Universidad Miguel Hernández de Elche

MASTER UNIVERSITARIO EN ROBÓTICA

“Programación con RobotStudio de robots ABB para la automatización de procesos de clasificación y paletizado”

Trabajo de Fin de Máster

2021/2022

Autor: Emilio Vicedo Cano

Tutor: Carlos Pérez Vidal

Índice

AGRADECIMIENTOS

En primer lugar, a mi familia, a esa familia con la que se establecieron mis cimientos como persona. En el mismo lugar, a mi otra familia, la que hemos creado mi mujer, Carolina, y yo. Todo el esfuerzo de este año ha sido y será para poder pasar más tiempo y de mejor calidad con vosotros, Rodri y Beltrán.

A Fran Soler Mora, por toda tu ayuda y apoyo desinteresado durante este máster, sin ti todo se hubiera dilatado en el tiempo.

A los profesores del máster, por su implicación y vocación para hacerlo posible, ha sido curioso reencontrarme con algunos casi 20 años más tarde.

Y, por último, a Carlos Pérez por brindarme la oportunidad de realizar este trabajo de fin de máster con él.

A todos vosotros, gracias de corazón.

RESUMEN

El trabajo mostrado a continuación pretende diseñar y simular el comportamiento de una estación de clasificación y pick and place de piezas. El sistema se compondrá de 4 robots industriales que se encargarán de clasificar las piezas en su cinta de descarga correspondiente y de paletizarla de la manera más eficiente posible.

La estructura del trabajo pretende seguir un orden lógico para su mejor comprensión, aumentando secuencialmente la complejidad de los contenidos a partir de la base explicada en capítulos anteriores.

Se presentará una introducción al problema, describiendo por encima el objeto del trabajo, después se realizará un estudio del arte, situando a la robótica industrial en la actualidad y describiendo sus posibilidades.

En los siguientes capítulos se explicará el proceso de adquisición de conocimiento del RobotStudio, utilizando para ello manuales de fabricante, videotutoriales y foros técnicos del programa. Una vez adquirido dicho conocimiento se procederá a describir el diseño de la estación desarrollada. Finalmente se expondrán los resultados y conclusiones obtenidas.

ABSTRACT

The work shown below aims to design and simulate the behavior of a sorting and pick and place station for parts. The system will be made up of 4 industrial robots that will be in charge of classifying the pieces on their corresponding discharge conveyor and palletizing it in the most efficient way possible.

The structure of the work tries to follow a logical order for its better understanding, sequentially increasing the complexity of the contents from the base explained in previous chapters.

An introduction to the problem will be presented, describing the object of the work above, then a study of the art will be carried out, placing industrial robotics today and describing its possibilities.

In the following chapters, the process of acquiring knowledge of RobotStudio will be explained, using manufacturer's manuals, video tutorials and technical forums of the program. Once this knowledge is acquired, the design of the developed station will be described. Finally, the results and conclusions obtained will be presented.

1. Introducción

El objeto principal de este trabajo es la programación y diseño de estaciones robóticas, robóticas industriales, por lo que se describirá este tipo de robots concretamente.

Según la norma ISO 8373:2012 de la IFR, un robot es un "mecanismo accionado programable en dos o más ejes con un grado de autonomía, moviéndose dentro de su entorno, para realizar el objetivo o tareas". Si se hila un poco más fino nos encontramos con la definición de robot industrial, que es un manipulador "automáticamente controlado, reprogramable y multipropósito, programable en tres o más ejes, que se puede fijar en su lugar o utilizar de forma móvil para su uso en aplicaciones de automatización industrial". Con esta definición ya se puede diferenciar de un robot normal, ya que este puede tener dos ejes, o de un robot de servicio en el que se expone que no es para uso industrial

1.1. Motivación

La industria, año tras año está evolucionando hacia una automatización cada vez mayor que permita impulsar la productividad, mejorar la continuidad de la fabricación y flexibilizar los procesos para mantener la producción y la calidad.

El efecto de la pandemia no ha hecho, sino confirmar esa transformación en todos los sectores. Los robots industriales continúan avanzando más allá de la fabricación tradicional hacia logística, laboratorios, talleres, incluidas las pymes que están apostando por primera vez por la automatización de sus procesos.

Cada vez se reconoce más la viabilidad y la escalabilidad que ofrecen los robots industriales debido al potencial que tienen para reducir costes y mejorar la productividad. Ya no solo la robótica industrial si no también la robótica de servicio y los robots colaborativos. Es, por tanto, necesario el estudio previo y el diseño del proceso que permita la valoración de su implantación.

Sin embargo, no todo son ventajas, la implantación de algunos procesos de la empresa con robótica industrial implica un alto coste económico inicial, con una amortización durante varios años, cubiertos, sin embargo, como se ha comentado, por una alta productividad y calidad, librando sin olvidarnos de las tareas repetitivas y peligrosas realizadas por los operarios.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Máster es el diseño de una estación robotizada para la clasificación y pick and place de piezas. Para ello se ha utilizado RobotStudio, un software propietario de la empresa ABB, uno de los mayores fabricantes mundiales de robots industriales. La adquisición de conocimientos y destrezas con el RobotStudio que se ha realizado implican una cercanía con el desarrollo industrial que se aplica hoy en día.

2. Estado del arte

Se van a describir algunos datos económicos globales de robots industriales para situarlos en la actualidad. Según el informe ejecutivo de IRF (International Federation of Robotics), a pesar de la situación de pandemia mundial, el 2020 fue el tercer año más exitoso para la industria de la robótica, después de 2018 y 2017, debido a que hubo un ligero crecimiento hasta llegar a las 383.545 unidades instaladas. A pesar de lo que pueda parecer, en 2020 la industria electrónica fue el principal motor de crecimiento de los robot industriales, por encima de la automotriz que había sido el principal cliente.

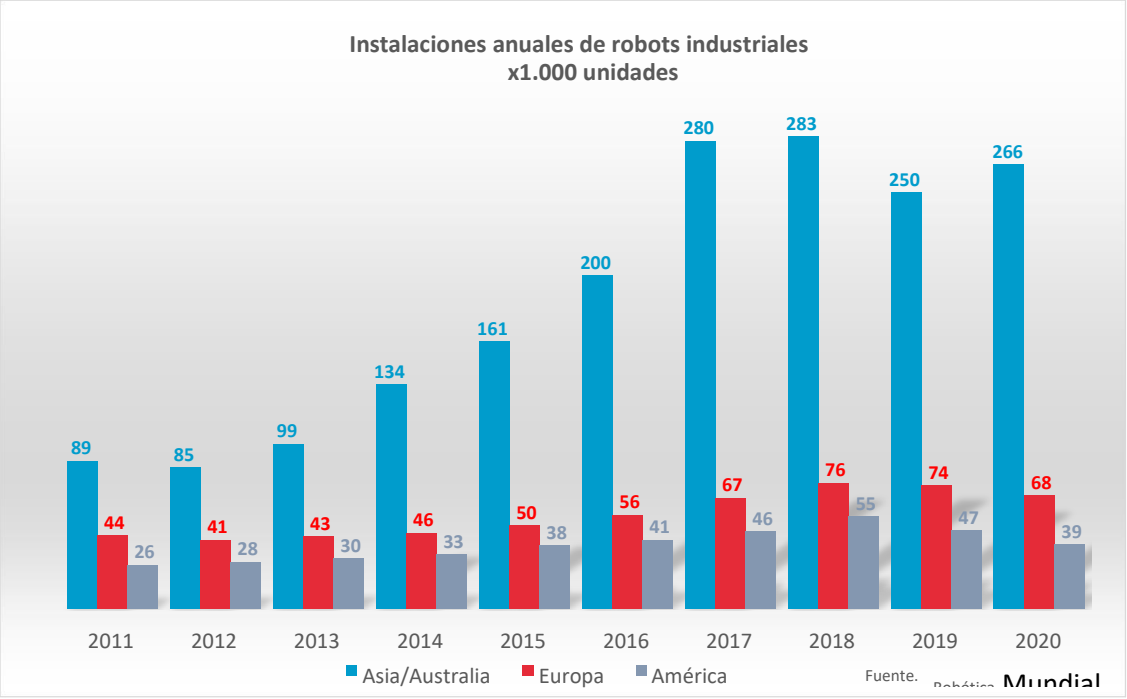


Figura 1. Instalaciones anuales de robots industriales

Asia, como se puede observar en la gráfica, es el mercado de robots industriales más grande del mundo, debido a sus industrias automoción y electrónica. Situando los datos más cercanos, Alemania es el primer mercado de robots en Europa (quinto en el mundo) y España es el décimo mercado de robots más grande del mundo, aunque como en muchos países europeos en 2020 se redujeron el número de instalaciones.

Se espera que en período que comprende desde el año pasado hasta 2024 las tasas de crecimiento sigan aumentando, aunque con una pequeña recesión en este año por la situación internacional. Con estos datos macroeconómicos se puede remarcar la importancia a nivel global de los robots industriales y justificar la evolución ascendente de este sector, contextualizándolo tanto a nivel europeo como a nivel nacional.

2.1. Sistemas pick and place

Para entender mejor que es un sistema pick and place, lo primero haremos será traducirlo, literalmente significa recoger y colocar, por lo que podemos relacionar todas las operaciones que tengan que ver con coger una pieza u objeto y ubicarlo en otro lugar. Las tareas de este sistema abarcan selección, clasificación, paletizado y empaquetado de cualquier pieza, objeto o producto terminado.

Debido a la naturaleza de estas actividades, poco ergonómicas y repetitivas dentro de la cadena de producción, hace que sean pesadas mental y físicamente para los operarios. Prácticamente ninguna industria se escapa estas tareas logísticas, de ahí la necesidad de automatizarlas para optimizar la producción.

Algunas de las ventajas que aportan este tipo de automatización con brazos robóticos son:

- Seguridad operativa.
- Ahorro de tiempo gracias a altas velocidades de trabajo.
- Precisión milimétrica.
- Minimización de los riesgos al personal por las condiciones descritas anteriormente.
- Fácilmente reprogramables y modulables.
- Reducción de costes de mano de obra.

Lo cierto es que todas estas ventajas tienen un inconveniente, este es la inversión inicial y la amortización de la instalación, pero eso ya queda fuera del estudio de este TFM.

3. Metodología

A continuación, se describirá apartado por apartado la metodología secuencial que se ha seguido para abordar este proyecto. Empezaremos con la introducción al software de programación.

3.1. Introducción al Programa RobotStudio

El software RobotStudio de ABB es una herramienta de programación y simulación de robots industriales, según la empresa es, “la herramienta de programación offline más utilizada del mundo”. Una de las ventajas que comenta ABB, es que su software permite la programación offline con lo cual tenemos la mejor manera de maximizar la eficiencia y los recursos ya que permite que se pueda programar el robot sin interrumpir la producción y desde cualquier ordenador sin estar en la propia instalación.

Esta característica es posible a que el software mencionado se basa en ABB VirtualController que es una copia idéntica del software real utilizado en sus robots. Esto nos va a permitir realizar simulaciones realistas, utilizando los programas realizados para descargarlos en robots reales con los mismos archivos de configuración que los utilizados en el programa.

Para instalar el programa se podían utilizar, en principio, dos opciones, la versión trial de 30 días que ofrecen o la licencia de UMH, se empezó con la primera a desarrollar el programa y cuando se caducó la versión trial se intentó poner la licencia de la UMH, pero no fue posible, ni desde casa, ni desde la universidad. Así que se decidió instalar otra versión de prueba en otro ordenador con algunos problemas en la migración del trabajo realizado.

La versión instalada fue la más reciente que había en ese momento. La versión 2022.1, a fecha actual ABB publicó una nueva versión, 2022.2 que corrige algunos bugs. La versión del VirtualController es la RobotWare 6.13.04.

En esta imagen se puede ver una pequeña descripción de las partes principales de RobotStudio.

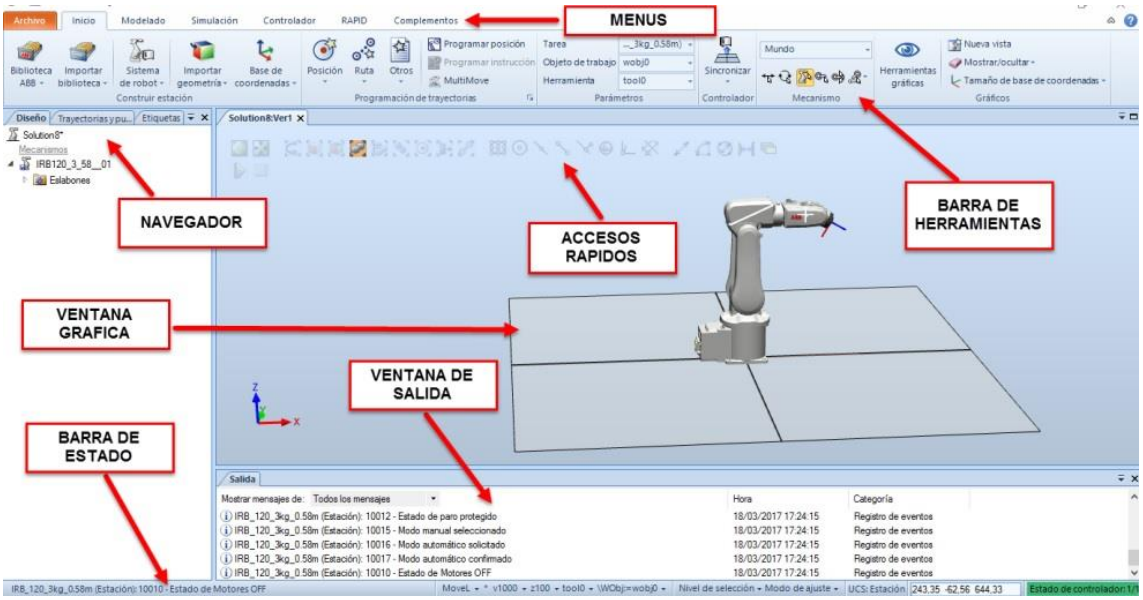


Figura 2. Organización RobotStudio

3.1.1. Creación de una nueva estación

Lo primero que se deberá realizar es crear una estación nueva (archivo del programa) para empezar a trabajar con el, en el programa se pueden escoger tres opciones para crear el archivo.

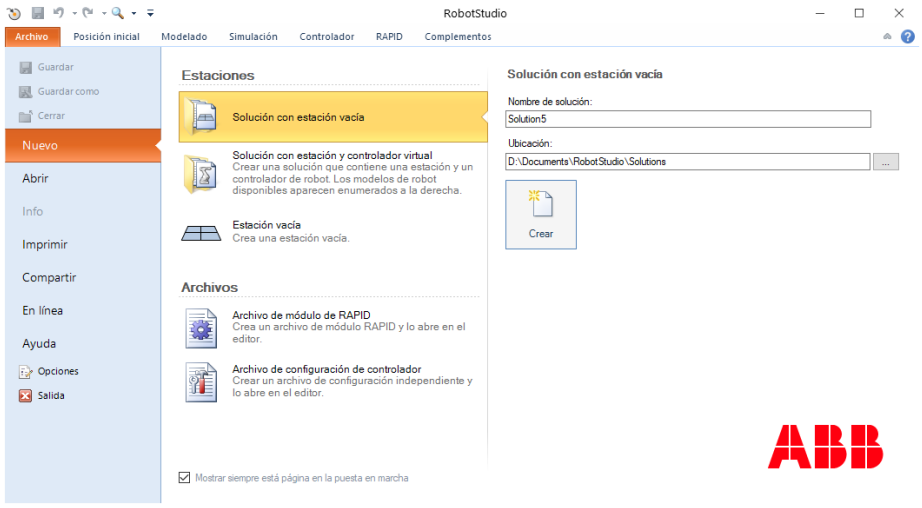


Figura 3. Menú creación de estación

- Solución con estación vacía: crea una estación vacía con la estructura de archivos necesarios.

- Solución con estación y controlador del robot: crea una solución que contiene una estación y un controlador del robot. Antes de crearla podemos elegir el robot y el controlador que se vaya a utilizar

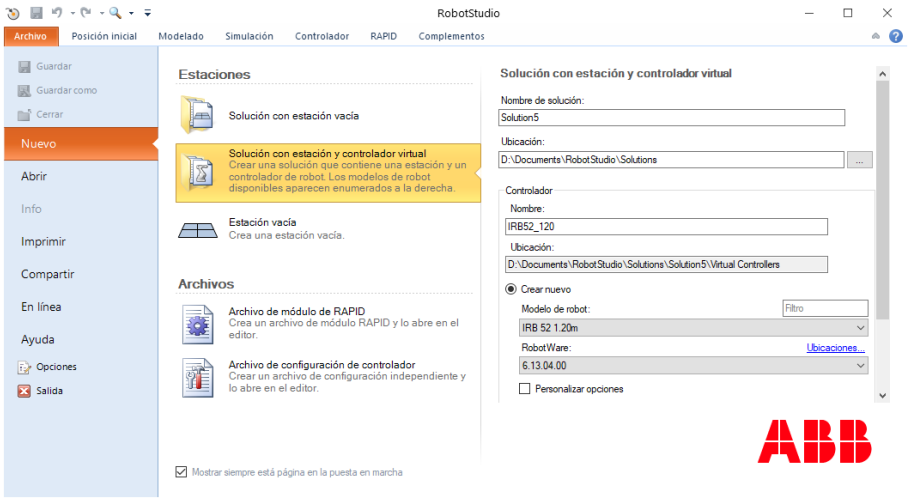


Figura 4. Menú solución con estación y controlador

- Estación vacía: Crea una estación vacía sin ningún componente.

Nosotros crearemos una estación vacía para tener un poco más de flexibilidad y abordar el proceso desde cero, por lo tanto, crearemos una estación vacía y elegiremos el robot desde el menú Biblioteca ABB,

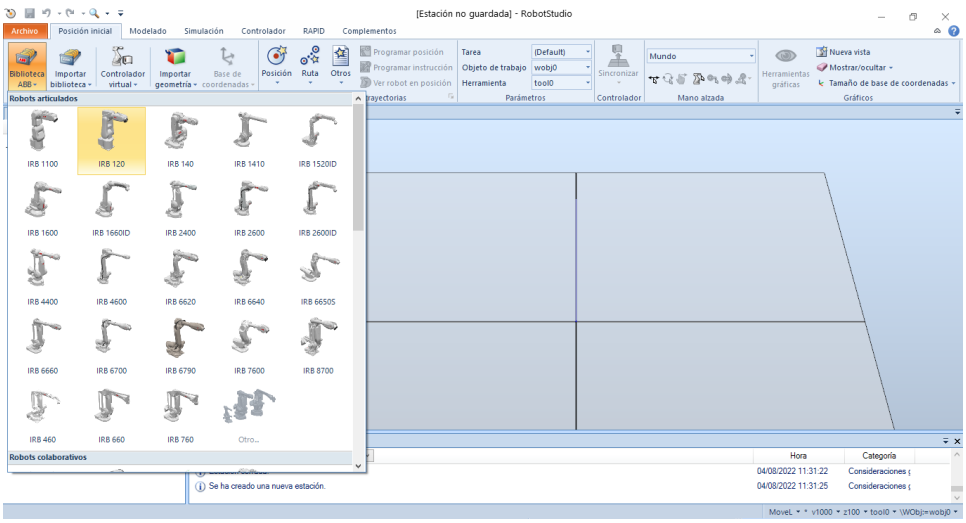


Figura 5. Elección del robot desde la Biblioteca ABB

3.1.2. Configuración de la herramienta de un robot

Para utilizar nuestro robot se necesita acoplarle una herramienta al efector final, este tipo de herramienta puede ser cualquiera que se vaya a utilizar, una ventosa, un gripper, un soldador...

En nuestro caso utilizaremos varias herramientas, un gripper o pinza y una ventosa, con ellas se podrá realizar todo el proceso de clasificación, pick and place y paletización. Las herramientas se pueden extraer de la biblioteca y configurar o crear desde cero, en nuestro caso utilizaremos las ya creadas y las configuraremos.

Para el robot clasificador, se ha elegido el robot IRB 120 y la herramienta ventosa que se importará de la biblioteca.

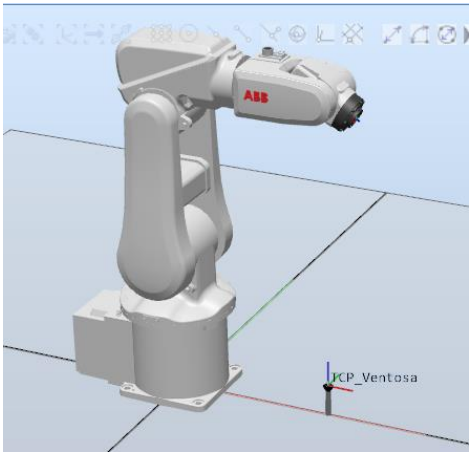


Figura 6. Robot IRB120 y herramienta ventosa

Una vez que tenemos el robot y la herramienta, se procederá a acoplarla al robot para que funcione como efector final. Una de las maneras de acoplarla es arrastrar la herramienta del menú de diseño al robot y automáticamente tendremos en nuestra muñeca del robot la herramienta preparada.

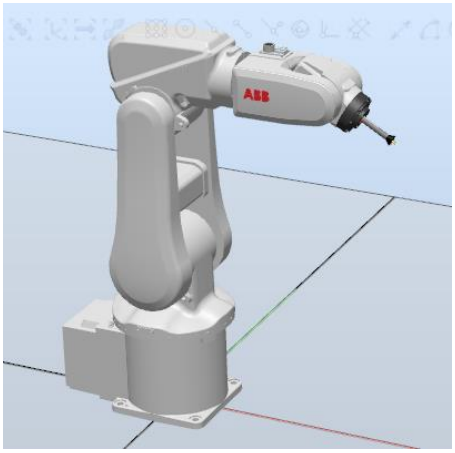


Figura 7. Robot IRB120 con ventosa conectada

Una vez conectada la ventosa, el programa nos sacará por la consola de salida el mensaje de que la herramienta se ha conectado al robot.

3.1.3. Creación del controlador de un robot

Con el robot y la herramienta acoplada nos falta dotar al robot de posibilidad de movimiento, para ello necesitamos configurarle un controlador para que facilite realizar todas las trayectorias y movimientos que le programemos.

Para ello pincharemos en controlador virtual y le diremos que nos cree un controlador desde diseño.

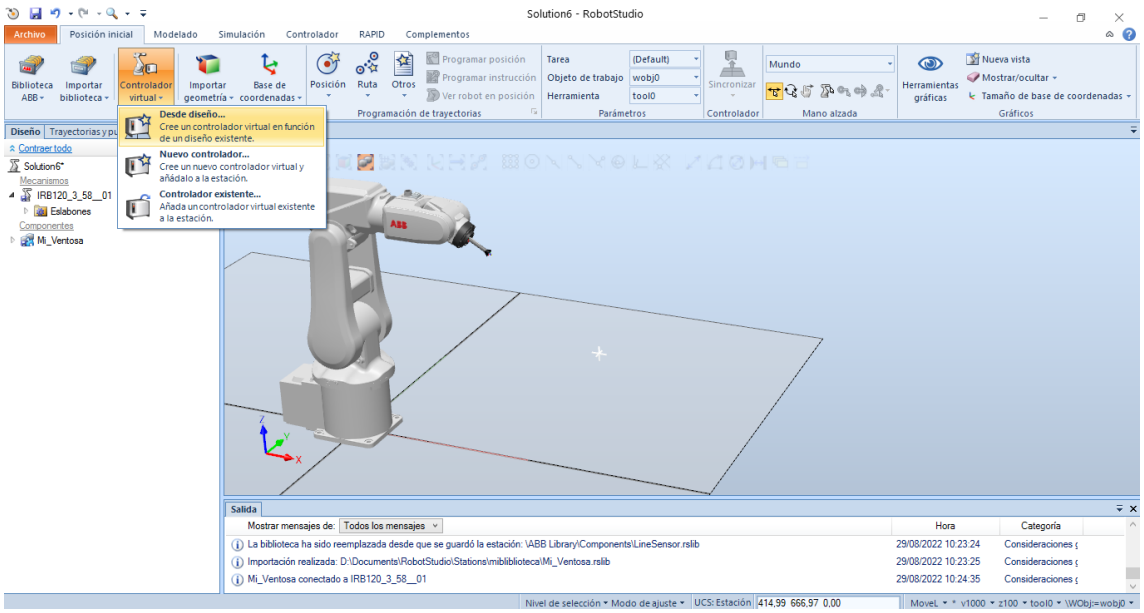


Figura 8. Asignación del controlador virtual al robot

Ajustaremos las opciones que nos interesen, las que existen por defecto de momento nos valdrán y crearemos el controlador necesario para poder dotar de movimiento a nuestro brazo robótico.

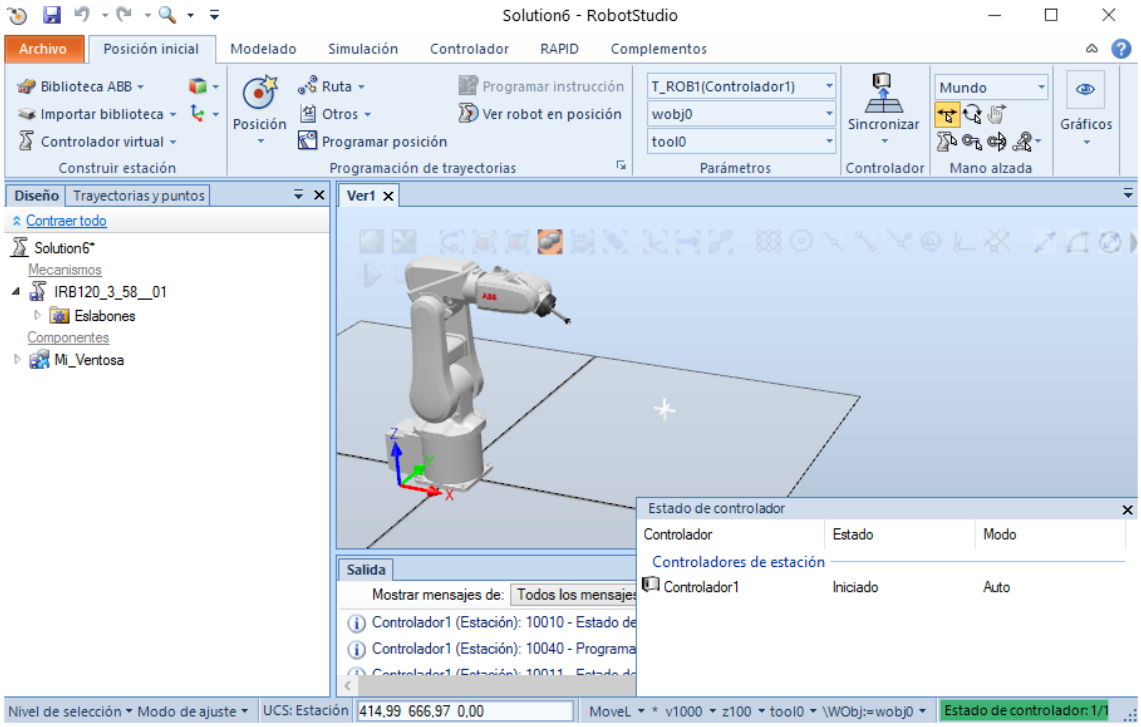


Figura 9. Estado del controlador activado

Una vez creado el controlador nos aparecerá como que ya lo tenemos instalado, indicando después de unos segundos que ya está listo para poder trabajar con el, como se ve en la esquina inferior derecha.

3.1.4. Creación de planos de trabajo y posiciones.

Para poder, en nuestro caso, coger un objeto con la ventosa, deberemos coger un plano de trabajo y una posición de la pieza donde la ventosa llegará para poder agarrarla y moverla. Para ello vamos a crear una pieza de test, desde el menú modelado -> crear sólido -> tetraedro, le daremos las medidas y se nos creará la pieza.

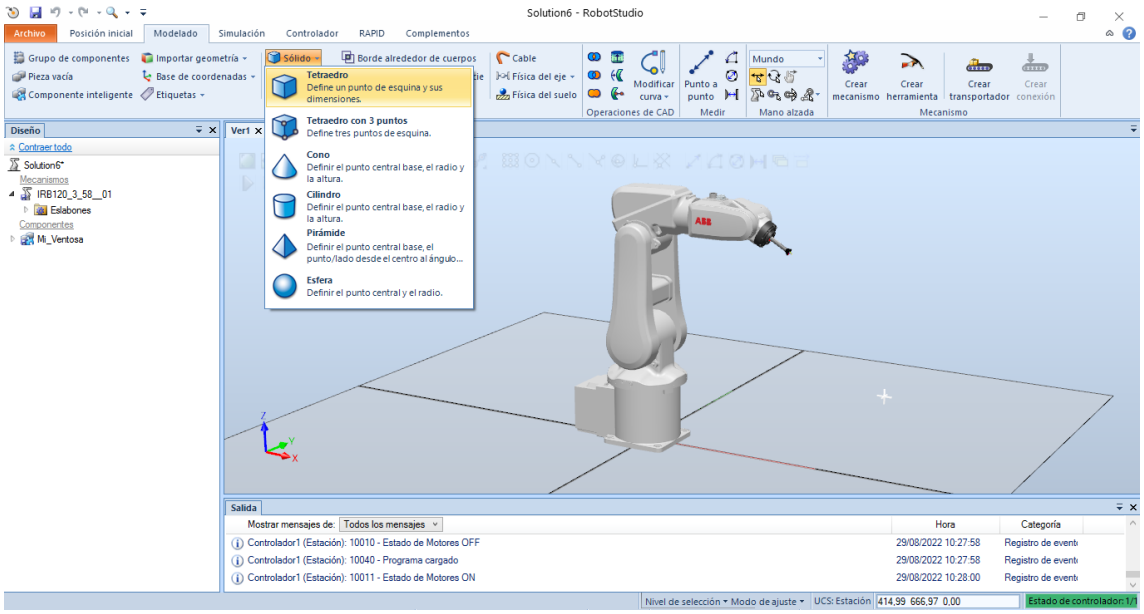


Figura 10. Menú insertar Sólido->Tetraedro

Una vez creada la pieza, si no le decimos lo contrario aparecerá en el origen de coordenadas del mundo 0, 0 ,0. En nuestro caso como no hemos movido el robot la pieza no se verá pues estará en la misma posición que el origen del robot, así que la moveremos para poder trabajar con ella.

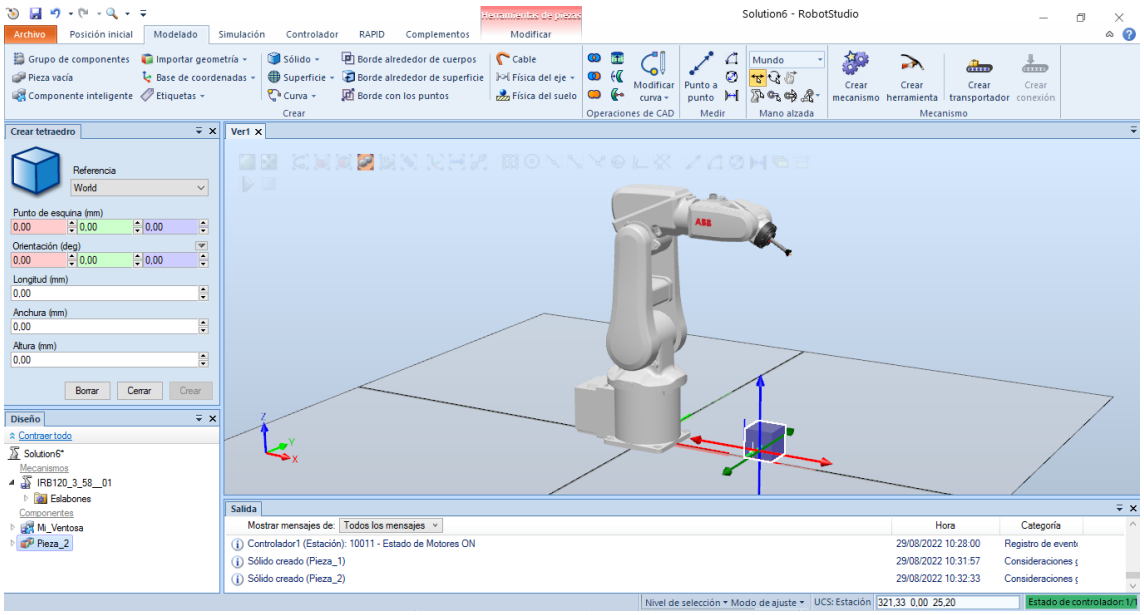


Figura 11. Colocación de la pieza en el plano de trabajo

Se puede observar que todos los componentes que vamos añadiendo a nuestra estación nos aparecen el árbol de diseño que aparece a la izquierda.

Para crear la primera posición donde cogerá la pieza, clicaremos en Posición -> crear Posición, comprobaremos que esta seleccionada una de las casillas de posición e iremos al menú de acceso rápido y seleccionaremos la opción ajustar a centro, la cual nos permitirá fijar el punto medio de la cara que escojamos del tetraedro, haciendo clic en ella, por manipulabilidad y estética elegiremos la cara superior del tetraedro. Estas acciones se pueden ver en las siguientes imágenes.

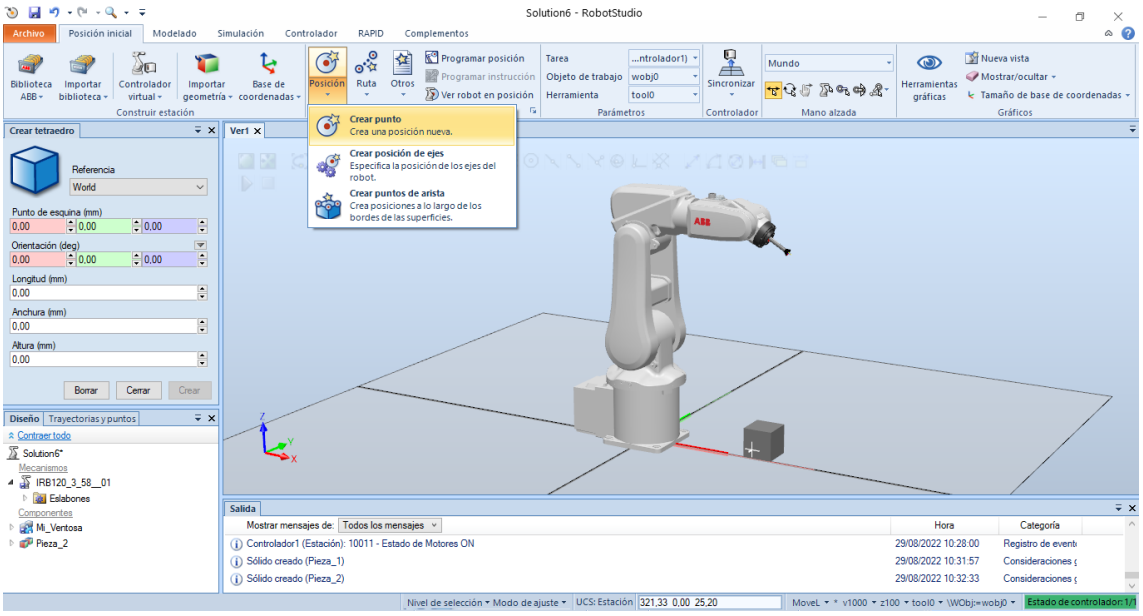


Figura 12. Menú crear punto o target

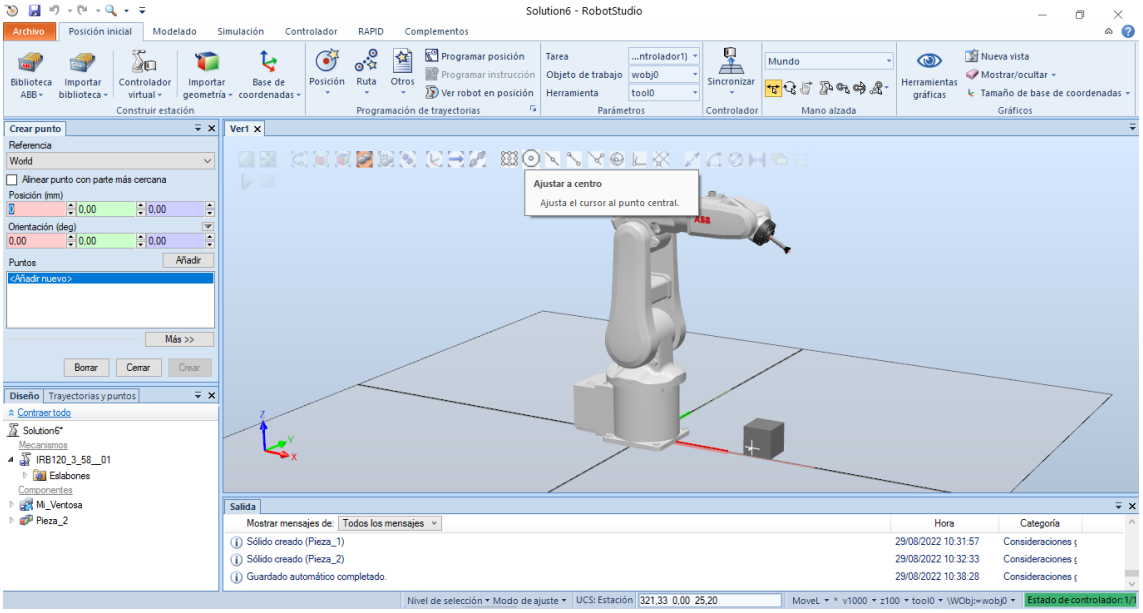


Figura 13. Ajustar a centro para obtener la posición

Nos fijaremos antes de seleccionar la cara superior, que este seleccionada la casilla de la posición en la pestaña de Crear punto, como se ve en la imagen siguiente. Así nos aseguraremos de crear el punto en la posición correcta.

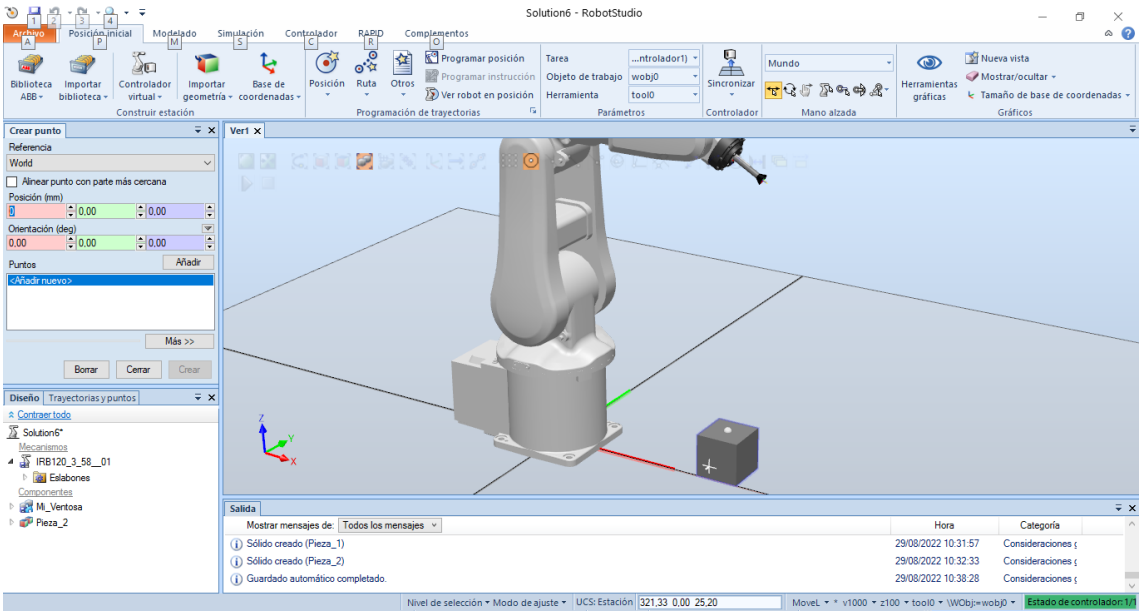


Figura 14. Creación de punto en la pieza

Es importante reseñar los WorkObjects, cuando se crea un punto va asociado a un WO, si no le decimos nada se creará en el WO por defecto. La gran ventaja de este sistema para encapsular targets o puntos es que, si creamos un WO para cualquier acción, por ejemplo la recogida en una cinta y los puntos los guardamos en ese WO nos va posibilitar poder mover el objeto asociado, por ejemplo la cinta y que los puntos vayan asociados a la posición respecto a la cinta, con lo que cualquier modificación de la posición de la cinta no nos supondrá volver a calibrar o a tomar esos puntos ya creados.

A la hora de fijar posiciones o targets se debe comprobar que la orientación de la posición creada es alcanzable por la herramienta o TCP. Cuando se crea una posición en un punto que sea de interés deberemos comprobar que el efector final puede orientarse con esa posición creada.

Si nos fijamos, después de crear el punto con la orientación por defecto, tenemos activa la pestaña de Trayectorias y puntos, se nos ha creado nuestro primer target (Target_10) o punto con la posición que le hemos dado.

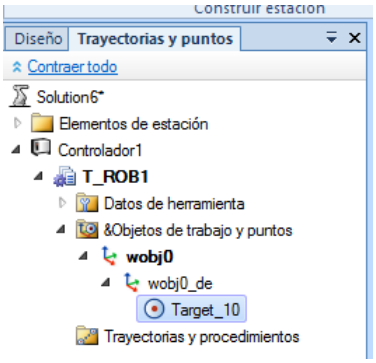


Figura 15. Creación de punto, Target_10

Ahora vamos a comprobar si ese punto es alcanzable para el robot, para ello pulsaremos en la pestaña Posición Inicial -> Ver robot en posición.

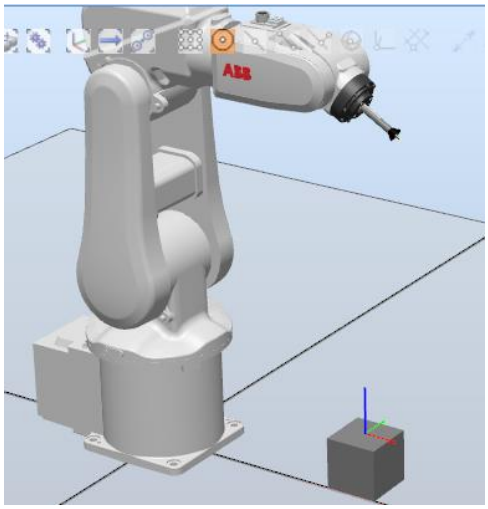


Figura 16. Orientación del punto configurado

El programa nos avisara si es alcanzable o no por el efector final, si no es alcanzable deberemos reorientarla para que pueda ser alcanzable. En este caso el robot no se posiciona y nos muestra por salida un mensaje diciendo:

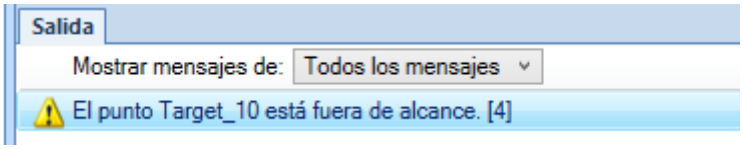


Figura 17. Mensaje en la consola de salida. Punto fuera del alcance

Por lo tanto, con esa orientación es imposible que el robot mediante la ventosa pueda coger la pieza. Para resolver este problema vamos a girar el target creado para reorientarlo. De la siguiente manera, pincharemos con el botón derecho en el Target_10 que hemos creado, y le daremos a Modificar posición -> Girar, como muestra la imagen

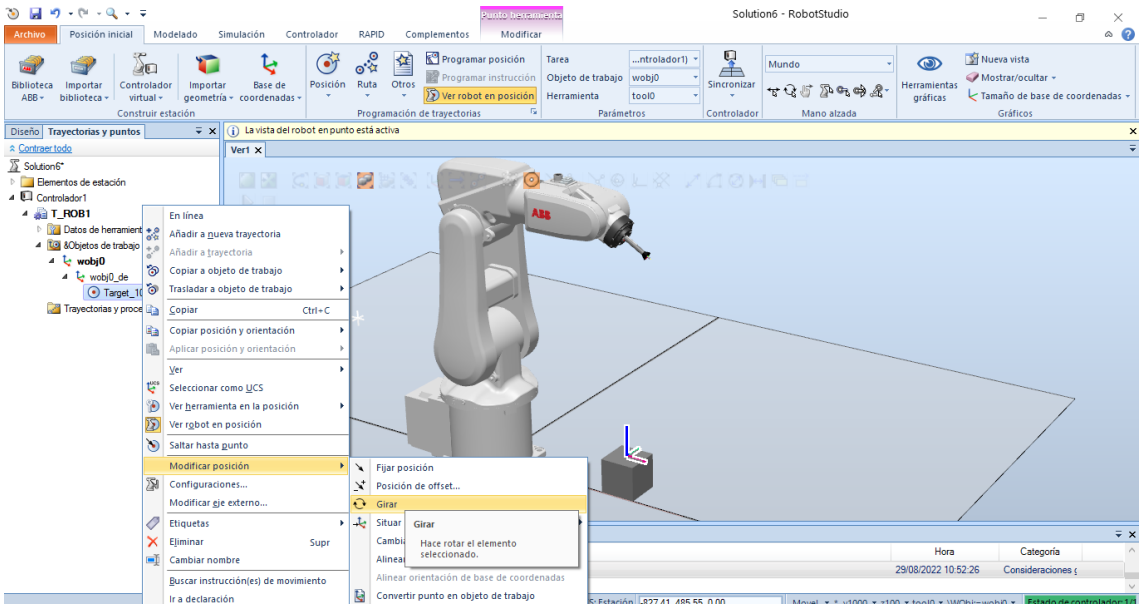


Figura 18. Modificación de la orientación del punto

Lo giraremos 180º en X y quedará como se muestra en la imagen inferior.

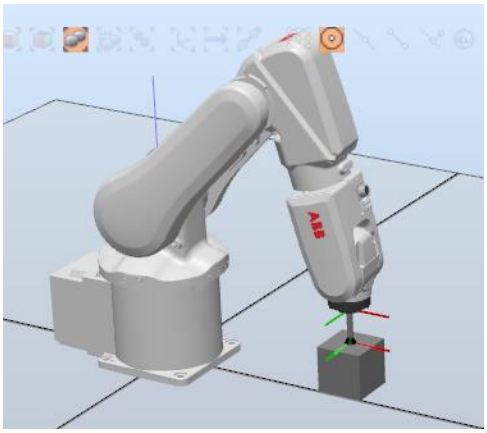


Figura 19. Punto reorientado. Robot en posición

Una vez reorientada y comprobado que puede ser alcanzable el punto podremos copiar esa orientación y aplicarla en los sucesivos puntos que hayamos creado.

Procederemos de la siguiente forma, botón derecho Target_10 -> Copiar posición y orientación -> Mundo relativo,

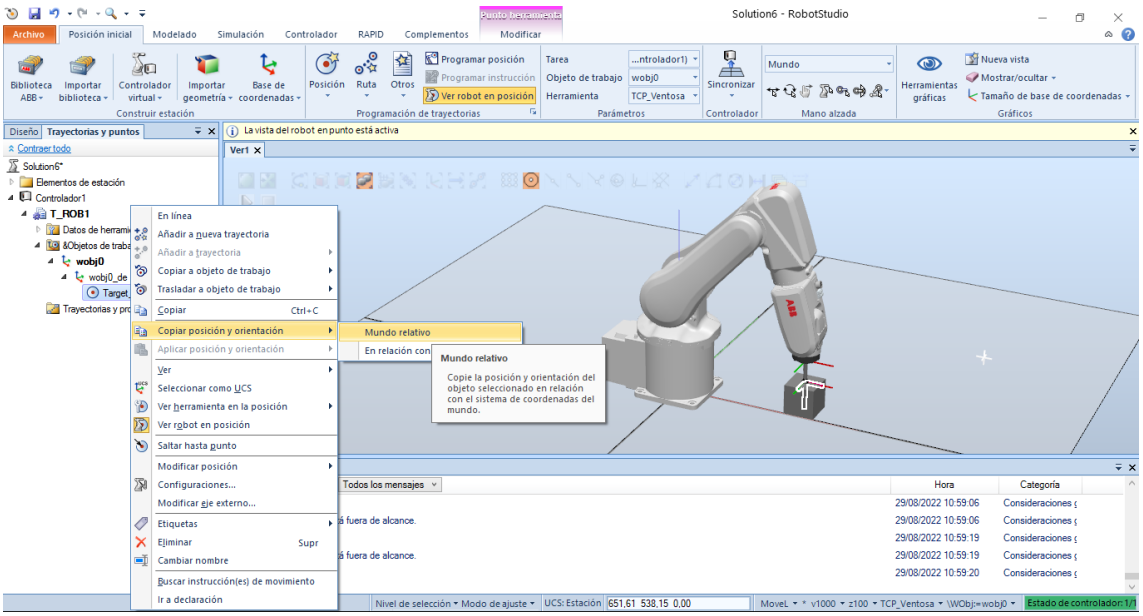


Figura 20. Menú copiar orientación y posición

Esta acción nos ahorrará mucho trabajo ya que cada vez que creemos un punto nos hará falta reorientarlo. A la hora de pegarlo podemos aplicar tanto orientación como posición, ya que cuando se copia se copian los dos vectores, posición y/o orientación, nos aseguraremos cuando vayamos a pegarlo que solo pegamos o apliquemos la orientación. Una vez que ya sabemos crear puntos o target podemos realizar las trayectorias.

3.1.5. Creación de trayectorias

Sabiendo crear posiciones o targets, las trayectorias serán muy fáciles de realizar, solo tendremos que pinchar en el menú trayectorias, crear una nueva e ir añadiendo targets o puntos ya creados.

En este ejemplo se utilizará la opción de crear trayectoria vacía, una vez creada, arrastraremos las posiciones o targets a la trayectoria colocándolas por el orden que nos interese.

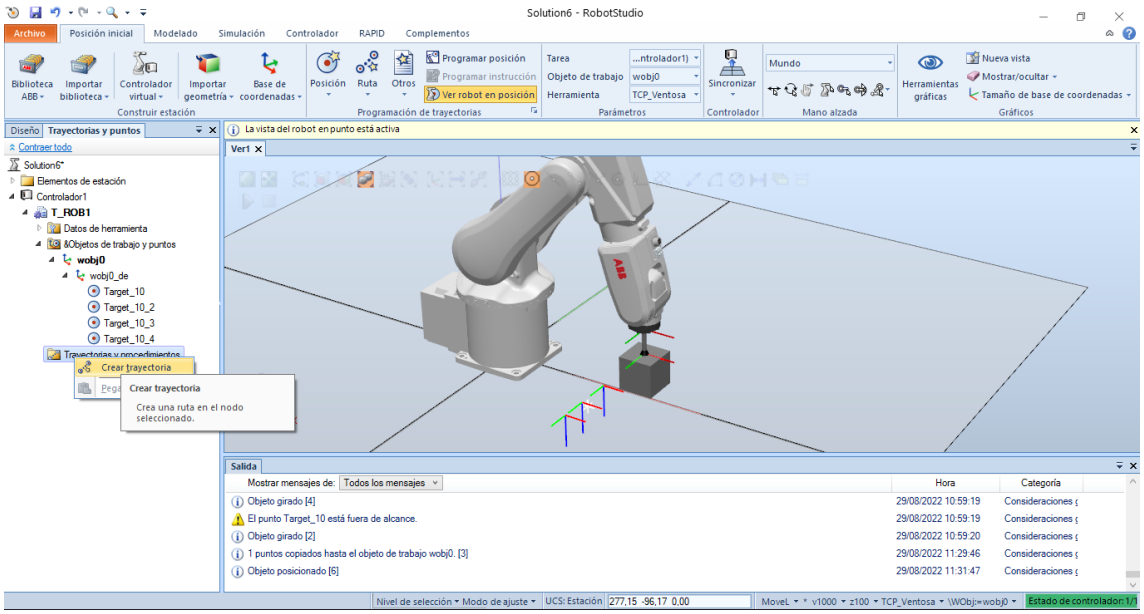


Figura 21. Puntos creados. Crear trayectoria

En la siguiente figura se puede ver como hemos arrastrado los targets establecidos con anterioridad a la trayectoria creada, así como, en amarillo, el recorrido de la trayectoria.

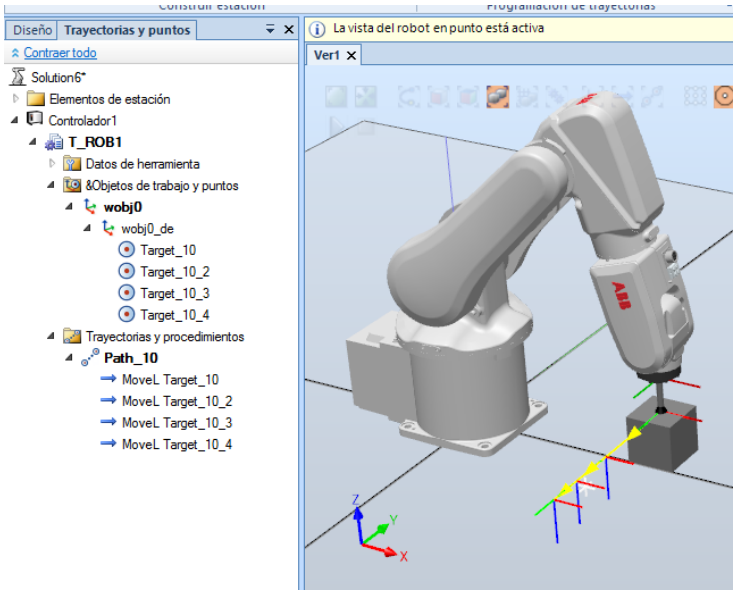


Figura 22. Trayectoria creada

A la hora de crear las trayectorias hay que tener en cuenta los diferentes parámetros que se pueden configurar, las posiciones de esta se irán estableciendo con los parámetros que existan por defecto, en la parte de inferior

de la ventana del programa se muestran de manera resumida los que están por defecto. Los distintos ítems que se pueden configurar son los siguientes:

- Tipo de movimiento:
    - MoveL: movimiento lineal, es decir desplazamiento del extremo del robot hasta el punto indicado siguiendo una línea recta.
    - MoveJ: movimiento de junta o eje, desplazamiento hasta el punto indicado sin garantizar cuál es la trayectoria seguida (no existe coordinación de velocidad entre los distintos ejes). Esta instrucción tiene un coste computacional menor que la anterior.
    - MoveC: Desplazamiento circular definiendo varios puntos.
- Velocidad del movimiento, caracterizado por la letra v y una cifra en mm/s, ejemplo v500.
- Exactitud o precisión del movimiento, caracterizado por la letra z y una cifra. Ejemplo z100, cuanto menor sea el número más exacto será el movimiento, pero mayor será el coste computacional de realizarlo. Es decir, la cifra indica el error en mm. Cuando necesitemos que el movimiento sea lo más exacto posible configuraremos “fine”, por ejemplo, ante de realizar acciones como agarrar una pieza o soltarla.

Todos los parámetros configurados podremos editarlos posteriormente para ajustarlos a las necesidades que más nos interesen. Se podrá realizar a través de la opción Editar instrucción pinchando con el botón derecho en la instrucción o instrucciones que nos interesen.

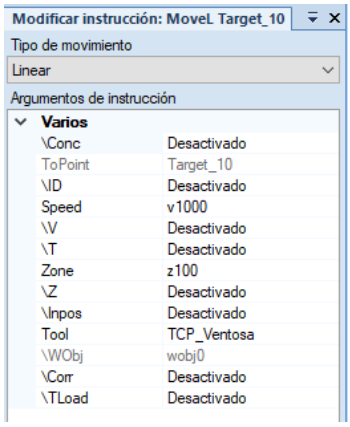


Figura 23. Modificar parámetros instrucción

O también a través del código RAPID que veremos a continuación.

3.1.6. Introducción a RAPID

RAPID es un lenguaje de programación de alto nivel que se utiliza para controlar los robots industriales de ABB. RAPID fue presentado por primera vez en 1994, sustituyendo al antiguo lenguaje de programación ARLA. Las siglas significan Robotics Application Programming Interface Dialogue.

Las características de este lenguaje son:

- Diferentes tipos de rutinas:
    - Procedimientos, usados como subprogramas
    - Funciones, que pueden devolver un valor de un tipo específico o que pueden utilizar como argumento de entrada una instrucción.
- Expresiones aritméticas y lógicas
- Manejo automático de errores
- Programas modulares
- Multitarea.

Dentro de un programa escrito en RAPID podemos encontrarnos tres partes diferenciadas:

- Datos del programa: se sitúan al principio del archivo, en el se definen posiciones, variables etc.
- Rutina o procedimiento principal llamado PROC main(), donde se programa la secuencia del programa.
- Subrutinas o procedimientos, espacio dónde se desarrollan los procedimientos secundarios que serán llamados por el procedimiento principal. Se utiliza la instrucción PROC nombreprocedimiento para iniciarlo y END PROC para finalizarlo.

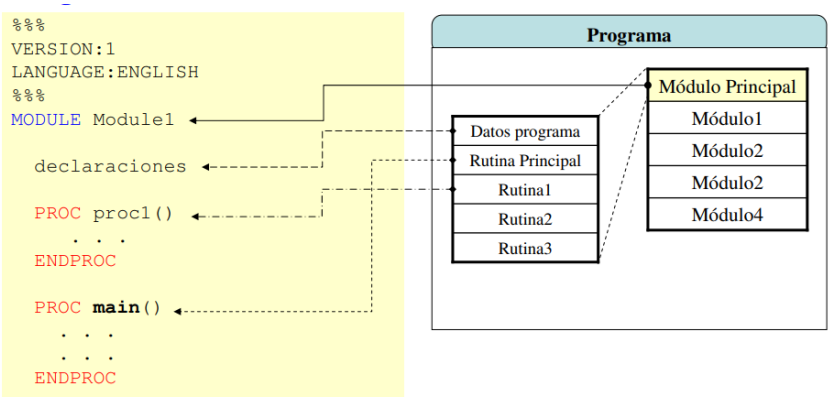


Figura 24. Esquema programa en RAPID

Una vez creado las posiciones y las trayectorias necesarias deberemos sincronizar esos datos con RAPID, para posteriormente poder modificarlos o ampliarlos en código. Deberemos pinchar en el menú de Posición Inicial -> Sincronizar -> Sincronizar con RAPID. Seleccionaremos todos los datos para asegurarnos de sincronizarlo todo y pulsaremos en aceptar.

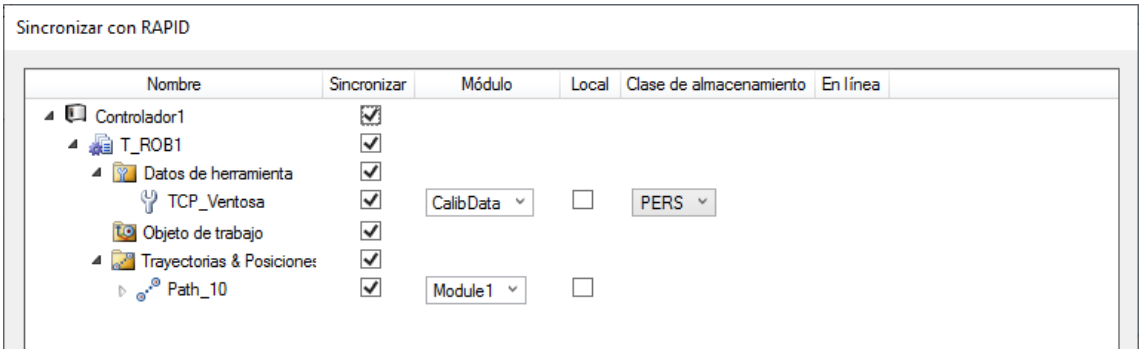


Figura 25. Menú sincronización con RAPID

En la imagen siguiente podemos ver el código generado en RAPID a partir de los puntos y la trayectoria creada anteriormente.

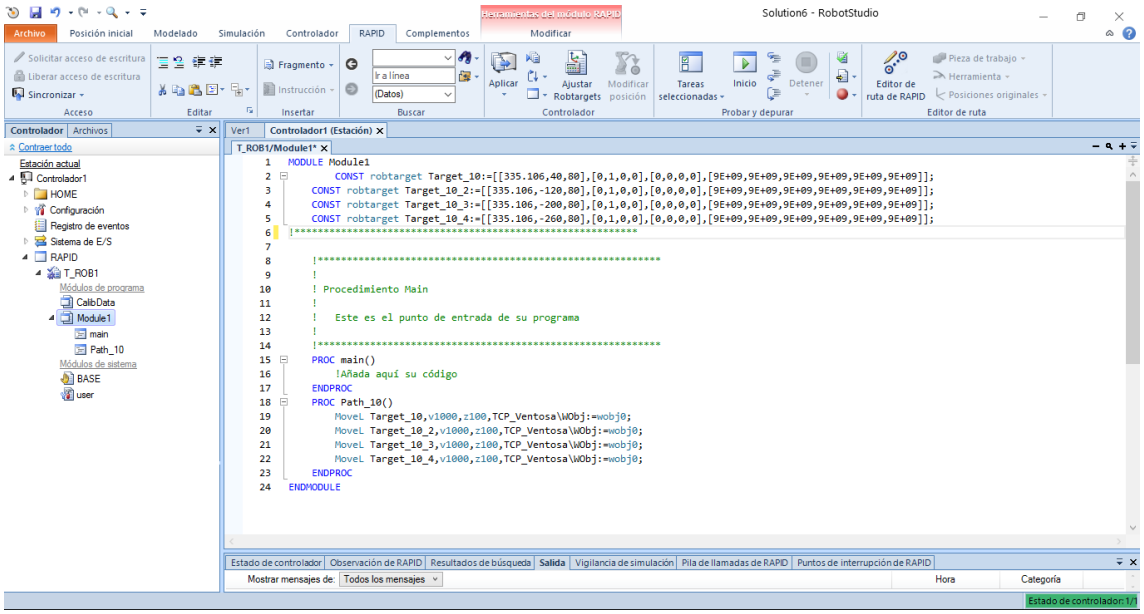


Figura 26. Programa ejemplo con RAPID

Cuando estemos en RAPID si hacemos cualquier modificación deberemos realizar la operación contraria para que se quede reflejado en la estación, este caso, dentro del menú de RAPID, pincharemos en Sincronizar -> Sincronizar con estación. De esta manera cualquier cambio realizado de las dos opciones que tenemos estará actualizado y podremos revisarlo desde cualquier parte del programa

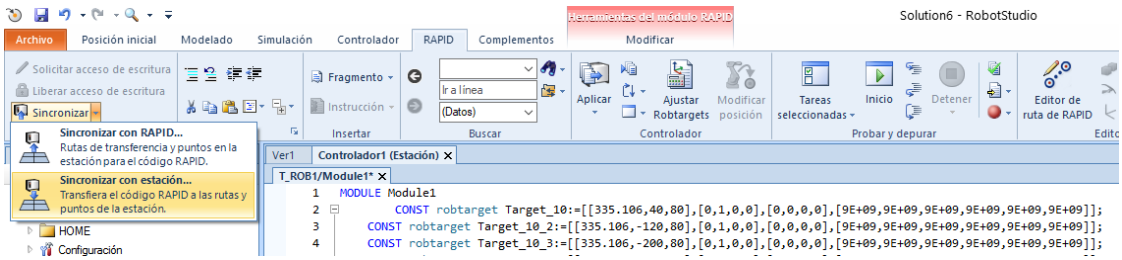


Figura 27. Menú sincronización de código con estación

3.2. Diseño y Programación de una Estación

Una vez visto parte del desarrollo y ítems que debemos configurar se puede decir que ya podríamos empezar a diseñar una estación, a parte de lo descrito anteriormente se va a describir tres puntos que completarían todo el proceso de diseño de la estación.

3.2.2. Creación y diseño de los Smart Components

Los Smart Components o componentes inteligentes son elementos creados o diseñados por nosotros que podemos dotar de cierto dinamismo o movimiento para realizar acciones o eventos que sean de interés, estos elementos se controlaran por señales o propiedades del sistema.

Los componentes inteligentes se añaden desde la pestaña Modelado -> Componente inteligente. Dentro de los componentes se pueden añadir bloques para configurarlos con nuestras especificaciones, estos bloques se clasifican en distintas categorías,

- Señales y propiedades: Aquí se encuentran elementos como puertas lógicas, multiplexores, contadores, temporizadores, expresiones matemáticas…
- Primitivos paramétricos: En esta categoría están los componentes necesarios para modificar o crear de forma automática sólidos y líneas.
- Sensores: Se encuentran todos los sensores para automatizar nuestra estación, sensores de colisión, sensor de línea, sensor de plano, sensores volumétricos…
- Acciones: Con estos componentes podemos conectar/desconectar objetos, crear copias de objetos, ocultar/mostrar/eliminar objetos etc, serán de mucha utilidad para nuestra estación.
- Manipuladores: Con estos componentes podremos mover un objeto de diferentes formas, lineal o angularmente, podremos también en caso de interesarnos mover los ejes del robot a una posición predeterminada.
- Controlador: En esta categoría solo existe un componente que nos permite establecer u obtener una variable de Rapid.
- Física: Podemos controlar las propiedades físicas de un objeto o de los ejes del robot.
- PLC: Este módulo se utiliza para crear conexiones con un OPC.
- Realidad Virtual: Módulos para trabajar con realidad virtual.
- Otros: En esta última categoría encontramos módulos muy interesantes, como cola de objetos, comparador de objetos, el módulo random, el módulo de eventos de la simulación …

Los componentes más interesantes se describirán en el apartado de diseño e implementación de la estación. Para crear un componente lo primero que debemos hacer es clicar en Componente inteligente como se ha comentado anteriormente, una vez hecho esto nos aparecerá en el árbol de la izquierda dentro del apartado de diseño. Como necesitamos diseñarlo pincharemos con el botón derecho -> editar componente.

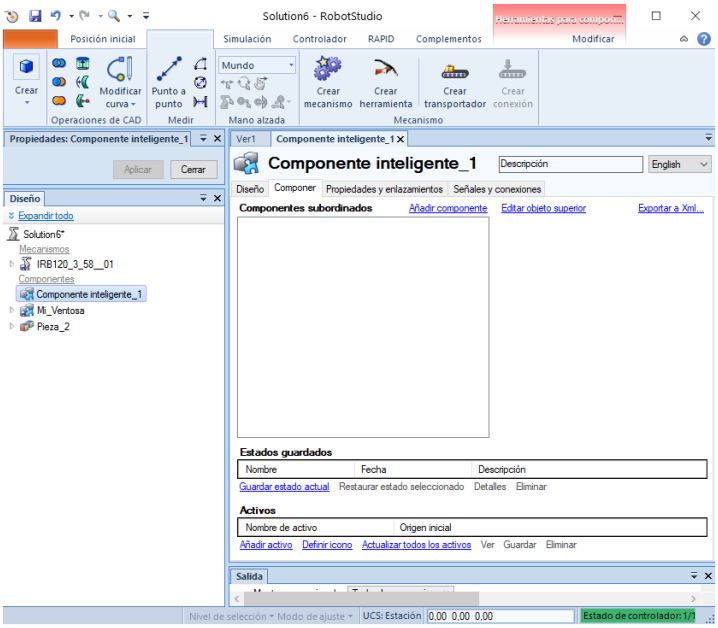


Figura 28. Ventana configuración de componente inteligente

Nos aparece una ventana con diferentes pestañas: diseño, componer, propiedades y enlazamientos y señales y conexiones, nosotros utilizaremos principalmente las dos primeras, Diseño y Componer, desde la segunda iremos agregando los componentes que nos interesen para, por ejemplo, diseñar una cinta transportadora o un gripper y desde la primera, Diseño, crearemos todas las conexiones y relaciones con otros componentes y con el controlador para poder controlar ese Smart Component desde RAPID.

3.2.3 Creación de señales E/S

Para poder controlar desde fuera de la estación parámetros o eventos podremos crear señales de E/S, en nuestro caso será para controlar la simulación, pero si fuera en una estación física podríamos asignar esas entradas y salidas a un módulo real del controlador. Las E/S que podemos utilizar son digitales y analógicas, nosotros utilizaremos principalmente las digitales.

Para acceder al módulo de E/S, tenemos que situarnos en la pestaña del Controlador, en el árbol de controlador situado en la izquierda, pincharemos en I/O System y nos aparecerá la ventana como se muestra en la imagen.

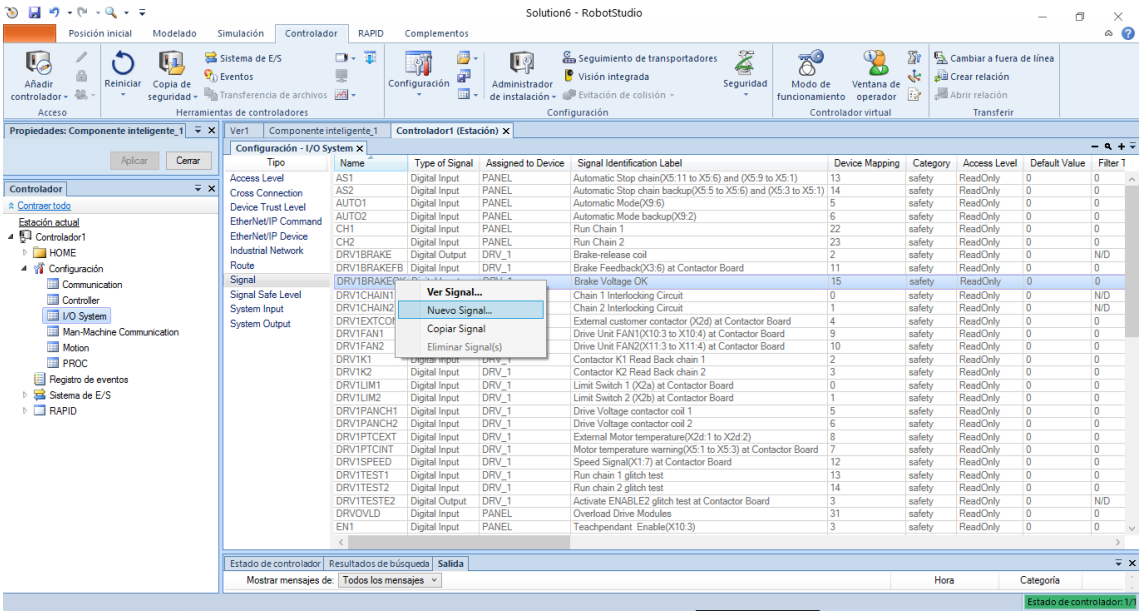


Figura 29. Creación de señales en el controlador virtual

Pincharemos como se observa con el botón derecho -> Nuevo Signal y podremos crear nuestra entrada o salida. El dialogo que nos aparece para configurarla es el que se muestra a continuación.

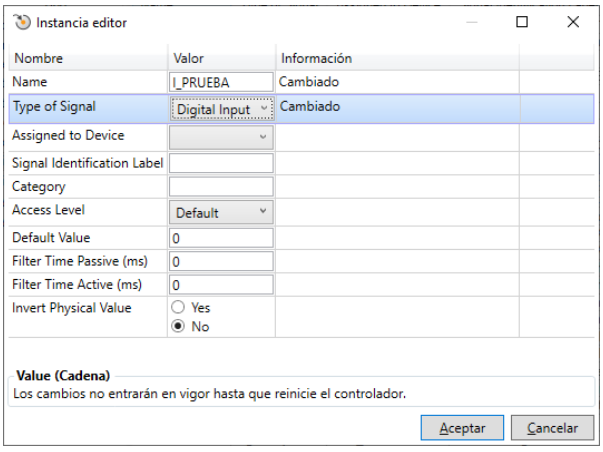


Figura 30. Diálogo de configuración E/S

Hay que indicar, aunque el programa nos lo recuerde, reiniciar el controlador para que los cambios tengan efectos, si no lo hacemos, no podremos trabajar

con las E/S creadas. Lo podremos hacer desde el mismo menú en el que nos encontramos pinchando en Reiniciar.

3.2.4. Lógica de estación

Una vez creadas las señales de E/S que nos interesen en el controlador y creado y diseñado el Smart Component tendremos que conectarlos de forma adecuada para hacerlos funcionar correctamente coordinados, es decir, diseñaremos la lógica de la simulación conectando los componentes, señales y conexiones. Para acceder a la lógica de la estación, pincharemos en la pestaña Simulación -> Lógica de la estación.

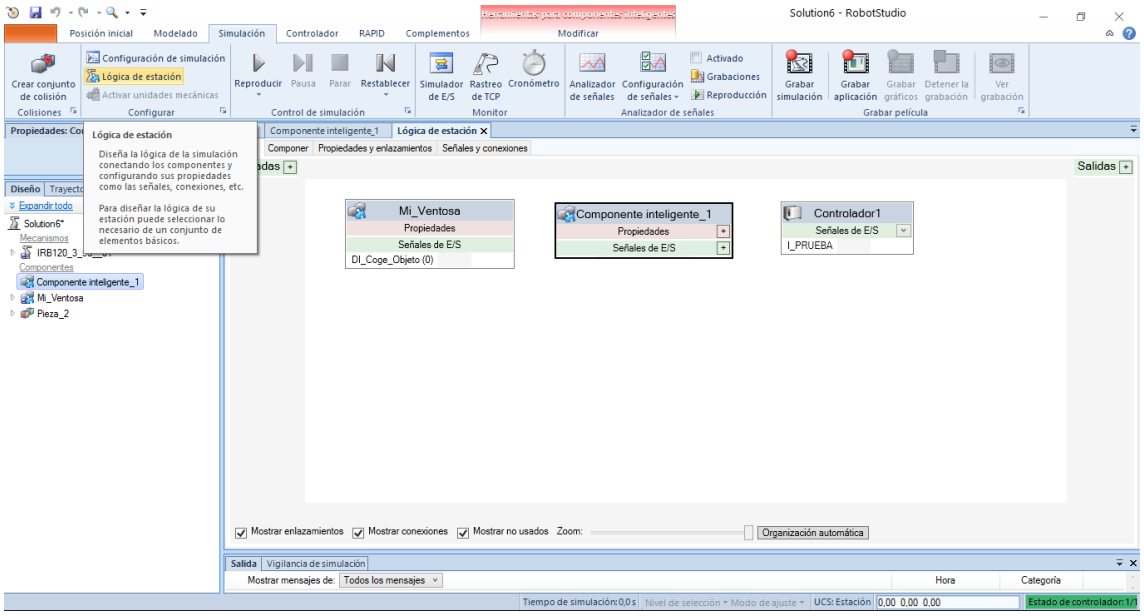


Figura 31. Pestaña 'Lógica de estación del controlador'

En el caso de la imagen superior tenemos los bloques que hemos ido creando, en primer lugar, tenemos el Smart Component de Mi_Ventosa, tenemos el bloque del otro SC, el que hemos creado sin nada dentro y tenemos el bloque del Controlador donde nos aparece, porque se lo hemos dicho, la entrada digital que configuramos. A partir de aquí habría que interconectar todos los bloques para el funcionamiento, como hemos ido creando los componentes para explicarlos no tienen funcionalidad por lo tanto no los conectaremos, este apartado se abordará en el de diseño e implementación de la estación.

3.3. Simulación de una estación

Llegados a este punto ya estaríamos preparados para lanzar la simulación de todo lo realizado para poder comprobar que todo funciona según lo diseñado y programado, y por supuesto, para hacer los ajustes necesarios.

Antes de simular nuestro sistema, comprobaremos en la pestaña de Configuración de la simulación que los parámetros coincidan con lo que queremos realizar. Desde esta pestaña podremos ajustar los escenarios de la simulación y estado inicial de los objetos que vamos a simular.

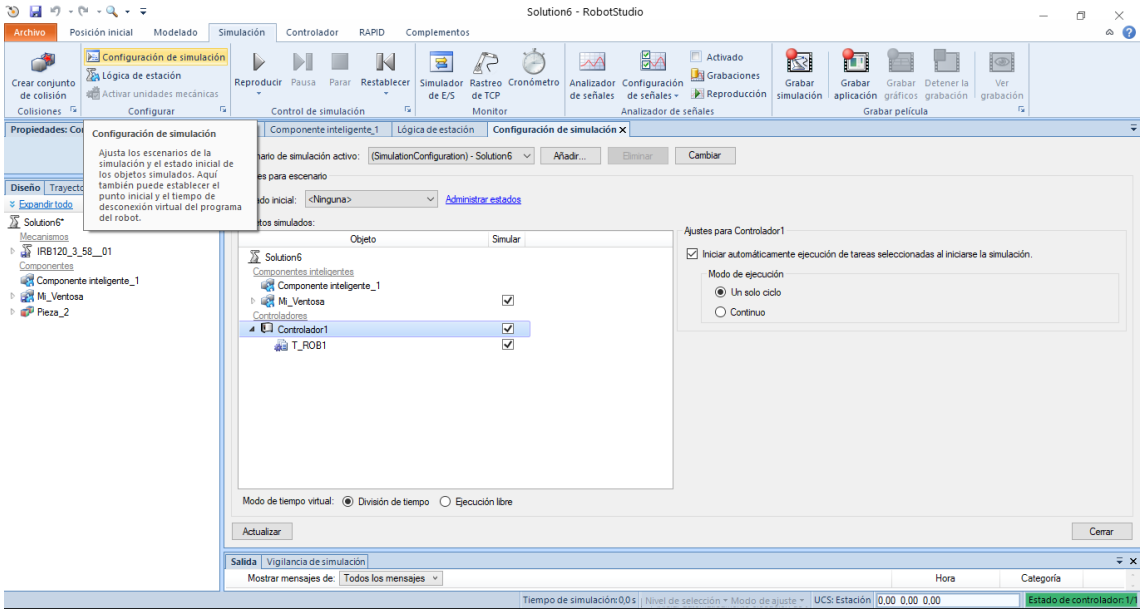


Figura 32. Pestaña 'Configuración de la estación'

En esta ventana configuraremos básicamente si queremos un solo ciclo de la simulación o si lo queremos continuo, tendremos el check activo de "Iniciar automáticamente la ejecución de las tareas seleccionadas al iniciarse la simulación" para poder escoger entre estas dos opciones. En la simulación de nuestra estación utilizaremos el modo continuo, en este ejemplo para ver la trayectoria del robot sin más, pincharemos en un solo ciclo.

Ahora sí, para simular nuestro sistema, dentro de la pestaña de Simulación pincharemos en el botón de Reproducir. Ahora ya estamos preparados para implementar nuestro proyecto.

4. Implementación de una estación clasificadora con pick and place.

Se va a explicar el diseño e implementación de una estación clasificadora con pick and place, se irá explicando todo el proceso de desarrollo y justificando las decisiones tomadas.

La estación se compone de los siguientes componentes principales

- Robot clasificador
- 3 robots para el pick and place (uno por cinta de descarga)
- Cinta transportadora principal
- 3 cintas de descarga

La estación clasificadora tendrá como elemento principal de generación de objetos una cinta clasificadora. En esta cinta se generarán los tres tipos de piezas que se serán clasificados por altura por los sensores del final de esta. Una vez sensada la altura de la pieza que le llegue al robot clasificador, este lo dejará en su cinta de descarga correspondiente. Cada cinta de descarga (tres en total) tendrá al final un robot que realizará el pick and place.

4.1. Elección del Robot

El modelo de robot que se va a utilizar es el IRB 120 de ABB, este modelo es el robot industrial multiusos más pequeño de ABB, tiene 6 ejes, pesa 25 kg y puede soportar una carga de 3 kg, 4 kg si la muñeca se encuentra en posición vertical. Tiene un alcance de 58 cm y ha sido diseñado específicamente para industrias de fabricación que utilizan una automatización flexible.

Se ha decidido utilizar este robot por ser el más versátil y pequeño de los robots industriales existentes, también por ser uno de los más económicos, aunque esta razón carezca de importancia para el proyecto desarrollado. Aunque el objeto del proyecto no tenga una finalidad específica, ya que se trata de aprender y conocer a manejar el RobotStudio con cierta soltura, una posible utilización podría ser la clasificación de pequeños componentes electrónicos.

Se muestran las especificaciones más relevantes que se completarán en el anexo III,

Figura 33. Imagen real IRB 120

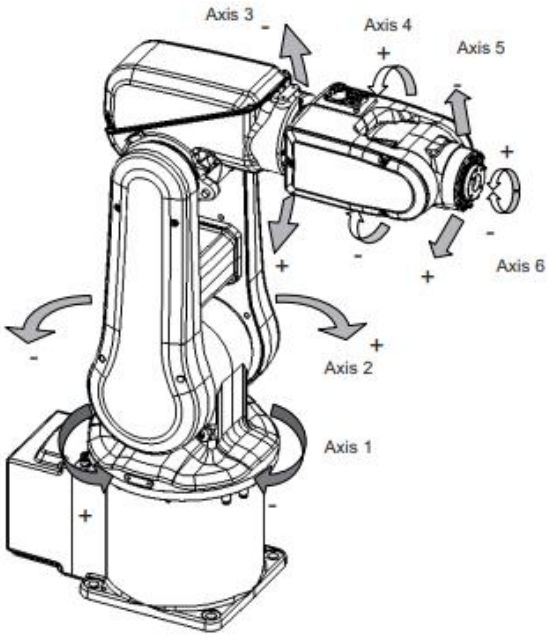


Figura 34. Ejes del IRB 120

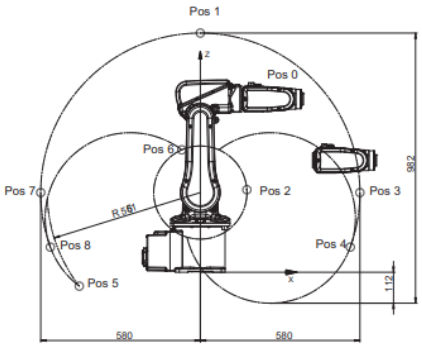


Figura 35. Área de trabajo del IRB 120

Posi-ción	Posición en el centro de la muñeca (mm)		Ángulo (grados)	
	X	Z	Eje 2	Eje 3
A	302 mm	630 mm	0°	0°
B	0 mm	870 mm	0°	-77°
C	169 mm	300 mm	0°	+70°
D	580 mm	270 mm	+90°	-77°
E	545 mm	91 mm	+110°	-77°
F	-440 mm	-50 mm	-110°	-110°
G	-67 mm	445 mm	-110°	+70°
H	-580 mm	270 mm	-90°	-77°
J	-545 mm	91 mm	-110°	-77°

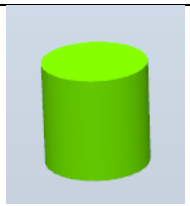
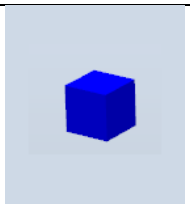
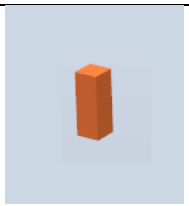
Figura 36. Posiciones del IRB 120

4.2. Elección de las herramientas

En principio se podía haber realizado todo el desarrollo con la herramienta ventosa, pero se ha querido mostrar el trabajo realizado con un gripper o pinza también como herramienta. El robot clasificador y dos del pick and place utilizaran una ventosa por las características de las piezas y el que paletiza las piezas más pequeñas utilizará un gripper programado como Smart Component.

4.3. Características de las piezas

La estación clasificará tres tipos de piezas con las siguientes características:

		
Cilindro Radio: 60 mm Altura: 120 mm	Tetraedro Ancho: 100 mm Largo: 100 mm Alto: 100 mm	Tiza Ancho: 30 mm Largo: 30 mm Alto: 80 mm

4.4. Creación de la Estación en RobotStudio.

El proceso de creación de la estación clasificadora y pick and place, se ha realizado de forma modular. Hemos seguido el patrón explicado en el apartado tres. Primero se puesto el robot clasificador y se le ha enlazado su controlador, después se ha diseñado el SC de la cinta de clasificación, se ha diseñado una cinta de descarga y se ha copiado para, en total, tener 3 cintas de descarga. Para hacer el programa más modular se han añadido los tres robots de paletización y se han asignado a otro controlador, controlador12.

A partir de aquí y con todos los SC diseñados se ha realizado la designación de puntos y posteriormente las trayectorias. A continuación, se describirán los SC diseñados.

4.4.1. SC Ventosa

Una de las herramientas que se ha utilizado es una ventosa. Los bloques utilizados para su diseño son los siguientes:

LineSensor: Se ha diseñado un sensor de línea muy pequeño para poder detectar la pieza que entra en contacto con la ventosa, este bloque activo cuando se lo digamos a través de RAPID mediante la entrada DI_Coge_Objeto, nos proporcionará el elemento en contacto, SensedPart(), que se lo pasaremos a los bloque de Attacher y Detacher.

Figura 37. Herramienta ventosa conectada

LogicGate: Este bloque de puerta lógica está configurado como puerta NOT, nos permitirá activar el bloque Detacher para soltar la pieza cuando pongamos a cero la entrada DI_Coge_Objeto.

Attacher: Nos permite coger la pieza en contacto con el sensor siempre y cuando la entrada DI_Coge_Objeto este activa.

Detacher: Nos permite soltar la pieza en contacto con el sensor cuando la entrada DI_Coge_Objeto este inactiva.

Podemos comprobar en la imagen las conexiones de los diferentes bloques explicados,

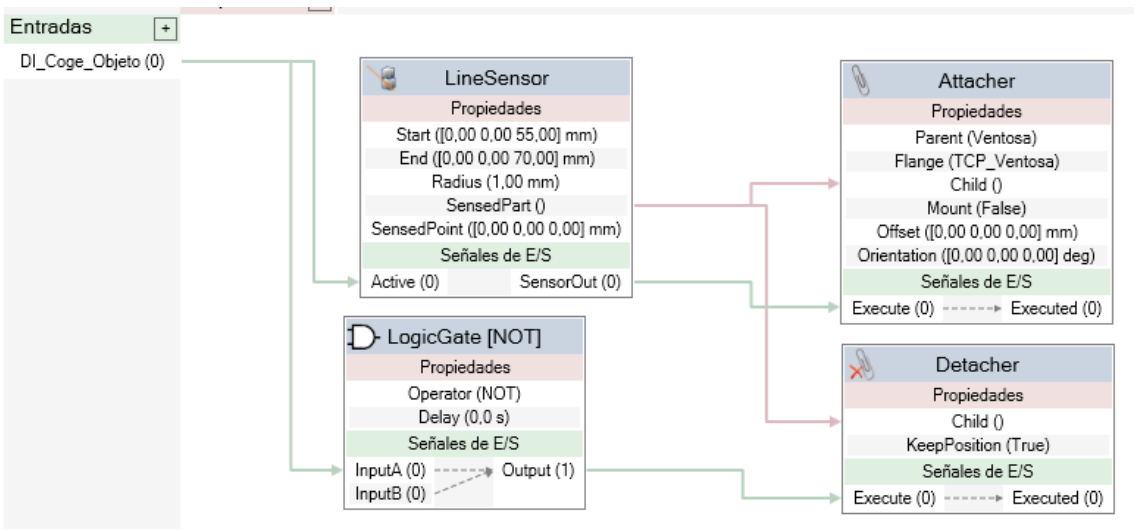


Figura 38. Conexionado Smart Component Ventosa

4.4.2. SC Pinza ABB (Gripper)

Al igual que todos los Smart Components, lo primero que tenemos que hacer es crearnos uno nuevo. Después iremos a la biblioteca y añadiremos la pinza ABB y la introduciremos dentro del componente creado, hasta aquí ya tendremos la parte física, ahora falta dotarla de movimiento.

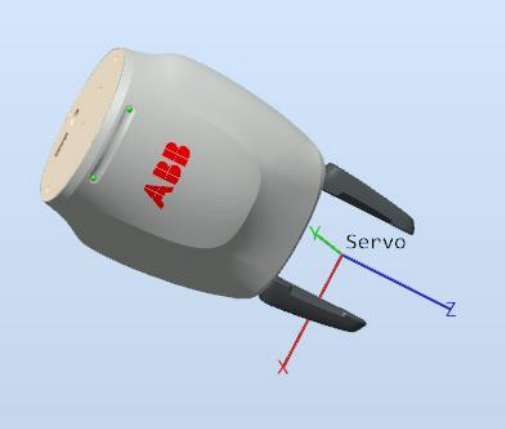


Figura 39. Pinza ABB sin conectar al brazo robótico.

En esencia la pinza diseñada es igual que la herramienta ventosa, solo tiene una diferencia reseñable, tiene una parte móvil o dinámica que es la pinza, esa funcionalidad, ya que de base es igual, se ha realizado con el bloque que se describe a continuación.

PosMover: Es un bloque que nos permite mover el eje de un mecanismo, en nuestro caso la pinza hasta una posición predefinida. Hemos definido dos posiciones abierto y cerrado para la pinza en dos bloques diferentes uno para cada posición como se puede ver en esquema de más abajo.

Antes de configurar nada del bloque tendremos que predefinir la dos poses, abierto y cerrado, debemos clicar con el botón derecho en árbol de diseño en Smart Gripper -> Modificar mecanismo. Crearemos dos nuevas poses, abierto y cerrado con las posiciones que nos interese, quedará así,

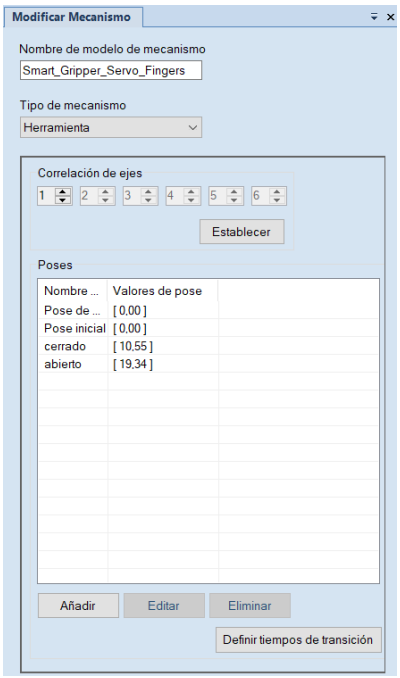


Figura 40. Configuración posiciones Smart Gripper

Con las dos poses predefinidas ya podremos ir a la ventana de nuestro componente inteligente y configurarlo. Primero le pasaremos que mecanismo queremos que mueva a través del dialogo de configuración, en nuestro caso de la pinza Smart Gripper Servo, después la pose, una de las que hemos predefinido y finalmente el tiempo que va a estar moviéndolo. Nosotros lo cambiamos a cero para que no ralentizase mucho la cogida y la descarga de la pieza. Habrá que hacer lo mismo, es decir, configurar otro bloque con la otra pose que nos ha quedado.

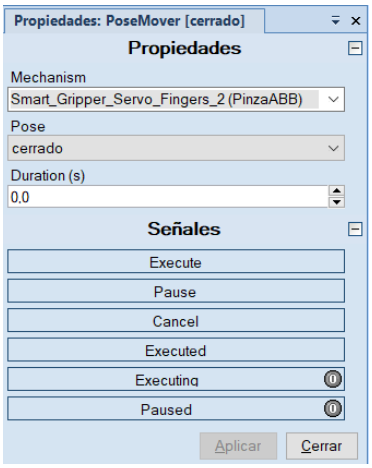


Figura 41. Configuración propiedades bloque PoseMover

Una vez configurados todos los bloques, nos quedará así el esquema de la pinza inteligente.

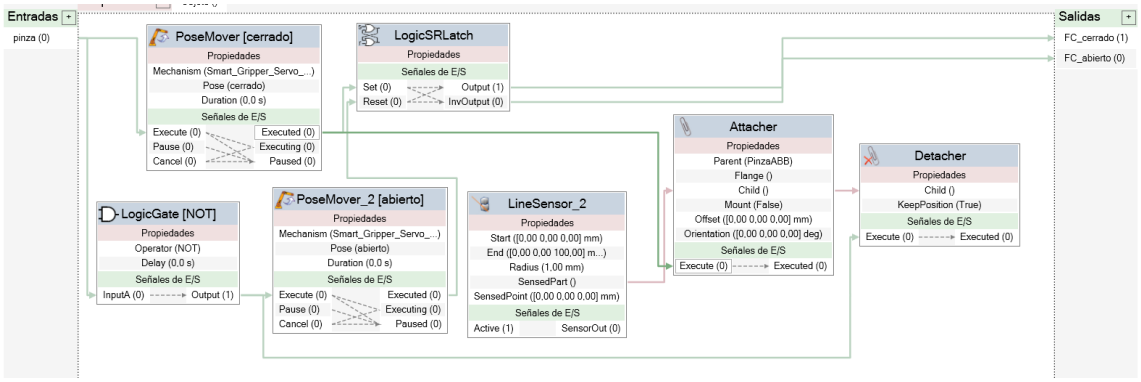


Figura 42. Conexionado Smart Component Pinza ABB

4.4.3. SC Cinta clasificadora

El componente inteligente de cinta clasificadora se ha diseñado desde cero, ya que las cintas existentes en la biblioteca del RobotStudio están pensadas para robots más grandes y no para el robot que hemos elegido.

En primer lugar, se ha diseñado en un programa CAD la estructura, que será el objeto visible de nuestro SC.

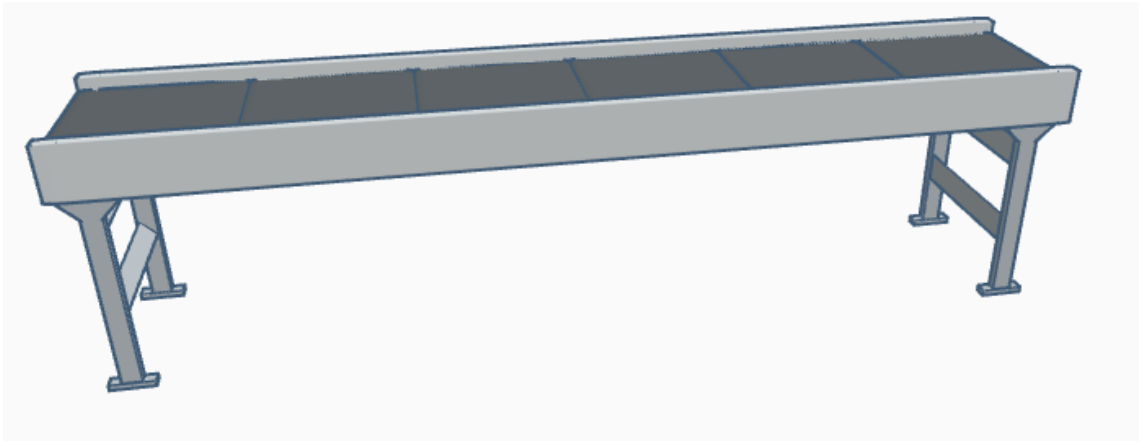


Figura 43. Diseño CAD de la cinta transportadora

Las medidas son 2000 x 378 x 500 mm

En el extremo izquierdo de la cinta aparecerán las piezas generadas y en el extremo derecho se recogerán con robot clasificador como se observa en la siguiente imagen.

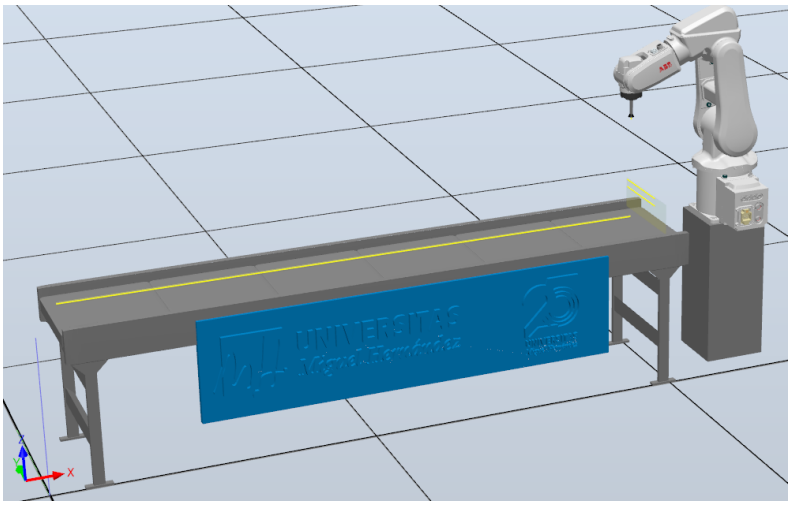


Figura 44. Smart Component cintra trasportadora

El SC tiene los siguientes componentes que posibilitan su funcionamiento:

LogicExpresion: Este bloque posibilita evaluar una expresión binaria. Se han utilizado tres bloques para generar una salida para cada bloque Source, dependiendo del valor de A y B. Se ha realizado la tabla de verdad y se ha calculado la función lógica, de dos entradas y 3 salidas, calculando esta para que en ningún momento puedan existir más de una salida con la misma combinación, el objetivo final es tener tres salidas que vayan a los tres bloques Source que son los que generan aleatoriamente las 3 piezas vistas anteriormente.

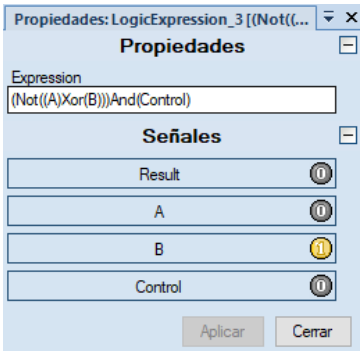


Figura 45. Configuración bloque LogicExpression

Los tres bloques tienen una entrada de control que van unidas a la entrada generada Control_Piezas, esta entrada irá conectada al controlador para controlar a través de RAPID cuando queremos suministrar piezas a la cinta transportadora.

Random: Existen dos bloques Random, este bloque genera un numero aleatorio, en nuestro caso entre 0 y 1. La entrada de este bloque la gestiona un Timer que lo activará cada cierto tiempo. Las salidas de los bloques irán a las entradas de los bloques de LogicExpression que serán binarias A y B. El bloque Random genera un número con decimales entre 0 y 1 y la entrada del bloque LogicExpression es binaria, al llegarle el número con decimales a la entrada binaria discretizará el valor como si de una puerta lógica se tratase, es decir por encima de un umbral será 1 y por debajo 0.

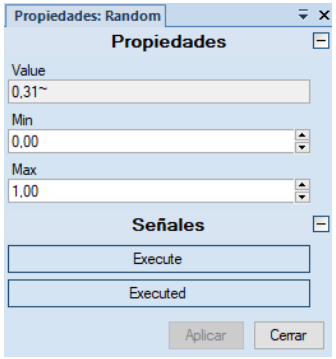


Figura 46. Configuración bloque Random

Timer: Es el temporizador encargado de temporizar el tiempo en que se generará una nueva pieza, está configurado que se active cada 6,5 segundos. Será la entrada de los bloques Random.

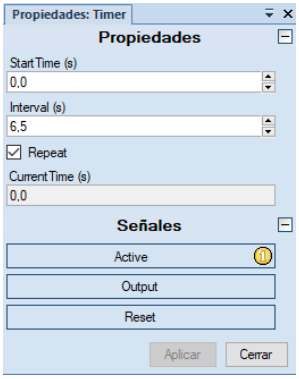


Figura 47. Configuración bloque Timer

Source: Tenemos tres bloques, uno para cada tipo de pieza. Este bloque genera una pieza preconfigurada cada vez que tenga una señal activa en su entrada. La entrada vendrá de los bloques LogicExpression comentados. La salida

Executed() de los tres bloques irá a la entrada de enqueue del bloque Queue. Tiene una salida llamada copy() que le pasará la pieza al bloque Queue

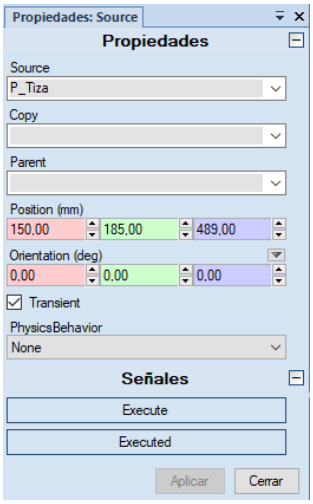


Figura 48. Configuración bloque Source

Queue: Este bloque pone en cola las piezas que vayan entrando por su entrada back() cuando la señal de su otra entrada enqueue este activa. Los bloques Source mandarán la pieza y la señal de poner en cola, enqueue(), cuando las vayan generando.

PlaneSensor: Este bloque es un sensor plano, si nos fijamos en la imagen de la cinta se puede observar en el extremo derecho al final de esta, está dispuesto verticalmente. Este sensor va a avisar, mediante su salida, que la pieza ha llegado al final de la cinta, la salida está conectada con la entrada dequeue() del bloque Queue para cuando llegue al final de la cinta se saque de la cola de piezas. También se ha generado una salida del SC de la cinta para poder trabajar con esa señal fuera del SC, la señal se llama Sensor1.

LineSensor: Es un sensor de línea, se han puesto dos al final de la cinta transportadora a diferente altura, estos dos sensores de línea tienen dos salidas, uno cada uno, que sensarán la altura de las piezas, están conectados a dos salidas digitales generadas del SC, llamadas SensorBajo y SensorAlto. Con las señales de estos dos sensores lograremos que el robot clasificador coloque la pieza en la cinta de descarga correspondiente.

LinearMover: Con este bloque conseguimos dotar a la pieza, en nuestro caso, a la de la cola (Queue), de movimiento hacia el eje y sentido que nosotros queramos, es lo que hace que las piezas se muevan sobre la cinta transportadora. Tiene una entrada de activación que siempre esta activa y mueve el objeto que este configurado, en nuestro caso en que este en Queue.

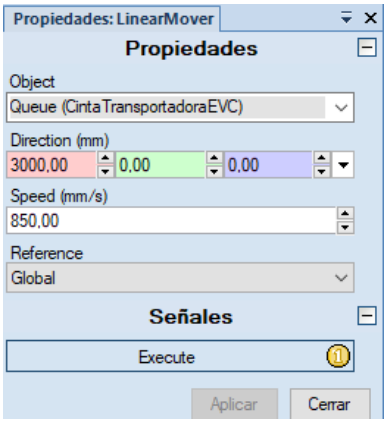


Figura 49. Configuración bloque LinearMover

También se le ha configurado la velocidad de movimiento. En nuestro caso todas las cintas estarán entre 700 y 850 mm/s.

Se muestra a continuación un esquema de la cinta transportadora con los bloques interconectados.

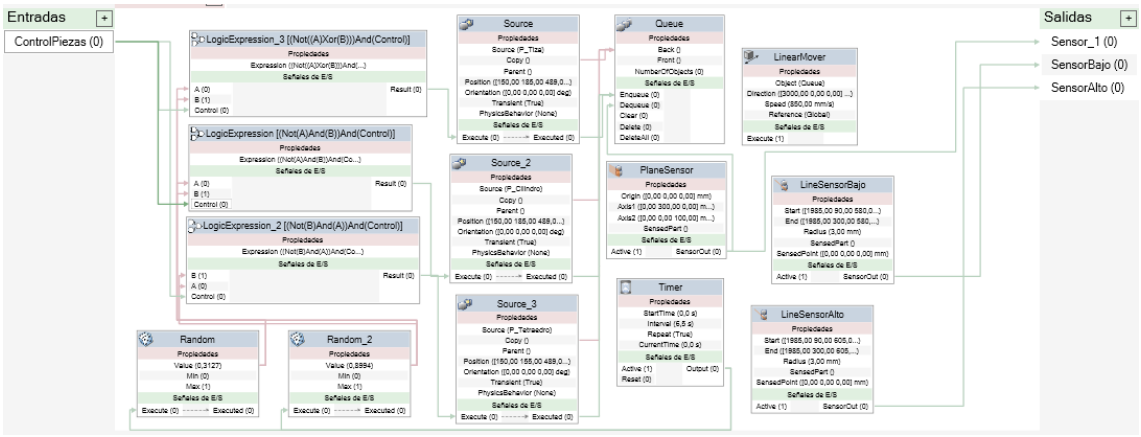


Figura 50. Conexionado Smart Component cintra transportadora

En el anexo I se adjunta el esquema en tamaño ampliado.

4.4.4. SC Cinta de descarga

Este SC es menos complejo que el anterior, es el mismo diseño CAD, pero de tamaño reducido, Las medidas de la cinta son 1320 x 300 x 500 mm

Se han utilizado tres una para cada tipo de pieza. Los componentes que han sido necesarios para el diseño son:

LineSensor: Sensor de línea puesto longitudinalmente con la cinta. Nos permite saber que pieza ha sido dejada en la cinta para enviarla a la cola del bloque Queque. También nos permite sacar del bloque Queue la pieza cuando ya no la detecta.

PlaneSensor: Al igual que en el anterior SC, permite saber cuando ha llegado al final de la cinta. Elimina de la cola la pieza y manda la señal mediante una salida digital generada para controlarla en la lógica de la estación.

Queue: El bloque de cola, al igual que en el apartado anterior, va metiendo en cola los objetos que le dice el LineSensor.

LineMover: Dota a la pieza de movimiento mientras este en la cola de Queue. No se conecta solo se configura para que el objeto que se mueva sea el que haya en la cola de Queue.

LogicGate: Este bloque permite hacer una operación lógica, en este caso no es ninguna operación o la operación NOP que es lo mismo. Este bloque genera un retraso de 1,5 segundos para que de tiempo al LineSensor a sensar la pieza que irá en cola.

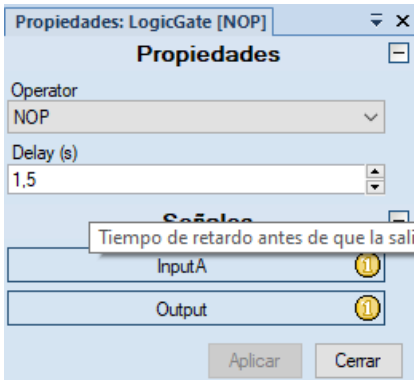


Figura 51. Configuración bloque LogicGate, NOP

Este es el cuadro de diseño de componentes conectados.

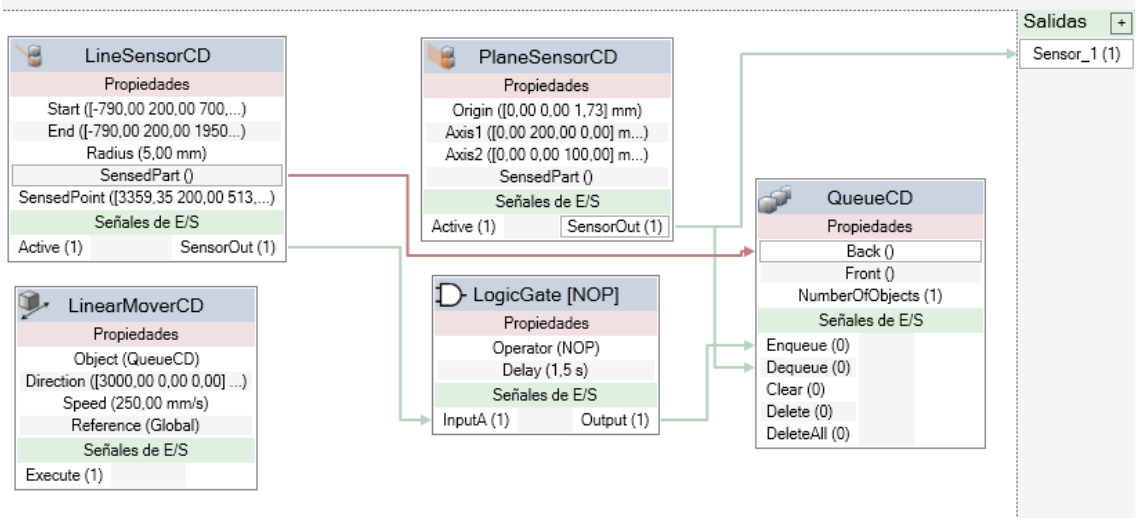


Figura 52. Conexionado Smart Component cinta de descarga

4.4.5. Caja de descarga

La caja de descarga se usará para introducir las piezas de los dos robots que no paletizan, serán las piezas tiza y cilindro.

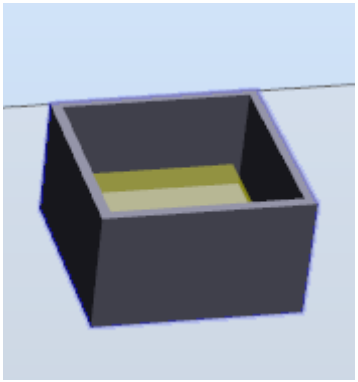


Figura 53. Caja de descarga

Este componente funciona mediante dos entradas que permiten abrir y cerrar la caja, una vez la pieza dentro, sensada por el Plane Sensor, el bloque Sink elimina la pieza, a través de un timer y actúa como si no hubiera ninguna la siguiente vez que se abre. Se ha utilizado para simular que las piezas se introducían en una caja.

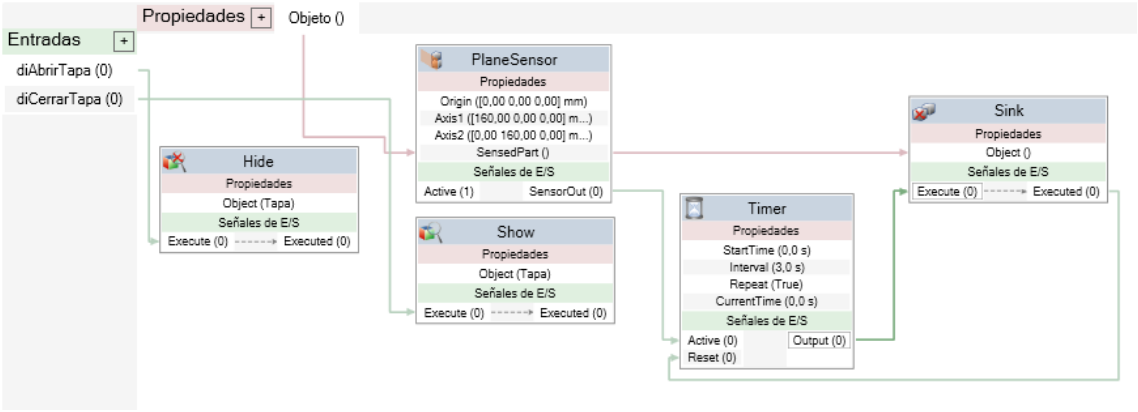


Figura 54. Esquema conexionado caja de descarga

Este componente junto con la herramienta de ventosa son los únicos Smart Components que no han sido diseñados por mí.

4.4.6. Controlador y Lógica de Estación.

Como se ha comentado anteriormente se ha utilizado dos controladores para hacer modulable el programa. El primero de ellos es el controlador10 que maneja al robot clasificador, el segundo es el controlador12 que se encarga de controlar los 3 robots restantes que se encuentran al final de las cintas de descarga.

En esta imagen se pueden observar la disposición de todos los elementos de la estación clasificadora y pick and place.

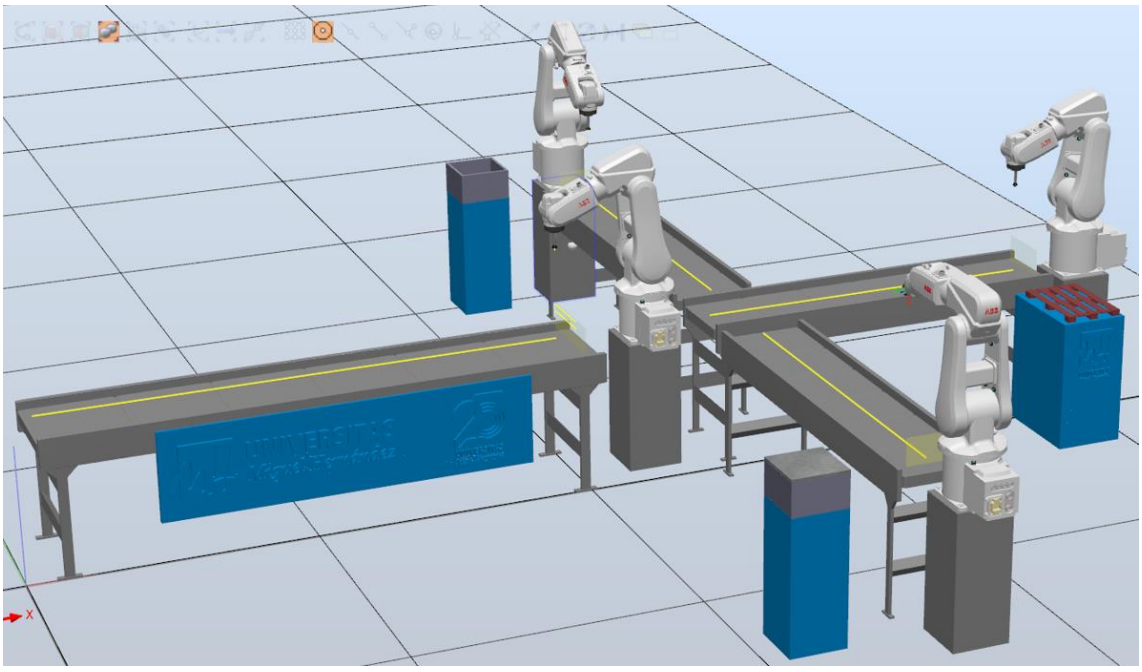


Figura 55. Disposición de todos los elementos de la estación

El diseño de la interconexión de la lógica de la estación se muestra a continuación.

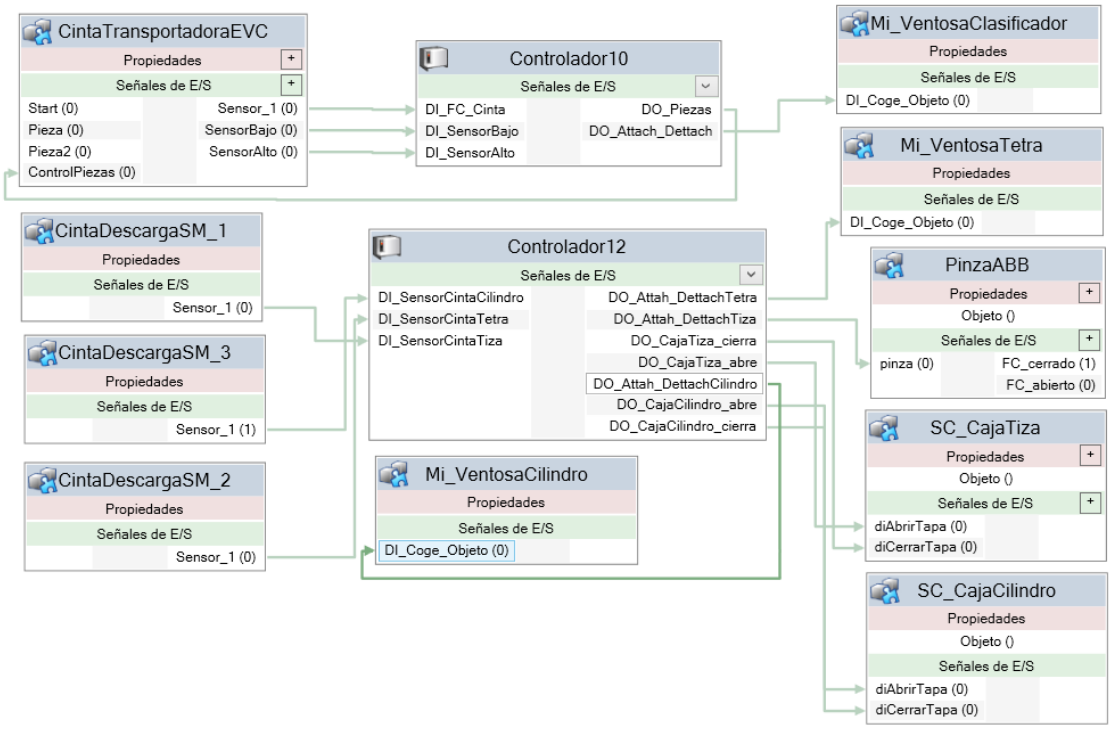


Figura 56. Conexionado bloques Lógica de la Estación

De todos los bloques que hay solo nos quedaría por ver los dos controladores.

Controlador10:

En el controlador10 se han creado tres entradas y dos salidas digitales, todas ellas son para controlar parte del proceso a través del lenguaje RAPID que veremos en el apartado siguiente.

<u>Entradas:</u>

- DI_FC_Cinta: Es una señal que viene de la cinta transportadora, si recordamos, está conectada al sensor de plano puesta al final de la cinta clasificadora. Nos va a permitir dentro de RAPID saber cuando hay una pieza disponible para clasificarla con el brazo robótico.
- DI_SensorBajo: Al igual que la anterior viene de la cinta transportadora pero ahora de un sensor lineal, el que pusimos a diferente altura, este en

concreto es el que hay más bajo se activará cuando la pieza sea de la misma altura o más alto.

- DI_SensorAlto: Igual que el anterior, pero a mayor altura para controlar si piezas más altas.

Salidas:

- DO_Piezas: Nos va a permitir decirle a la cinta transportadora a través de su entrada Control_Piezas que ya puede suministrar piezas.
- DO_Attach_Dettach: Esta salida la vamos a utilizar para decirle a la herramienta ventosa cuando tiene que agarrar la pieza y cuando soltarla.

Controlador12:

En este controlador están conectados los tres brazos robóticos al final de las cintas de descarga. Tiene 3 entradas digitales, DI_SensorCintaCilindro, DI_SensorCintaTiza y DI_SensorCintaTetraedro, las 3 entradas están conectadas a los respectivos bloques de cinta de descarga, estas señales nos avisan cuando ha llegado al final de la cinta de descarga la pieza, nos servirá para saber cuando podremos poner en marcha el robot que las paletice.

Este controlador tiene siete salidas digitales, que de manera similar a las del controlador10 nos van a permitir decirle cuando queremos que agarre la pieza y cuando queremos que la suelte. Las salidas digitales para controlar las herraminetas son, DO_Attach_DettachTetra, DO_Attach_DettachTiza, DO_Attach_Dettach Cilindro, las otras cuatro son las dos parejas, que permiten abrir y cerrar las cajas de descarga de los dos brazos robóticos que no paletizan.

4.4.7. Programación en RAPID.

La programación en lenguaje RAPID se divide en dos partes, cada uno en un controlador. El controlador10 tenemos la programación que va a realizar el brazo robótico clasificador y en el controlador12 tenemos tres archivos de programación RAPID donde van a ir las acciones que realizarán los tres robots restantes.

En el controlador10, como se ha comentado, tenemos la mecánica de la cinta clasificadora y el robot que clasifica. El código se describe por partes esquematizado, se han ocultado posiciones para verlo más claro, es el siguiente:

```
1  MODULE Module1
2      CONST robtarget AproxR:=[[0.054,299.174,370],[0,0,1,0],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
3      CONST robtarget AproxCercaR:=[[0.054,299.174,270],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
4      CONST robtarget CogeR:=[[0.054,299.174,89],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figura 57. Declaración de posiciones con RAPID

En la primera parte del archivo Module1 nos encontramos con todas las declaraciones de las posiciones o targets, aquí se encontrarán todas las posiciones declaradas que se utilizarán después para realizar las trayectorias o paths.

```
53      PROC main()
54          IF DI_FC_Cinta=1 THEN
55              IF DI_SensorBajo=0 AND DI_SensorAlto=0 THEN     !Pieza baja (Tiza)
56                  Path_recogidatiza;
57                  PulseDO DO_Piezas;
58                  Path_descargatiza;
59              ELSEIF DI_SensorBajo=1 AND DI_SensorAlto=0 THEN !Pieza mediana (Tetraedro)
60                  Path_recogidatetra;
61                  PulseDO DO_Piezas;
62                  Path_descargatetra;
63              ELSEIF DI_SensorBajo=1 AND DI_SensorAlto=1 THEN !Pieza alta (Cilindro)
64                  Path_recogidacilindro;
65                  PulseDO DO_Piezas;
66                  Path_descargacilindro;
67              ENDIF
68          ENDIF
69      ENDPROC
```

Figura 58. Procedimiento main() robot clasificador

Dentro del procedimiento del main() nos encontramos la lógica que se irá repitiendo cíclicamente, en este módulo lo que tenemos es diferentes if para comprobar que tamaño tiene la pieza para descargarla en su cinta correspondiente. Primero comprueba que hay una pieza esperando al final de la cinta de clasificación mediane la entrada que ya comentamos, DI_FC_Cinta, si hay pieza tenemos que discernir entre 3 diferentes, si la lectura de los sensores bajo (DI_SensorBajo) y alto (DI_SensorAlto) nos dan 0, tendremos la pieza más baja, llamada tiza, si el sensor bajo está activo y el sensor alto a 0, tendremos la pieza mediana, pieza llamada tetraedro y si tenemos los dos sensores activos, tendremos la pieza más alta, el cilindro.

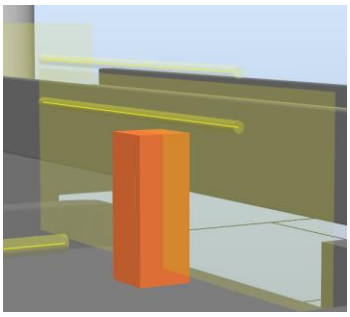


Figura 59. Pieza tiza, sin activar ningún sensor

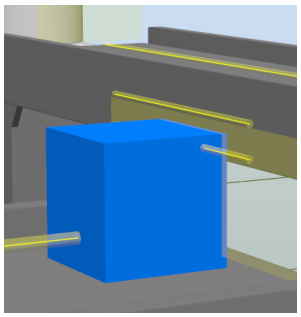


Figura 60. Pieza tetraedro activando el sensor bajo

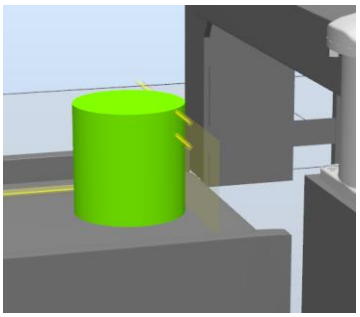


Figura 61. Pieza cilindro activando los dos sensores

Dentro de cada if tenemos la misma forma de trabajar, llamamos a la función que recoge la pieza (Path_recoge'nombrepieza'), con las posiciones ajustadas a esa pieza, una vez ya la hemos recogido le damos un pulso a la salida DO_Piezas que nos genera una nueva pieza y después descargamos la pieza en su cinta correspondiente a través de función Path_descarga'nombrepieza'.

```
73      PROC Path_recogidatiza()
74          MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
75          MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
76          MoveJ Target_20,v500,fine,TCP_Ventosa\WObj:=WO_recogidatiza;
77          SetDO DO_Attach_Dettach,1;
78          WaitTime 1;
79          MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
80          MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
81      ENDPROC
```

Figura 62. Procedimiento Patch_recogidatiza()

Ahora pasamos a los procedimientos que ejecutan las trayectorias, vamos a ver como ejemplo una de recogida, ya que las otras dos, las correspondientes a las otras dos piezas son iguales, solo se diferencian en los puntos que serán diferentes. Se ha programado 3 puntos principales. El primero en este caso en Target_20_2_2 será la posición más alta e inicial del robot clasificador, la siguiente posición será una posición de aproximación a la pieza, Target_20_2, y la tercera la posición que agarra la pieza. Aquí se puede observar, en amarillo, la trayectoria marcada comentada,

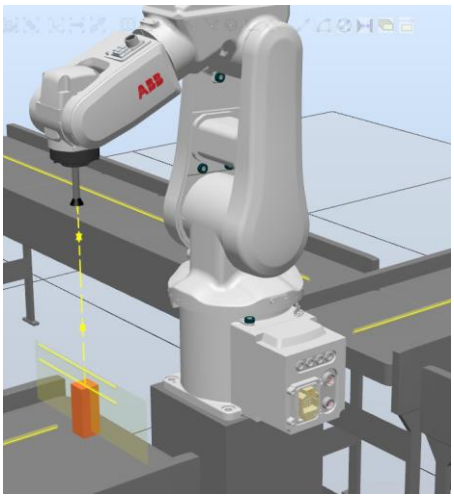


Figura 63. Trayectoria de recogida pieza tiza

La única diferencia entre las tres posiciones es en la altura, eje Z, que se irá decrementando en 100 mm. Una vez realizada las tres instrucciones, la herramienta ya estará en contacto con la pieza, entonces se ejecutará la instrucción SetDO DO_Attach_Dettach, 1, que agarrará la pieza, esperaremos un segundo con la instrucción WaitTime, y ejecutaremos las dos instrucciones de movimiento que nos subirán la pieza hasta la posición inicial.

Las instrucciones de movimiento tienen el formato siguiente MoveJ, tipo de movimiento, v500, velocidad en mm/s, z100, precisión del movimiento, TCP_Ventosa\Wobj:=recogida_tiza, herramienta y workobject asociado. Si nos fijamos en la instrucción de movimiento justo antes de agarrar la pieza se puede comprobar que la precisión está a fine, en vez de tener un valor cualquiera o cercano a 0, esto se debe a que queremos que a la hora de agarrar la pieza la instrucción tenga en mínimo error posible.

El procedimiento para descarga de la pieza es prácticamente igual,

```
100     PROC Path_descargatiza()
101         MoveJ Target_40_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
102         MoveJ Target_40_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
103         MoveJ Target_40,v500,fine,TCP_Ventosa\WObj:=WO_descargatiza;
104         SetDO DO_Attach_Dettach,0;
105         WaitTime 1;
106         MoveJ Target_40_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
107         MoveJ Target_40_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
108     ENDPROC
```

Figura 64. Procedimiento Path_descargatiza()

La única diferencia es que las posiciones son distintas, y en vez de coger la pieza, activando la señal SetDO DO_Attach_Dettach, lo que hacemos es desactivarla poniendo la señal generada en el controlador a 0 para poder soltarla en la cinta de descarga. Cada función tiene un workobject con los puntos asociados a el, esta manera de asociar puntos a workobjects nos permite de una forma muy cómoda modificar la posición de todos ellos si fuera necesario por algún ajuste de la estación, como cinta transportadora de otro tamaño, herramienta diferente con otra longitud, etc.

El resto de procedimientos son repetitivos, pues hacen la misma tarea que los explicados. Se muestra una captura del árbol de controlador donde se pueden observar los procedimientos.

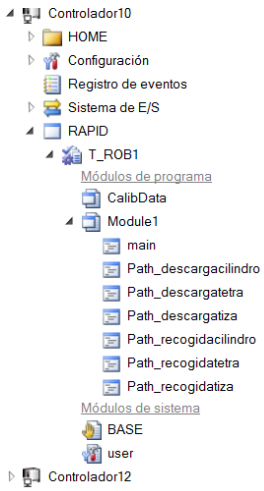


Figura 65. Árbol de procedimientos del controlador10

En el anexo II del TFM se encuentra el código de los programas en RAPID completo.

En cuanto en el controlador12, tenemos la acción de los tres robots que reciben las piezas de las cintas de descarga. Dos de los robots colocan las piezas en las cajas de descarga y el otro robot paletiza, están divididos de la siguiente manera dentro del controlador.

- T_ROB1: Programación del robot que paletiza, las piezas paletizadas son el tetraedro, la herramienta utilizada es una ventosa.

- T_ROB2: El robot que descarga las piezas tiza en la caja de descarga, este robot lleva una herramienta gripper o pinza.
- T_ROB3: Misma función que el anterior, pero con una ventosa y con las piezas cilindro.

En esta imagen se puede observar el árbol de los archivos comentados,

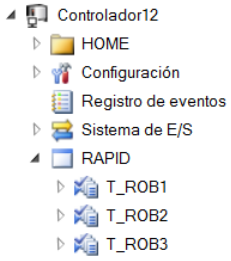


Figura 66. Árbol de archivos del controlador12

T_ROB1

Empezaremos por la programación del brazo que paletiza los tetraedros, describiremos las partes diferentes a las ya explicadas.

```
33    PROC main()
34        WaitDI DI_SensorCintaTetra, 1;
35        Path_recogidaTetra;
36        Path_descargaTetra;
37    ENDPROC
```

Figura 67. Procedimiento main() robot paletizador

La función main es muy parecida, esperamos con la instrucción WaitDI a que la pieza llegue al sensor de plano de la cinta de descarga para poder realizar la trayectoria de recogida y después la de descarga.

```
38    PROC Path_recogidaTetra()
39        MoveJ Target_10_2_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
40        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
41        MoveJ Target_10,v500,fine,TCP_Ventosa\WObj:=WO_rTetra;
42        SetDO DO_Attah_DettachTetra, 1;
43        WaitTime 1;
44        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
45        MoveJ Target_10_2_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
46    ENDPROC
```

Figura 68. Procedimiento Path_recogidaTetra

El procedimiento de recogida hace los movimiento programados, punto inicial, una aproximación y mediante la instrucción SetDO le decimos que ponga a la salida correspondiente a 1 que irá enlazada con la herramienta, le damos un segundo para que pueda realizar la acción, después realizamos el movimiento contrario, subimos un poco lo que era la aproximación y situamos el brazo robótico en la parte alta que estaba al principio.

```
47    PROC Path_descargaTetra()
48        MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
49        !Hacemos offset también de la aproximación para que no se choquen las cajas
50        MoveL Offs (Target_20_2,P1_X,P1_Y,P1_Z),v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
51        MoveL Offs (P1,P1_X,P1_Y,P1_Z),v500,fine,TCP_Ventosa\WObj:=WO_dTetra;
52        SetDO DO_Attah_DettachTetra, 0;
53        WaitTime 1;
54        P1_X := P1_X - 100;
55        !Compruebo que no me haya pasado en X
56        IF P1_X < -100 THEN
57            P1_X := 0;
58            P1_Y := P1_Y + 100;
59        ENDIF
60        IF P1_Y > 300 THEN
61            P1_Y:=0;
62            P1_Z:= P1_Z + 2;
63        ENDIF
64        MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
65        MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
66        MoveJ Target_10_2_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
67    ENDPROC
```

Figura 69. Procedimiento Path_descargaTetra. Paletizado con offset

La parte de la descarga de la pieza tiene más sustancia ya que aquí hacemos un paletizado con offset, la instrucción MoveL Offs nos permite ir aumentando la posición de referencia en una cantidad que vamos incrementado a través de los bucles if.

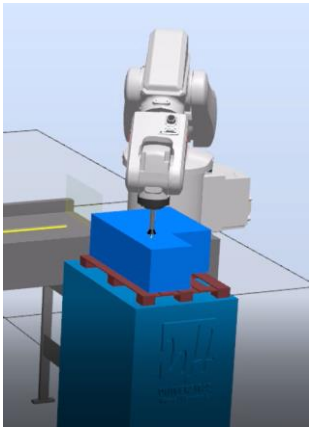


Figura 70. Robot IRB120 paletizando

Como se puede comprobar se van aumentando las posiciones de x e y para ir colocando las cajas en forma de 3 x 2, después incrementará la z para ponerlas encima. Una vez que realiza el ciclo vuelve a la posición inicial a espera la próxima pieza.

T_ROB2 y T_ROB3

La programación de los dos brazos robóticos es prácticamente igual, con la única diferencia que han programado puntos diferentes, la forma de trabajar es igual aun teniendo dos herramientas diferentes. Aquí se muestra el código del brazo con gripper,

```
27      PROC main()
28          PulseDO DO_CajaTiza_cierra;
29          WaitDI DI_SensorCintaTiza,1;
30          Path_recTiza;
31          Path_desTiza;
32      ENDPROC
33      PROC Path_recTiza()
34          MoveJ Target_10_2,v500,z100,Servo\WObj:=WO_recTiza;
35          MoveJ Target_10,v500,fine,Servo\WObj:=WO_recTiza;
36          SetDO DO_Attah_DettachTiza,1;
37          WaitTime 1;
38          MoveJ Target_10_2,v500,z100,Servo\WObj:=WO_recTiza;
39      ENDPROC
40      PROC Path_desTiza()
41          MoveJ Target_20_2,v500,fine,Servo\WObj:=WO_desTiza;
42          PulseDO DO_CajaTiza_abre;
43          MoveJ Target_20,v500,fine,Servo\WObj:=WO_desTiza;
44          SetDO DO_Attah_DettachTiza,0;
45          WaitTime 3;
46          MoveJ Target_20_2,v500,fine,Servo\WObj:=WO_desTiza;
47          PulseDO DO_CajaTiza_cierra;
48          MoveJ Target_10_2,v500,z100,Servo\WObj:=WO_recTiza;
49      ENDPROC
```

Figura 71. Programa RAPID robot control pieza tiza

La diferencia con los anteriores es que aquí hemos utilizado las salidas digitales DO_CajaTiza_abre y DO_CajaTiza_cierra para abrir y cerrar la caja de descarga mediante la aplicación de un pulso con la instrucción PulseDO.

5. Resultados

Los resultados obtenidos son más o menos los esperados, partiendo del objetivo que era saber utilizar el programa con cierta habilidad y destreza, se ha

conseguido, aun así hay que decir que aunque pueda parecer sencillo el software se va complicando por la cantidad de acciones y coordinaciones que hay que tener en cuenta cada vez que se programa un brazo robótico, no por el diseño de los puntos y trayectorias, algo que al final se acaba mecanizando después de muchas veces, si no, de la parte de diseño de los componentes inteligentes. Estos han sido, la mayoría, diseñados íntegramente desde cero, para poder dotar de movimiento a las cintas, diseñar la colocación de los sensores, ajustar las lecturas, coger piezas etc.

A parte de lo descrito en el proyecto salieron varías opciones como objetivo de la estación, se hicieron pruebas para clasificar en cuanto a color las piezas, en vez de tamaño, pero se encontraron muchos problemas con el módulo de lectura de color, buscando en foros, en concreto en el mismo de ABB aseguraban que para la versión 2022.2 estaría resuelto, cuando se empezó el proyecto la versión más reciente que había era la 2022.1 que es la que se ha utilizado, a mediados de julio sacaron la versión en cuestión, pero se decidió seguir con la clasificación por tamaño.

Otra de las opciones que se manejaron y que se estudiaron fue paletizar automáticamente, ABB dispone de un complemento o aplicación para el software RobotStudio para paletizar, se puede descargar de su página web, ellos lo llaman PowerPac Palletizing. Es un complemento muy potente que permite paletizar con gran cantidad de configuraciones y patrones, eso sí, está limitado a robots más grandes que sean expresamente para paletizar como el IRB 460 y IRB 760, también tiene el inconveniente de que hay que utilizar sus cintas transportadoras, pallets y herramientas, esta todo más normalizado.

6. Conclusiones y líneas futuras de trabajo

Se ha conseguido aprender y trabajar en un espacio de tiempo no muy largo con un programa de gran relevancia por su utilización en el mundo industrial. Usando una tecnología cada vez más implantada en todos los procesos de la industria como pueden ser la logística, automoción y la electrónica.

Como líneas futuras de trabajo sería interesante poder pasar de la simulación a la realidad, en nuestro caso el brazo escogido es uno de los más económicos.

También es verdad que la estación consta de 4 brazos robóticos y dos controladores, algo inalcanzable en cuanto a presupuesto.

7. Bibliografía

- Guide for ABB RobotStudio. M. Natapon, Aalborg University. 2017
- Manual del operador RobotStudio ABB 2022.1
- Revista de electricidad, electrónica y automática. J.C.M. Castillo http://reea-blog.blogspot.com/p/robotstudio-abb.html
- Executive Summary World Robotics 2021 Industrial Robots. IFR https://ifr.org/
- ISO 8373:2012 Robots and robotic devices - Vocabulary; http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55890.
- https://en.wikipedia.org/wiki/RAPID
- https://new.abb.com/products/robotics/es/robots-industriales/irb-120

9. Índice de figuras

8. Anexos

ANEXO I: Esquema conexionado Smart Component cinta clasificadora

ANEXO II: Código de los programas en RAPID

Controlador10. T_ROB1. (Robot clasificador). Main()

```
MODULE Module1
    CONST robtarget AproxR:=[[0.054,299.174,370],[0,0,1,0],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
AproxCercaR:=[[0.054,299.174,270],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget
CogeR:=[[0.054,299.174,89],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Aprox1:=[[-372.548055495,-360.173438569,370],[0,0,1,0],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget AproxCerca1:=[[-372.548,-360.173,269.555429952],[0,0,1,0],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Deja1:=[[-372.548055495,-360.173438569,100],[0,0,1,0],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Aprox2:=[[10.179,-
360.173,370],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget AproxCerca2:=[[10.179,-
360.173,270],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Deja2:=[[10.179,-
360.173,108.161],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+0
9,9E+09]];
    CONST robtarget AproxCerca3:=[[370.531,-
360.173,270],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Deja3:=[[370.531,-
360.173,108.161],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+0
9,9E+09]];
    CONST robtarget Aproxgiro:=[[-1048.992151931,-
404.489026678,369.999974883],[0,0.997550466,0.069950473,0.000000031],[1,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_2_2:=[[-
0.132,286.397,370],[0,0.707106781,0.707106781,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_2:=[[-
0.132,286.397,270],[0,0.707106781,0.707106781,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[-
0.132,286.397,88.542],[0,0.707106781,0.707106781,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40_2_2:=[[-320.066,-
360.173,370],[0,0.707106781,0.707106781,0],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40_2:=[[-320.066,-
360.173,270],[0,0.707106781,0.707106781,0],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[-320.066,-
360.173,108.161],[0,0.707106781,0.707106781,0],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50_2_2:=[[2.449,-
397.77,370],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Target_50_2:=[[2.449,-
397.77,270],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Target_50:=[[2.449,-
397.77,128.61],[0,0.707106781,0.707106781,0],[-
1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60_2_2:=[[312.854,-
369.743,370],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Target_60_2:=[[312.854,-
369.743,270],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Target_60:=[[312.854,-
369.743,144.576],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+0
9,9E+09]];
```

```
    CONST robtarget Target_10_3:=[[-
13.096,340.231,370],[0,0.707106781,0.707106781,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10_2:=[[-
13.096,340.231,270],[0,0.707106781,0.707106781,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[-
13.096,340.231,130.235],[0,0.707106781,0.707106781,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_30_2_2:=[[5.276,334.22,370],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+0
9,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_30_2:=[[5.276,334.22,270],[0,0.707106781,0.707106781,0],[0,0,1,0],[9E+09,9E+09,
9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
Target_30:=[[5.276,334.22,108.821],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+0
9,9E+09,9E+09,9E+09,9E+09]];
    !***********************************************************
    !
    ! Módulo:  Module1
    !
    !***********************************************************


    !***********************************************************
    !
    ! Procedimiento Main
    !
    !
    !***********************************************************
    PROC main()

        etiqueta:
        IF DI_FC_Cinta=1 THEN
            IF DI_SensorBajo=0 AND DI_SensorAlto=0 THEN     !Pieza baja (Tiza)
                Path_recogidatiza;
                PulseDO DO_Piezas;
                Path_descargatiza;
            ELSEIF DI_SensorBajo=1 AND DI_SensorAlto=0 THEN !Pieza mediana (Tetraedro)
                Path_recogidatetra;
                PulseDO DO_Piezas;
                Path_descargatetra;
            ELSEIF DI_SensorBajo=1 AND DI_SensorAlto=1 THEN !Pieza alta (Cilindro)
                Path_recogidacilindro;
                PulseDO DO_Piezas;
                Path_descargacilindro;
            ENDIF
        ENDIF
        GOTO etiqueta;
    ENDPROC
    PROC Path_recogidatiza()
        MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
        MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
        MoveJ Target_20,v500,fine,TCP_Ventosa\WObj:=WO_recogidatiza;
        SetDO DO_Attach_Dettach,1;
        WaitTime 1;
        MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
        MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatiza;
    ENDPROC
    PROC Path_recogidacilindro()
        MoveJ Target_10_3,v500,z100,TCP_Ventosa\WObj:=WO_recogidacilindro;
        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidacilindro;
        MoveJ Target_10,v500,fine,TCP_Ventosa\WObj:=WO_recogidacilindro;
        SetDO DO_Attach_Dettach,1;
        WaitTime 1;
        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidacilindro;
        MoveJ Target_10_3,v500,z100,TCP_Ventosa\WObj:=WO_recogidacilindro;
    ENDPROC
    PROC Path_recogidatetra()
        MoveJ Target_30_2_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatetra;
        MoveJ Target_30_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatetra;
        MoveJ Target_30,v500,fine,TCP_Ventosa\WObj:=WO_recogidatetra;
        SetDO DO_Attach_Dettach,1;
        WaitTime 1;
        MoveJ Target_30_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatetra;
        MoveJ Target_30_2_2,v500,z100,TCP_Ventosa\WObj:=WO_recogidatetra;
```

```
    ENDPROC
    PROC Path_descargatiza()
        MoveJ Target_40_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
        MoveJ Target_40_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
        MoveJ Target_40,v500,fine,TCP_Ventosa\WObj:=WO_descargatiza;
        SetDO DO_Attach_Dettach,0;
        WaitTime 1;
        MoveJ Target_40_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
        MoveJ Target_40_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatiza;
    ENDPROC
    PROC Path_descargatetra()
        MoveJ Target_50_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatetra;
        MoveJ Target_50_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatetra;
        MoveJ Target_50,v500,fine,TCP_Ventosa\WObj:=WO_descargatetra;
        SetDO DO_Attach_Dettach,0;
        WaitTime 1;
        MoveJ Target_50_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatetra;
        MoveJ Target_50_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargatetra;
    ENDPROC
    PROC Path_descargacilindro()
        MoveJ Target_60_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargacilindro;
        MoveJ Target_60_2,v500,z100,TCP_Ventosa\WObj:=WO_descargacilindro;
        MoveJ Target_60,v500,fine,TCP_Ventosa\WObj:=WO_descargacilindro;
        SetDO DO_Attach_Dettach,0;
        WaitTime 1;
        MoveJ Target_60_2,v500,z100,TCP_Ventosa\WObj:=WO_descargacilindro;
        MoveJ Target_60_2_2,v500,z100,TCP_Ventosa\WObj:=WO_descargacilindro;
    ENDPROC
ENDMODULE
```

Controlador12. T_ROB1. (Robot paletiza pieza tetra). Main()

```
MODULE Module1
    CONST robtarget Target_10_2_2:=[[263.765,-
2.449,370],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10_2:=[[263.765,-
2.449,270],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[263.765,-2.449,128.61],[0,1,0,0],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_2_2:=[[263.203,262.137,370],[0,1,0,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_2:=[[263.203,262.137,270],[0,1,0,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[263.203,262.137,128.61],[0,1,0,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget P1:=[[263.203,262.137,128.61],[0,1,0,0],[0,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR num P1_X:=0;
    VAR num P1_Y:=0;
    VAR num P1_Z:=0;
    !***********************************************************
    !
    ! Módulo:  Module1
    !
    !
    !***********************************************************


    !***********************************************************
    !
    ! Procedimiento Main
    !
    !***********************************************************
    PROC main()
        WaitDI DI_SensorCintaTetra, 1;
        Path_recogidaTetra;
        Path_descargaTetra;
    ENDPROC
    PROC Path_recogidaTetra()
        MoveJ Target_10_2_2,v500,z0,TCP_Ventosa\WObj:=WO_rTetra;
        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
        MoveJ Target_10,v500,fine,TCP_Ventosa\WObj:=WO_rTetra;
        SetDO DO_Attah_DettachTetra, 1;
```

```
        WaitTime 1;
        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
        MoveJ Target_10_2_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
    ENDPROC
    PROC Path_descargaTetra()
        MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
        !Hacemos offset también de la aproximación para que no se choquen las cajas
        MoveL Offs (Target_20_2,P1_X,P1_Y,P1_Z),v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
        MoveL Offs (P1,P1_X,P1_Y,P1_Z),v500,fine,TCP_Ventosa\WObj:=WO_dTetra;
        SetDO DO_Attah_DettachTetra, 0;
        WaitTime 1;
        P1_X := P1_X - 100;
        !Compruebo que no me haya pasado en X
        IF P1_X < -100 THEN
            P1_X := 0;
            P1_Y := P1_Y + 100;
        ENDIF
        IF P1_Y > 300 THEN
            P1_Y:=0;
            P<1_Z:= P1_Z + 2;
            WaitDI DI_SensorCintaTiza,1;
        ENDIF
        MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
        MoveJ Target_20_2_2,v500,z100,TCP_Ventosa\WObj:=WO_dTetra;
        MoveJ Target_10_2_2,v500,z100,TCP_Ventosa\WObj:=WO_rTetra;
    ENDPROC
ENDMODULE
```

Controlador12. T_ROB2. (Robot pieza tiza). Main()

```
MODULE Module1
    CONST robtarget
Target_10_2:=[[1406.024,1531.375,395],[0,1,0,0],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+
09,9E+09]];
    CONST robtarget
Target_10:=[[1406.024,1626.3748,108.161],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,
9E+09,9E+09]];
    CONST robtarget Target_20_2:=[[1860.38,1762.688,395],[0,1,0,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[1860.38,1762.688,120],[0,1,0,0],[1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
!*********************************************************
    !
    ! Módulo:  Module1
    !
    !*********************************************************


    !*********************************************************
    !
    ! Procedimiento Main
    !
    !
    !*********************************************************
    PROC main()
        PulseDO DO_CajaTiza_cierra;
        WaitDI DI_SensorCintaTiza,1;
        Path_recTiza;
        Path_desTiza;
    ENDPROC
    PROC Path_recTiza()
        MoveJ Target_10_2,v500,z100,Servo\WObj:=WO_recTiza;
        MoveJ Target_10,v500,fine,Servo\WObj:=WO_recTiza;
        SetDO DO_Attah_DettachTiza,1;
        WaitTime 1;
        MoveJ Target_10_2,v500,z100,Servo\WObj:=WO_recTiza;
    ENDPROC
    PROC Path_desTiza()
        MoveJ Target_20_2,v500,fine,Servo\WObj:=WO_desTiza;
        PulseDO DO_CajaTiza_abre;
        MoveJ Target_20,v500,fine,Servo\WObj:=WO_desTiza;
        SetDO DO_Attah_DettachTiza,0;
        WaitTime 3;
```

```
        MoveJ Target_20_2,v500,fine,Servo\WObj:=WO_desTiza;
        PulseDO DO_CajaTiza_cierra;
        MoveJ Target_10_2,v500,z100,Servo\WObj:=WO_recTiza;
    ENDPROC
ENDMODULE
```

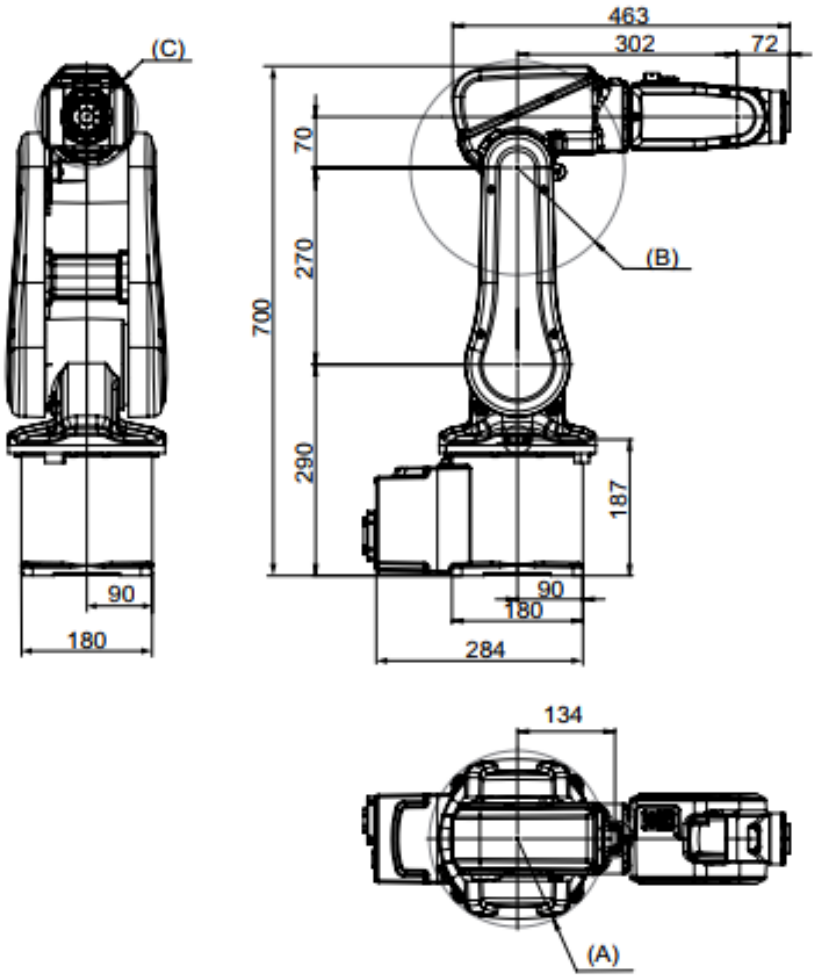
Controlador12. T_ROB3. (Robot pieza cilindro). Main()

```
MODULE Module1
    CONST robtarget Target_10_2:=[[1396.454,-1486.825,320],[0,1,0,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[1396.454,-1486.825,144.576],[0,1,0,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_2:=[[1791.371,-1734.299,320],[0,1,0,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[1791.371,-1734.299,140],[0,1,0,0],[-1,0,-
2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
!*************************************************************
    !
    ! Módulo:  Module1
    !
    !
    !*************************************************************


    !*************************************************************
    !
    ! Procedimiento Main
    !
    !
    !*************************************************************
    PROC main()
        PulseDO DO_CajaCilindro_cierra;
        WaitDI DI_SensorCintaCilindro,1;
        Path_recCilindro;
        Path_desCilindro;
    ENDPROC
    PROC Path_recCilindro()
        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_rCilindro;
        MoveJ Target_10,v500,fine,TCP_Ventosa\WObj:=WO_rCilindro;
        SetDO DO_Attah_DettachCilindro,1;
        WaitTime 1;
        MoveJ Target_10_2,v500,z100,TCP_Ventosa\WObj:=WO_rCilindro;
    ENDPROC
    PROC Path_desCilindro()
        MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_dCilindro;
        PulseDO DO_CajaCilindro_abre;
        MoveJ Target_20,v500,fine,TCP_Ventosa\WObj:=WO_dCilindro;
        SetDO DO_Attah_DettachCilindro,0;
        WaitTime 1;
        MoveJ Target_20_2,v500,z100,TCP_Ventosa\WObj:=WO_dCilindro;
        MoveJ Target_10_2,v500,fine,TCP_Ventosa\WObj:=WO_rCilindro;
        PulseDO DO_CajaCilindro_cierra;
    ENDPROC
ENDMODULE
```

ANEXO III: Características técnicas relevantes IRB120

Dimensiones IRB 120-3/0.6

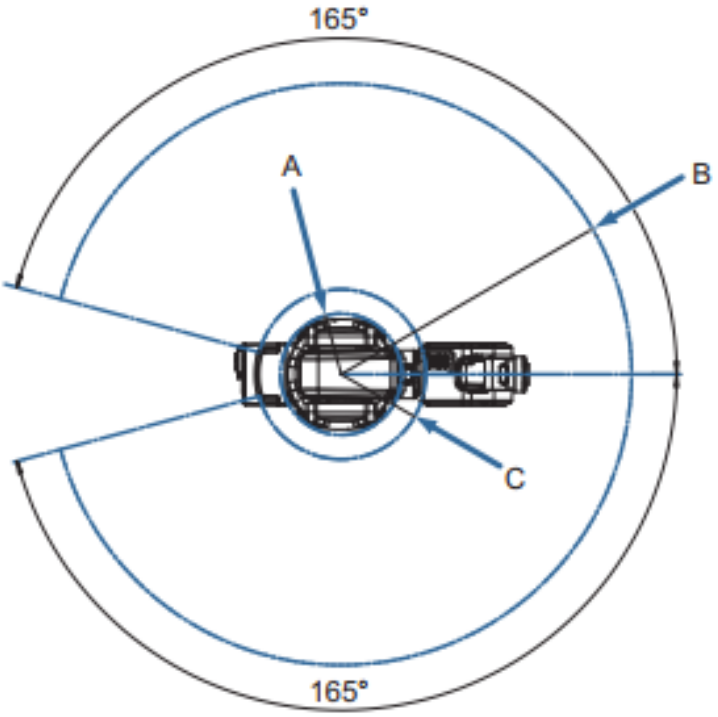


xx0900000256

Posición	Descripción
A	Radio mínimo de giro del eje 1 R=121 mm
B	Radio mínimo de giro del eje 3 R=147 mm
C	Radio mínimo de giro del eje 4 R=70 mm

Radio de giro

El radio de giro del robot se muestra en la figura.



xx0900000157

Variante de robot	Pos. A	Pos. B	Pos. C
IRB 120-3/0.6	R121 i	R580	R169.4

i Radio de giro mínimo del eje 1.

Movimiento del robot

En la tabla se especifican los tipos y áreas de movimiento de todos los ejes.

Ubicación del movimiento	Tipo de movimiento	Área de movimiento
Eje 1	Movimiento de rotación	De +165° a -165°
Eje 2	Movimiento del brazo	De +110° a -110°
Eje 3	Movimiento del brazo	De +70° a -110°
Eje 4	Movimiento de la muñeca	De +160° a -160°
Eje 5	Movimiento de doblado	De +120° a -120°
Eje 6	Movimiento de giro	De +400° a -400° (de forma predeterminada) De +242 a -242 revoluciones como máximo i

i El área de trabajo predeterminada para el eje 6 puede ampliarse mediante el cambio de valores de parámetros en el software.

La opción 610-1 "Independent axis" puede utilizarse para restablecer el cuentarrevoluciones tras el giro del eje (sin necesidad de "rebobinar" el eje).