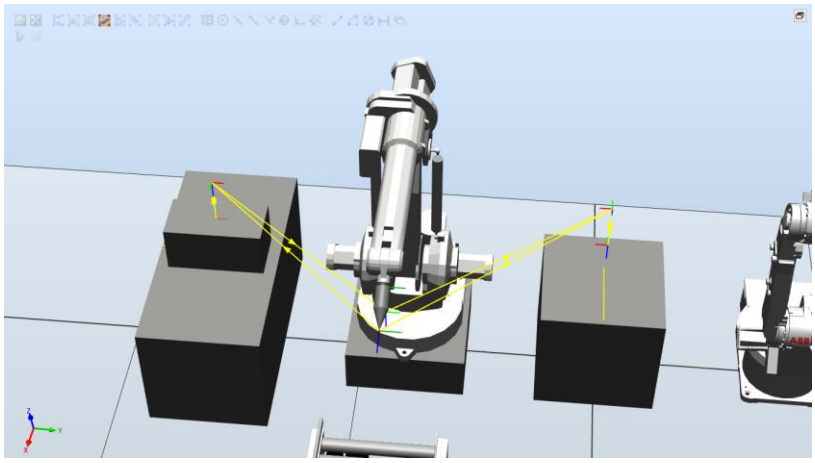

Ilustración 51 Path10 Sistema7

Trayectoria llevar pieza a cinta (Path_20):

En esta trayectoria utilizaremos los puntos *target_50 [100,3400,200*] y *target_40 [100,3400,0]* para coger la pieza. Además, se necesita el punto auxiliar *target_80 [977.417557498,2600,460.117093341]*, otro punto auxiliar que sirve de aproximación, *target_70 [1324.028,2597.668,500]*, para dejar la pieza en *target_60 [1424.028,2597.668, 274.083]*. La orientación sigue un patrón similar al del *Path_10*. Trayectoria en amarillo en la Ilustración 52.

La secuencia queda: reposo → punto aproximación → punto agarre → señal activar herramienta → punto aproximación → punto auxiliar → punto aproximación → punto soltar pieza → señal desactivar herramienta → punto aproximación → reposo.

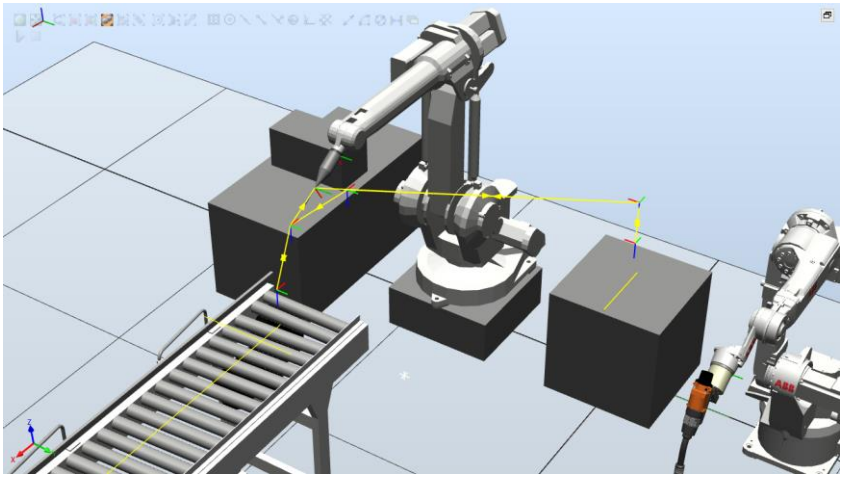

Ilustración 52 Path20 Sistema7

Sistema9:

Trayectoria tratar pieza (Path_10):

En este caso la herramienta debe incidir con un ángulo de 45º sobre la superficie de la pieza, por lo que todos los puntos tendrán esta orientación para que durante el movimiento por ellos se mantenga la herramienta en la misma posición. Como aproximación se crea *target_60 [300,0,300]*, y los punto del recorrido son: *target_20 [200,0,200]*, *target_30 [200,0,100]*, *target_40 [200,200,100]*, *target_50 [200,400,100]*, *target_70 [200,300,200]* y *target_80 [300,400,100]*. Ver la trayectoria en amarillo en la Ilustración 53.

La sucesión queda: reposo → punto aproximación → sucesión de puntos → reposo.

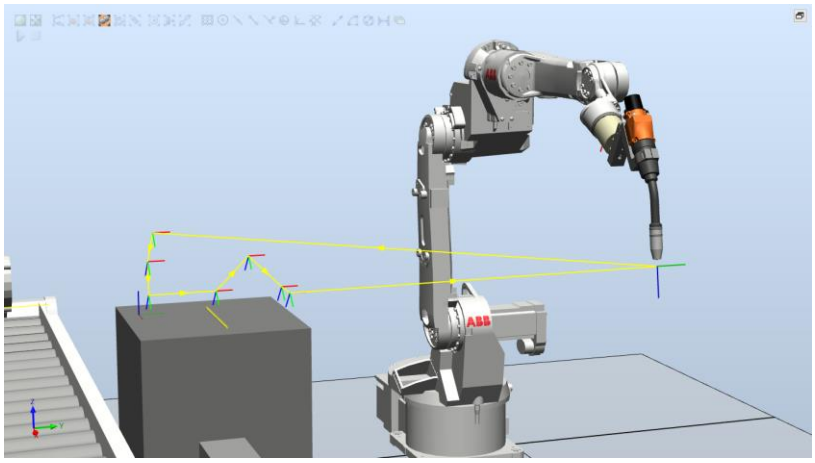

Ilustración 53 Path10 Sistema9

Sistema8:

Trayectoria coger pieza (Path_10):

Es similar a las anteriores trayectorias de transporte pero con un punto de espera, *target_100 [-285.459217565,1008.970109218,1016.016708033]*. El punto de aproximación *target_120 [-992.86,1703.256,1056]* y el punto de coger pieza *target_110 [-992.86,1703.256, 829.553]*. Ver la trayectoria en amarillo en la Ilustración 54.

La secuencia queda: reposo → punto aproximación → punto coger pieza → señal activar herramienta → punto aproximación → punto espera.

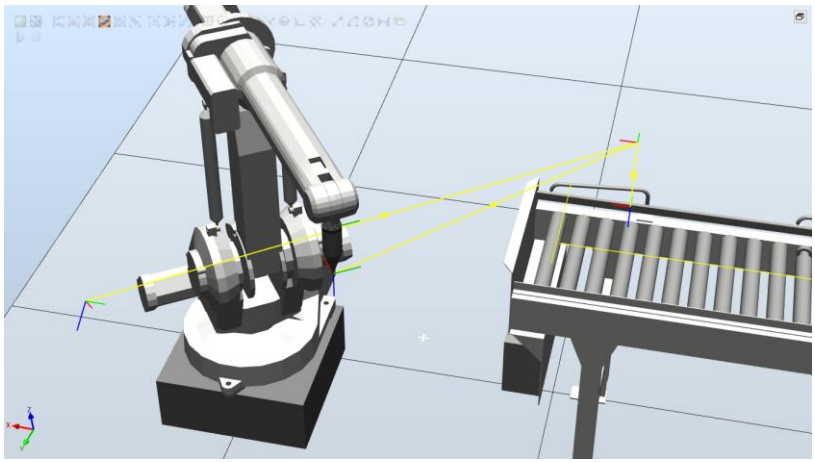

Ilustración 54 Path10 Sistema8

Trayectoria dejar pieza en palé (Path_20):

Para esta trayectoria se tiene como punto auxiliar de referencia a *target_30 [200,100,356]* y punto de referencia para dejar la pieza a *target_20 [200,100,200]*. Estos serán los puntos iniciales donde se colocará la primera pieza apilada. Cada tarea aumenta la distancia en el eje '*y*' de los puntos un valor constante nombrado como "*DISTANCIA_CAJAS*". De esta forma, las piezas se apilan de forma consecutiva hasta que el palé se llena. Ver la trayectoria en amarillo en la Ilustración 55.

La secuencia es igual para todos: espera → punto aproximación → punto dejar pieza → señal desactivar herramienta → punto aproximación → reposo.

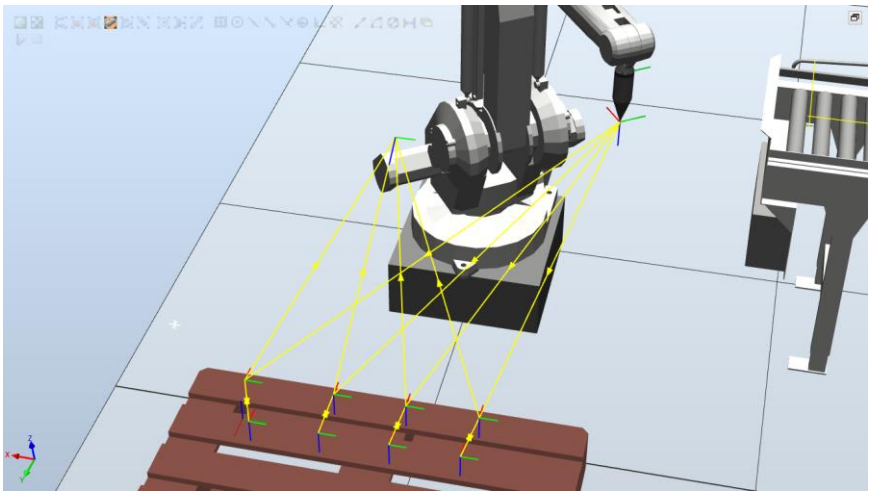

Ilustración 55 Path20 Sistema8

- Ficheros Rapid

La comunicación entre el robot de transporte y el de tratado se realiza a través de la señal "*EstadoTarea*" y "*OcuparMesa*". Esta es usada por el "*Sistema9*" para comunicar que el tratado de la pieza ha finalizado, cambiando la señal "*EstadoTarea*" a 1. Una vez que se retira la pieza se reinicializa la señal a 0. De esta forma el "Sistema7" lee el estado de la tarea del otro sistema. Por otra parte, el "*Sistema7*" comunica al "*Sistema9*" el uso del espacio de la mesa con la señal "*OcuparMesa*". Cuando el "*Sistema9*" lee la señal a 1, este no iniciará la tarea al saber que la mesa está ocupada y esperará al cambio de la señal a 0.

El robot de apilado, por su parte, recogerá las piezas y las apilará, hasta cuatro de ellas, a lo largo del palé. Para ello el programa incrementará los valores `*y*' de los puntos con una constate, "*DISTANCIA_CAJAS*". Esta constante es la distancia de separación que hay entre un punto similar entre dos piezas. Una vez hay cuatro cajas apiladas el programa espera la señal de cambio de palé para reiniciar el proceso y las variables de los puntos. Los programas en ***RAPID*** se encuentran en el Anexo 5.3.

Para los procesos de interrupción se han generado dos procesos '***TRAP***'. El primero será la interrupción de parada de emergencia y el segundo la señal de reanudación. Sucede que las interrupciones tiene mayor prioridad que las señales comunes del sistema, por lo que la reanudación deberá tener una prioridad similar o mayor para que el sistema la reconozca durante la parada. En la parada las instrucciones "*Stopmov*", "*Storepat*" y "*Clearpath*"; realizan las acciones de detener el robot, guardar en memoria el '*Path*' y limpiar la última tarea de la cola parando y dejando al robot en estado de espera. En la reanudación las instrucciones "*RestorePath*" y "*StartMove*" cargan el '*Path*' guardado y reanudan el movimiento del robot en el punto donde se realizó la parada.

3.3.3. Comprobación

Se cargan los ficheros y seleccionamos ejecutar la simulación en ciclo continuo. Reproducimos la simulación y observamos que no hay problemas. El sistema esperará hasta que indiquemos que hemos cambiado el palé una vez este esté lleno. Una vez que se indica la señal de que hay un cambio de palé, en la simulación se vaciará el palé y el robot de apilado continuará con la tarea.

4. Réplica del laboratorio

Siguiendo los pasos para generar el escenario, se creará una estación vacía con las áreas y jaulas del laboratorio (Ilustración 56) siguiendo lo dispuesto en el plano de la Ilustración 2. Se procede posteriormente a introducir las librerías creadas de los componentes y los sistemas de los robots.

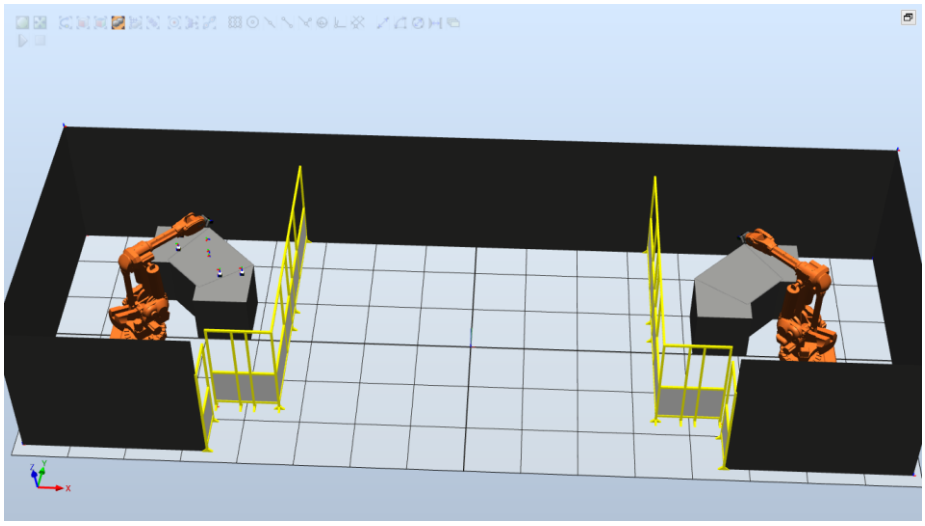

Ilustración 56 Escenario Laboratorio

4.1. Librerías creadas

Mediante la creación de librerías, se nos permite implementar objetos o componentes para incluir en nuestros escenarios de forma genérica y sin la necesidad de crearlos directamente cada vez que realizamos un escenario. Para generar una librería basta simplemente con crear el mecanismo, pieza o componente que necesitemos; con el botón derecho del ratón seleccionamos la opción **Guardar como biblioteca** (Ilustración 57).

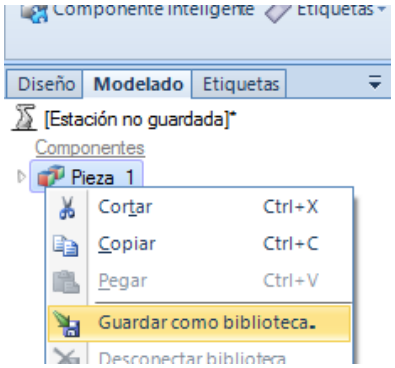

Ilustración 57 Pestaña Guardar biblioteca

- Cubo de rubik:

El cubo de *rubik* (Ilustración 58) se crea como tetraedros de lado 65mm.

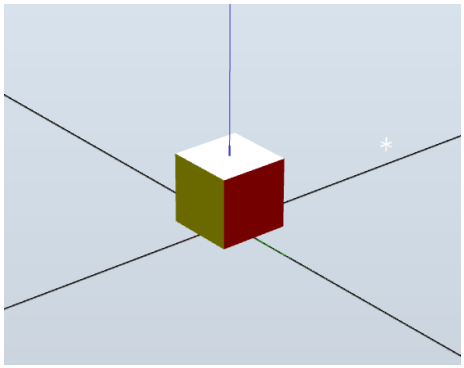

Ilustración 58 modelo cubo de Rubik

- Mesas de trabajo:

Las mesas de trabajo comprenden un cuarto de circunferencia sobre el robot, que será el volumen que operará en las jaulas. Está formada por tres cubos de lado 1m y situados sus centros en (500,1500,0), (1500,500,0) y un tercero entre ambos en (1060,1060,0) girado a 45º (Ilustración 59).

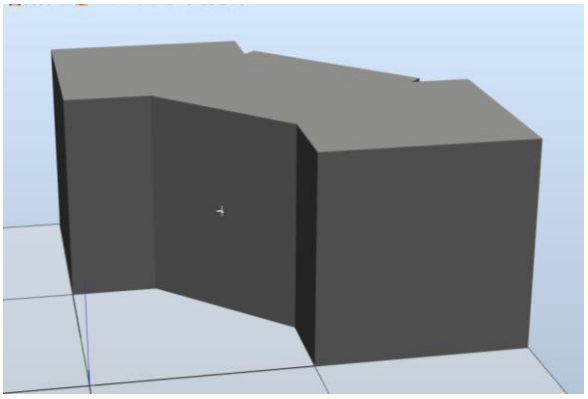

Ilustración 59 bloques Mesa

Para ampliar el área de la superficie de la mesa se añade una superficie creada mediante un segmento. El segmento comprende el área marcada en la Ilustración 60, delimitada por las aristas de los cubos.

En la creación de la mesa se sigue el esquema del modelo del Anexo 2.

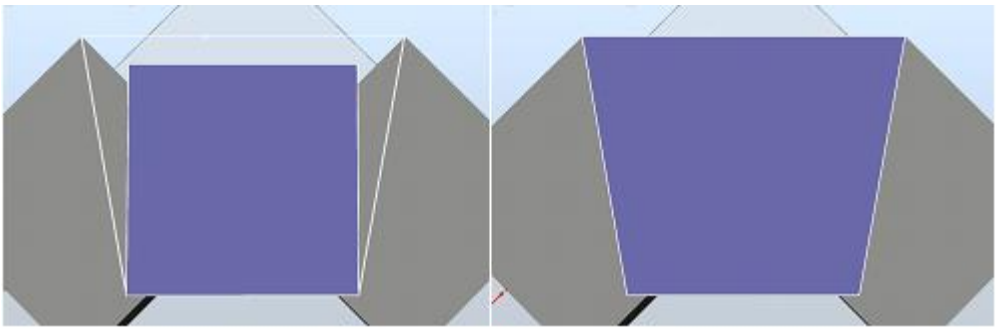

Ilustración 60 Secuencia creación de la superficie Mesa

- Aros y soporte:

Con la opción crear **Superficie con curva** se genera la superficie de la mesa. Posteriormente se conectan todos los cuerpos y se agrupan para su guardado.

Para los aros se tendrá que realizar un proceso por el cual creamos una **Superficie** circular de 25mm de diámetro y varias **Curvas** circulares de 60mm, 67.5mm, 72.5mm, 80mm y 87.5mm. Estas herramientas están en la pestaña de modelado junto a la de **Crear Sólido**. Se coloca el centro de la superficie circular en la curva y con la opción **Extrudir superficie** se selecciona la superficie y la curva para que cree el nuevo cuerpo (Ilustración 61). La Ilustración 62 muestra el ejemplo del modelo de un aro después de aplicar el procedimiento.

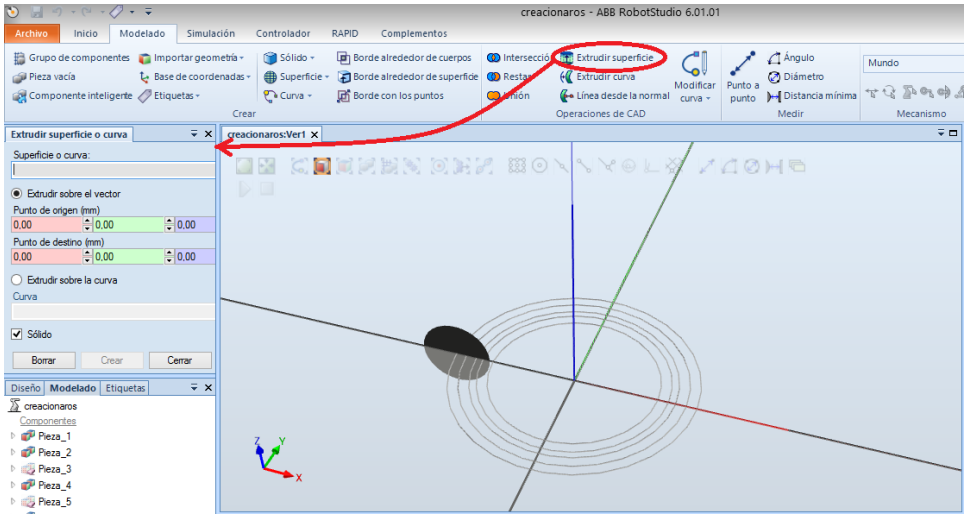

Ilustración 61 Herramienta extrudir superficie

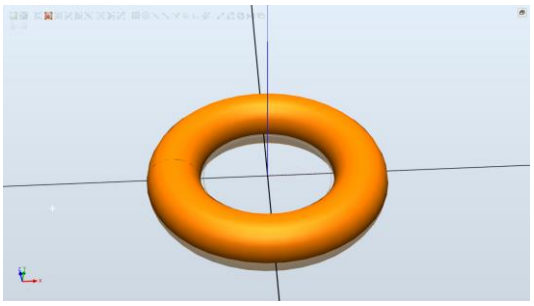

Ilustración 62 Modelo aro

El soporte para colocar los aros es más complejo de realizar, sus cálculos del teorema de Tales se encuentran en el Anexo 3. La base se puede asemejar a un cilindro sencillo de altura 26mm y diámetro 130mm. Para el poste se crea un cono con una base de diámetro 55mm y una altura de 345mm. Con **Resta** se deja una altura de 125mm.

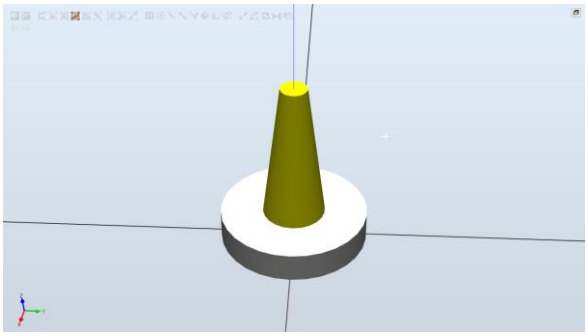

Ilustración 63 Modelo soporte aros

- Herramienta robot:

La herramienta **MHL2-16D2(SMC)** ha sido recreada mediante **Crear mecanismo**, seleccionando la opción de mecanismo **herramienta** (Ilustración 64). Los eslabones se han recreado por separado de forma aproximada a la herramienta real con su soporte para herramientas. El eslabón L1 corresponde a la base con el soporte incluido y los eslabones L2 y L3 como los dedos de la pinza. En los datos herramienta se introducen la masa igual a 1kg (dato aproximado), centro de gravedad desplazado 1cm en '*z*' (dato aproximado) y **TCP** en 16cm sobre el eje '*z*'. Para darle mayor realismo, se ha añadido un movimiento de apertura y cierre de los deode de la pinza. Para ello, se introducen dos ejes para cada eslabón de los dedos. Se introduce como eslabón principal la base y el secundario el dedo con ejes prismáticos sobre la coordenada '*y*', con un movimiento del eje de 41mm (positivo o negativo según el dedo).

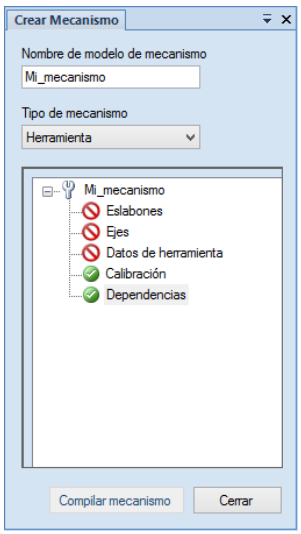

Ilustración 64 Ventana Crear mecanismo

Para que al activar una señal se abra y cierre la herramienta hace falta crear un componente inteligente que contenga el mecanismo herramienta e introducir el componente ***JointMover***. Este componente permite que se muevan los ejes creados en el mecanismo seleccionado. Simplemente se le indica el mecanismo y el movimiento que tienen que realizar los ejes cuando se introduzca una señal de ejecución en el mismo. La Ilustración 65 muestra el modelo funcional de la herramienta listo para conectarse.

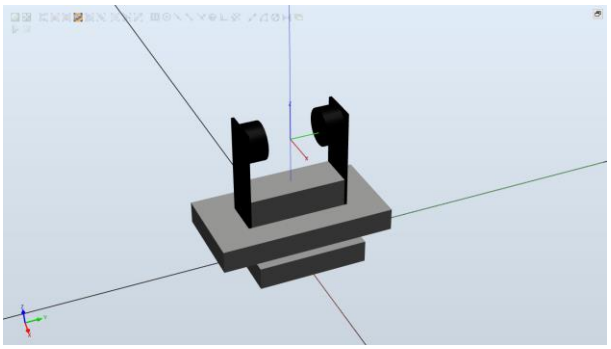

Ilustración 65 Modelo herramienta MHL2-16D2

4.2. Simulación robots reales

El modelo de los brazos del laboratorio es **IRB6400F 2.4-150**. Como este modelo ya no está disponible en las versiones actuales de ***RobotStudio***, se realizó una selección, eligiendo finalmente un modelo con unas dimensiones similares, el **IRB6400R 2.5-150**, Anexo 1.2.

El **IRB6400R 2.5-150** requiere de la instalación de sus controladores para ***RobotStudio***, ya que estos no están disponibles en los controladores por defecto. En la pestaña de **Controlador** en **Administrador de instalación**, se puede crear un sistema específico para el funcionamiento del modelo (Ilustración 66).

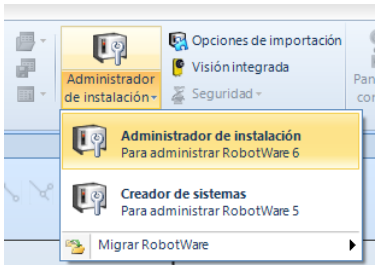

Ilustración 66 Opciones Administrador de instalación

Se crea un nuevo sistema virtual vacío en la ventana de administración que aparece. La siguiente ventana permite introducir los componentes para el control del sistema, en este caso ***RobotWare*** en su versión adecuada (normalmente la instalada con ***RobotStudio***). Hay varios tipos de controles según los mecanismos y robots a utilizar, por lo que según nuestras necesidades podemos descargar los complementos que necesitemos con la pestaña **Complementos** de ***RobotStudio***. La ventana siguiente permite visualizar las licencias que se requerirán instalar en el proceso y añadir si fuera necesario.

En las siguientes opciones se pueden introducir más parámetros a los sistemas. Un complemento necesario es la opción de ***world zone***, para indicar los límites de la zona de trabajo del robot. Los drivers se tendrán que ajustar al modelo del robot a controlar, como se muestra en la Ilustración 67. Las aplicaciones especiales no se requieren para este sistema.

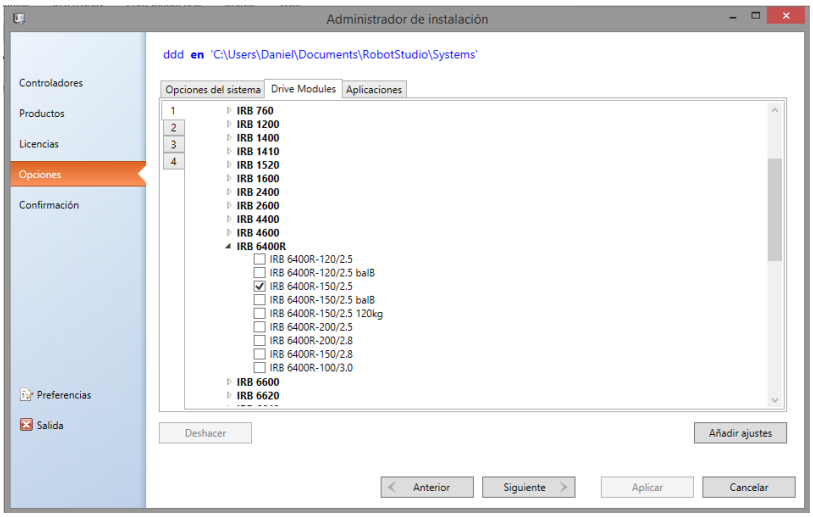

Ilustración 67 Ventana selección de drivers Sistema virtual

Una vez finalizado, pulsando **Aplicar** se descargarán e instalarán las librerías y licencias para el controlador. Simplemente en un escenario, con cargar el nuevo sistema guardado en **Sistemas existente** (Ilustración 68) el programa preparará el controlador y el modelo del robot para su utilización.

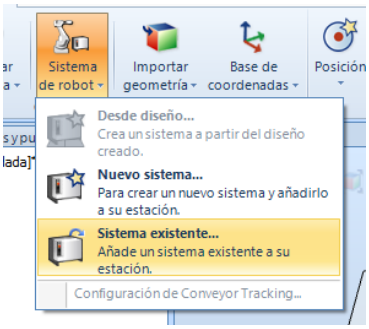

Ilustración 68 Opción de importación de Sistemas existentes

El funcionamiento de las pinzas seguirá el mismo proceso que en los otros escenarios. La herramienta tendrá dos eventos, uno de conectar y otro de desconectar. Este proceso no sirvió para que la herramienta recogiera los aros, debido a limitaciones del programa, por lo que se tuvo que implementar con componentes inteligentes. Esto se explica en detalle en el siguiente apartado. Las conexiones de las señales son similares para ambas herramientas, las señales de activación de la herramienta se conectan con la señal de control de cada sistema, señal "*herramienta*".

4.3. Programa de ejemplo de uso de la estación

4.3.1. Programa 1

Como ejemplo del uso de componentes disponibles en el laboratorio, se ha realizado un proceso de apilado de varios cubos de *rubik*. Estos están dispuestos sobre la mesa y se apilan sobre el centro de la mesa (Ilustración 69).

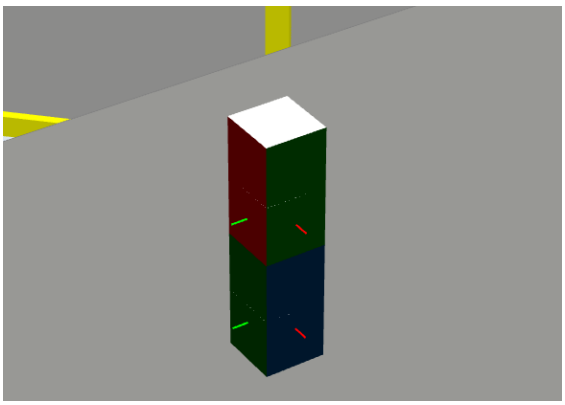

Ilustración 69 Tarea apilado de cubos

Para el programa se disponen puntos de aproximación a 100mm sobre cada cubo *target_70 [32.5, 32.5, 165]*, *target _80 [33.416, -351.87, 165]*, *target _90 [1336.73, -970.855, 165]*, *target _100 [928.204, -1027.415, 165]*; y posiciones de aproximación de apilado *target _110 [653.889, -515.749,165]*, que varía al igual que en el escenario3, según los cubos apilados, una distancia fija "*DISTANCIA_CUBOS*". Lo mismo sucede con el punto de apilado *target _60 [653.888999899, -515.748999912, 45]*. El punto de agarre estará sobre el centro de cada cubo *target_20 [32.5, 32.5, 45], target _30 [33.416, -351.87, 45], target _40 [928.204, -1027.415, 45], target _50 [1336.73, -970.855, 45]*. Se requiere un punto auxiliar *target _120 [653.889, -515.749, 465]*, para que las piezas al ser transportadas no choquen con las apiladas. Los puntos están expresados con respecto a un objeto de trabajo creado en la base de uno de los cubos dispuestos en la mesa.

Como será necesario aplicar también limites articulares en los robots de la simulación se usará la función "*WZLimJointDef*". Esta función permite crear una zona mundo con los limites articulares del robot para limitar la zona de trabajo. Se aplican los límites del **anillo *Balluf*** a los robots de forma que solo se puedan mover sobre el cuarto de circunferencia de sus respectivas mesas de trabajo. Para evitar que choquen con las vallas, las paredes y el techo; se han generado cajas mundo, o ***worldbox***, delimitando el área de trabajo con un margen de 300mm de seguridad. Con "*WzBoxDef*" se genera la caja delimitada por dos puntos de esquina que la definen, [-3700, 0, 0] y [-6150, 2400, 2500] para el robot del programa 1 y [3700, 0, 0] y [6150, 2400, 2500] para el robot del programa 2.

Los programas 1 y 2 del escenario del laboratorio están disponibles en el Anexo 5.4.

4.3.2. Programa 2

Para el segundo robot se realiza una tarea de apilado de aros. Los aros están dispuestos por la mesa y el robot los apilará en el centro de la mesa sobre el soporte en orden de mayor a menor diámetro (Ilustración 70).

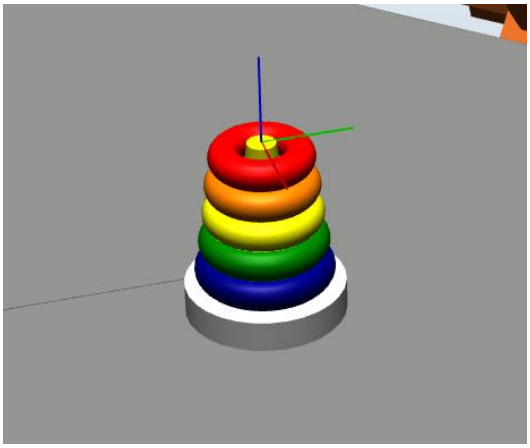

Ilustración 70 Tarea apilado aros

Las trayectorias se han creado sobre un objeto de trabajo situado en el extremo superior del soporte. Los aros se disponen en la mesa en los puntos *target_20 [708.833,97.746,-138.5]*, *target_30 [103.436,397.836,-138.5]*, *target_40 [-588.627,-327.953,-138.5]*, t*arget_50 [-429.489,-548.171,-138.5]* y *target_60 [-41.069,-535.798,-138.5]*. Así mismo se apilan en los puntos *target_70 [0,0,-12.5], target_80 [0,0,-37.5], target_90 [0,0,-62.5], target_100 [0,0,-87.5], target_110 [0,0,-112.5]*. Los puntos de aproximación estarán dispuestos sobre los aros en *target_120 [103.436,397.836,-38.5]*, *target_130 [708.833,97.746,-38.5]*, *target_140 [-47.06,-553.06,-38.5]*, *target_150 [-429.489,-548.171,-38.5]* y *target_160 [-588.627,-327.953,-38.5]*. Para el apilado se hará uso del punto *target_170 [0,0,87.5]*. Este robot dispondrá de los mismos límites articulares que en la tarea anterior.

Para el caso de simular el agarre de la pinza se ha tenido que aplicar otro método debido a la poca precisión que presenta el programa con los eventos. Para ello se ha colocado un sensor cerca del **TCP** de la herramienta (ver sensor en amarillo en la Ilustración 71) y se ha conectado este sensor al robot y no a la herramienta. De esta forma, se solucionaron los problemas de conexión de objetos entre la herramienta y el robot. Las librerías se han desconectado con el fin de poder manipularlas y modificarlas para esta tarea. En la lógica de estación de la herramienta conectaremos los componentes, como se dijo en el escenario 1, de forma que la pieza detectada por el sensor se conecte o desconecte a la pinza con ***attach*** y ***detach***. Las conexiones entre los distintos componentes y sensores que forman la pinza final puede observarse en la Ilustración 72.

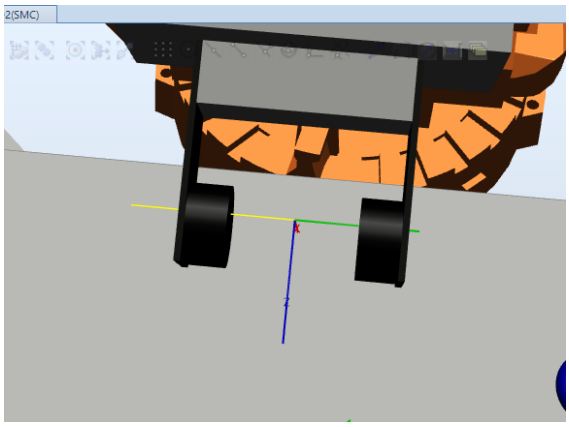

Ilustración 71 Disposición del Sensor en la herramienta

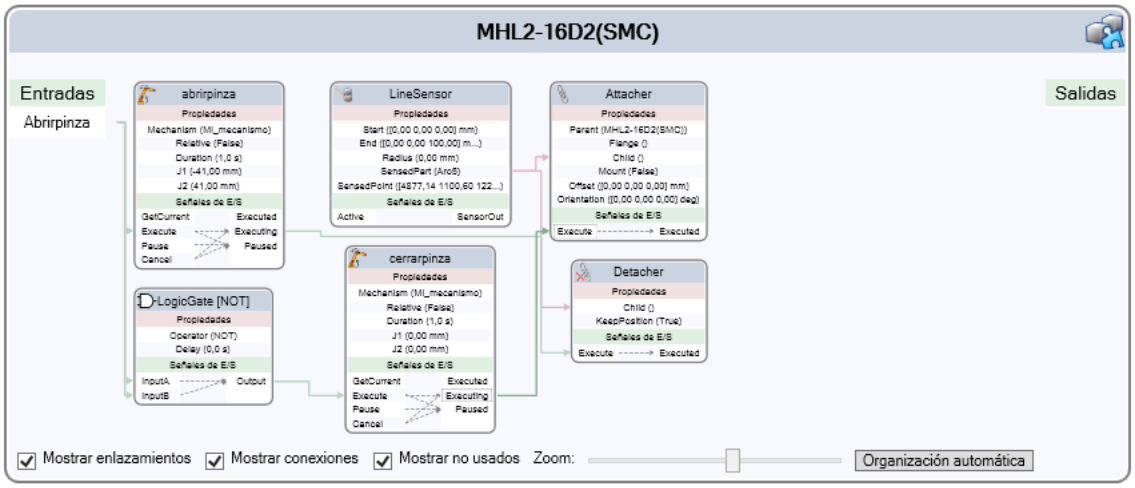

Ilustración 72 Lógica de la estación de la herramienta

4.3.3. Comprobación

Finalmente se simulan estos programas en ambos robot. Se comprueba que se corresponden con bastante fidelidad a los elementos físicos que se encuentran realmente en el laboratorio, así como la interacción de la pinza con estos elementos. La buena implementación de la pinza y con las limitaciones del área de trabajo correctamente implementadas, permiten usar el modelo para probar distintas aplicaciones para posteriormente probarlas en los robots reales.

5. Conclusiones y trabajo futuro

Para finalizar vamos a enunciar las conclusiones de este trabajo, los principales problemas detectados así como las líneas de trabajo futuro que podrían derivarse del mismo.

5.1. Conclusiones

Podemos concluir que se han cumplido los objetivos planteados inicialmente. El desarrollo de tres escenarios ha permitido adquirir destrezas en el manejo de ***RobotStudio***, mejorando la visualización. Esto ha permitido determinar una serie de pasos para el desarrollo de un escenario:

1) Creación de la estación.
2) Introducción de objetos, robots y controladores.
3) Asignación de entradas y salidas.
4) Implementación de la lógica de estación.
5) Creación de trayectorias.
6) Creación de ficheros Rapid.

Todo este conocimiento ha sido utilizado para la creación del escenario virtual del laboratorio. Este escenario dispone de todo lo necesario para implementar las tareas que se pudieran llegar a realizar en el laboratorio real. Al haber tenido que utilizar modelos de robots un poco más pequeños que los reales, el escenario servirá como una pre-visualización de la tarea y método de aprendizaje. Esto no será un problema a la hora de utilizar los robots reales, ya que se realizarán las programaciones de las tareas a través del ***Flexpendant***. Este dispositivo permite realizar un control manual del movimiento del robot. Asimismo, permite adaptar adecuadamente los programas que se ejecutan finalmente en los robots.

5.2. Problemas detectados

A continuación vamos a enumerar los principales problemas detectados en la implementación del trabajo.

-Versiones

Ciertos complementos y herramientas del software requieren del uso específico de versiones del programa o las aplicaciones. Esto llega al punto de tener que descargarlas por separado de forma que sean compatibles con la versión de ***RobotStudio*** en cuestión. Todo ello origina problemas de ejecución si no se realizan de forma correcta. Con la herramienta ***Pack and Go*** sucede, sin embargo, que las versiones deben ser similares a las de los complementos guardados en el mismo para que se pueda ejecutar en otros equipos.

-Ausencia de documentos

Dada la antigüedad de los equipos robóticos los datos de los mismos son más difíciles de encontrar. **ABB** dispone de una remesa de robots de nueva generación, de forma que los modelos antiguos están descatalogados y apenas se dispone de librerías para ellos, como es el caso del modelo **IRB6400F**. También hay que remarcar el problema con los manuales, ya que parece ser más fácil el encontrarlos en páginas no oficiales antes que en la oficial de **ABB**. Un ejemplo de esto es que en la página oficial se ofrecía como versión revisada del manual de ***RobotStudio*** una versión más antigua, que no corresponde con el programa. Esto y los escasos ejemplos del manual generaron una falta de información importante sobre el manejo del programa.

-Necesidad de creación de modelos

Ninguno de los materiales físicos a usar en el laboratorio estaba disponible en las librerías de ***RobotStudio*.** En algunos casos ni siquiera hay recursos que tengan una forma aproximada. Por lo tanto, se han tenido que crear librerías específicas para los mismos con las herramientas de modelado. Estas herramientas nos permiten una aproximación bastante realista de los objetos. Sin embargo, si se desease un diseño más cercano, se podrían utilizar programas de diseño como ***Autocad*** en su lugar. Los elementos usados de las librerías se limitan a vallas y herramientas de robots comunes en la industria, lo que casi siempre requiere de la creación de librerías propias.

-Problemas con la web

Durante bastante tiempo los video-tutoriales que facilitó **ABB** estuvieron restringidos en su página web oficial. Posiblemente los motivos podrían ser por mantenimiento de la web o debido a cambios en la misma. Esto sucedió durante un periodo de desarrollo del trabajo que obligó a buscar otros tutoriales más extensos. Los tutoriales fueron necesarios debido a la complejidad del programa, que en partes del manual requerían una visión más explicativa y detallada de cada una de las herramientas. Un claro ejemplo es el uso de **Componentes Inteligentes**, cuya utilización requiere de amplios conocimientos y ejemplos de uso, ya que el manual ofrece ejemplos muy sencillos.

5.3. Líneas de trabajo futuro

Finalmente vamos a enunciar varias líneas de trabajo futuro que podrían derivarse de este trabajo.

-Desarrollo de tareas de mayor complejidad con el escenario.

Dado que los robots están operativos en el laboratorio, se pueden realizar diversas tareas con ellos. Los robots tienen instalada la herramienta **MHL2-16D2 (SMC)**, la cual permite manipular objetos cúbicos. Esto permite desarrollar toda una serie de tareas de apilado y des-apilado. En caso de incluir algún mecanismo con el que interactuar como una cinta o herramienta para la tarea, se tendría un trabajo más completo. Con el escenario desarrollado, en este trabajo se haría la comprobación con el escenario virtual de la tarea y se incluirían las nuevas librerías desarrolladas con las ya disponibles.

-La creación de un modelo de robot funcional en RobotStudio.

Uno de los aspectos más problemáticos de este trabajo ha sido la carencia de un modelo funcional en ***RobotStudio*** del robot **IRB 6400F**. Las herramientas del programa permiten la creación de cualquier mecanismo, incluido un robot. Esto da la posibilidad de desarrollar un trabajo exclusivo en la creación del mecanismo de los robots. De esta forma el trabajo futuro ampliaría las características del escenario desarrollado aquí, abriendo así la posibilidad de una actualización continua del mismo. Esta línea también abre la posibilidad de que el trabajo se oriente a otros mecanismos que se deseen incluir en el escenario real.

Bibliografía

Página oficial de ABB donde se disponen los manuales, programas y complementos de RobotStudio de ABB:

[1] *Software RobotStudio de ABB: página oficial.*

http://new.abb.com/products/robotics/es/robotstudio, ABB, 2016.

Video-tutoriales:

[2] *Curso de RobotStudio. Tema1.* https://www.youtube.com/watch?v=uISH8ksfIX4, Canal alonsoprofe, Youtube, 2013.

[3] *Curso de RobotStudio. Tema2.* https://www.youtube.com/watch?v=pQnbdfHLJSg, Canal alonsoprofe, Youtube, 2013.

[4] *Curso de RobotStudio. Tema3.* https://www.youtube.com/watch?v=CXRGOxS4p9I, Canal alonsoprofe, Youtube, 2013.

[5] *Curso de RobotStudio. Tema4.* https://www.youtube.com/watch?v=6aOQNPS7yUw, Canal alonsoprofe, Youtube, 2013.

[6] *Curso de RobotStudio. Tema5.* https://www.youtube.com/watch?v=5hwdRBQUaRY, Canal alonsoprofe, Youtube, 2013.

[7] *Curso de RobotStudio. Tema6.* https://www.youtube.com/watch?v=NbiilV6pIrE, Canal alonsoprofe, Youtube, 2013.

[8] *Curso de RobotStudio. Tema7.* https://www.youtube.com/watch?v=C18yf6irv0w, Canal alonsoprofe, Youtube, 2013.

Video-tutoriales proporcionados por ABB:

[9] *Tutoriales para RobotStudio. La herramienta de programación fuera de línea más usada en el mundo de la robótica.*

http://new.abb.com/products/robotics/es/robotstudio/tutoriales, ABB, 2015.

Manuales de refencia:

[10] *Manual del operador* RobotStudio6.0, ABB Robotics, 2008-2014. Document id: 3HAC032104-005, Revisión N.

[11] *Manual de referencia técnica. Instruccione, funciones y tipos de datos de RAPID. RobotWare6.0*, ABB Robotics. Document id: 3HAC050917-005.

[12] *Product specification IRB6660*, ABB Robotics, 2004-2014. Document id: 3HAC028207-001.

[13] *Product specification IRB6400 M98 BWOS3.2*, ABB Robotics. Document id: 3HAC4019-1.

[14] *Product specification IRB 6640ID*, ABB Robotics. Document id: 3HAC028284-001.

[15] *Product On-line Manual IRB 6400R*, ABB Robotics.3HAC6264-1.

[16] *Manual Pinza neumática de gran apertura serie MHL2*, SMC.

[17] Para la referencia de funciones RAPID y de herramientas del programa:

Ayuda RobotStudio 6.01.

Trabajos de Fin de Grado relacionados con el Laboratorio de Tecnologías Industriales:

[18] Sergio Lorente Doñate. "*Diseño e Instalación de Medidas de Seguridad en un Robot ABB IRB 6400".* Trabajo de Fin de Grado, 2015. Graduado en Ingeniería en Electrónica y Automática. Universidad de Zaragoza, Escuela Universitaria Politécnica de Teruel.

Índice de ilustraciones