Universidad Miguel Hernández de Elche

MASTER UNIVERSITARIO EN ROBÓTICA

“Programación de un robot ABB para una estación de trabajo de soldadura de chasis para maquinaria”

Trabajo de Fin de Máster

2018-2019

Autor: David José Roldán Román

Tutor/es: Carlos Pérez Vidal

ÍNDICE GENERAL

Gracias en primer lugar a mi familia, por las horas que no les he podido dedicar y su comprensión.

Gracias a mi tutor de proyecto Carlos Pérez Vidal por su ayuda y sus valiosas indicaciones y gracias a los profesores del Master de Robótica de la UMH por su dedicación e implicación.

RESUMEN.

En el presente Trabajo Fin de Master se lleva a cabo la robotización de una estación de soldadura manual de chasis metálicos para maquinaria mediante un robot ABB.

En primer lugar se realiza un estudio del estado del arte de las celdas robotizadas de soldadura MIG/MAG y de los sistemas para la ampliación de la zona de trabajo del robot tales como Tracks y posicionadores.

En segundo lugar se realiza la fase de adquisición de conocimientos y habilidades para poder utilizar el software de programación de robots RobotStudio de ABB haciendo uso de los manuales y tutoriales que se pueden encontrar en la web.

En tercer lugar se realiza la fase de estudio de las necesidades de la empresa y en base a ello la selección del robot, equipo de soldadura, track, posicionadores y sistemas de seguridad necesarios y la definición del layout de la estación de soldadura.

Finalmente se realiza el diseño del útil de fijación de las piezas y la programación del robot y la evaluación de los resultados mediante simulaciones en RobotStudio.

Palabras clave: ABB, chasis, estación de limpieza de soldadura, estación de soldadura MIG/MAG, posicionador, RAPID, RobotStudio, simulación de robots, track.

ABSTRACT.

In this Master's Final Project the robotization of a manual welding station of metal chassis for machinery is carried out by means of an ABB robot.

In the first place, a study of the state of the art of the MIG / MAG welding robot cells and of the systems for the expansion of the robot's working area such as Tracks and Positioners is carried out.

Secondly, the knowledge and skills acquisition phase is carried out in order to use ABB RobotStudio robot programming software using the manuals and tutorials that can be found on the web.

Thirdly, the study phase of the needs of the company is carried out and based on this, the selection of the robot, welding equipment, track, positioners and safety systems necessary and the definition of the layout of the welding station.

Finally, the design of the tool for fixing the parts and the programming of the robot and the evaluation of the results through simulations in RobotStudio is carried out.

Keywords: ABB, chassis, welding cleaning station, MIG / MAG welding station, positioner, RAPID, RobotStudio, robot simulation, track.

ÍNDICE DE FIGURAS III

ÍNDICE DE TABLAS IV

Página intencionadamente en blanco

1. INTRODUCCIÓN.

La necesidad de adaptación y búsqueda de nuevos nichos de mercado en la pequeña y mediana empresa del sector del metal hace necesaria la modernización de su proceso productivo y la implantación de sistemas de producción mediante células robóticas es, en algunos casos, una solución que puede ser rentable para la empresa y que posibilita avances en la mejora en la calidad, la seguridad y la producción.

La implantación de una célula robótica supone una elevada inversión para este tipo de empresa y es interesante cuando la producción va dirigida a poca variedad de productos pero elevado número de piezas anuales, por lo que es necesario un estudio previo para determinar su necesidad y viabilidad.

La sustitución de tareas repetitivas penosas y/o peligrosas para las personas es uno de los objetivos buscados en la automatización mediante robots aunque, para este tipo de empresa, lo son el aumento de la producción y la disminución de costes. Por tanto, las estaciones de soldadura manual son un caso claro de prioridad de automatización en la pequeña y mediana empresa.

2. ESTADO DEL ARTE.

La soldadura robotizada es una tecnología extensamente utilizada en la industria y probada sobradamente. Prácticamente todos los fabricantes de robot disponen de soluciones especializadas en soldadura de diferentes tipos que abarcan desde los propios robots hasta los equipos de soldadura, ejes externos, celdas de soldadura completamente equipadas y , por supuesto, paquetes de software especializados en diferentes funciones como coordinación de varios robots, pick and place, soldadura, etc.

El software de programación disponible, permite diseñar una réplica 3D de la estación de soldadura y programar las trayectorias del robot a partir de estos modelos mediante un controlador virtual que funciona igual que el real. Podemos simular y depurar los programas sin interrumpir el trabajo en la estación robotizada y finalmente podemos implementarlos en muy poco tiempo.

El proceso de soldadura es proceso complejo en el sentido de que es difícil de parametrizar ya que entran en juego altas temperaturas que afectan a zonas pequeñas y que producen fenómenos de expansión y contracción del metal además de producir cambios que afecten a su comportamiento mecánico.

Estas dilataciones y contracciones así como la exactitud en el posicionamiento del robot y las piezas de trabajo complican la realización de los cordones de soldadura, por ello, en los últimos años se han ido desarrollando sistemas para el control y autoajuste de los cordones en la soldadura robotizada. Dichos sistemas se basan básicamente en sensores de arco y sensores ópticos aunque existen otros basados en infrarrojos, ultrasonidos, electromagnéticos, etc.

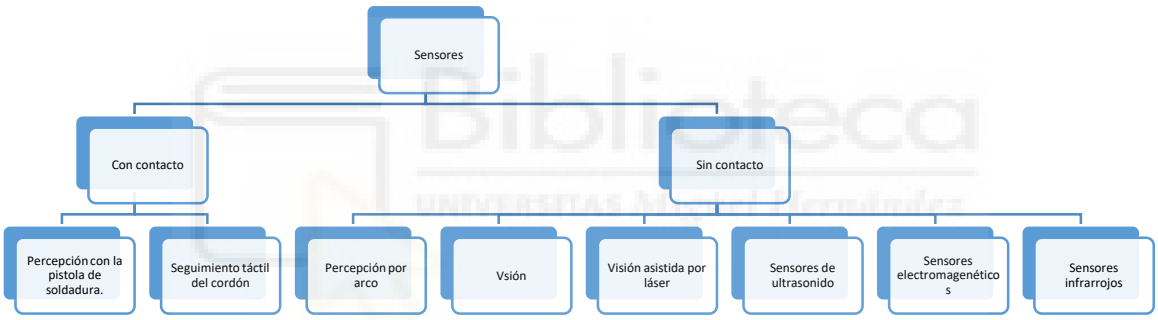


Figura 1: Tipos de sensores de seguimiento de cordones de soldadura.

Los sensores de arco son propios de la soldadura por arco y se basan en las variaciones en el voltaje y la intensidad del arco.

Los sistemas de visión basados en cámaras CCD o en cámaras CCD y láser se pueden utilizar para reconocer y encontrar las trayectorias de los cordones y finalmente corregir las desviaciones.

Figura 2: Sensor visión + laser.

La soldadura MIG/MAG es una de los métodos de soldadura más extensamente utilizados en la soldadura robotizada. En la soldadura MIG se aporta un gas inerte (Argón y/o CO2) para la protección del cordón de soldadura, lo cual evita la creación de escoria y reduce las operaciones de limpieza del cordón posteriores. Además permite una gran velocidad de soldadura (40 a 70 cm/min según material y espesor). El hilo de aportación suele ser de composición similar a la del metal base.

En el mercado se dispone de una amplia gama de equipos de soldadura MIG/MAG que permiten su integración con un robot. Un equipo básico para soldadura MIG/MAG robotizada estaría compuesto por:

- Equipo de potencia MIG/MAG
- Arrastradores de hilo y accesorios.
- Pistola de soldadura con soporte para TCP del robot.
- Interfaces controladora del robot/equipo de potencia MIG/MAG

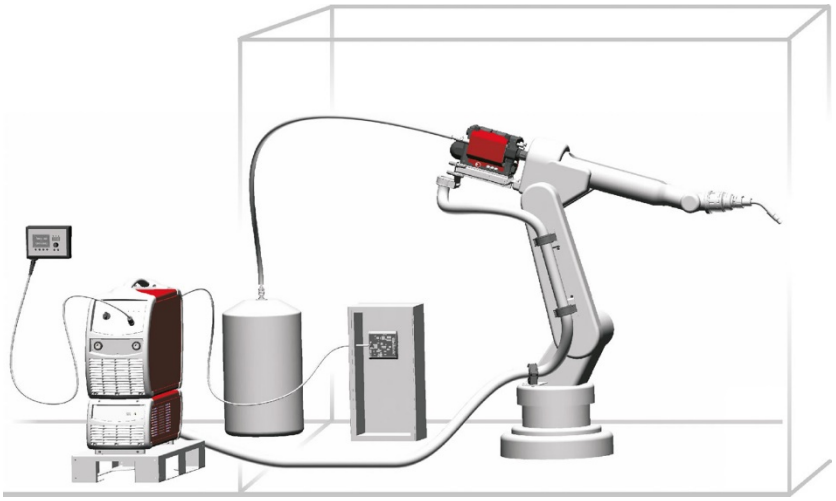


Figura 3: Equipos básicos de una estación robotizada de soldadura.

3. OBJETIVOS.

El presente Trabajo Fin de Master (en adelante TFM) tiene como objetivo adquirir los conocimientos y habilidades necesarios para diseñar y programar mediante un software de simulación y programación de robots industrial una estación de soldadura MIG manual existente en una pequeña y mediana empresa dedicada a la fabricación de chasis para maquinaria.

4. METODOLOGÍA.

4.1 Entorno de programación y simulación software: RobotStudio.

El entorno de programación y simulación que se va emplear en el presente TFM es RobotStudio de la empresa ABB en su versión 6.08, del cual dispone licencia de uso escolar esta universidad.

RobotStudio incorpora todas las herramientas necesarias para modelar, programar offline y simular mediante un controlador virtual exactamente igual al del robot real una estación completa robotizada permitiendo realizar animaciones realistas mediante programación de todos los objetos que interactúan en la estación real, de forma que es posible verificar su correcto funcionamiento antes de su implantación, reducir los riegos, permitir una implantación más rápida e incrementar la productividad.

4.1.1 Términos y conceptos empleados en RobotStudio.

Algunos de los términos y conceptos más importantes en RobotStudio son:

- **Flexcontroller**: Se denomina así al armario del controlador denominado IRC5 (5ª generación de controlador de robot ABB). Está compuesto por el módulo de control y el módulo accionamiento de los motores. Contiene, por tanto, el hardware y el software necesario para el control, comunicaciones y accionamiento de un robot o eje externo (track, manipulador, herramienta estacionaria).
- **FlexPendant**: Es la unidad de programación conectada al módulo de control del FlexController y permite la programación “on line” del robot.
- **Herramienta**: El dispositivo montado en el TCP (punto de inserción de la herramienta en el brazo del robot) que permite realizar las diversas tareas necesarias (sujetar, cortar, soldar, pintar,….).
- **Track**: soporte móvil que sostiene al robot para dotarle de una mayor área de trabajo. En el caso de que el controlador del robot controle el movimiento del track éste actúa como un “eje externo” del robot.
- **Posicionador**: manipulador con uno o más ejes que sostiene a la pieza de trabajo y la coloca en la posición necesaria para que el robot realice las tareas programadas. Si el controlador del robot lo acciona, actúa como un “eje externo” del robot.
- **FlexPositioner**: Se denomina así a un robot que actúa como posicionador de la pieza de trabajo y es controlado por el IRC5.
- **Herramienta estacionaria**: Un dispositivo que permanece en una posición fija. El robot lleva la pieza de trabajo y la mueve sobre la herramienta estacionaria para realizar la tarea, como por ejemplo aplicar adhesivo, soldar, etc.
- **Pieza de trabajo**: es la pieza sobre la cual realiza la tarea.
- **Útil**: Es una construcción que sostiene la pieza de trabajo en una posición determinada de forma que permite la repetitividad de la tarea.
- **RobotWare**: Se refiere al software del controlador utilizado para crear las estaciones en RobotStudio. Es posible tener instaladas diferentes versiones.

4.1.2 RAPID básico.

RAPID es el lenguaje de programación de alto nivel empleado en los robots de ABB. El lenguaje consta de un conjunto de instrucciones que describen la actividad del robot en tareas de movimiento, lectura y escritura de salidas. Dichas instrucciones, generalmente, disponen de argumentos asociados en los que se define como debe actuar la instrucción en concreto, por ejemplo, la velocidad o la salida a activar.

Se trata además de un lenguaje de programación completo en el que podemos definir variables, bucles de decisión y de iteración, operaciones lógicas y aritméticas, etc.

Las instrucciones se agrupan para formar rutinas (subprogramas) que pueden ser de tres tipos:

- Procedimientos: se utilizan como subprogramas.
- Funciones: devuelven un valor de tipo concreto y se utilizan como argumento de una instrucción.
- Rutinas TRAP: se emplean para responder a las interrupciones. Una rutina TRAP se asocia a una interrupción determinada, por ejemplo, una entrada.

Un programa **RAPID** se divide en *programa* y *módulos de sistema*, a su vez, el *programa* puede dividirse en *módulos*.

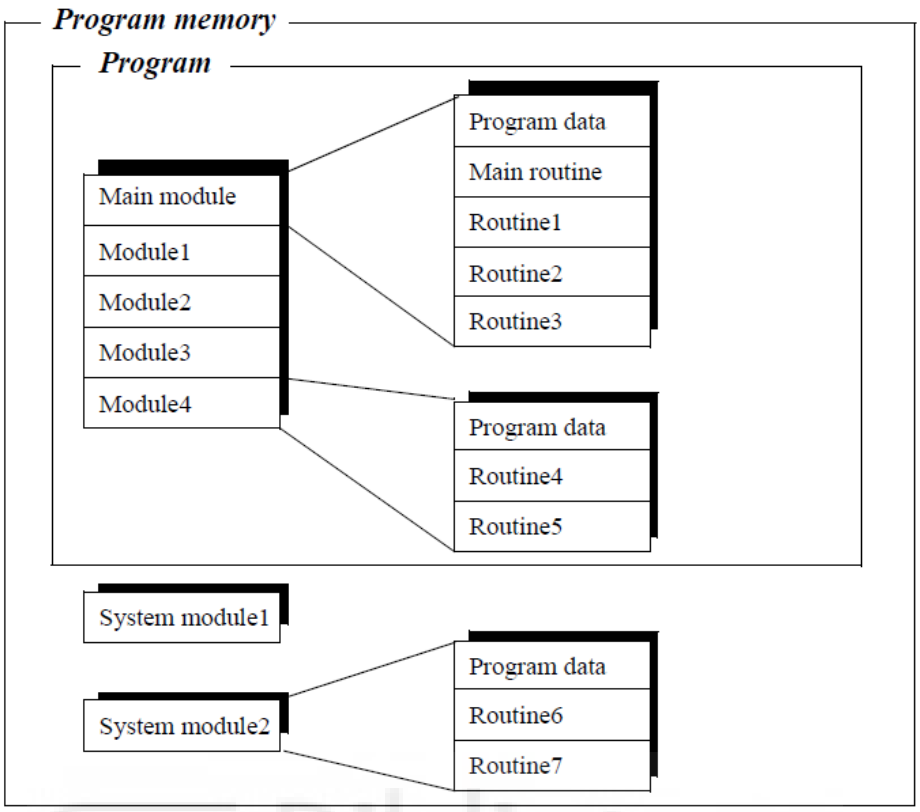


Figura 4: Estructura de un programa RAPID, fuente ABB

Cada módulo de programa puede estar compuesto por datos y rutinas diferentes, pero solo uno de ellos contiene el procedimiento de entrada denominado *main* que es por el que empieza la ejecución del programa.

Los módulos de sistema se usan para definir datos y rutinas comunes y específicos del sistema tales como datos de herramientas. Los módulos de sistema no se guardan con el programa, por lo que si se modifican afectará a todos los programas en memoria o que se carguen posteriormente en ella.

La declaración de un módulo sería:

MODULE module_name (SYSMODULE, VIEWONLY)
!data type definition
!data declarations
!routine declarations
ENDMODULE

Donde el significado de alguno de los atributos opcionales para el módulo son:

- SYSMODULE: El módulo es un módulo de sistema (si no se indica, es un módulo de programa).
- NOSTEPIN: No se permite la activación del módulo durante la ejecución paso a paso.
- VIEWONLY: No se permite la modificación del módulo.
- READONLY: No se permite la modificación del módulo, pero sí la eliminación del atributo.
- NOVIEW: No se permite la visualización del módulo, sino solamente su ejecución. Las rutinas globales pueden utilizarse desde otros módulos y se ejecutan siempre como NOSTEPIN. Los valores de los datos globales pueden usarse en cada momento desde otros módulos o desde la ventana de datos del FlexPendant. El atributo NOVIEW sólo puede definirse fuera de línea desde un PC.

RAPID dispone de múltiples tipos de datos para manejar por ejemplo datos de tipo reloj, fechas…, pero todos se basan en tres siguientes tipos de datos:

- *num*: datos numéricos tanto enteros como decimales, por ejemplo 2 o 3,14.
- *string*: cadenas de texto, por ejemplo “Caracteres”
- *bool*: tipo booleano (lógico), puede tomar valores TRUE o FALSE.

Las variables se declaran con la palabra clave VAR y la siguiente sintaxis:

VAR datatype identifier;

Por ejemplo:

VAR num longitud;
VAR string nombre;
VAR bool finalizado;

Para asignar un valor a una variable, se utiliza la instrucción “:=”, por ejemplo:

longitud:=300;
nombre:=”Juan”;
finalizado:=1;

También posible asignar el valor en la declaración:

VAR num longitud:=300;

En general las variables almacenan los datos si se detiene el programa, pero si se mueve el puntero al programa *main*, se pierde su valor, por ello, en **RAPID** existen un tipo de variables denominadas persistentes que conservan su valor aunque se reinicie el programa.

Las variables persistentes se declaran con la palabra clave PERS. En el momento de la declaración es necesario indicar un valor inicial.

PERS num nbr := 1;

PERS string string1 := "Hello";

En **RAPID** también es posible definir constantes, que como a cualquier variable se le puede asignar un valor pero solo en el momento de la declaración y posteriormente ya no es posible cambiarlo. Son útiles para definir valores que no deben cambiar en la ejecución del programa y permiten a su vez una fácil actualización de los programas, por ejemplo para definir offsets.

La constante se declara con la palabra clave CONST seguida del tipo de dato, el identificador y la asignación de un valor.

CONST num gravity := 9.81;

CONST string greating := "Hello"

Los operadores usados en **RAPID** se clasifican en:

- numéricos: suma (+), resta (-), producto (*), división (/).
- relacionales: igual a (=), menor que (<), mayor que (>),menor que o igual a (<=), mayor que o igual a(>=), distinto que (<>).
- de cadena: concatenación de cadena de string (+).

Para el control del flujo de programa dispone de las instrucciones:

- IF:

```
VAR string string1 := "Hello";
IF string1 <> "" THEN
TPWrite string1;
```

ENDIF

- ELSE:

VAR string string1 := "Hello";
IF string1 <> "" THEN
TPWrite string1;
ELSE
TPWrite "The string is empty";
ENDIF

- ELSEIF

VAR num time := 38.7;
IF time < 40 THEN
TPWrite "Part produced at fast rate";
ELSEIF time < 60 THEN
TPWrite "Part produced at average rate";
ELSE
TPWrite "Part produced at slow rate";
ENDIF

Es posible también anidar sentencias IF.

- FOR: La sintaxis es la siguiente.

FOR contador FROM valorinicial TO valorfinal DO
código de programa a repetir
ENDFOR

La variable utilizada en el contador (por ejemplo i) no es necesario declararla antes, se inicializa con el valor indicado en *valorinicial* y se incrementa en 1 cada vez hasta llegar al valor indicado en *valorfinal* ejecutándose a continuación el código que hay tras ENDFOR. No se permite asignar un valor al contador en el bucle FOR.

- WHILE: La sintaxis es la siguiente.

WHILE condición DO
código de programa a repetir
ENDWHILE

Si la condición no se cumple al inicio, no se ejecuta el código dentro del WHILE. Mientras será cierta la condición se repetirá el código

Para insertar comentarios en el código del programa que no se ejecutarán se utiliza el símbolo “!”, por ejemplo:

```
! Calcula la suma de dos números
suma := num1 + num2;
```

RAPID dispone de diversas funciones de movimiento que permiten mover el robot y realizar diferentes tipos de acciones simultáneamente tales como accionar salidas.

Una instrucción de movimiento simple como *MoveL* que realiza una trayectoria lineal del TCP del robot desde el punto actual al punto especificado en su argumento *ToPoint* tiene la siguiente sintaxis:

MoveL [\Conc] ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj] [\Corr] [\TLoad]

Donde los argumentos entre corchetes son opcionales, siendo obligatorios:

- *ToPoint* : Punto de destino del robot y de los ejes externos si existen. Es un dato de tipo *robtarget*.
- *Speed*: Velocidad que se aplica a los movimientos del TCP, la reorientación de la herramienta y los ejes externos en mm/s.
- *Zone*: Especifica con que exactitud debe alcanzarse el objetivo con una contante de tipo *zonedata*. Por ejemplo con *fine* se alcanza el objetivo con exactitud y con *Z10* con un margen alrededor del punto de máximo 10 mm. Existen múltiples valores predefinidos.
- *Tool*: Especifica la herramienta utilizada por el robot, definida por una variable persistente del tipo de dato tooldata. El TCP del robot pasa a la punta de la herramienta. Esto se realiza automáticamente si se declara, asigna y utiliza un valor de tooldata en la instrucción MoveL. tool0 es una herramienta predefinida que representa al robot sin ninguna herramienta montada en él y no debe ser declarada ni asignada. Cualquier otra herramienta debe ser declarada y asignada antes de usarla.

Para más detalles sobre la instrucción, consultar el Manual de Referencia Técnica de **RAPID**.

Por ejemplo, las siguientes instrucciones darán lugar a la trayectoria de la figura:

MoveL p10, v1000, z50, tool0;

MoveL p20, v1000, fine, tool0;

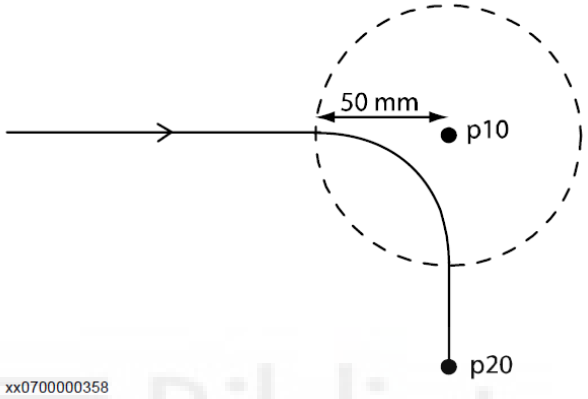


Figura 5: Zona en una trayectoria, fuente ABB.

RAPID cuenta con otras instrucciones de movimiento y entre las más utilizadas habitualmente, destacan las siguientes:

- *MoveAbsJ*: mueve el robot y los ejes externos a una posición absoluta definida por la posición de los ejes con un movimiento no lineal. Todos los ejes alcanzan la posición al mismo tiempo.

  MoveAbsJ [\Conc] ToJointPos [\ID] [\NoEOffs] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj] [\TLoad]

- *MoveJ*: mueve el robot y los ejes externos desde el punto actual al objetivo definido con un movimiento no lineal. Todos los ejes alcanzan la posición al mismo tiempo.

MoveJ [\Conc] ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj] [\TLoad]

- *MoveC*: mueve el TCP del robot en un movimiento circular desde el punto actual al de destino. Durante el movimiento la orientación de la herramienta permanece fija respecto al círculo. Se debe dar un objetivo de paso *CirPoint* y el de destino *Topoint*. Para realizar un círculo completo son necesarias dos instrucciones *MoveC*.

MoveC [\Conc] CirPoint ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj] [\Corr] [\TLoad]

Los sistemas de coordenadas en **RAPID** son importantes porque la posición de destino de una instrucción de movimiento depende de ellos. Si no se especifica nada en la instrucción, la posición objetivo se toma como relativa al sistema de coordenadas de la base del robot denominada w*obj0*.

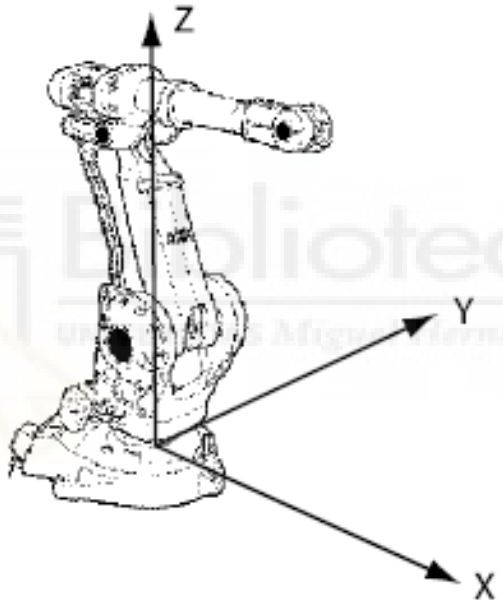


Figura 6: Sistema de coordenadas de la base, fuente ABB.

Es posible definir y utilizar otros sistemas de coordenadas en las instrucciones de movimiento mediante el argumento *[\Wobj]*, en el siguiente ejemplo el sistema de coordenadas definido por el usuario y sobre el que se referencia el objetivo *p10* es *wobj1*:

MoveL p10, v1000, z50, tool0 \WObj:=wobj1;

En el apartado 4.1.2.4 se amplía la información sobre los sistemas de referencia y como usarlos en RobotStudio.

4.1.3 Flujo de trabajo en RobotStudio.

En RobotStudio cuando iniciamos el trabajo creamos soluciones y el término SOLUCIÓN lo define como “el nombre colectivo para la carpeta que contiene la estructura para estaciones, bibliotecas y todos los elementos relacionados”. Dicha carpeta se crea cuando el usuario crea una nueva solución y contiene:

- **Estaciones**: las estaciones creadas como parte de la solución.
- **Sistemas**: controladores virtuales creados.
- **Bibliotecas**: definidas por el usuario u utilizadas en la estación
- **Archivo de solución**: el archivo que abre la solución.

Robot Studio permite iniciar una solución de diferentes formas y en función del tipo de estación (solo con robot o con posicionador, track, Transportador, sistemas Multimove, etc) la forma de iniciar la solución recomendada varía básicamente en añadir los elementos de la estación antes o después de crear el sistema del controlador virtual con las opciones necesarias.

El flujo de trabajo para crear una solución **básica** con un robot en RobotStudio es el siguiente:

1. Creación de una estación desde la pestaña Archivo
    a. Seleccionamos la opción “Solución con estación y controlador de robot”

    En la parte derecha:

    - Nos permite dar un nombre a la solución (por defecto Solution X, donde X es el número consecutivo que va añadiendo RobotStudio).
    - Modificar la ubicación de la carpeta de la solución.
    - Modificar el nombre del sistema del controlador virtual. Por defecto escoge el del robot seleccionado.
    - Crear una solución nueva o una a partir de una copia de seguridad existente.
    - Seleccionar la versión del controlador si tenemos varias instaladas.
    - Seleccionar un robot, y si marcamos la opción “Personalizar opciones” nos abre la pantalla de las opciones para configurar el controlador virtual.

    De esta forma se crea una solución con su estructura de carpetas con un robot y un sistema controlador virtual con las opciones elegidas.
2. Incorporar componentes a la estación:

a. Importar otros equipos desde archivos CAD.
b. Piezas de trabajo desde archivos CAD.

3. Colocar los objetos y mecanismos:
    a. Colocar objetos y piezas de trabajo en la posición exacta.
    b. Conexión de herramientas a robots.
    c. Comprobar alcanzabilidad.
4. Creación de objetivos y trayectorias.
5. Crear programa **RAPID**.
6. Simulación.

Si la solución que estamos creando está compuesta por más de un robot, posicionadores, Transportadores, Tracks, etc., el flujo de trabajo puede variar un poco y en apartados posteriores se describen algunos de esos casos.

4.1.3.1 Creación de una estación.

Al iniciar la aplicación RobotStudio, nos muestra la pestaña “Archivo” desde la cual nos ofrece tres opciones para crear una nueva solución.

- **Solución con estación vacía**: Crea una solución con la estación vacía y la estructura de carpetas anteriormente mencionada, sin robots ni sistemas de controlador virtual, posteriormente se añaden.

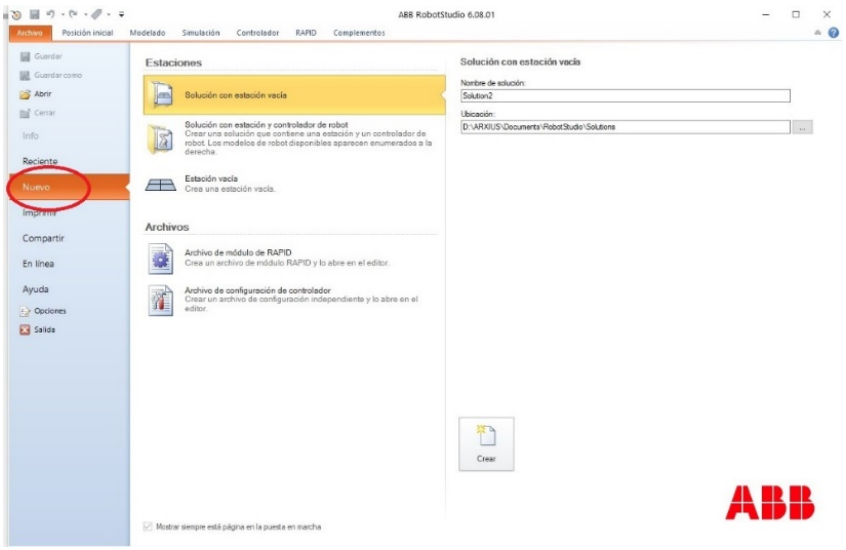


Figura 7: Solución con estacón vacía.

- **Solución con estación y controlador de robot**: Crea una solución con la estructura de carpetas anteriormente mencionada, Añadimos el robot que seleccionemos y creamos el sistema de controlador virtual con la versión deseada si hay varias instaladas. El nombre del robot que añadamos a la estación lo aplica también el sistema creado. Es posible seleccionar las opciones del sistema del controlador virtual.

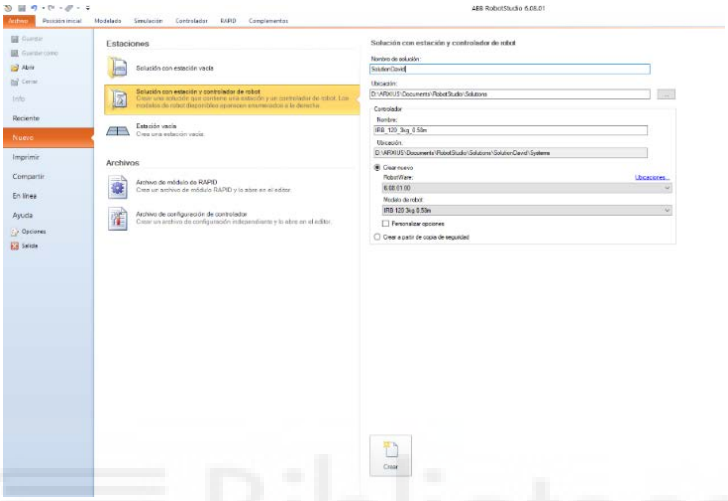


Figura 8: Solución con estación y controlador.

- **Estación vacía**: Crea una estación vacía. Posteriormente añadimos el robot y el sistema del controlador virtual. Usa las carpetas “Stations” y “Systems” en /Documentos/RobotStudio creadas por defecto, no se crea la estructura de carpetas de una solución completa.

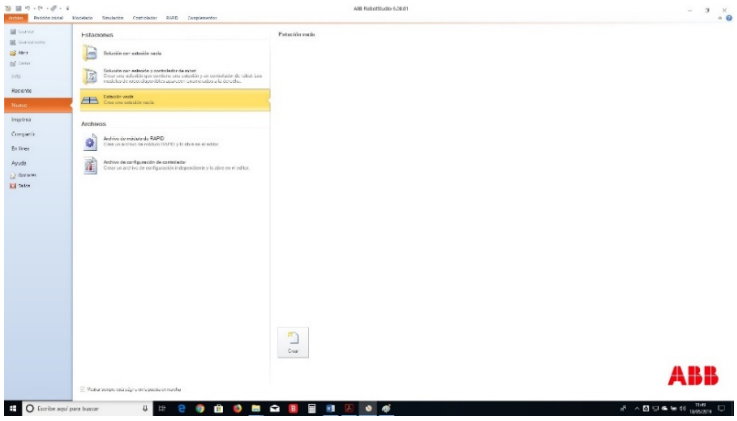


Figura 9: Estación vacía.

4.1.3.2 Inserción de componentes en la estación.

Una vez creada la solución o estación vacía conforme al apartado anterior, podremos incorporar los componentes necesarios:

- Importación de un modelo de robot:

En la pestaña “Posición inicial” hacer clic en “Biblioteca ABB” y seleccionar el robot en el menú desplegable.

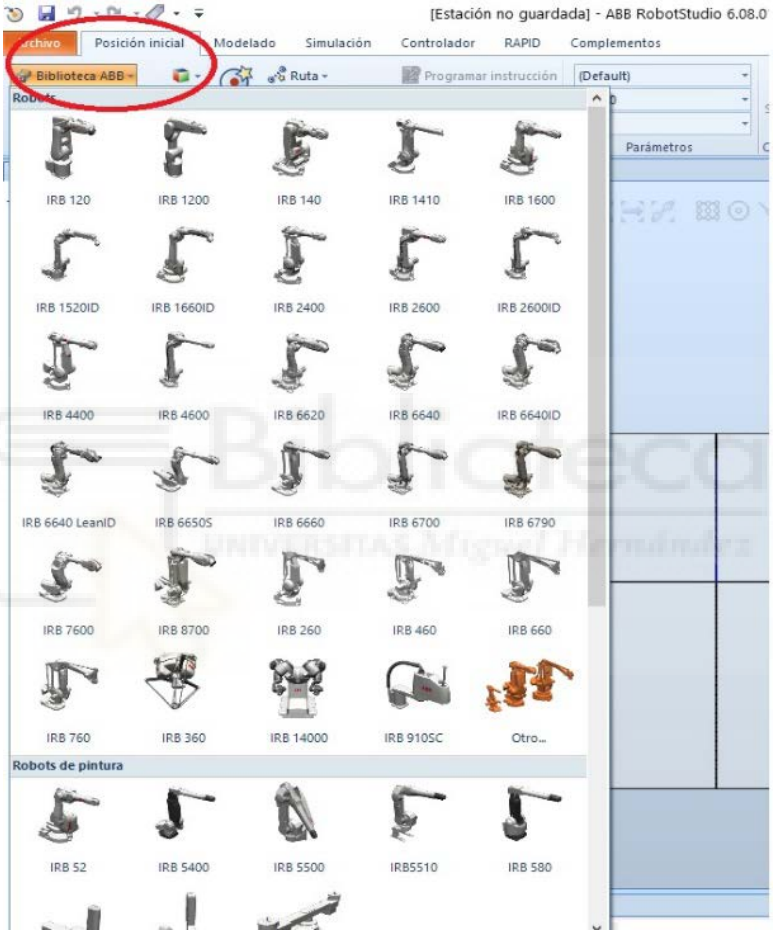


Figura 10: Biblioteca ABB.

Dependiendo del robot seleccionado, es posible que tenga diferentes versiones (capacidad y/o alcance) que podremos escoger en la pantalla que nos aparece al seleccionar un robot:

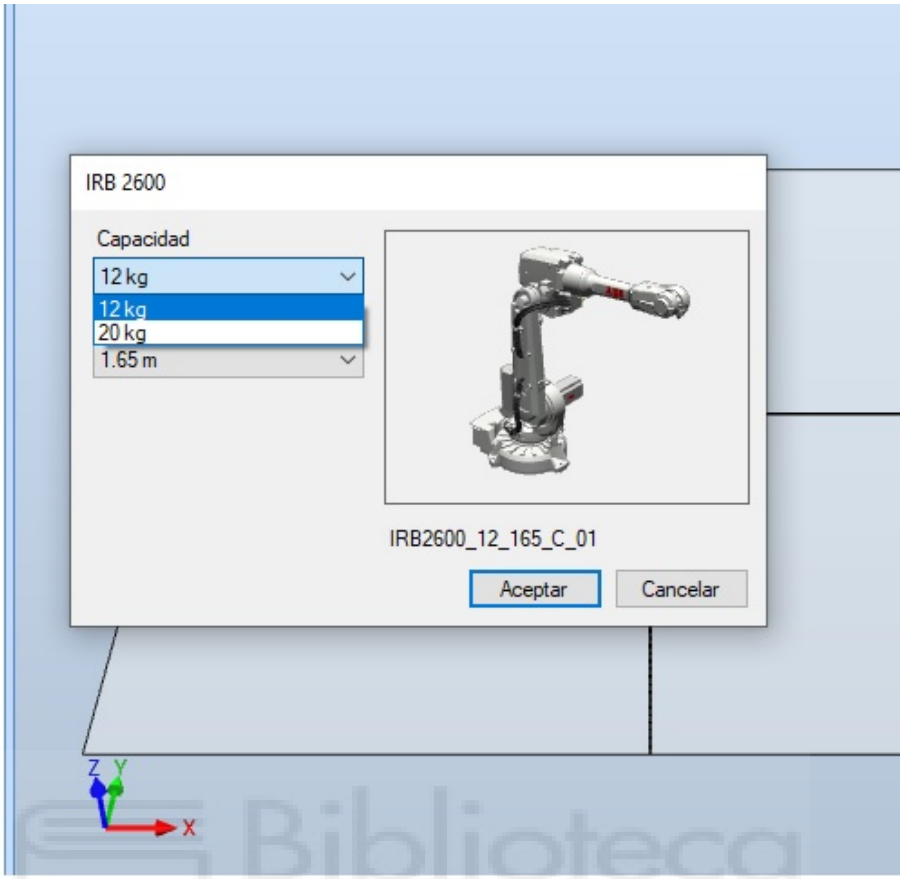


Figura 11: Selección del robot.

El robot insertado es solo un modelo gráfico, debe asociarse a un sistema (controlador virtual) para que pueda programarse y realizar movimientos, etc.

- Importación de una herramienta:

Una herramienta es un objeto que actúa sobre una pieza de trabajo, por ejemplo una pistola de soldadura, un taladro, una pulidora, etc.

Para poder obtener el posicionado correcto del robot con la herramienta es necesario definir correctamente los parámetros de la herramienta y entre ellos su TCP es el más importante y que es la posición del punto central de la herramienta respecto al TCP del robot denominado “tool0”.

RobotStudio dispone de algunas herramientas creadas que podemos usar. Desde la pestaña “Posición inicial” hacemos clic sobre “Importar biblioteca” y seleccionamos una de las herramientas disponibles.

Si es necesario podemos crear una herramienta a partir de un modelo CAD importado o dibujado en RobotStudio.

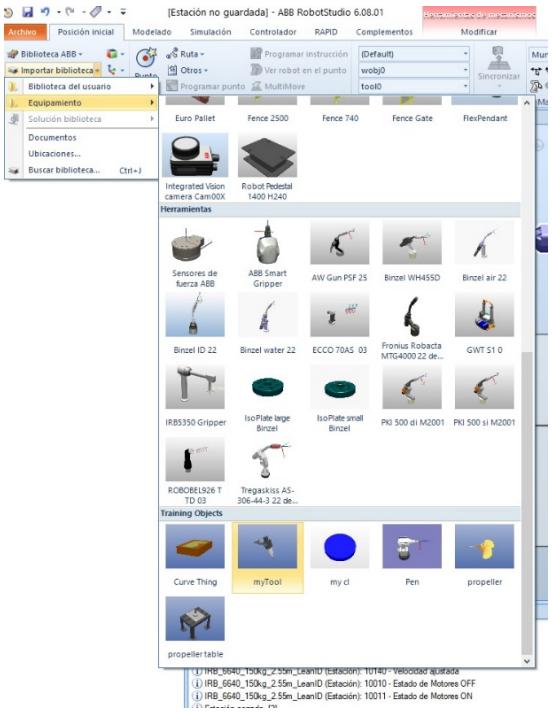


Figura 12: Importar herramienta.

Al importarse la herramienta esta se sitúa en el origen de coordenadas gráfico de la estación de RobotStudio, por lo que es necesario conectarla al robot posteriormente para poder usarla.

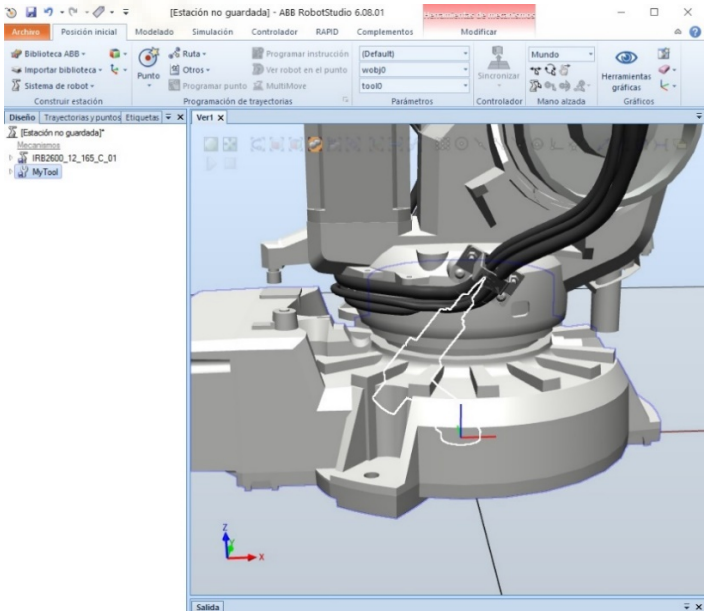


Figura 13: Inserción de la herramienta en la estación.

- Importación de un posicionador.

En la pestaña “Posición inicial” hacer clic en “Biblioteca ABB” y seleccionar el posicionador en el menú desplegable.

Figura 14: Importar posicionador.

- Importación de un track:

Básicamente hay dos formas de importar un track a una estación, la primera y más sencilla es importar el robot y el track y luego crear el sistema seleccionando ambos en la pantalla de configuración del sistema. La segunda opción es añadir el robot, crear el sistema y luego añadir el track. Luego debemos editar el sistema y añadir el archivo de configuración del track correspondiente y reiniciar el controlador .Posteriormente en el apartado 5.4.4 .de describen con detalle estos procedimientos, comentando ahora solo como importar in track.

Para importar un track, desde la pestaña “Posición inicial” hacer clic en “Biblioteca ABB” y seleccionar el track en el menú desplegable.

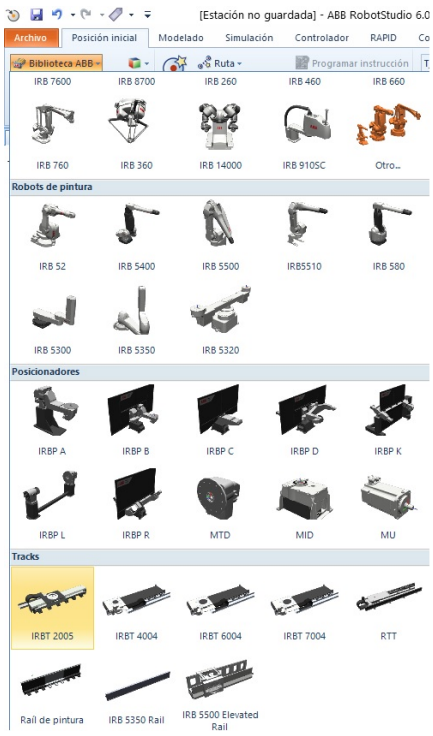


Figura 15: Importar track.

- Importación de una biblioteca, geometría o elemento de equipo:

Para la simulación de la estación podemos crear objetos tales como piezas de trabajo, equipos…con las herramientas de que dispone RobotStudio o realizarlos en un software externo de diseño 3D e importarlos posteriormente a la estación.

Un archivo de biblioteca es un archivo guardado desde RobotStudio como un archivo externo y se diferencia de importar un archivo CAD externo en que se crea un enlace entre la estación y archivo de biblioteca y por tanto no aumenta el tamaño que ocupa en memoria la estación.

En cuanto a los formatos de archivo de CAD que admite RobotStudio, en la ayuda del programa puede consultarse una tabla con los formatos admitidos, entre los cuales encontramos los más habituales como .3ds, .sat, .dxf, .dwg, .sldrpt, .sldasm, .stl (ASCII), .step, etc.

Para importar una geometría externa, desde la pestaña "**Posición inicial**", hacemos clic en el icono "**Importar Geometría**"

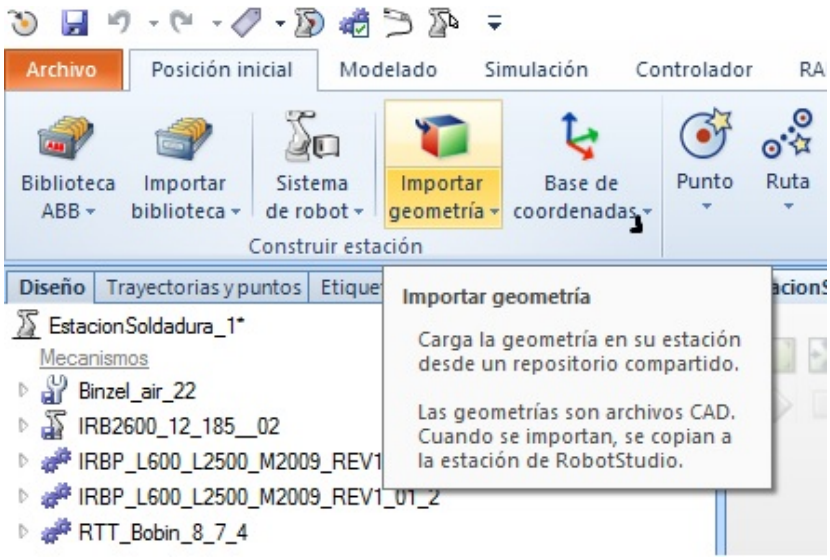


Figura 16: Importar geometría.

Para crear una biblioteca a partir de una geometría creada en RobotStudio o importada, seleccionamos la geometría en el árbol de navegación "**Diseño**" haciendo clic sobre ella con el botón derecho del ratón, en la ventana flotante que aparece seleccionamos "**Guardar como biblioteca**"

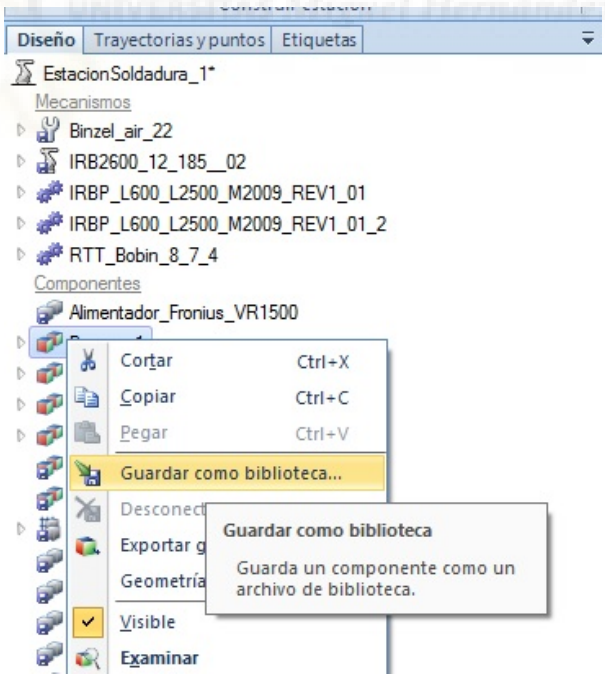


Figura 17: Guardar como biblioteca.

Podemos importar una biblioteca desde la pestaña “**Posición inicial**”, haciendo clic en el icono “**Importar Biblioteca**”.

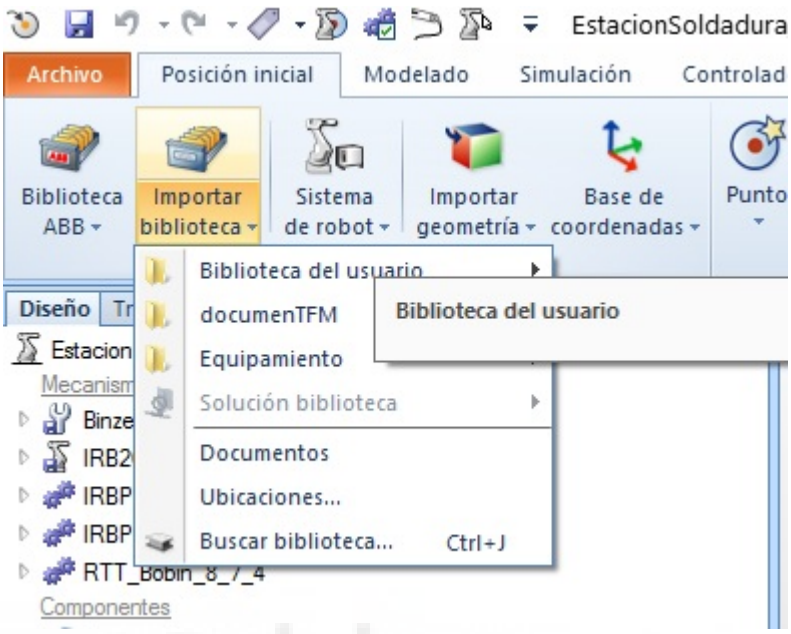


Figura 18: Importar biblioteca.

4.1.3.3 Creación de sistemas.

El sistema o controladora virtual se puede crear en el inicio cuando se crea una estación tal como se describe en el apartado **4.1.3.1** mediante la opción “**Solución con estación y controlador de robot”** o bien se puede crear la estación sin sistema y crearlo posteriormente como se describe en este apartado.

Es muy recomendable, antes de crear el sistema, incorporar a la estación todos los sistemas a controlar tales como robots, posicionadores, Tracks y transportadores y colocarlos en la estación en su posición de trabajo, ya que al crear el sistema RobotStudio nos incluirá automáticamente la opciones de configuración necesarias.

Desde la pestaña “Posición inicial” el botón “Sistema Robot” nos permite crear el sistema controlador a través de tres opciones distintas:

- “**Desde diseño**”: permanece desactivada si no se ha insertado en la estación previamente algún robot, posicionador o track. Es la opción más sencilla de iniciar un sistema. Los pasos a seguir se describen a continuación:

Para crear un sistema a partir de una estación vacía, creamos primero la estación como se describe en **4.1.3.1** y luego, desde la pestaña "**Posición inicial**" seleccionamos "**Bibliioteca ABB**" y añadimos, al menos, un robot, posicionador o track, por ejemplo el IRB 120.

Figura 19: Importar robot.

Desde la pestaña "**Posición inicial**", hacemos clic sobre "**Sistema Robot**" y seleccionamos "Desde diseño…"

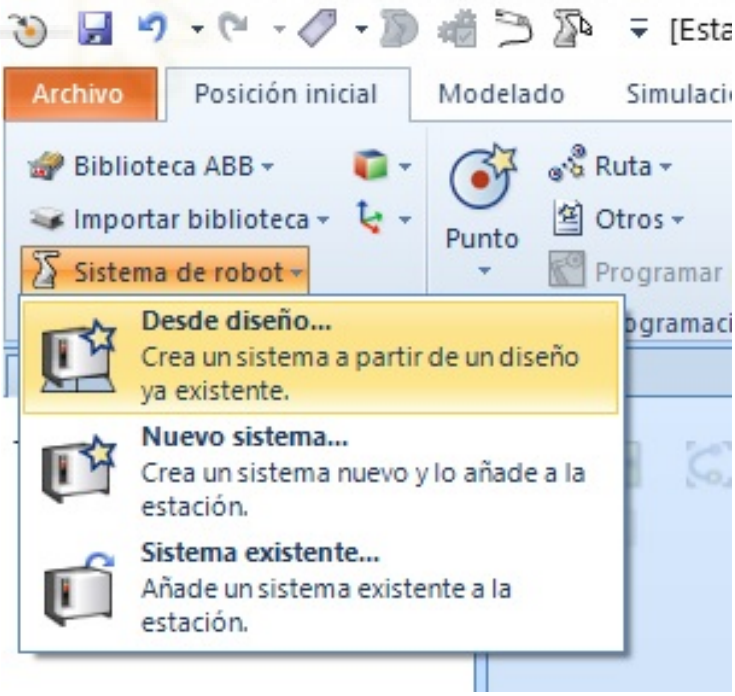


Figura 20: Crear estación "Desde diseño..."

Las siguientes pantalla nos permiten cambiar el nombre del sistema, seleccionar la ruta de guardado, seleccionar los robots, Tracks, posicionadores o transportadores a incluir en el controlador (en caso de que se hayan insertado en la estación ya) y mediante el botón "**Opciones**" incluir todas las opciones de configuración del sistema necesarias tales como idioma, tarjetas de comunicación, opciones de coordinación entre robots, Tracks, etc. de acuerdo a las licencias que se tengan adquiridas.

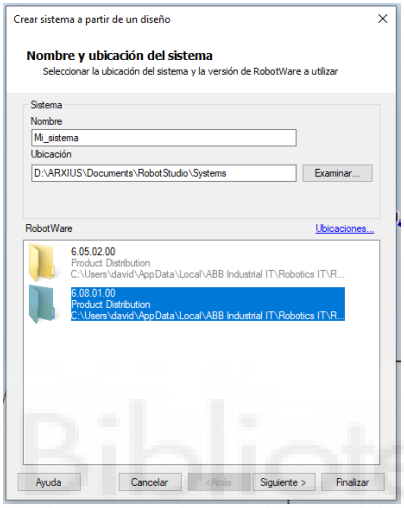


Figura 21: Nombre y ubicación del sistema.

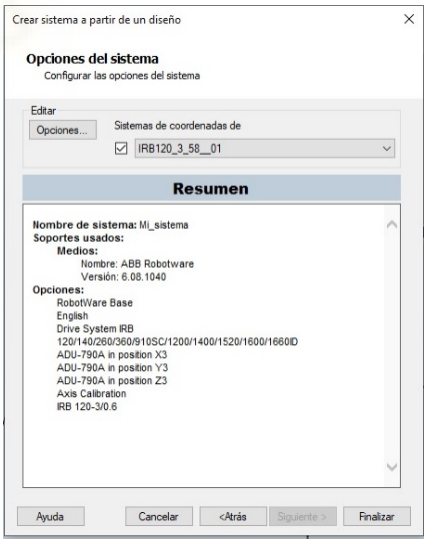


Figura 22: Opciones del sistema.

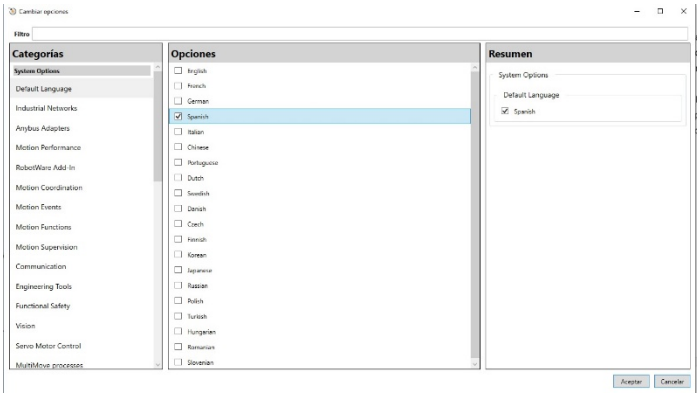


Figura 23: Opciones del sistema: idioma.

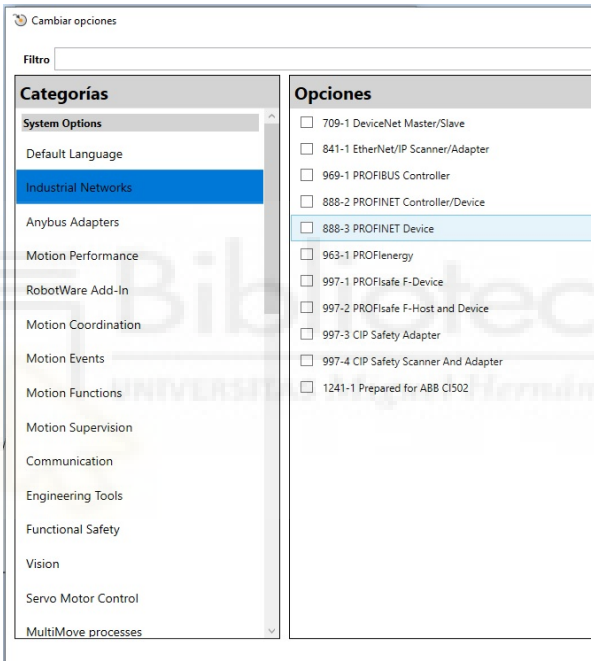


Figura 24: Opciones del sistema disponibles.

- "**Nuevo sistema**": Mediante esta opción creamos un sistema desde cero, seleccionando la versión RobotWare (si tenemos más de una instalada), el robot y además nos permite seleccionar las opciones necesarias.

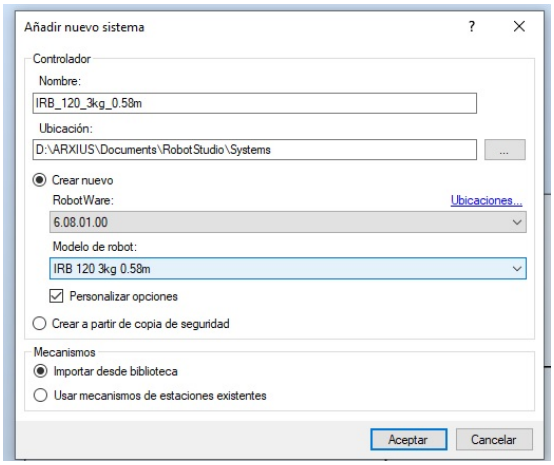


Figura 25: Nuevo sistema.

- "**Sistema existente**": Esta opción nos permite añadir a la estación un sistema previamente creado y guardado.

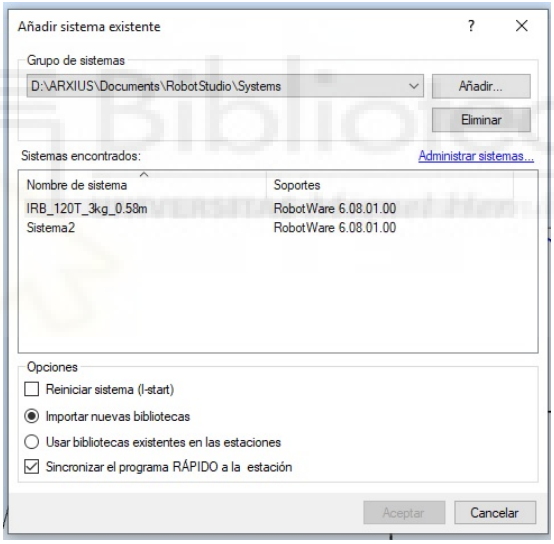


Figura 26: Sistema existente.

4.1.3.4 Sistemas de referencia en RobotStudio.

En el entorno virtual de RobotStudio usa un sistema de coordenadas jerárquico, en el cual el origen de cada sistema de coordenadas se define como una posición en cada uno de sus predecesores en el orden jerárquico. Los sistemas de coordenadas en RobotStudio son:

- Sistema de coordenadas de la estación (**RS-WCS**): Las siglas provienen de **R**obot**S**tudio **W**orld **C**oordinate **S**ystem y está en la cúspide de la jerarquía. Todos los sistemas de coordenadas de la estación dependen de él. Cuando se inserta un objeto (robot, posicionador, etc) en la estación se sitúa en el origen de RS-WCS.

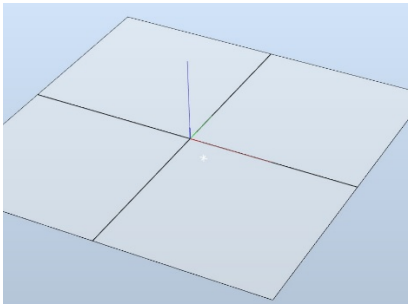


Figura 27: Sistema de coordenadas RS-WCS.

- Sistema de coordenadas del punto central de la herramienta (**TCP / tool0**): Es el punto donde queremos que la herramienta se sitúe en el espacio, por defecto un robot sin herramienta colocada en su extremo tiene su TCP en el punto de montaje de la herramienta denominado "**tool0**". Cada herramienta tiene definido su TCP con un nombre y además es posible tener definidos varios TCP distintos. Cuando se definen los puntos que debe alcanzar el robot, es necesario indicar en que sistema de coordenadas de herramienta queremos que estén situados.

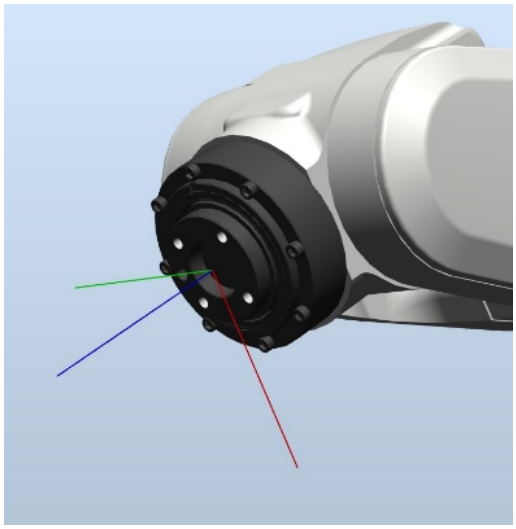


Figura 28: Sistema de coordenadas del TCP del robot.

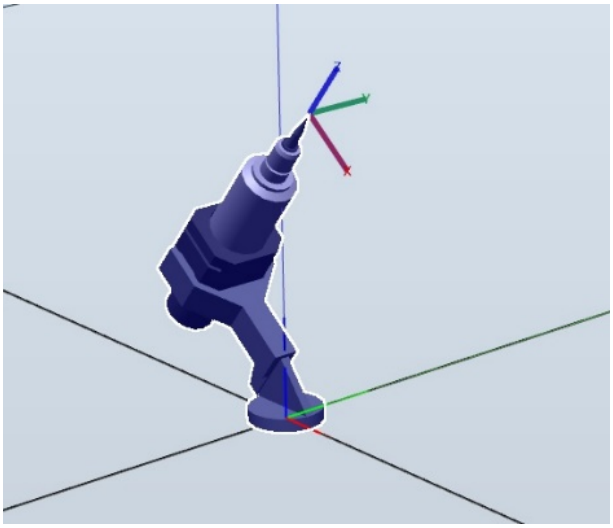


Figura 29: Sistema de coordenadas del TCP de la herramienta.

- Sistema de coordenadas de la base (**BF**): Es el sistema de coordenadas del robot que está situado en su base, tanto en la estación virtual de RobotStudio como en el mundo real.

- Sistema de coordenadas de la tarea (**TF**): Este sistema es el origen del sistema de coordenadas mundo del **controlador**, tanto en la estación virtual como en el mundo real, es decir, es el sistema de coordenadas en el cual se basa el controlador para mover y situar la herramienta en el punto deseado del espacio. Inicialmente cuando se sitúa un robot en la estación, los sistemas de coordenadas **RS-WCS**, **BF** y **TF** coinciden. El **TF** es posible cambiarlo de posición respecto de la base del robot. Si movemos el robot de posición, nos pregunta si deseamos mover también el sistema de coordenadas **TF** o no.

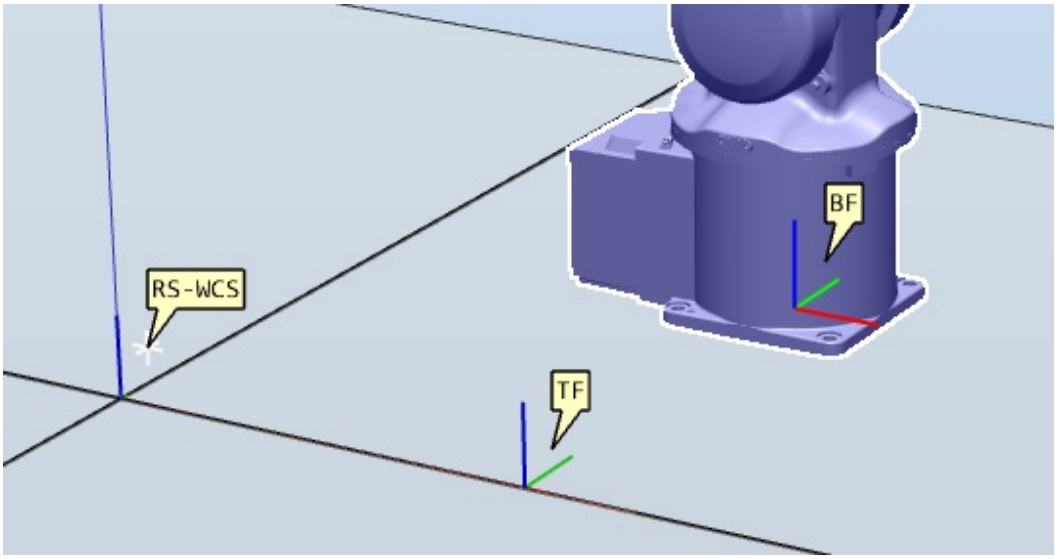


Figura 30: Sistemas de coordenadas RS-WCS, BF y TF.

- Sistemas de coordenadas del objeto de trabajo y del usuario: La pieza física sobre la que se va a realizar la tarea se denomina en el entorno de RobotStudio como "**objeto de trabajo**" y es posible crear un sistema de coordenadas denominado "**objeto de trabajo**" y asociarlo a la pieza de forma que cuando se crean los objetivos o posiciones que debe alcanzar el robot, estos estén referidos a dicho sistema de coordenadas, lo cual simplifica mucho la programación, ya que si se mueve el objeto, de forma automática, se mueven los objetivos. También permite en el caso de no asociarlo a la pieza, utilizar un offset para realizar ajustes en caso de que la posición de la pieza en el mundo real no coincida exactamente con su posición en la estación respecto del **TF**. En el caso de que no se especifique ningún objeto de trabajo, se toma por defecto el denominado **Wobj0**, que coincide con el sistema de coordenadas de la tarea.

  En realidad, el sistema de coordenadas denominado "objeto de trabajo" consta de dos sistemas de coordenadas denominados sistemas de coordenadas del usuario (**workobject_2_uf**) y el sistema de coordenadas del objeto (**workobject_2_of**) en la imagen. El sistema de coordenadas del usuario depende del sistema de coordenadas de la tarea (**TF**) y el sistema de coordenadas del objeto depende del sistema de coordenadas del usuario. Cuando programamos un punto objetivo, este toma sus coordenadas en el sistema de referencia de objeto.

En la siguiente imagen podemos observar que se ha creado un **workobject** en el que se ha situado el sistema de coordenadas del usuario en las coordenadas (50,50;0) respecto de **TF** y el sistema de coordenadas del objeto en (100,100,0) respecto del sistema de coordenadas del usuario.

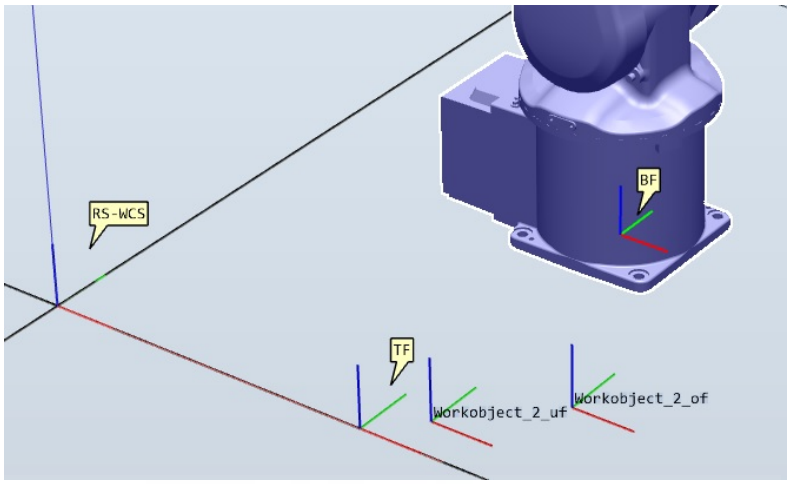


Figura 31: Sistema de coordenadas de un objeto de trabajo en un TF desplazado del BF.

En la siguiente figura, podemos observar el sistema de coordenadas mundo de la estación (RS-WCS) coincidiendo con el sistema de coordenadas de la tarea (TF) en el suelo, en una esquina de la mesa de apoyo de la pieza se ha situado el sistema de coordenadas del usuario y en una esquina de la pieza de trabajo el sistema de coordenadas del objeto

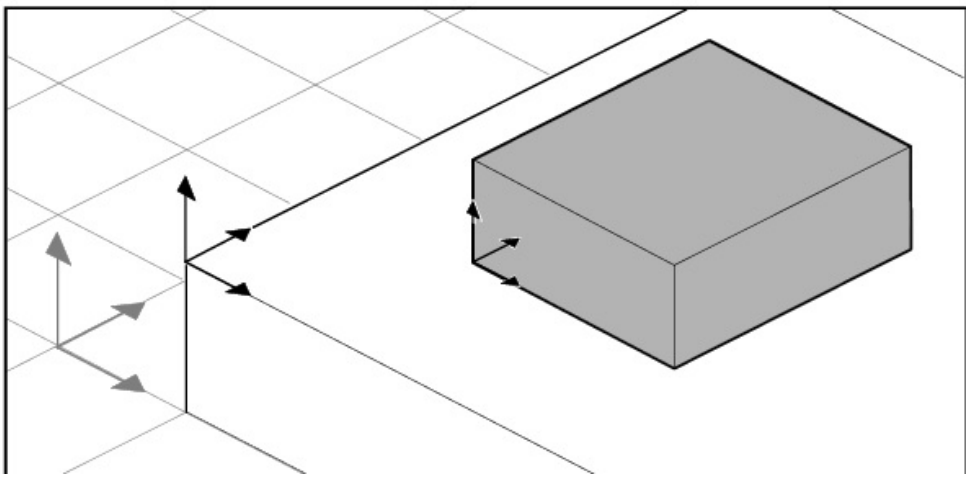


Figura 32: Sistema de coordenadas "objeto de trabajo".

En la siguiente figura podemos observar la relación entre los sistemas de coordenadas de la base y de la tarea en la estación virtual de RobotStudio.

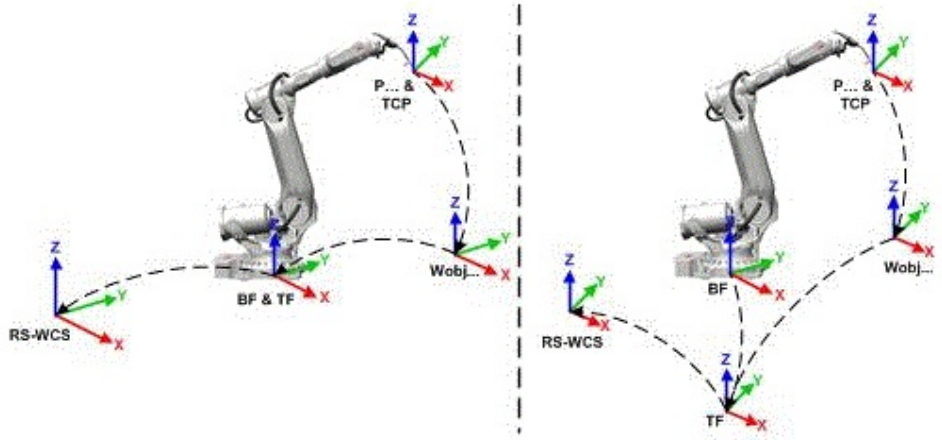


Figura 33: Relación entre los sistemas de coordenadas en RobotStudio.

La relación entre los sistemas de coordenadas de la estación virtual de RobotStudio (a la izquierda) y el mundo real (a la derecha) se pueden observar en la siguiente figura.

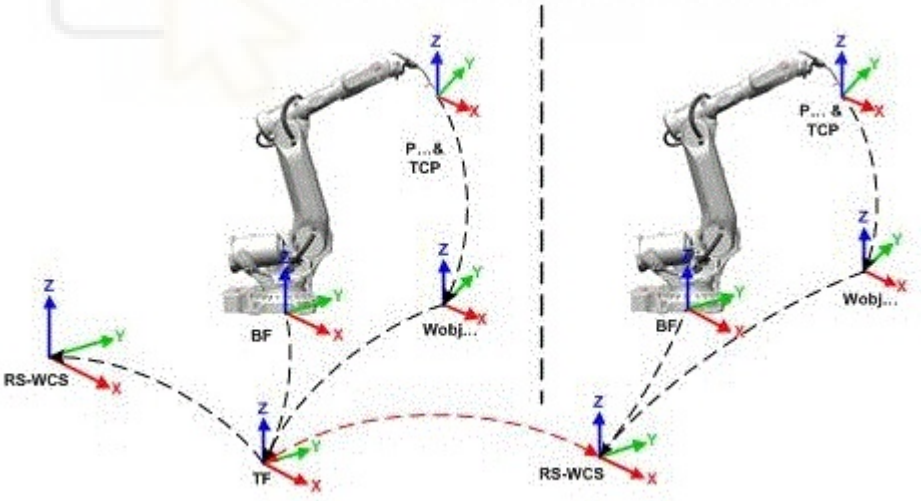


Figura 34: Relación entre los sistemas de coordenadas de RobotStudio y la estación real.

Hacer notar que en la figura del manual hay una errata, en la parte derecha donde pone **RS-WCS** (Sistema de Coordenadas Mundo de RobotStudio) debe poner **RC-WCS** (Sistema

de Coordenadas Mundo del Controlador). Como podemos observar en el mundo real el sistema de coordenadas mundo de la controladora coincide con el sistema de coordenadas de la tarea (**TF**) en la estación virtual de RobotStudio, y el resto de sistemas de coordenadas están referidos a él directa o indirectamente.

4.1.3.5 Creación de una herramienta y conexión al robot.

En RobotStudio podemos importar una herramienta ya creada y conectarla al TCP del robot, o bien, crear nuestra propia herramienta haciendo uso de la pestaña modelado de que dispone. Una vez creado el objeto gráfico que constituye la herramienta seleccionaremos es asistente "**Crear herramienta**" de la pestaña modelado que nos permitirá crear la herramienta para su uso en RobotStudio. El flujo de trabajo es el siguiente:

- Dibujar la herramienta en un software externo e importarla a RobotStudio o crearla con las herramientas de dibujo disponibles en la pestaña modelado. Creamos un cono de radio 13 mm y longitud 150 mm que se inserta en el (0,0,0) del sistema coordenadas mundo de la estación virtual de RobotStudio (**RS-WCS).**

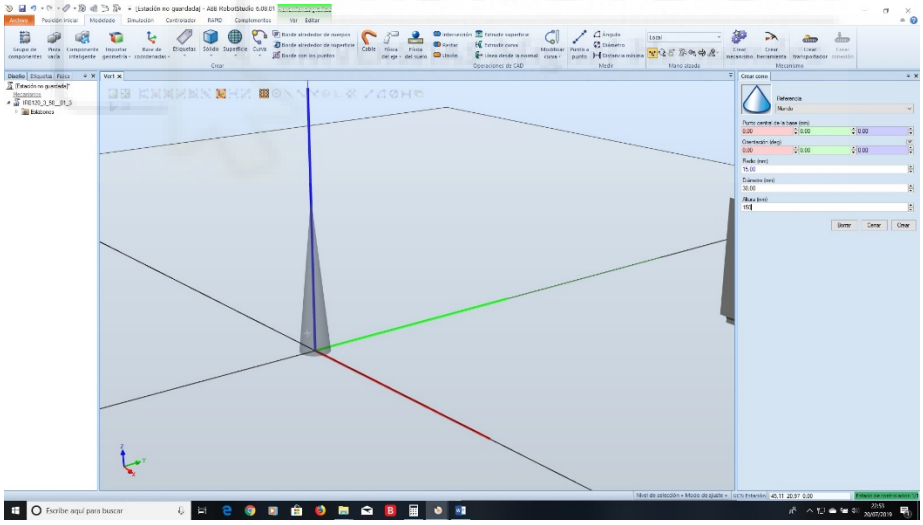


Figura 35: Modelado de una herramienta ejemplo.

- En la pestaña modelado hacer clic sobre el icono "Crear herramienta"

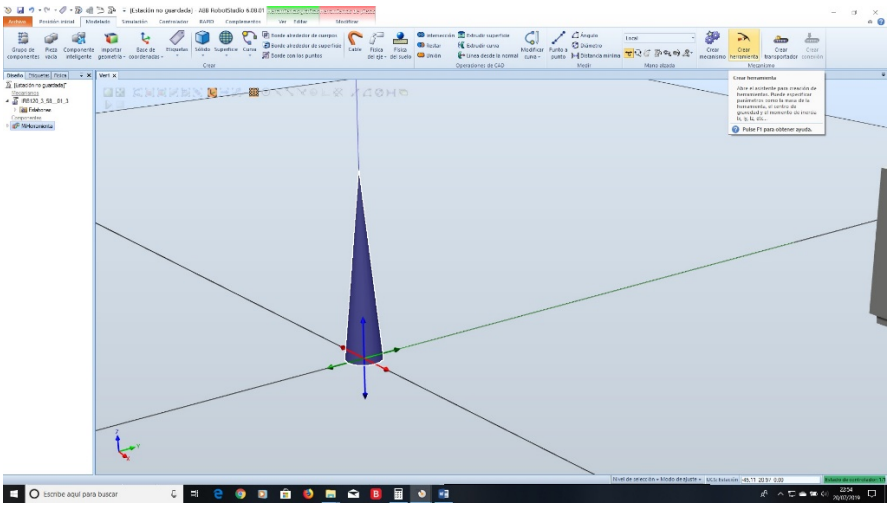


Figura 36: Creación de una herramienta I.

- Se abre una ventana flotante, seleccionamos “Usar existente” para que podamos elegir el objeto creado e iremos introduciendo la información requerida. En este caso el nombre que hemos dado a la herramienta, la masa y el centro de gravedad dejando a cero el momento de inercia por no disponer de datos.

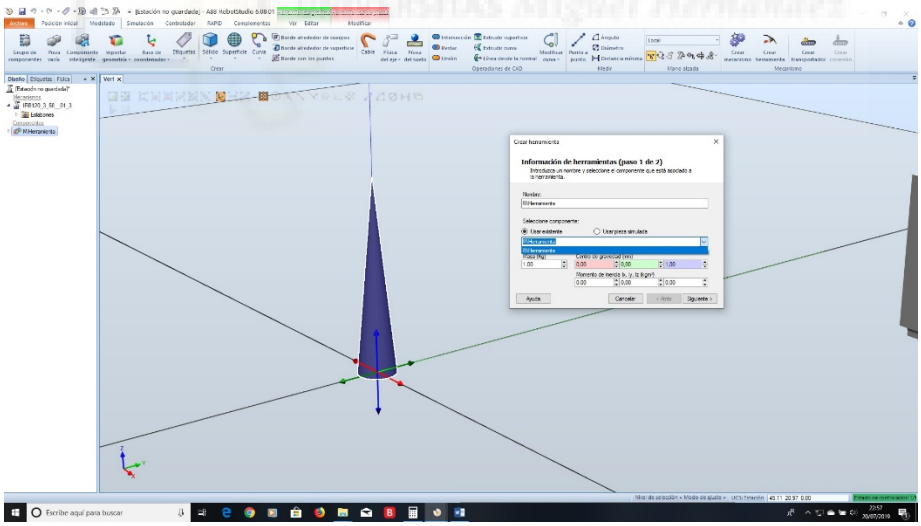


Figura 37: Creación de una herramienta II

- En el siguiente paso debemos especificar el nombre que le damos al TCP de la herramienta, su posición (0,0,150) y orientación (0,0,0) y lo añadimos a la lista de

TCPs clicando sobre el botón alargado. Pulsado sobre el botón “Terminado” creamos la herramienta (en el caso de que se hubiera creado una herramienta más compleja con más de un TCP podríamos crearlos en esta ventana).

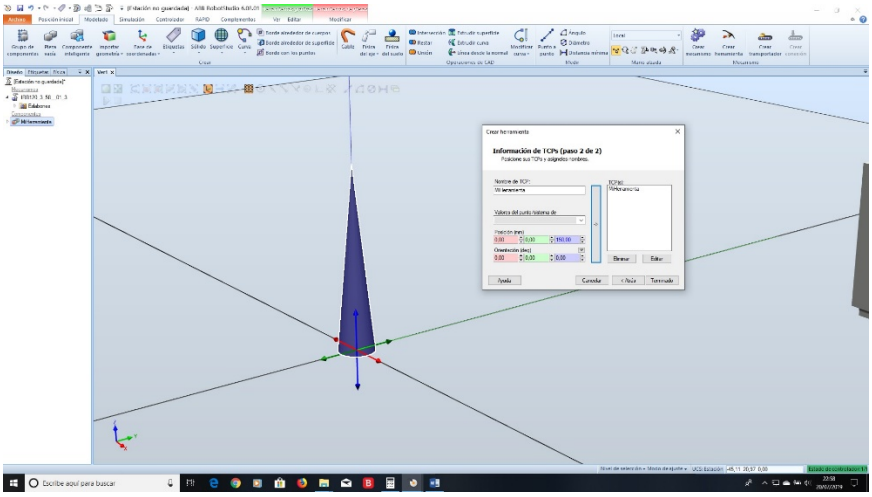


Figura 38: Creación de una herramienta III

Una vez acabado el proceso de creación de la herramienta podemos observar esta y su TCP.

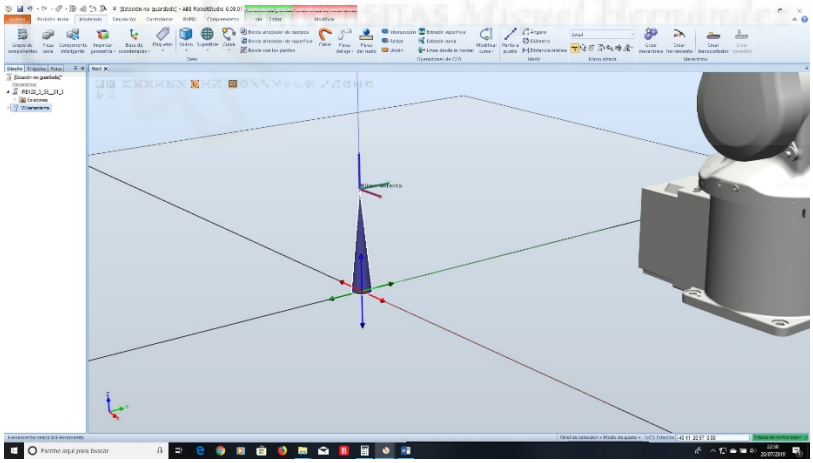


Figura 39: Creación de una herramienta IV.

- Para añadir la herramienta al TCP del robot clicamos sobre ella en árbol de navegación de la izquierda y la arrastramos sobre el icono del robot en dicho árbol. Contestamos si a la pregunta de si queremos actualizar su posición. La herramienta se coloca en el TCP del Robot. Otra forma de realizar este proceso es seleccionar la

herramienta en el árbol izquierdo, hacer clic sobre su nombre con el botón derecho del ratón y en la ventana que aparece seleccionar “Conectar a” y seleccionar el robot en la ventana de selección que aparece. Para desconectar la herramienta repetimos este último proceso pero seleccionamos la opción “Desconectar”.

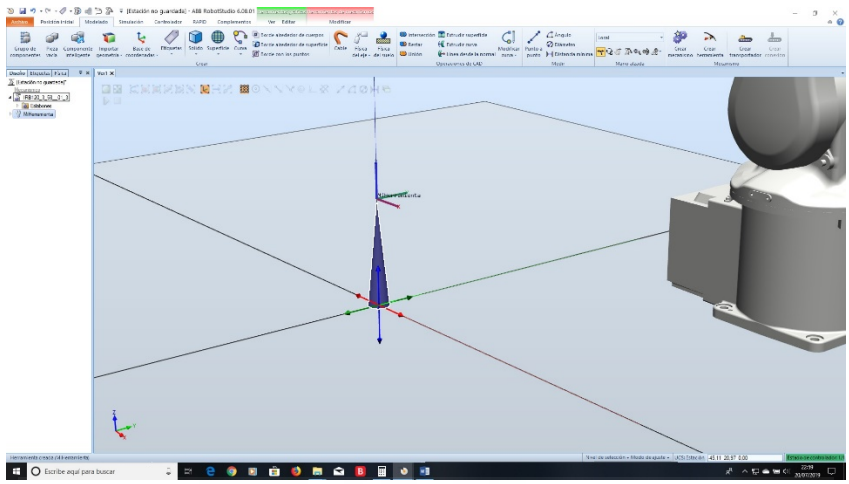


Figura 40: Creación de una herramienta V

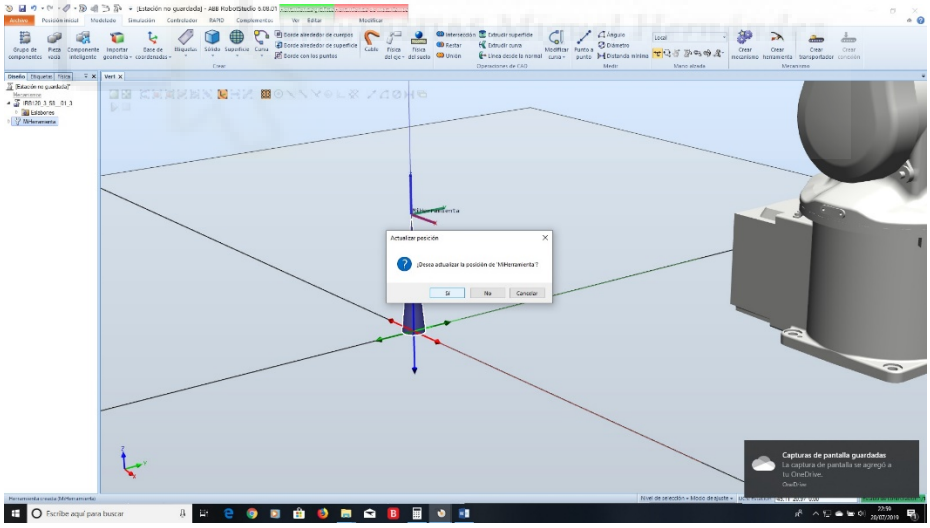


Figura 41: Creación de una herramienta VI.

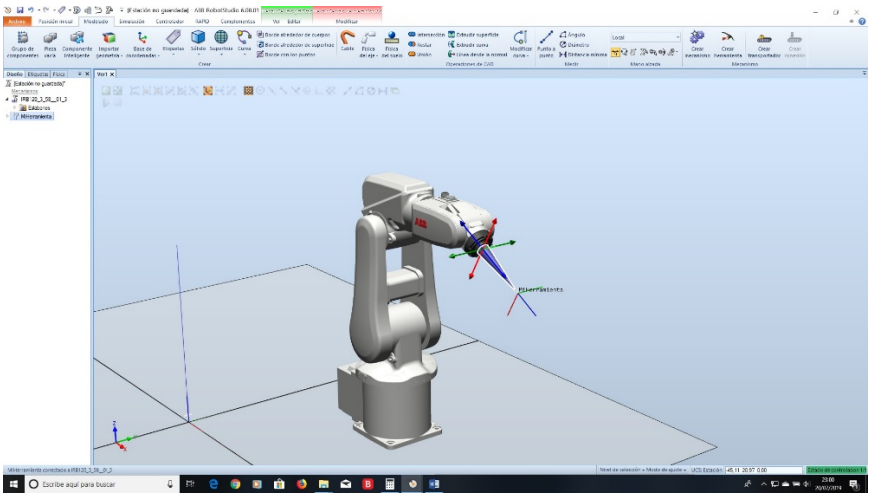


Figura 42: Creación de una herramienta VII.

- Si queremos reutilizar la herramienta creada en otros proyectos, la podemos guardar como biblioteca.

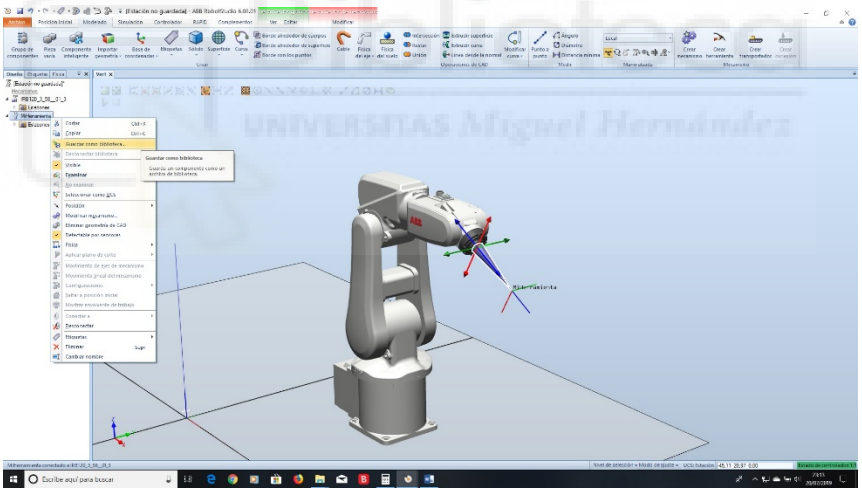


Figura 43: Creación de una herramienta VIII.

4.1.3.6 Creación de Piezas de trabajo.

Como ya se ha comentado, el gran de ventaja de RobotStudio es la facilidad con que podemos crear los puntos y trayectorias objetivo de la herramienta del robot gracias a su entorno 3D. La creación de réplicas exactas de las piezas de trabajo reales nos permite posteriormente situarlas en la estación virtual en el punto exacto en que se van a situar y de esa

forma realizar la programación offline sin tener que interrumpir el trabajo de la estación real. Además, si la pieza se mueve de la posición será muy fácil modificar el programa sin tener que rehacerlo desde cero como veremos en el siguiente apartado.

Para la creación de piezas de trabajo disponemos de dos opciones:

- Usar el módulo **Modelado** de RobotStudio.
- Usar un software externo y luego importar el archivo (ver tipos de archivo admitidos en la ayuda)

En el entorno de modelado de RobotStudio encontramos herramientas de modelado similares a las de otros programas modelado 3D standard en la industria tales como Autocad, SolidWorks, FreeCad, etc.

Como ejemplo vamos a crear una pieza de trabajo utilizando alguna de las herramientas disponibles.

- Creamos una superficie poligonal con la herramienta crear superficie.

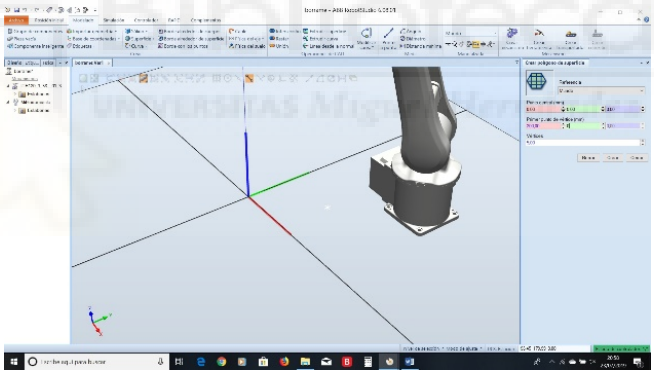


Figura 44: Modelado de una pieza de trabajo I.

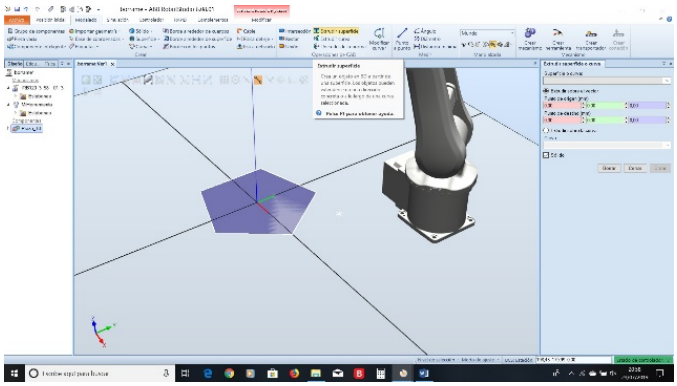


Figura 45: Modelado de una pieza de trabajo II.

- Extruimos la superficie.

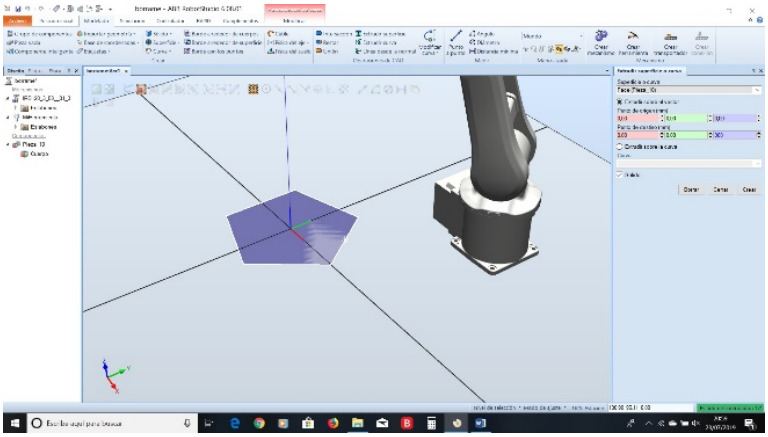


Figura 46: Modelado de una pieza de trabajo III.

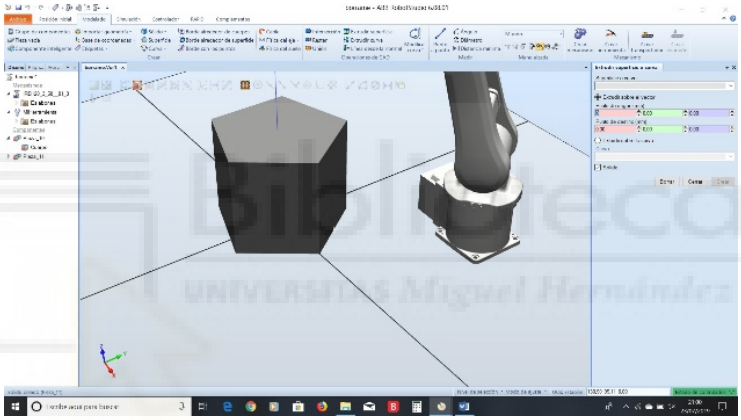


Figura 47: Modelado de una pieza de trabajo IV.

- Creamos un cilindro en el centro del prisma pentagonal.

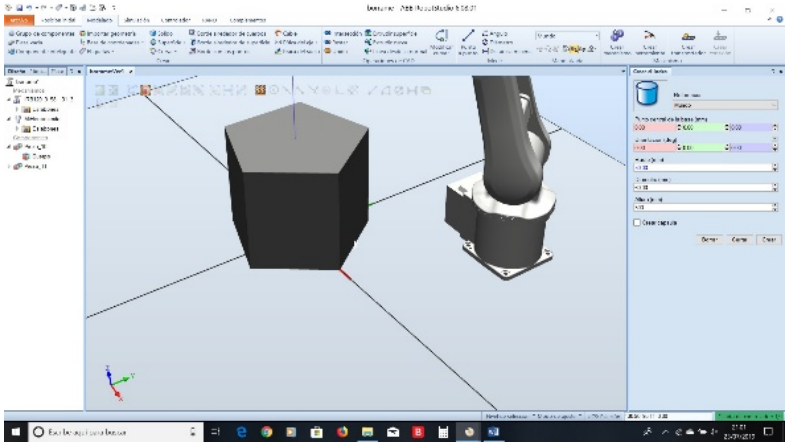


Figura 48: Modelado de una pieza de trabajo V.

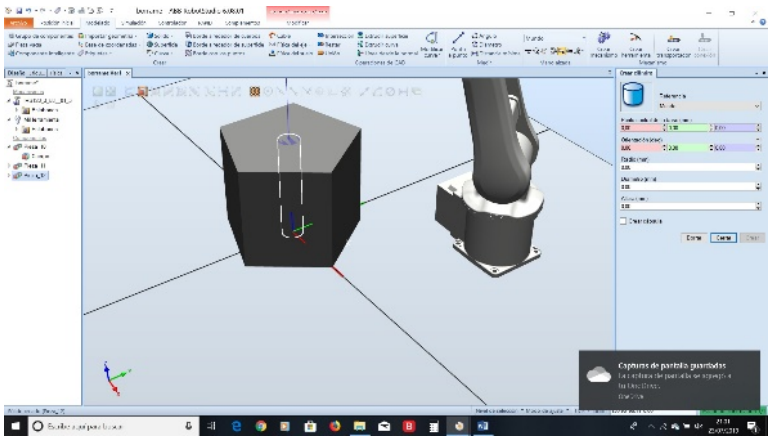


Figura 49: Modelado de una pieza de trabajo VI.

- Restamos el cilindro al prisma (desmarcar conservar original").

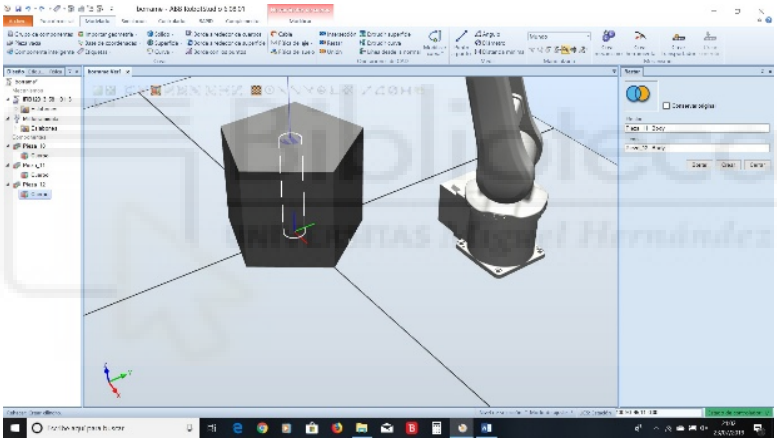


Figura 50: Modelado de una pieza de trabajo VII.

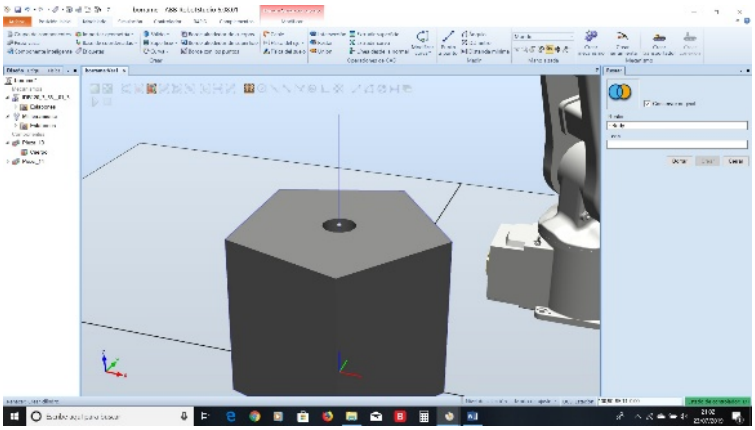


Figura 51:Modelado de una pieza de trabajo VIII.

Por ultimo podemos cambiar el nombre y agrupar todos los objetos/pieza en un “Grupo de componentes” para tratarlo como un solo objeto creando un grupo de componentes y arrastrando los objetos dentro del grupo.

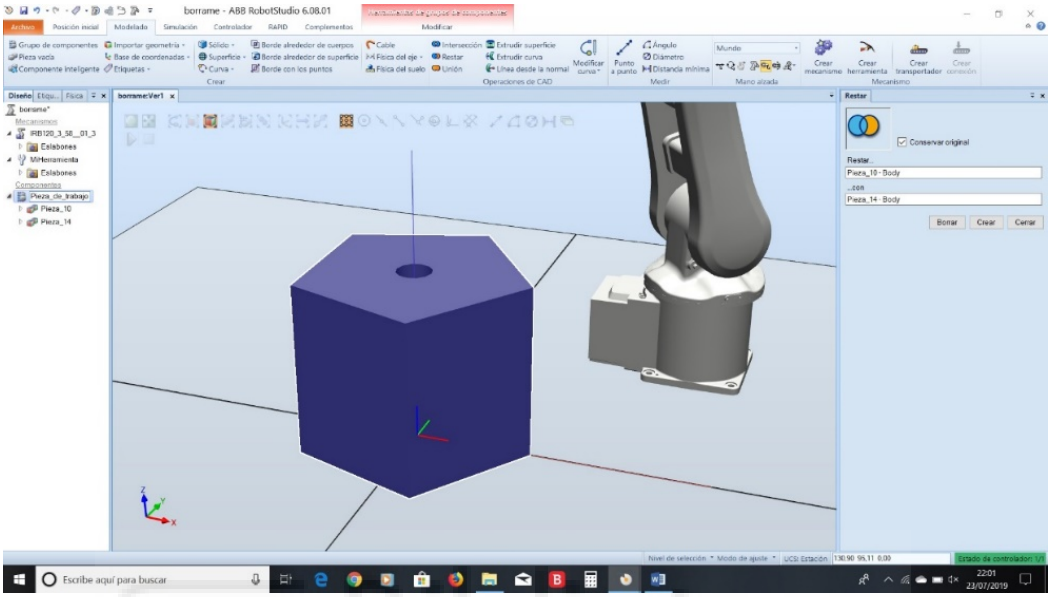


Figura 52: Modelado de una pieza de trabajo IX.

4.1.3.7 Creación de objetos de trabajo y objetivos.

Los objetos de trabajo son los sistemas de coordenadas en que se sitúan los puntos objetivo del robot. Un objeto de trabajo está compuesto por dos sistemas de coordenadas denominados “sistema de coordenadas del usuario” y “sistema de coordenadas del objeto” siendo éste último descendiente del anterior en la jerarquía.

Para crear un punto objetivo en RobotStudio, antes debemos elegir los siguientes parámetros en la cinta superior:

- Tarea: Sistema/Tarea en que estamos creándolo (puede haber más de un sistema/tarea).
- Objeto de trabajo: Si no hay creado ninguno, por defecto, está el **wobj0**. Dicho sistema coincide con el sistema de coordenadas de la tarea del sistema (**TF**).
- Herramienta: Define el **TCP** que queremos situar en el punto objetivo. Si no hay definida ninguna herramienta, el **tool0** corresponde al **TCP** del robot.

Se recomienda crear siempre los objetivos de trabajo en objetos de trabajo del usuario, ya que esto facilita los ajustes de la posición de la pieza en caso de ser necesario sin la necesidad de volver a crear de nuevo los objetivos y trayectorias, de hecho el propio robot estudio nos alerta cuando creamos objetivos de trabajo en el **wobj0**.

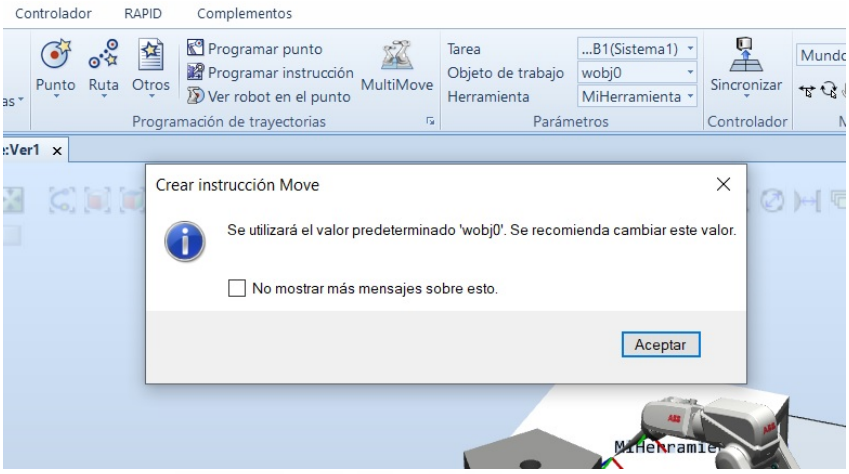


Figura 53: Objetivo en "wobj0".

Para crear un objeto de trabajo:

- Seleccionamos "Crear objeto de trabajo" en el menú desplegable "Otros" en la pestaña "Posición inicial".

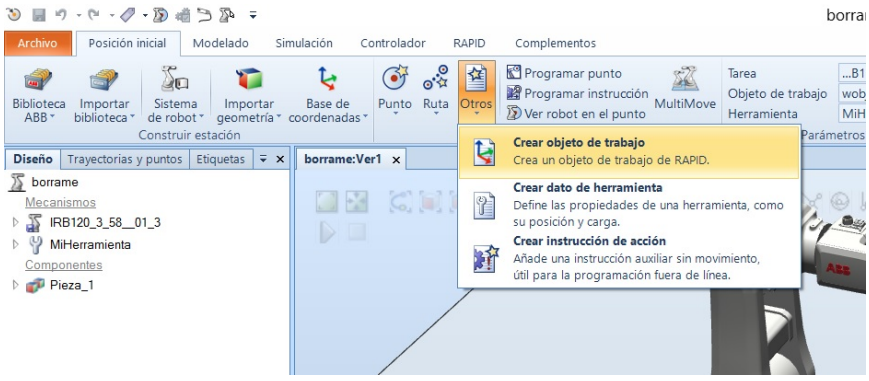


Figura 54: Crear "objeto de trabajo"

- Aparece la ventana de creación de objetos de trabajo, que nos permite especificar la posición y orientación de los sistemas de coordenadas del usuario y del objeto

para que coincidan o que se sitúen en diferentes puntos. También es posible mediante la opción “Sistema de por puntos” situar el origen de ambos sistemas de coordenadas y su orientación empleando “Tres puntos” situados en los ejes X,Y, Z deseados, o mediante “Posición” situando primero el origen del sistema y para su orientación un punto en el eje X deseado y un punto en el plano XY deseado. Si solo especificamos el sistema de coordenadas del usuario y dejamos a cero el del objeto, este último coincidirá con el del usuario ya que depende jerárquicamente de el.

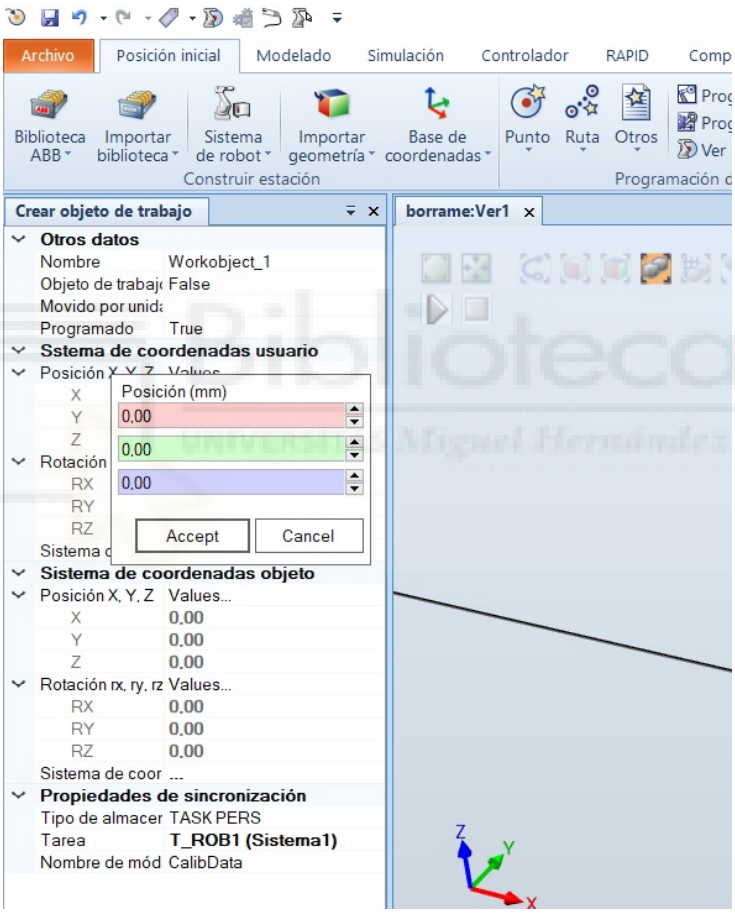


Figura 55: "Objeto de trabajo" mediante posición XYZ.

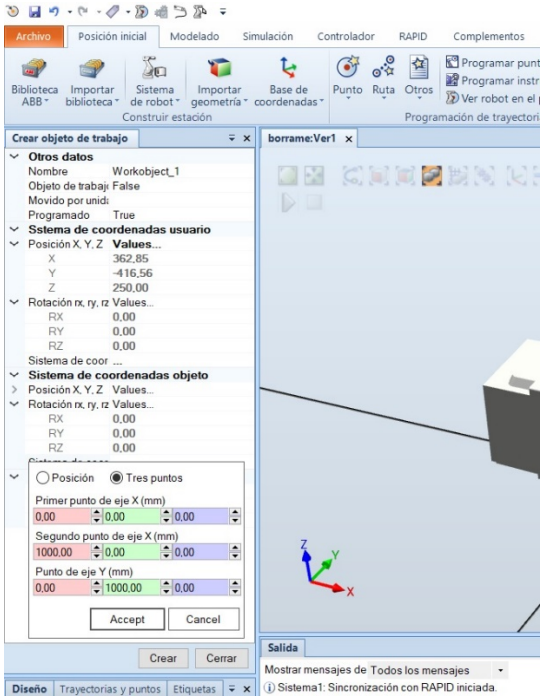


Figura 56: "Objeto de trabajo" mediante tres puntos.

Una vez creado podemos ver que ya aparece en la pestaña “Trayectorias y puntos” Al seleccionar el objeto de trabajo en el árbol de la izquierda comprobamos que se seleccionan el sistema del usuario y el del objeto que corresponden a dicho objeto de trabajo (en este caso se han situado separados como ejemplo).

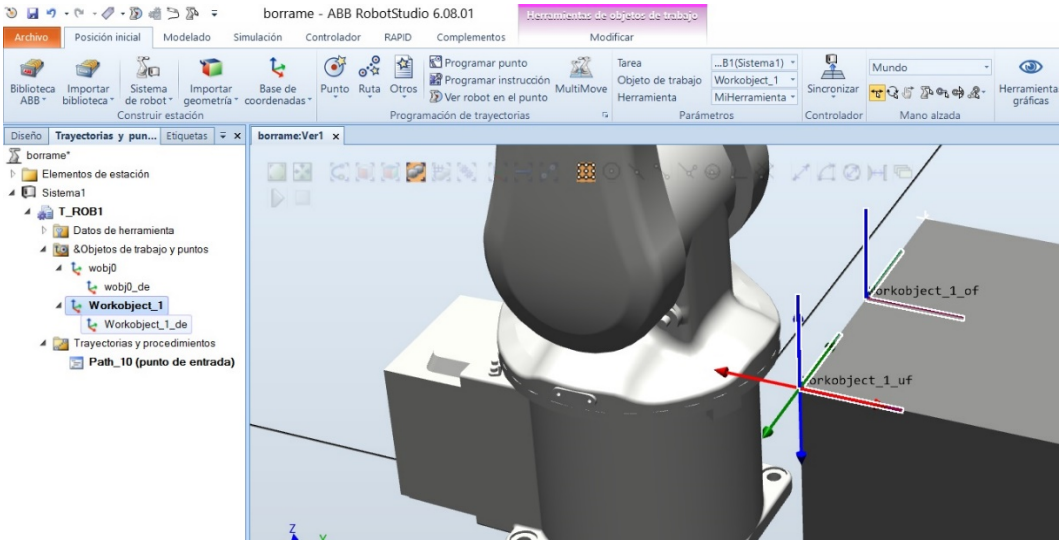


Figura 57: Sistemas del usuario y del objeto.

El objeto de trabajo es posible modificarlo una vez creado seleccionándolo en el árbol de la izquierda con el botón derecho del ratón y elegir la opción “Modificar objeto de trabajo” en el menú que aparece.

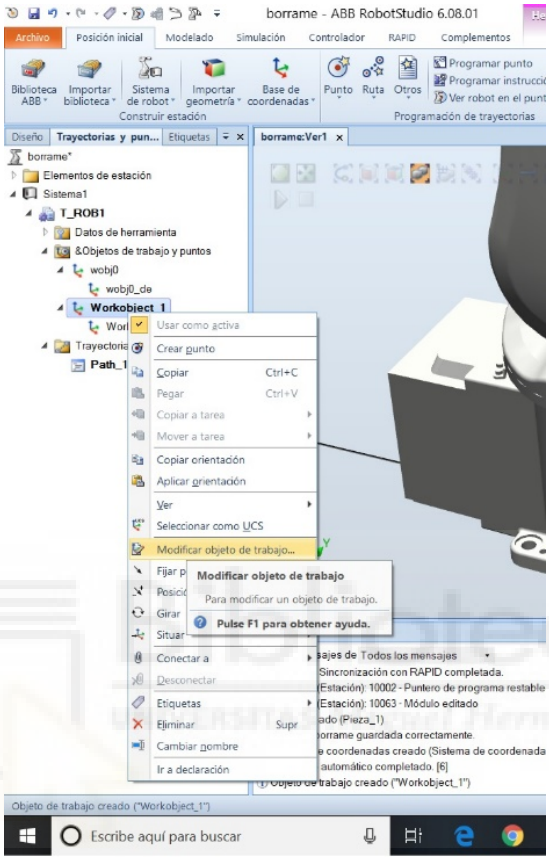


Figura 58: Modificar "objeto de trabajo"

Para crear un objetivo:

- Primero debemos seleccionar los parámetros de sistema/tarea, objeto de trabajo y herramienta en el cual queremos que se sitúe el objetivo, como ya se ha mencionado, en la pestaña “Posición inicial”.
- En la misma pestaña seleccionamos el icono punto, se abre una ventana que nos permite crear puntos objetivo seleccionando la opción “Crear punto” y con la ayuda de las herramientas de selección gráfica para localizar con exactitud extremos de línea, etc.

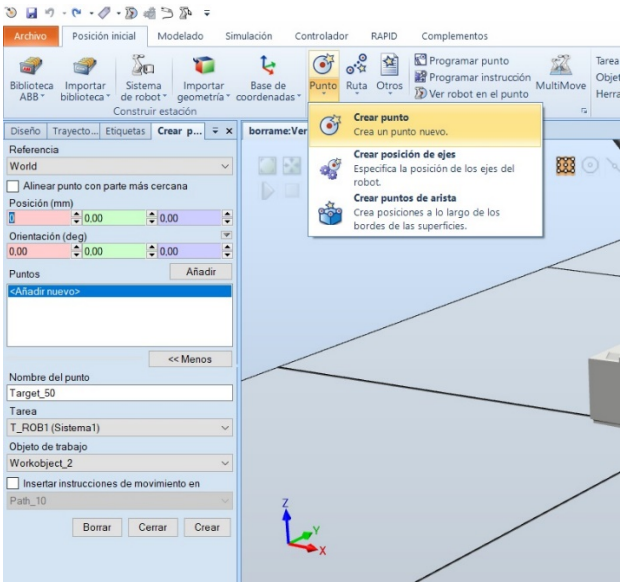


Figura 59: Crear punto.

- Los puntos creados, aparecen en el árbol de la izquierda dentro del workobject en el que se han creado. Es posible copiar y/o trasladar los puntos de un workobject a otro.

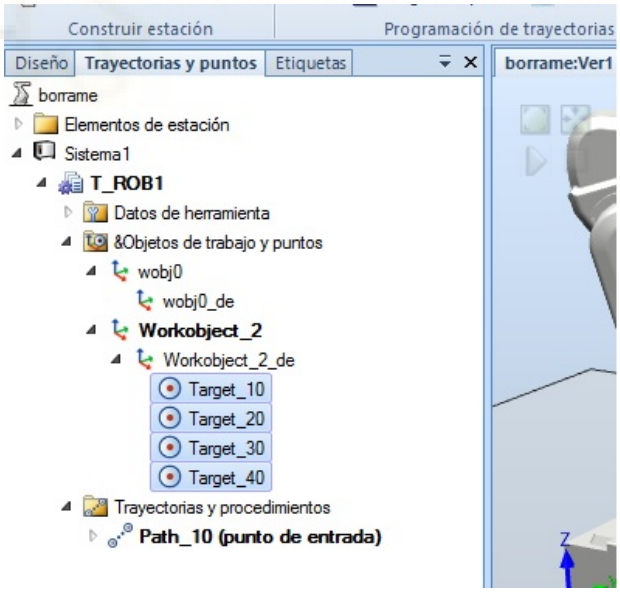


Figura 60: Selección de objetivos.

Un objetivo al sincronizar con **RAPID** tiene el siguiente código:

```
CONST robtarget Target_30:=[[250,250,0],[0.000000001,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

El siguiente paso es reorientar la herramienta para posicionarla de forma que sea alcanzable el objetivo para el robot. RobotStudio usa un sistema de alertas en el que un triángulo amarillo nos avisa de que el objetivo es alcanzable con otra configuración de ejes del robot y con un triángulo rojo que indica que el objetivo no es alcanzable. En el primer caso, podemos cambiar la configuración del robot haciendo clic sobre con el botón izquierdo del ratón sobre el objetivo y seleccionar “Configuraciones en el menú que aparece. En el segundo caso debemos modificar la posición de la pieza de trabajo o del robot o bien usar posicionadores para conseguir la alcanzabilidad. Suponemos que se ha estudiado previamente el espacio de trabajo necesario y seleccionado el robot correctamente para los trabajos a desarrollar en la estación. Que el sistema nos indique que dos puntos son alcanzables, no presupone que la trayectoria entre esos dos pueda realizarse, ya que entre los puntos que crea el motor de trayectorias de la controladora pueden existir configuraciones no alcanzables.

- Para cambiar la orientación de la herramienta seleccionamos el **target** en el árbol de la izquierda clicando con el botón derecho del ratón y en el menú emergente elegimos “Ver herramienta en el punto” para ver la orientación de la herramienta, repetimos la acción seleccionando ahora “Modificar punto” >>”Girar”. Elegimos el eje de giro deseado y vamos aplicando giros hasta situar la herramienta en la orientación adecuada.

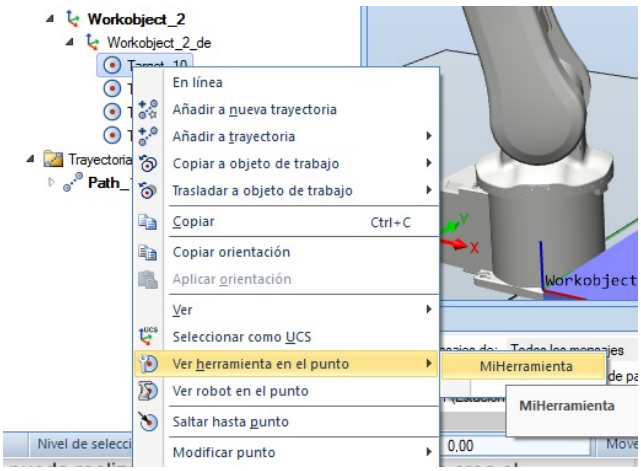


Figura 61: Ver herramienta en el punto.

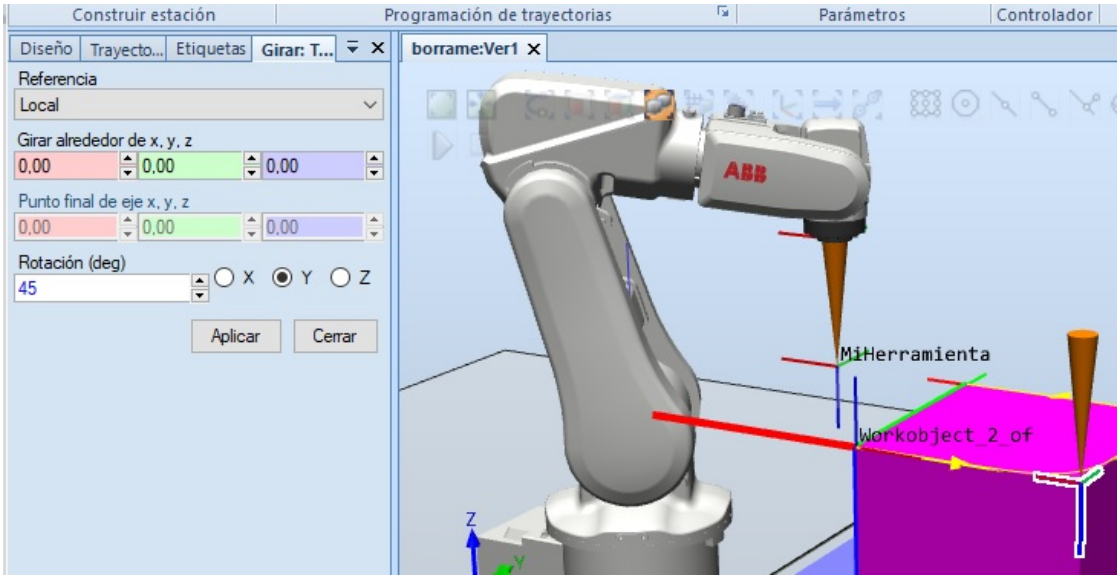


Figura 62: Girar herramienta.

Si queremos que todos los puntos tengan la misma orientación de herramienta que un **target,** en el árbol de la izquierda clicamos con el botón derecho del ratón sobre el y en el menú emergente elegimos “Copiar orientación” para ver copiar la orientación de la herramienta, seleccionamos el resto de puntos donde queremos y repetimos la acción seleccionando ahora “Aplicar orientación”

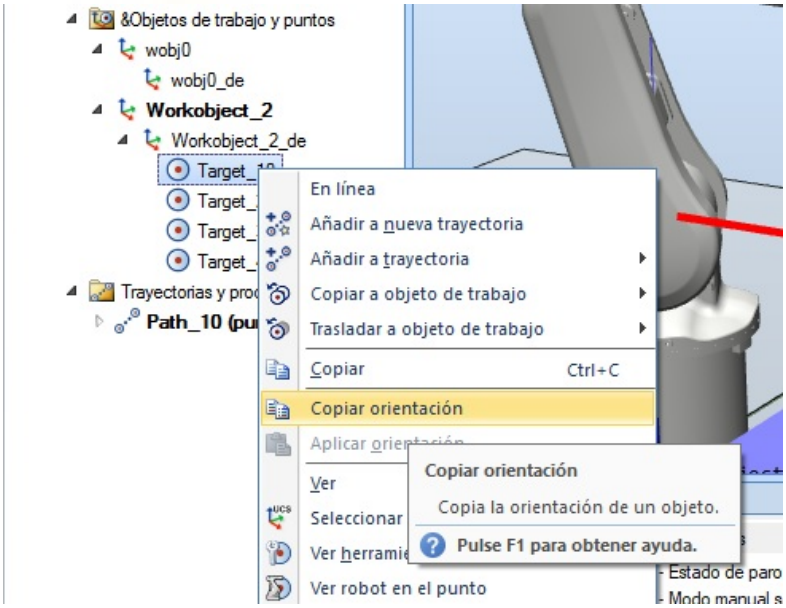


Figura 63: Copiar orientación.

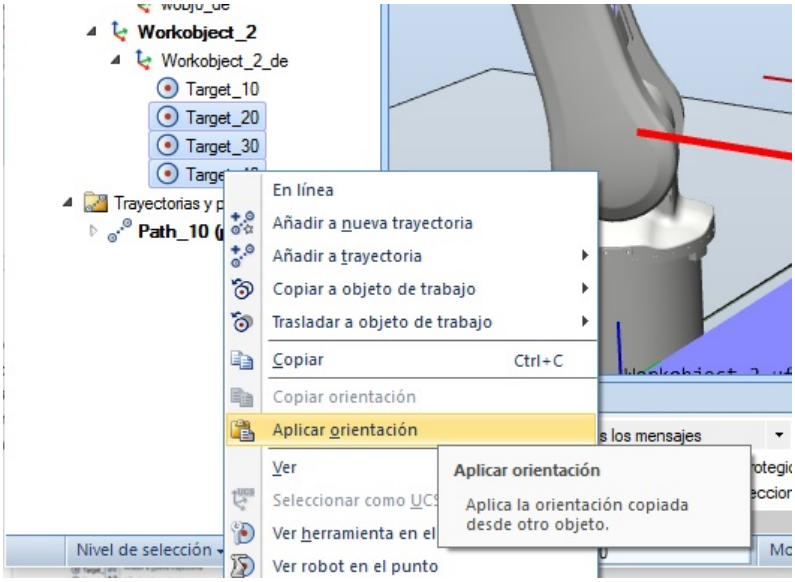


Figura 64: Aplicar orientación.

Existen otras posibilidades para programar puntos **especificando la posición de los ejes del robot** para alcanzar dicha posición.

- En el icono "**Punto**" seleccionamos "**Crear posición de ejes**" en el menú desplegable. Aparece una ventana en la cual podemos especificar los ángulos o desplazamientos en que se deben situar las articulaciones del robot o ejes externos. La posición de ejes creada "**JointTarget**" aparece en el árbol de la izquierda dentro de la carpeta "Posición de ejes".

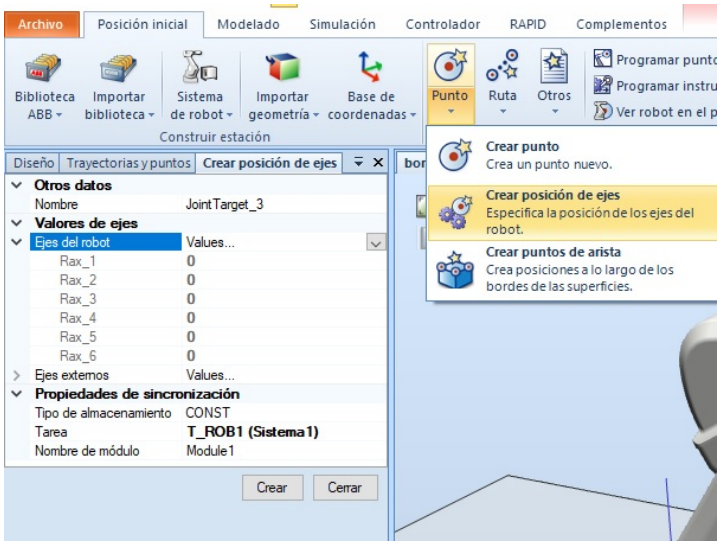


Figura 65: Crear posición de ejes.

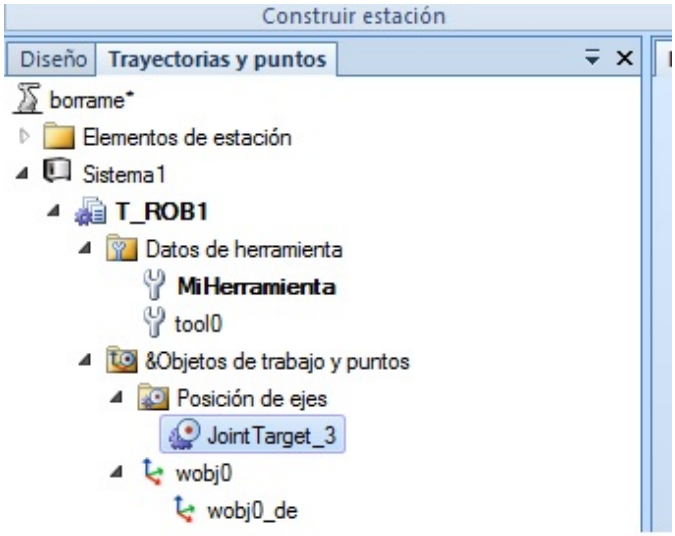


Figura 66: Posición de ejes.

- Podemos programar un punto moviendo las articulaciones del robot manualmente (herramientas del grupo "**Mano alzada**" de la cinta superior) o especificar lo ángulos mediante la herramienta "Movimiento de ejes de mecanismo" y clicar sobre el icono "**Programar punto**" de la cinta superior.

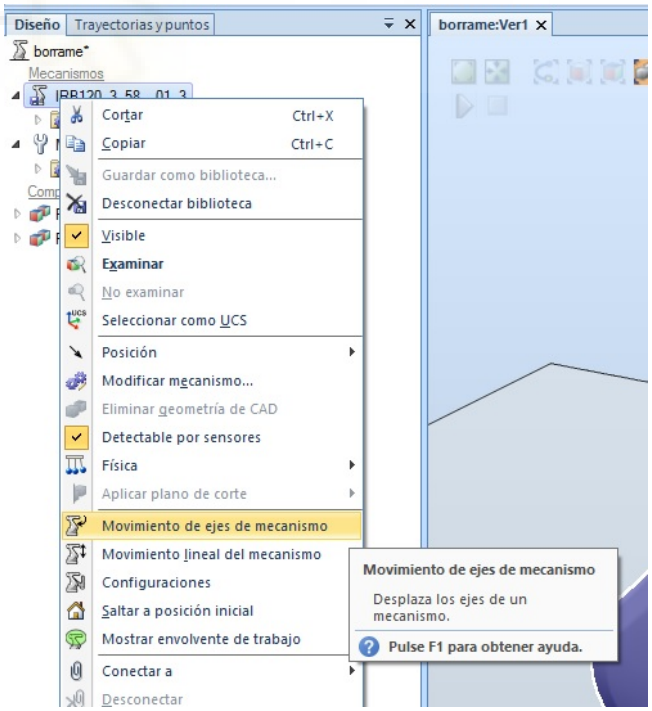


Figura 67: Movimiento de ejes de mecanismo.

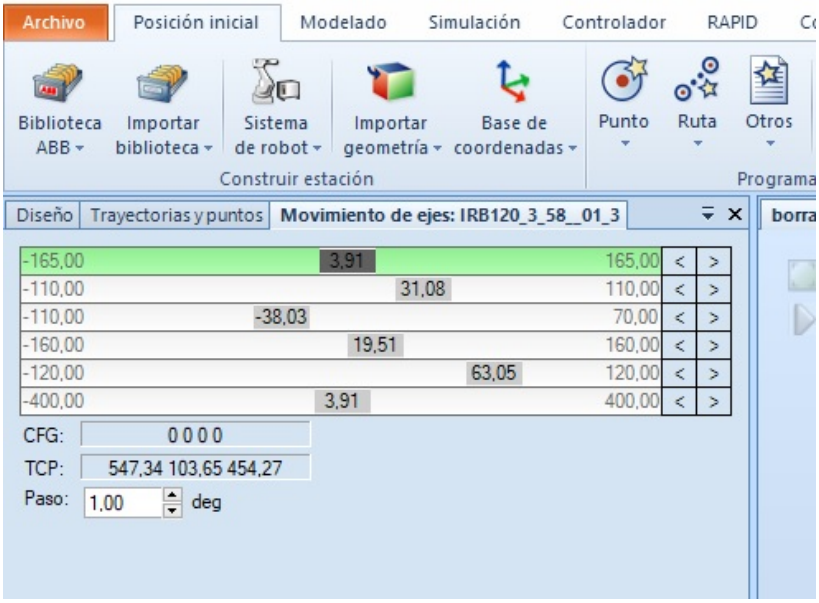


Figura 68: Ventana movimiento de ejes.

- Y por último podemos programar un punto y la instrucción de movimiento al mismo tiempo mediante el icono "**Programar instrucción**" En la barra inferior del programa debemos seleccionar la plantilla de la instrucción de movimiento deseada antes de programarlo (tipo de movimiento, velocidad, precisión, etc.). Una vez creado también es posible posteriormente cambiar todas esas opciones si se necesita.

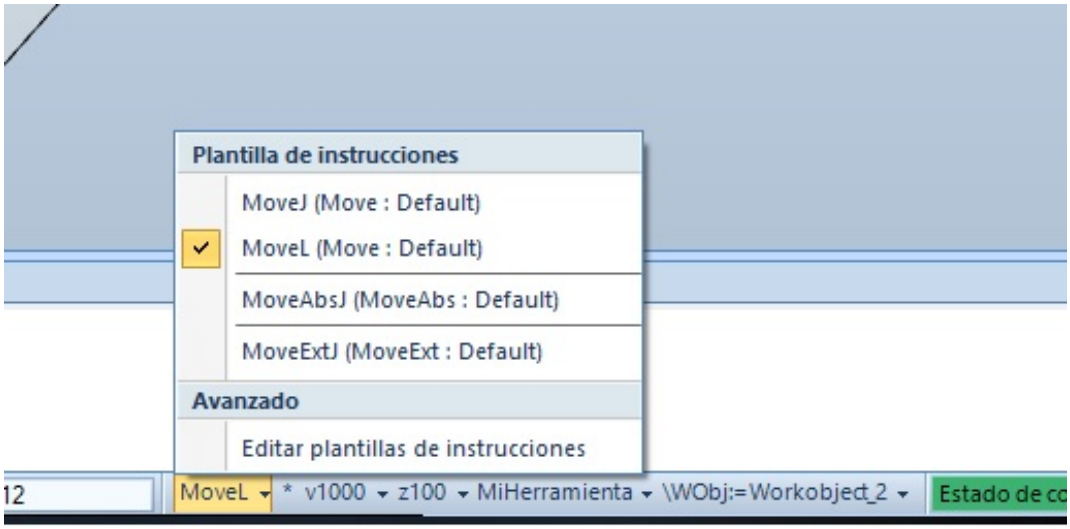


Figura 69: Plantilla de instrucciones de movimiento.

4.1.3.8 Creación de trayectorias.

Las trayectorias son secuencias de objetivos con instrucciones de movimiento que debe seguir el robot. Las trayectorias se pueden crear manualmente o mediante las herramientas de que dispone RobotStudio

- **Trayectoria vacía**: Para crear trayectorias manualmente desde la pestaña “Posición inicial” podemos, en primer lugar, crear una trayectoria vacía desde la cinta superior y clicando en el icono “**Ruta**” se crea una carpeta dentro de la carpeta “**Trayectoria y movimientos**” del árbol de la izquierda denominada “**Path_20**” si es primera que creamos y que RobotStudio va nombrando sucesivamente como “Path_30”, etc. Posteriormente antes de arrastrar los objetivos en el orden que queramos que se realice el movimiento, debemos seleccionar el tipo de movimiento (MoveL, MoveJ,…) la velocidad del movimiento (v1000, v500,….) y la precisión (z100, fine,…) aunque posteriormente es posible cambiar todas estas opciones.

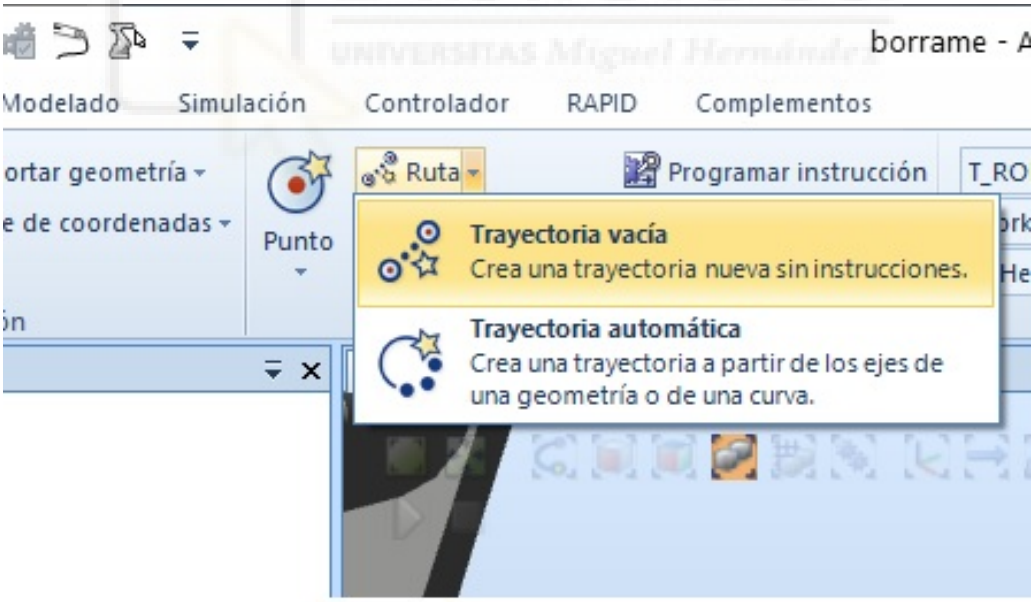


Figura 70: Crear trayectoria vacía.

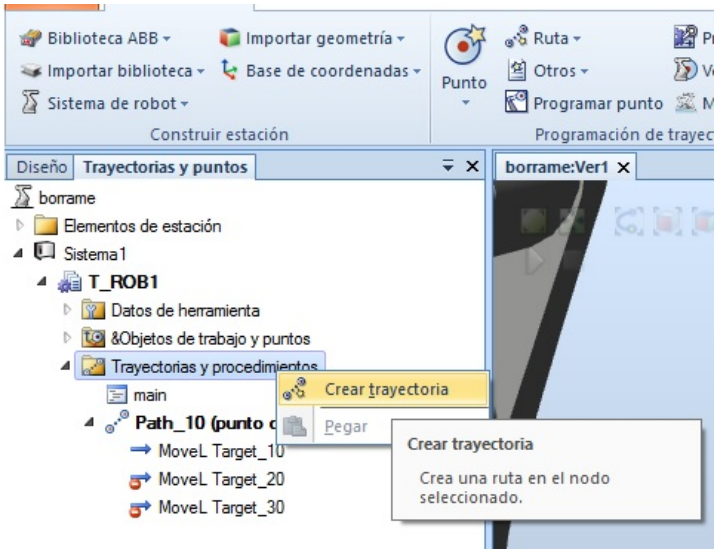


Figura 71: Crear trayectoria.

- **Trayectoria a partir de un borde/curva**: RobotStudio dispone de herramientas para crear trayectorias automáticamente (objetivos e instrucciones de movimiento) siguiendo curvas o contornos de la pieza de trabajo. Desde la pestaña "**Posición inicial**" pulsamos sobre el icono "**Ruta**" y seleccionamos la opción "**Trayectoria automática**". Se abre una ventana para definir que aristas/curvas debe seguir, el tipo de movimiento a realizar y diferentes opciones de exactitud y offset de inicio, fin, aproximación y partida a la trayectoria.

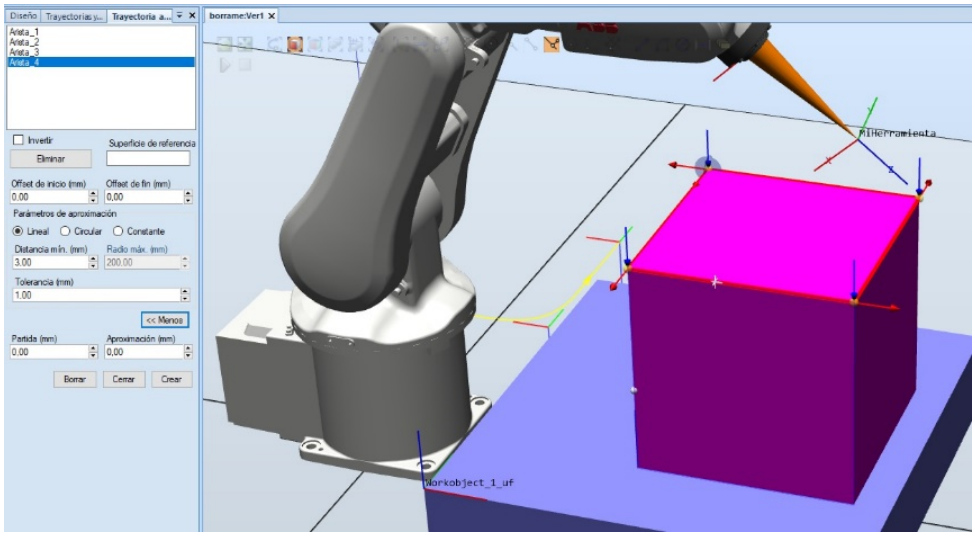


Figura 72: Trayectoria de borde I.

Pulsando en “**Crear**” se crea la trayectoria especificada.

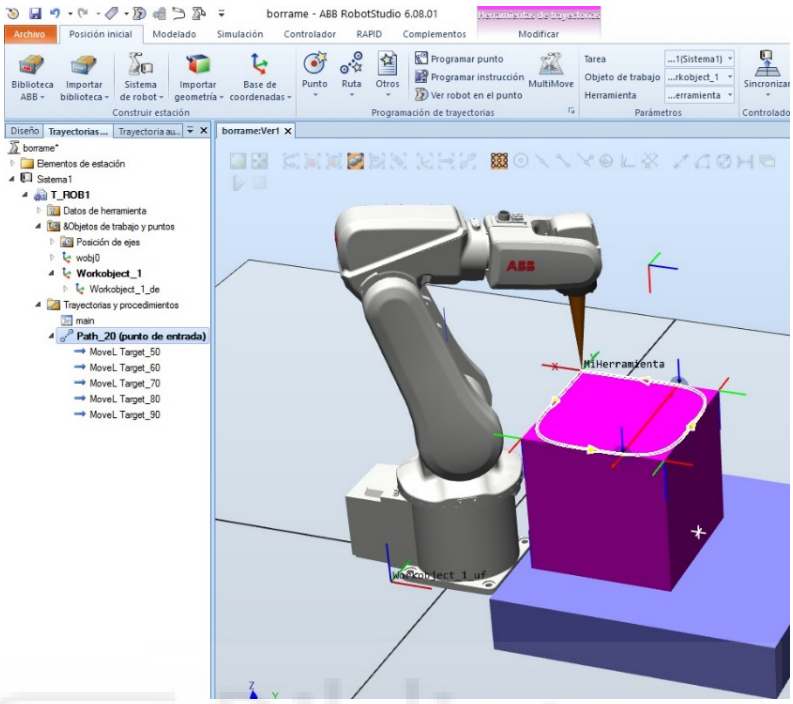


Figura 73: Trayectoria de borde II.

Una vez creados los objetivos y las trayectorias hay que sincronizar con **RAPID** para convertirlo a código ejecutable por la controladora del sistema. Para ello pulsamos sobre el icono “**Sincronizar**” de la cinta superior

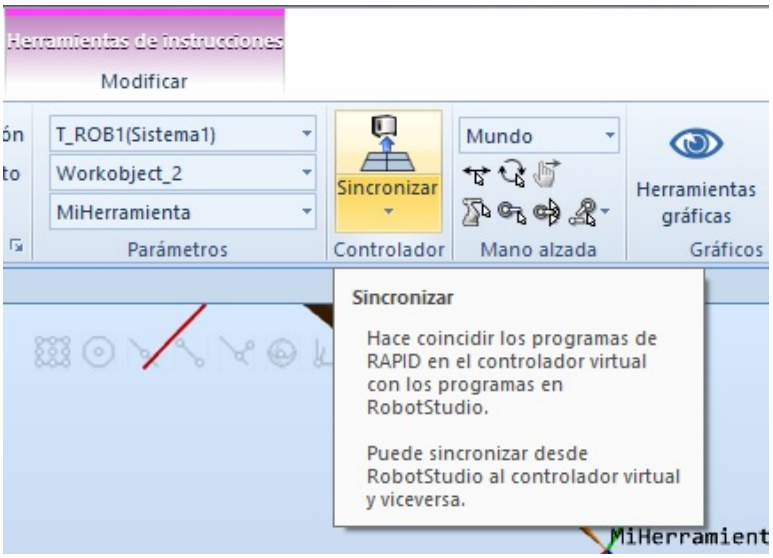


Figura 74: Sincronizar con RAPID.

- **Configuración de ejes para trayectorias**: Ya se ha comentado que es necesario que el robot tenga la herramienta correctamente orientada, y las trayectorias a realizar estén dentro del área de trabajo del robot, además, es posible que al realizar una trayectoria el robot pase cerca o por uno de sus **puntos singulares** y no pueda realizar el movimiento, con lo que hay que ir solucionando todas estas alertas que nos ofrece el programa antes de dar por bueno el código y poder simularlo. Si un objetivo no es alcanzable en la configuración actual pero si en otra configuración RobotStudio muestra un icono de advertencia amarillo, en ese caso, hacemos clic con el botón derecho del ratón sobre el path y seleccionamos "**Configuración automática**" <<"**Todas las instrucciones de movimiento**". En el caso de que no sea alcanzable en ninguna configuración muestra un icono rojo y en ese caso debemos optar por cambiar la orientación de la herramienta, posición de la pieza o del robot para lograr la alcanzabilidad.

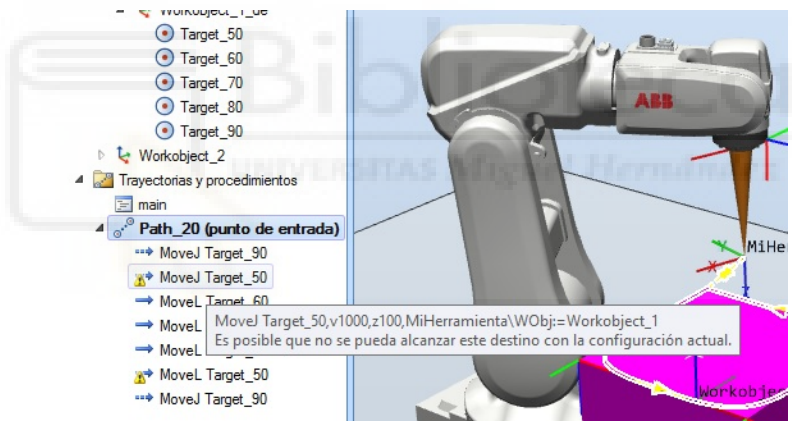


Figura 75: Instrucciones de movimiento posibles en otra configuración del robot.

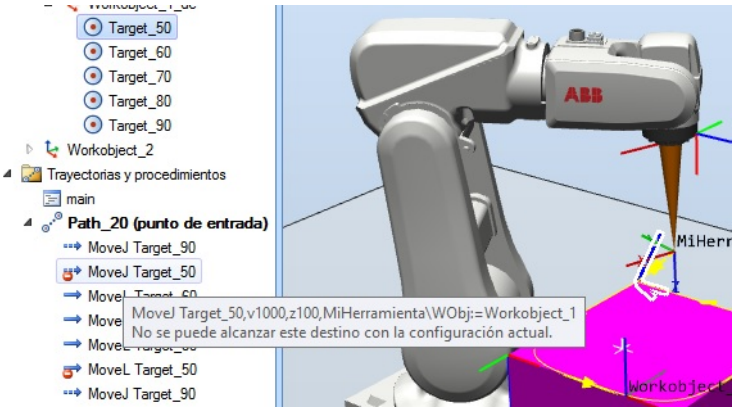


Figura 76:Instrucciones de movimiento no posibles en ninguna configuración del robot.

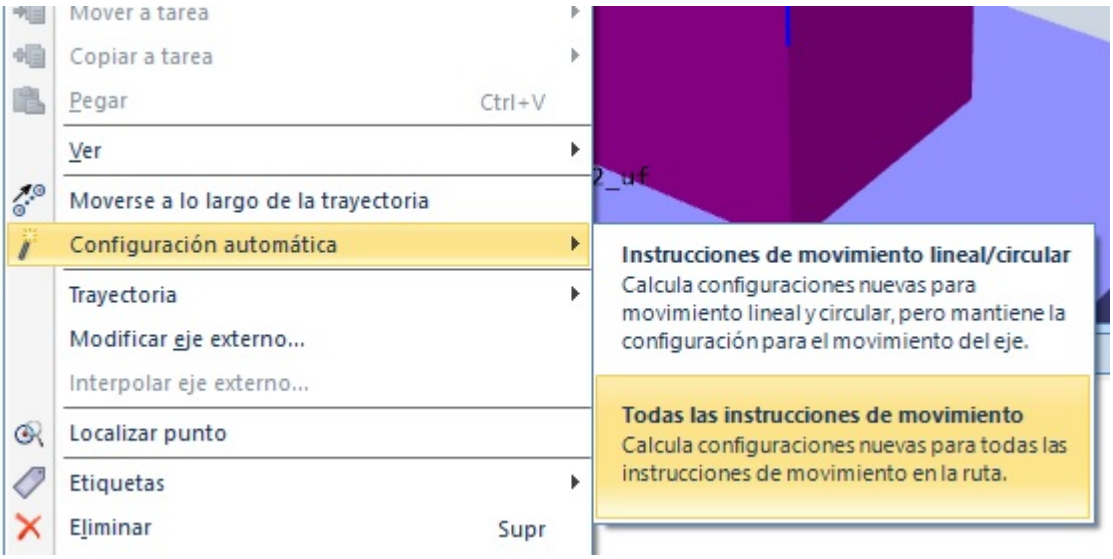


Figura 77: Configuración automática.

RobotStudio dispone, además, de las siguientes herramientas para el trabajo con trayectorias, comentando las usadas con mayor frecuencia:

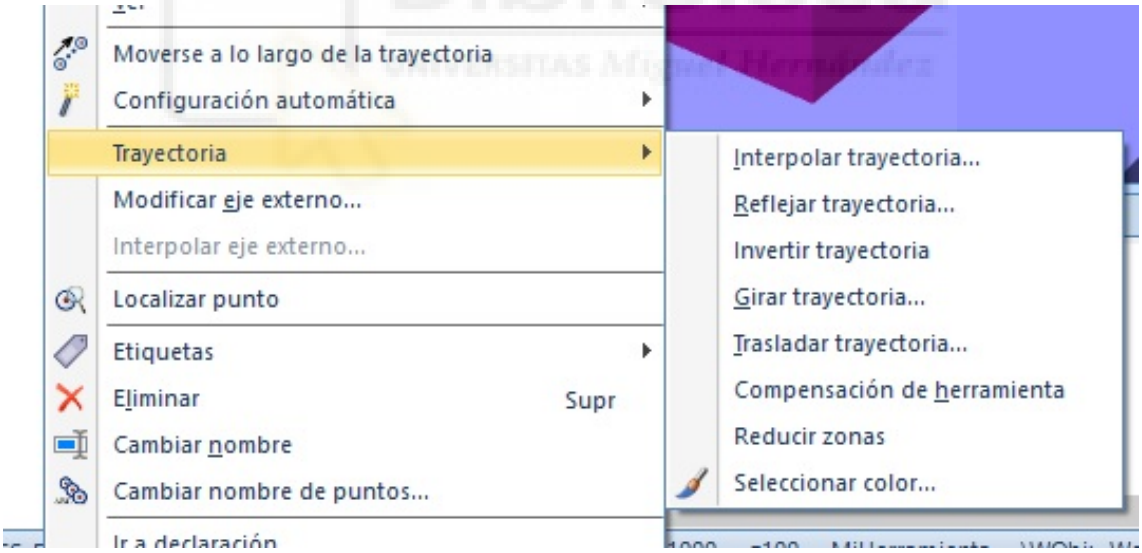


Figura 78: Opciones de trayectoria.

- **Interpolar trayectorias**: En el caso de que la orientación de la herramienta sea distinta en los dos puntos objetivo inicial y final de una trayectoria, podemos mediante esta herramienta reorientar los objetivos de la trayectoria de forma uniforme mediante una interpolación lineal o absoluta. En la figura izquierda podemos ver

una trayectoria sin interpolación donde el último objetivo se orienta de forma distinta a los demás. En la figura central una interpolación lineal donde los objetivos intermedios de la trayectoria se orientan según su posición respecto de los objetivos inicial y final. Si movemos un objetivo y ejecutamos de nuevo la interpolación lineal, la orientación cambiará. Si se insertan objetivos entre los existentes y se ejecuta de nuevo la interpolación lineal la orientación de los objetivos existentes no se ve afectada. Finalmente en la figura de la derecha tenemos una interpolación absoluta de la orientación de la herramienta en la que los objetivos se orientan según su secuencia en la trayectoria. Cada objetivo se reorienta por igual independientemente de su ubicación. Si movemos un objetivo y ejecutamos de nuevo la interpolación absoluta, la orientación no cambiará. Si se insertan objetivos entre los existentes y se ejecuta de nuevo la interpolación absoluta la orientación de los objetivos existentes cambiará.

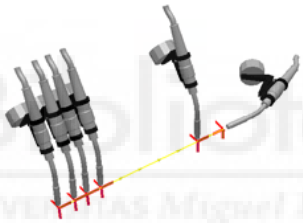


Figura 79: Sin interpolación.

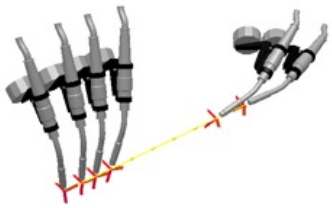


Figura 80: Interpolación lineal.

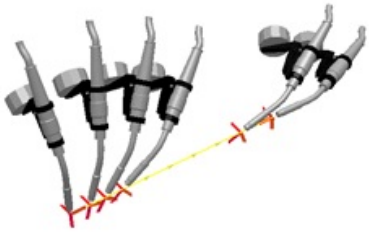


Figura 81: Interpolación absoluta.

- **Reflejar trayectorias:** mediante esta opción todas las instrucciones de movimiento y sus objetivos de crean en una nueva trayectoria especular de la anterior.

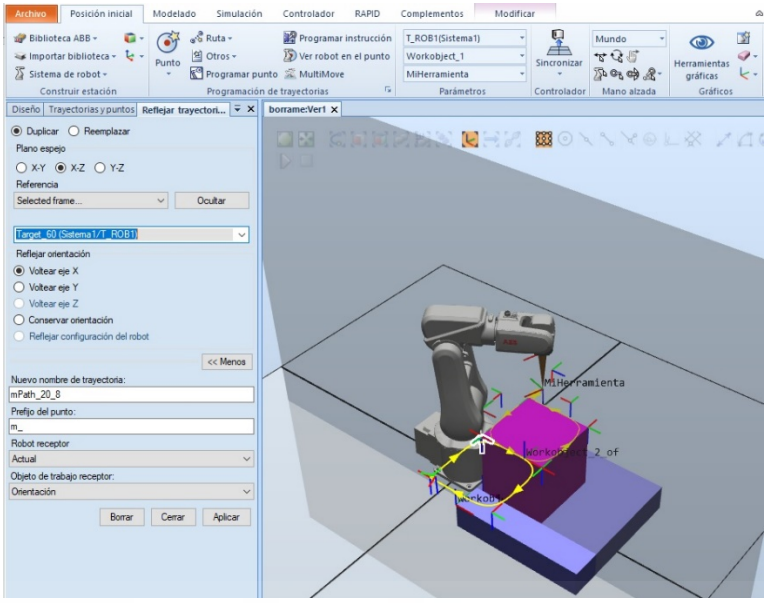


Figura 82: Reflejar trayectoria.

- **Invertir la trayectoria:** permite cambiar el orden de los objetivos de tal manera que el robot avanzará desde el último objetivo al primero.

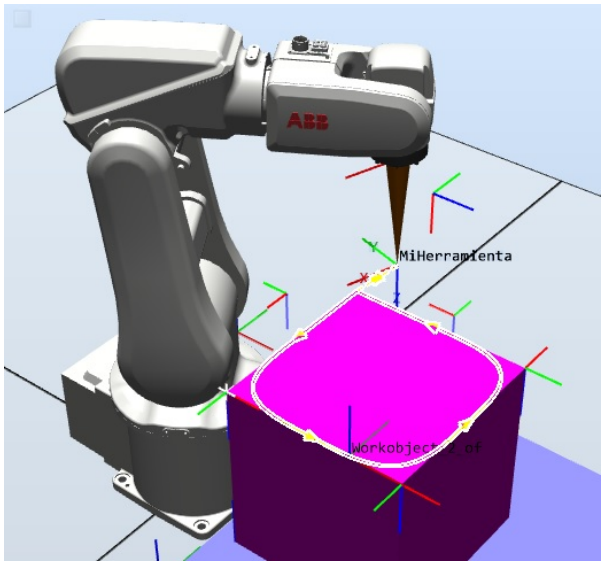


Figura 83: Invertir trayectoria I.

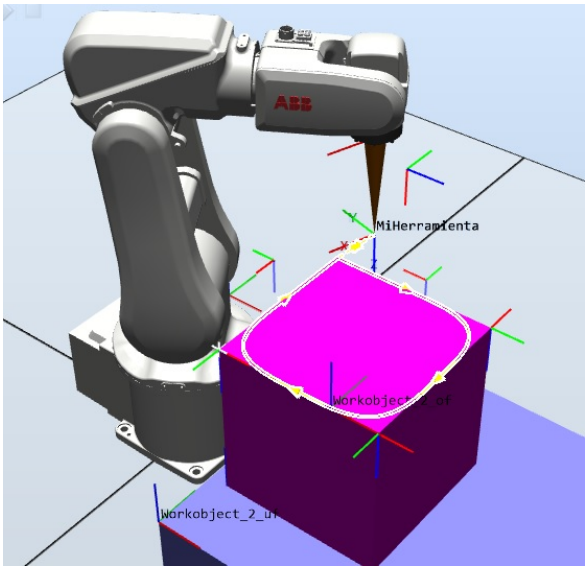


Figura 84: Invertir trayectoria II.

- **Girar trayectoria:** permite rotar una trayectoria completa un ángulo respecto de un eje y en un sistema de referencia elegido. Tras el giro se pierden la configuración de ejes si se hubiesen asignado.

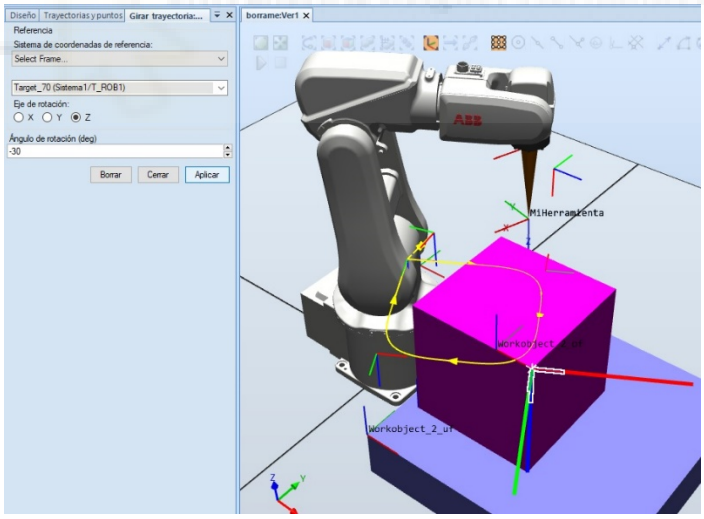


Figura 85: Girar trayectoria.

- **Trasladar trayectoria:** podemos trasladar una trayectoria y todos sus objetivos incluidos. Permite varias opciones, entre ellas trasladar desde un punto de inicio a un punto final seleccionados.

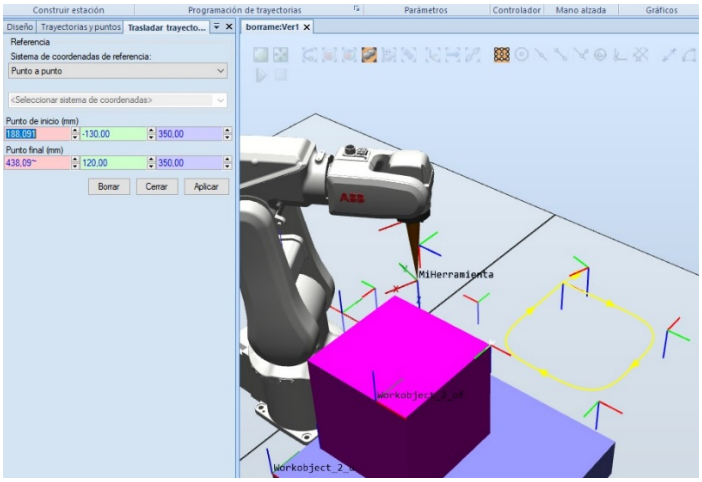


Figura 86: Trasladar trayectoria.

- **Selección de color:** podemos cambiar el color con que se representa la trayectoria seleccionada en RobotStudio.

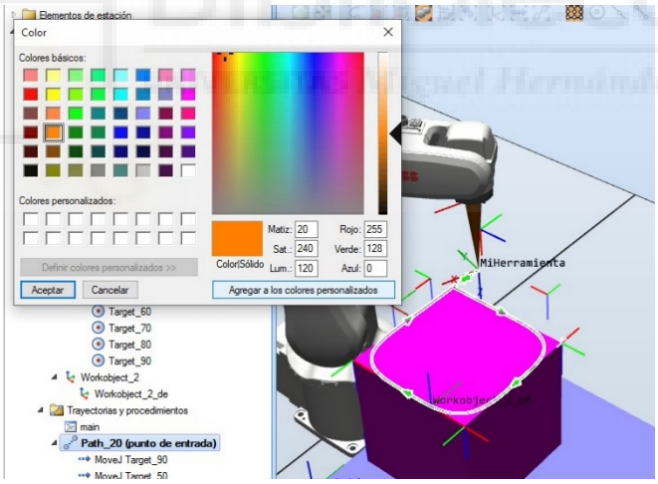


Figura 87: Selección de color.

4.1.3.9 Tracks y posicionadores.

Para aumentar el espacio de trabajo y/o la alcanzabilidad del robot podemos incluir en nuestra estación un track para mover el robot o un posicionador para mover la pieza de trabajo o ambos si es necesario. La controladora del robot tratará dichos elementos como ejes

externos y controlará su movimiento como si se tratara de un eje más del robot. Será posible almacenar en los puntos objetivo no solo la pose del robot, sino también la de dichos ejes externos.

RobotStudio dispone en la "**Biblioteca ABB**" de diferentes modelos de track y posicionadores, algunos con diferentes configuraciones de cargas y/o alcance.

Para facilitar la configuración del sistema de forma que RobotStudio añada automáticamente las opciones necesarias, podemos seguir el siguiente flujo de trabajo:

- Iniciaremos una nueva estación con opción "**Estación vacía**".
- Importaremos a la estación el robot, el posicionador y el track desde "**Biblioteca ABB**" en la pestaña "**Posición inicial**" de la cinta superior.
- Usando las herramientas de "**Posición**" colocaremos el robot, posicionador y track según el diseño que tengamos previsto.
- Para colocar el robot sobre el track, en el árbol de la izquierda, arrastramos el robot sobre el track. A las preguntas de si debemos actualizar la posición del robot y de si debe estar coordinado el robot con el track respondemos "Si".

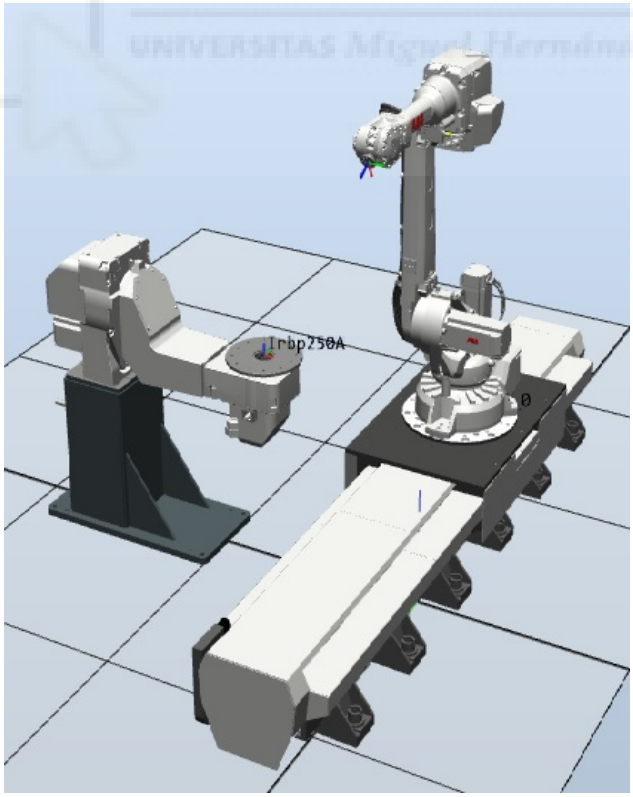


Figura 88: Robot sobre track.

- Crearemos el sistema desde “**Sistema de robot**”<<”**Desde diseño**”. Al tener seleccionados los mecanismos robot, track y posicionador, RobotStudio nos añadirá automáticamente las opciones necesarias para su correcta configuración.

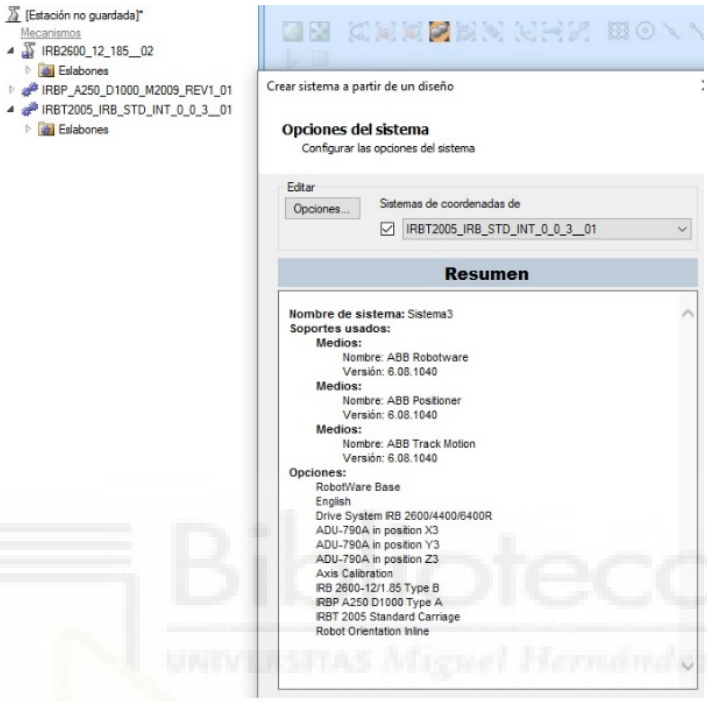


Figura 89: Creación de un sistema con track.

- Para colocar la herramienta en el robot, importarla a la estación o crearla y en el árbol de la izquierda arrastrar sobre el robot.
- Para colocar la pieza de trabajo en el posicionador, importarla a la estación o crearla y en el árbol de la izquierda arrastrar sobre el posicionador. Debe tenerse en cuenta que el posicionador cuenta con un sistema de coordenadas donde se situará y alineará el sistema de coordenadas de la pieza de trabajo. Es posible modificar la posición del sistema de coordenadas de la pieza de trabajo para colocarlo donde y con la orientación que se necesite.

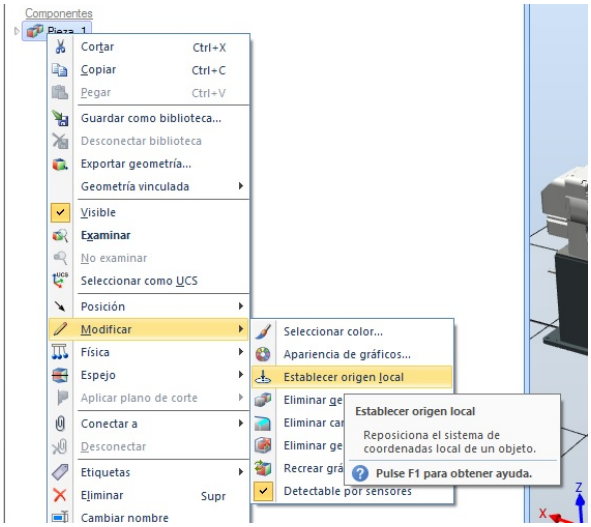


Figura 90: Establecer origen local.

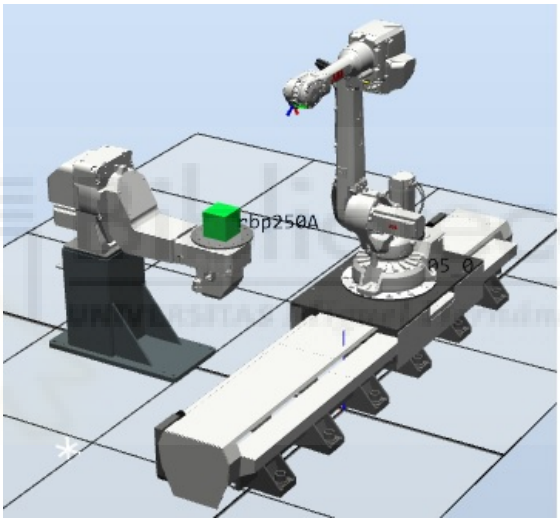


Figura 91: Pieza de trabajo sobre posicionador.

- Para programar los puntos objetivos y trayectorias, debemos colocar en primer lugar en la posición requerida el track y el posicionador. Es necesario activar antes el mecanismo posicionador en RobotStudio para que almacene su posición actual en el objetivo, para ello desde la pestaña "**Simulación**" en el grupo "**Configurar**" seleccionamos el icono "**Activar unidades mecánicas**" y en la ventana que nos aparece activamos el posicionador con el que vamos a trabajar, en este caso "**STN1**". Como vemos, el track está seleccionado por defecto.

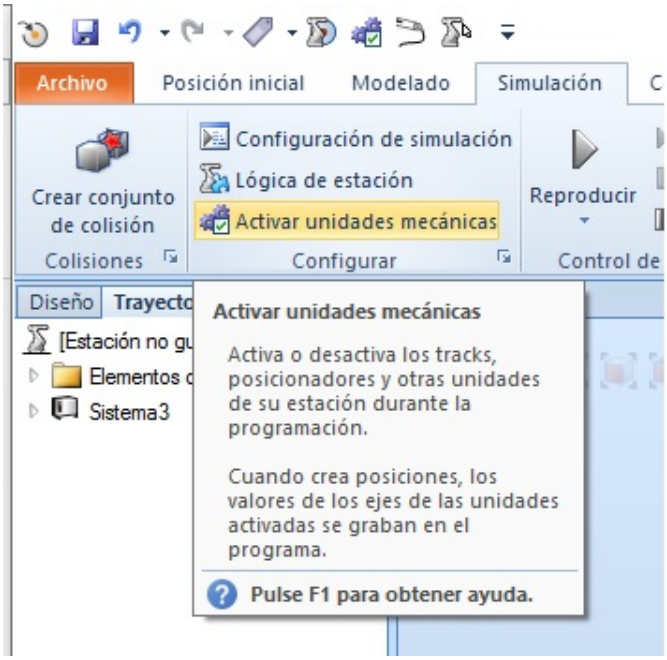


Figura 92: Activar unidades mecánicas I.

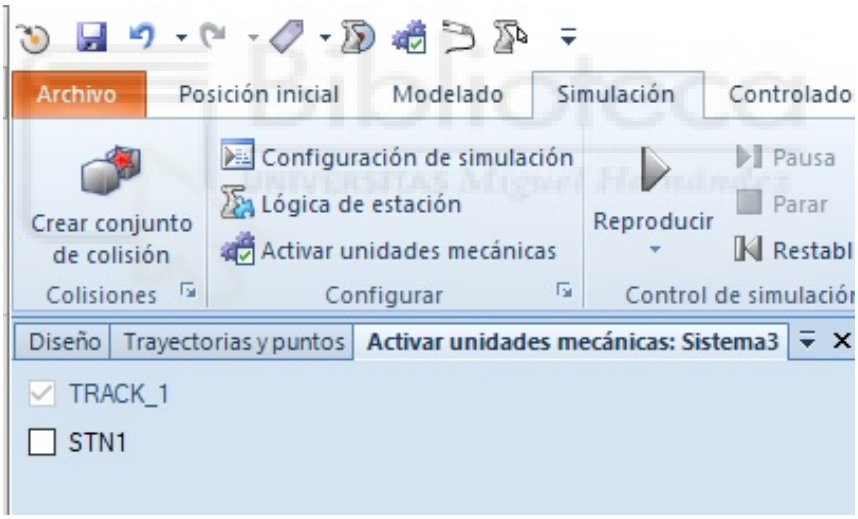


Figura 93: Activar unidades mecánicas II.

- Una vez activado el posicionador y el track podemos colocarlos en la posición requerida mediante la herramienta "**Movimiento de ejes**" a la que podemos acceder haciendo clic con el botón derecho de ratón sobre el robot, track o posicionador. Si la abrimos desde el robot en la persiana de selección podremos elegir el eje externo a mover.

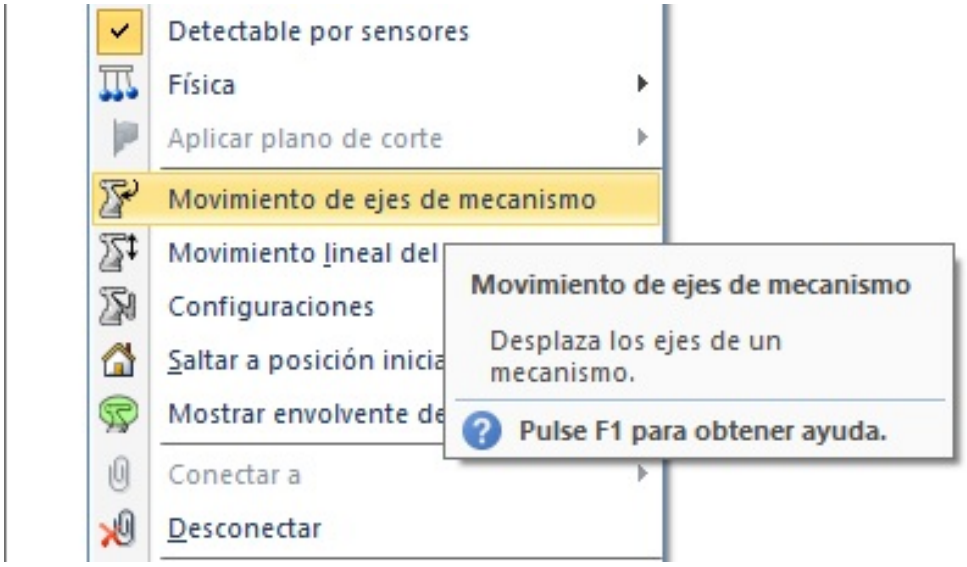


Figura 94: Movimiento de ejes del mecanismo I.

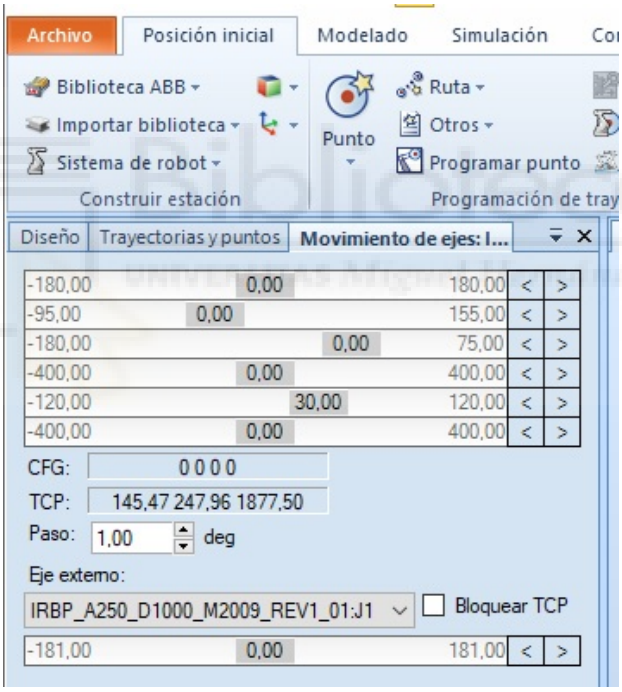


Figura 95: Movimiento de ejes del mecanismo II.

- Una vez colocado el track y el posicionador en la posición deseada, los puntos objetivo que se programen almacenarán la configuración del eje externo del posicionador. Es importante crear un **workobject** y conectarlo a la pieza de trabajo para crear los objetivos y trayectorias en él, de

forma que si movemos el posicionador los objetivos y trayectorias se mueven con la pieza de trabajo.

- Para cambiar la posición del posicionados o track almacenada en los objetivos de una trayectoria, en la pestaña “**Trayectorias y puntos**” del árbol de la izquierda hacemos clic con el botón derecho del ratón sobre el path o el punto y en la ventana seleccionamos al opción “**Modificar eje externo**” que nos permite modificar la posición de los ejes externos.

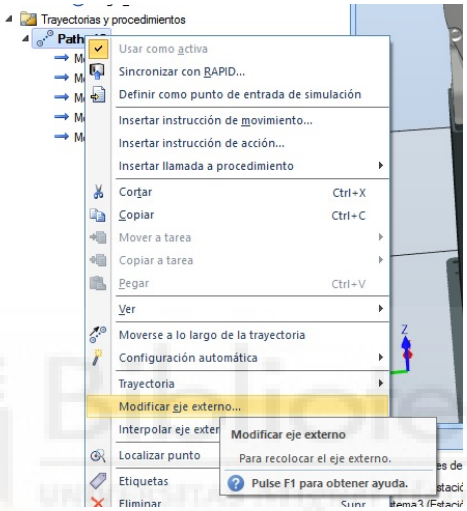


Figura 96: Modificar eje externo I.

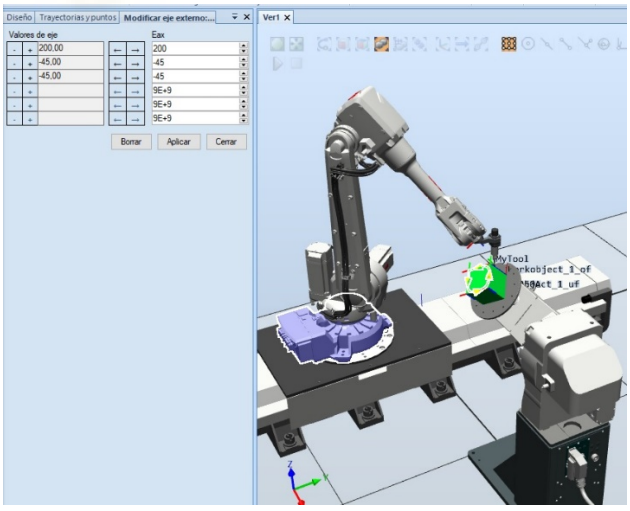


Figura 97: Modificar eje externo II.

Recordar que para que todos los cambios que se vayan efectuando queden reflejados en el controlador hay que pulsar sobre el botón “**Sincronizar con RAPID**”

En RAPID mediante las órdenes “ActUnit STN1;” y “DeactUnit STN1;”, donde “STN1” es el posicionador que queremos controlar, podemos activar o desactivar el posicionador a voluntad.

```
28  PROC main()
29      ActUnit STN1;
30      Path_10;
31      DeactUnit STN1;
32
33  ENDPROC
34  PROC Path_10()
35      MoveL Target_10,v1000,z100,MyTool\WObj:=Workobject_1;
36      MoveL Target_20,v1000,z100,MyTool\WObj:=Workobject_1;
37      MoveL Target_30,v1000,z100,MyTool\WObj:=Workobject_1;
38      MoveL Target_40,v1000,z100,MyTool\WObj:=Workobject_1;
39      MoveL Target_50,v1000,z100,MyTool\WObj:=Workobject_1;
40  ENDPROC
41  ENDMODULE
```

Si incluimos un posicionador en una tarea distinta a la del robot es posible usar la orden **RAPID** “**MoveExtJ**” para mover el eje externo a la posición deseada.

4.1.3.10 Transportadores.

En RobotStudio es posible simular el trabajo de un robot sobre piezas en movimiento sobre un trasportador. El flujo de trabajo para crear una estación con un sistema robot y un transportador es el siguiente:

- Crear una estación vacía.
- Añadir un robot a la estación desde la “Biblioteca ABB”.
- Crear el sistema desde “Sistema robot”<<”Desde diseño” con la opción “606-1 Transportador tracking” y la opción “709-1 DeviceNet Master/Slave”.

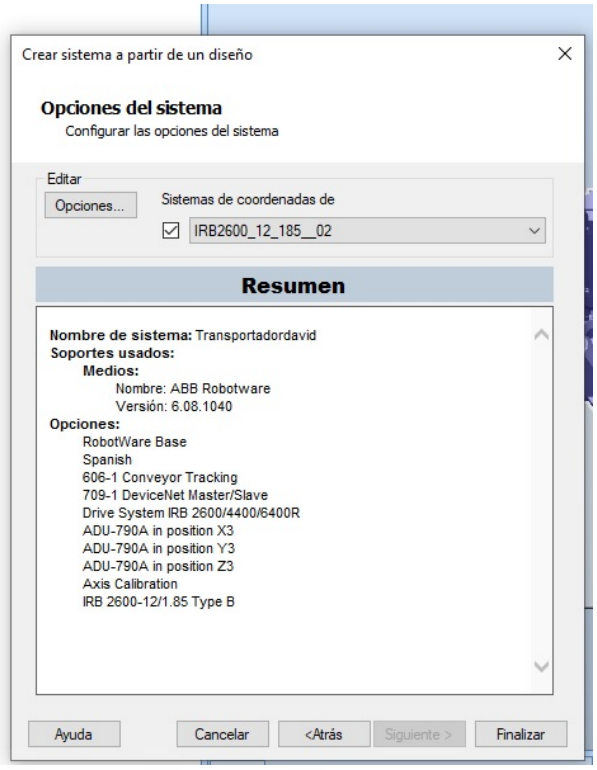


Figura 98: Sistema con transportador.

- Crear un transportador desde modelado:
    - Crear el objeto transportador mediante un prisma rectangular de 5000 mm de largo, 500 mm de ancho y 100 mm de alto.

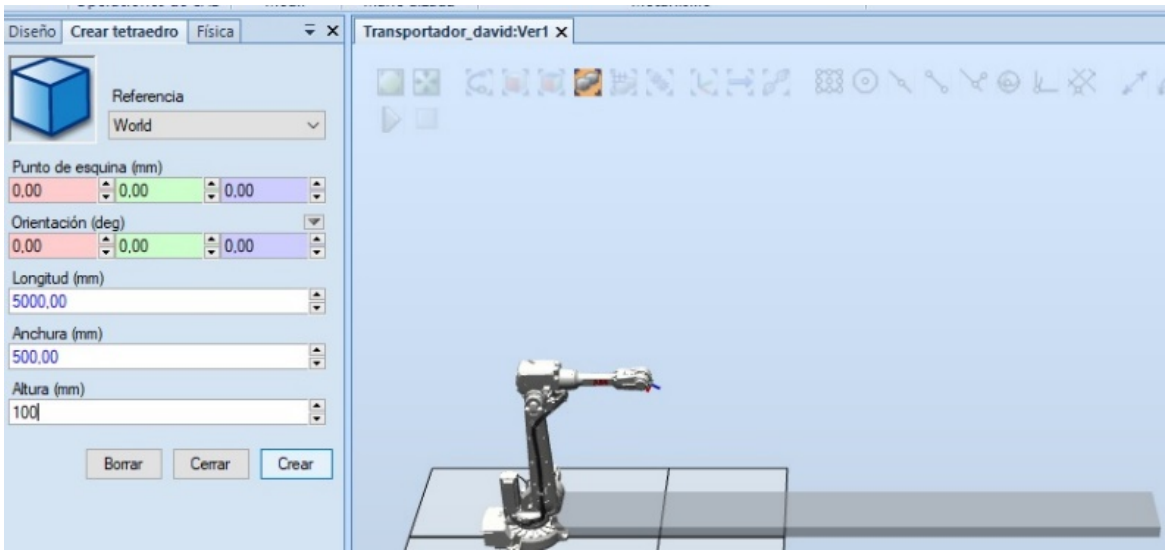


Figura 99: Crear objeto que modela el transportador.

- Mover robot y el transportador a la posición deseada.

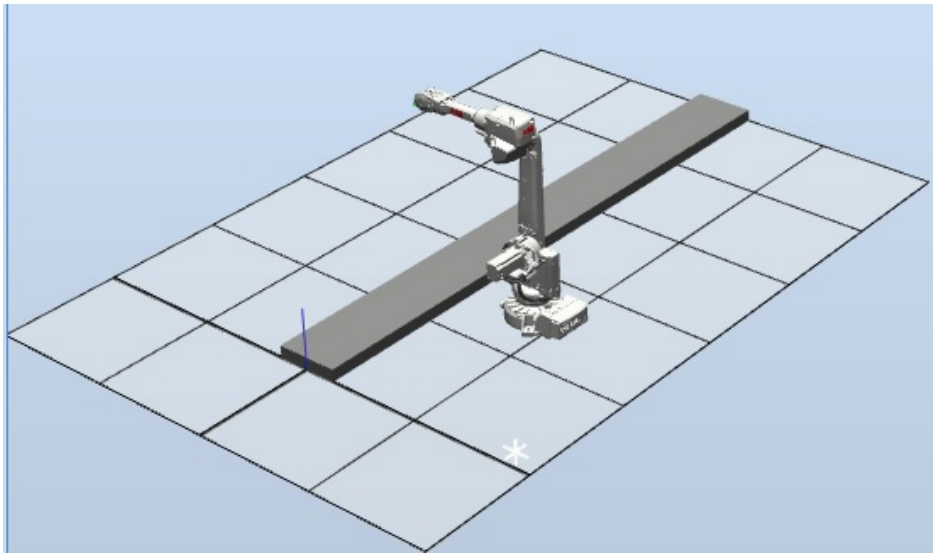


Figura 100: Situar robot y objeto modelo del transportador en la estación.

- Seleccionar el prisma y en la pestaña “Modelado” de la cinta superior hacer clic sobre “Crear transportador”. En la ventana de diálogo que aparece, seleccionar el prisma en “Transportador y geometría”, en “Tipo” seleccionar “Lineal” y en “Longitud del transportador” escribir 5000 y finalmente hacer clic sobre “Crear”.

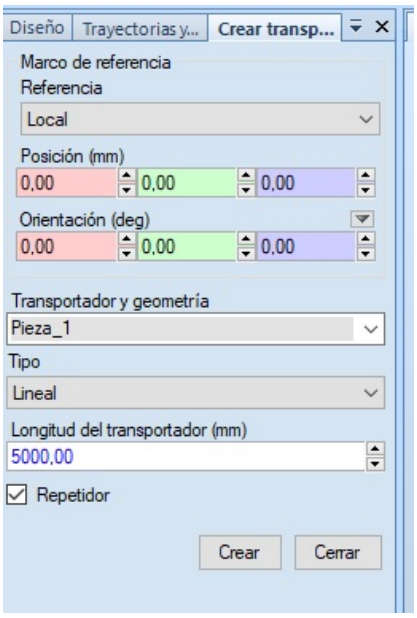


Figura 101: Crear transportador.

- Hacer clic con el botón derecho del ratón sobre el transportador creado y seleccionar “Crear conexión”. En la ventana que aparece especificamos 500 mm en el “Offset”, 2000 mm en la “Anchura de la ventana de inicio”, -1000 mm en la “Distancia mínima” del área de trabajo y 4500 en la “Distancia máxima”

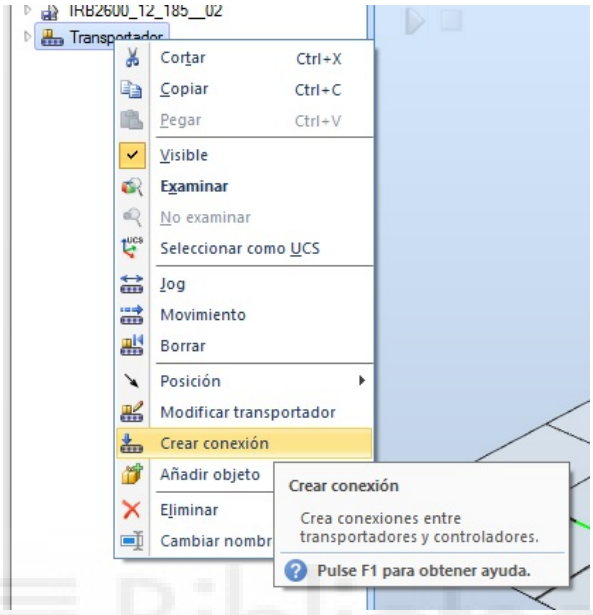


Figura 102:Crear conexión del transportador I.

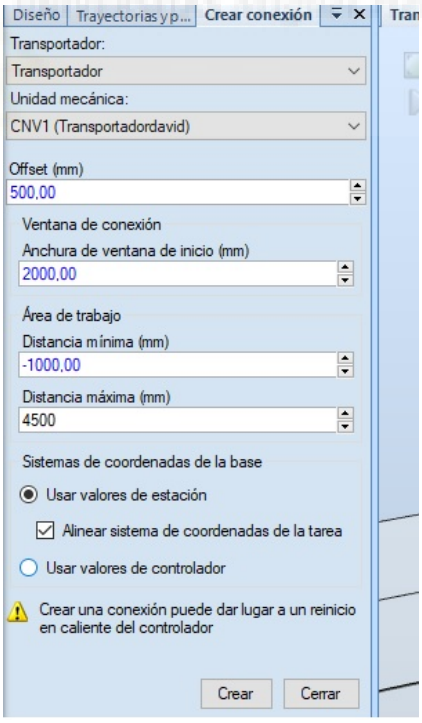


Figura 103: Crear conexión del transportador II.

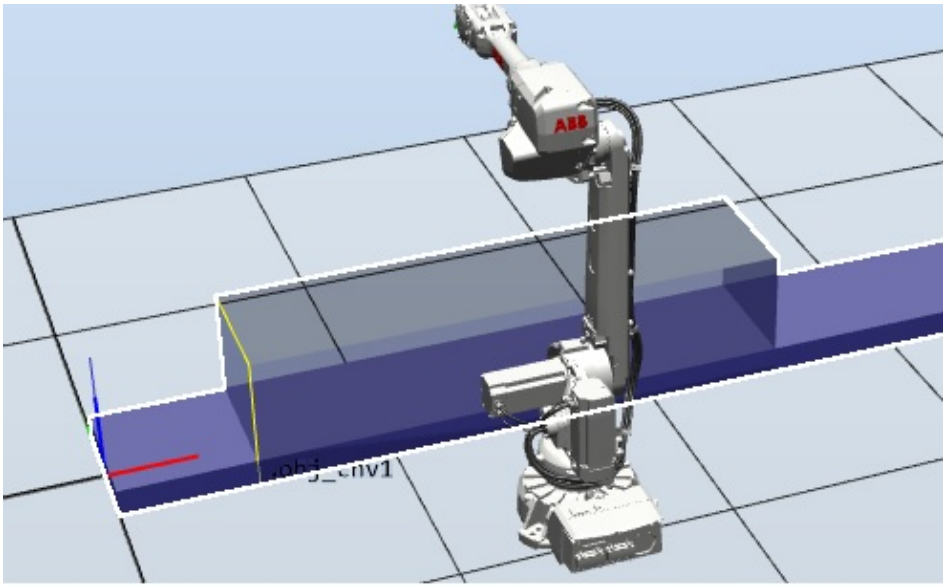


Figura 104: Crear conexión del transportador III.

- Importamos una herramienta desde “**Importar biblioteca**”<<”**Equipamiento**”<<”**Traininig objects**” y la añadimos al robot.

- En la pestaña “**Modelado**” creamos la pieza a trabajar, en este caso crearemos un cilindro de 250 mm de diámetro y 500 mm de altura. Situaremos su sistema de coordenadas de origen sobre el transportador y centrado.

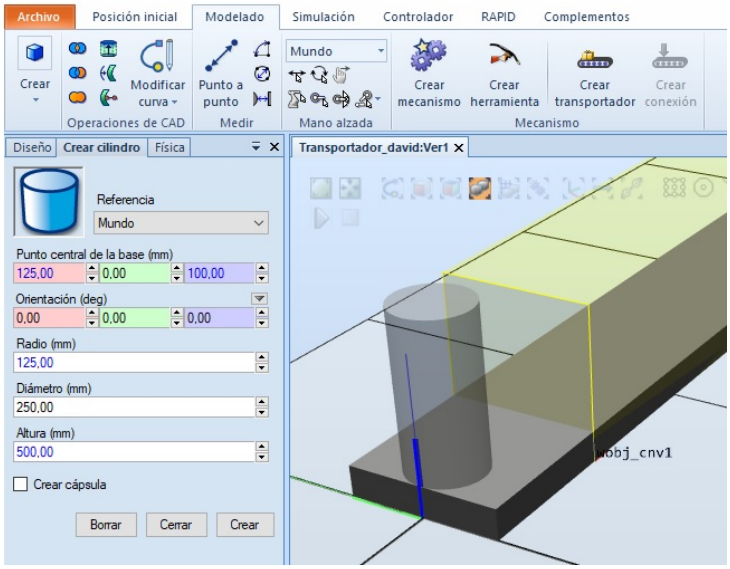


Figura 105: Crear pieza a trabajar sobre el transportador.

- En la pestaña “**Posición inicial**”, hacemos clic con el botón derecho sobre el transportador y seleccionamos la opción “**Añadir objeto**”. En la ventana que aparece, seleccionamos la pieza de trabajo y establecemos una separación entre piezas consecutivas sobre el transportador de 1000 mm, y para que aparezcan centradas sobre el transportador una compensación de la posición en “**Y**” de 250 mm (respecto del sistema de coordenadas “**wobj_cnv1**”.

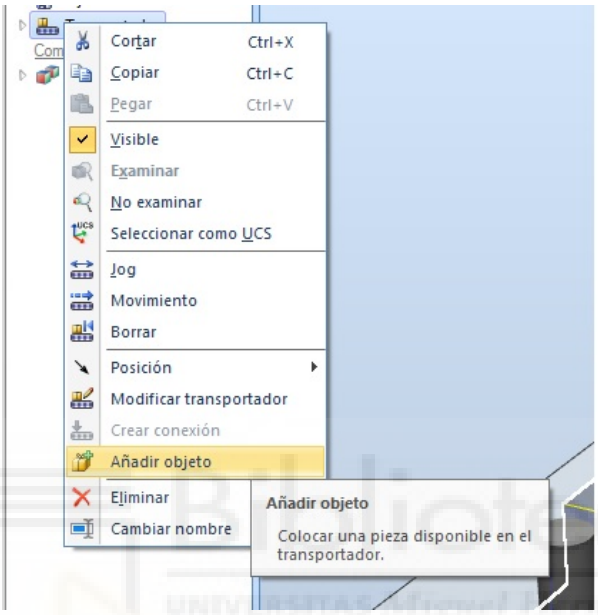


Figura 106: Colocación de la pieza sobre el transportador I.

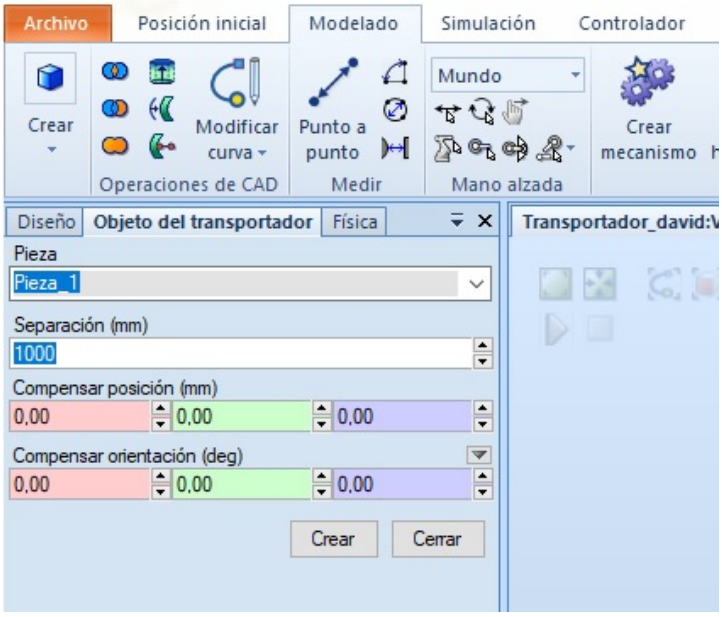


Figura 107: Colocación de la pieza sobre el transportador II.

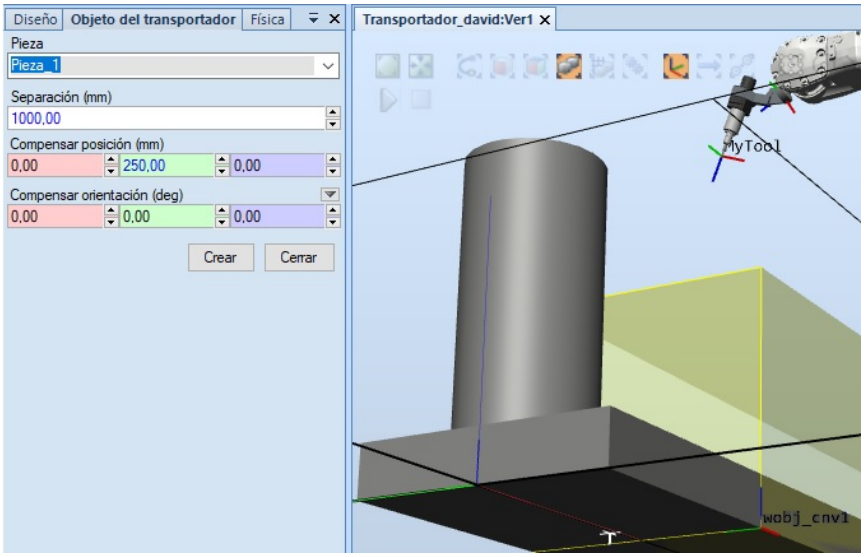


Figura 108: Colocación de la pieza sobre el transportador III.

- En el árbol de la izquierda, dentro de la carpeta "**Fuente de objeto**" seleccionamos la pieza haciendo clic con el botón derecho del ratón y seleccionamos "**Colocar en el transportador**". Repetimos la operación y ahora seleccionamos "**Adjuntar objeto de trabajo**"<<"**wobj_cnv1**".

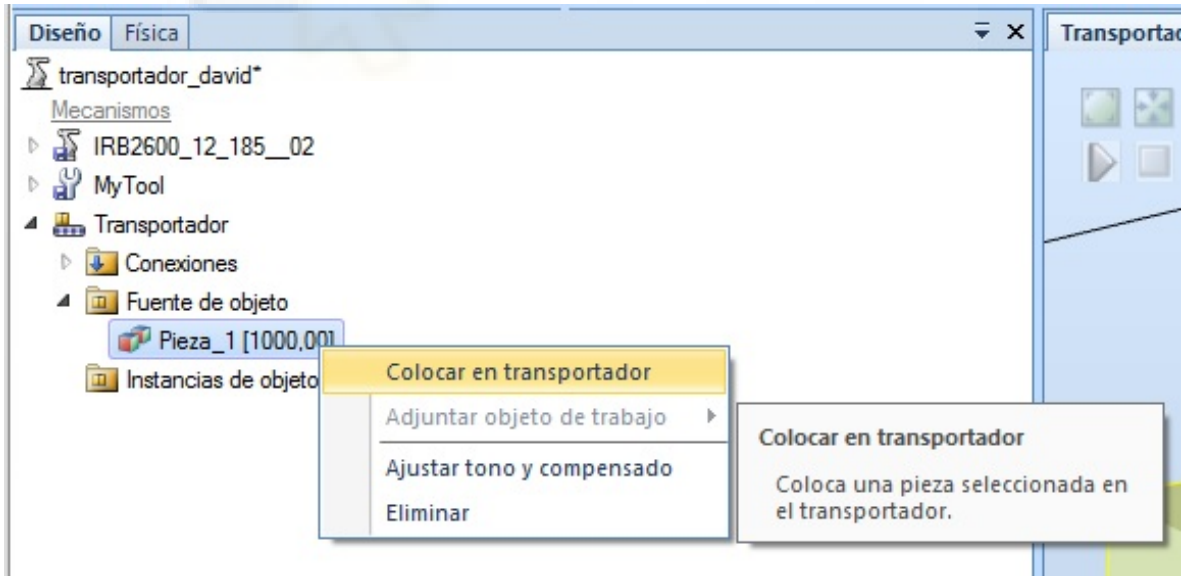


Figura 109: Colocación de la pieza sobre el transportador IV.

Figura 110: Colocación de la pieza sobre el transportador V.

- Con el “Jog” del transportador llevamos pieza a la posición de inicio de la zona de seguimiento anteriormente creada, que podemos ver dibujada sobre el transportador, donde el robot comenzará a trabajar sobre la pieza. En esa posición, crearemos las trayectorias de trabajo del robot sobre la pieza.

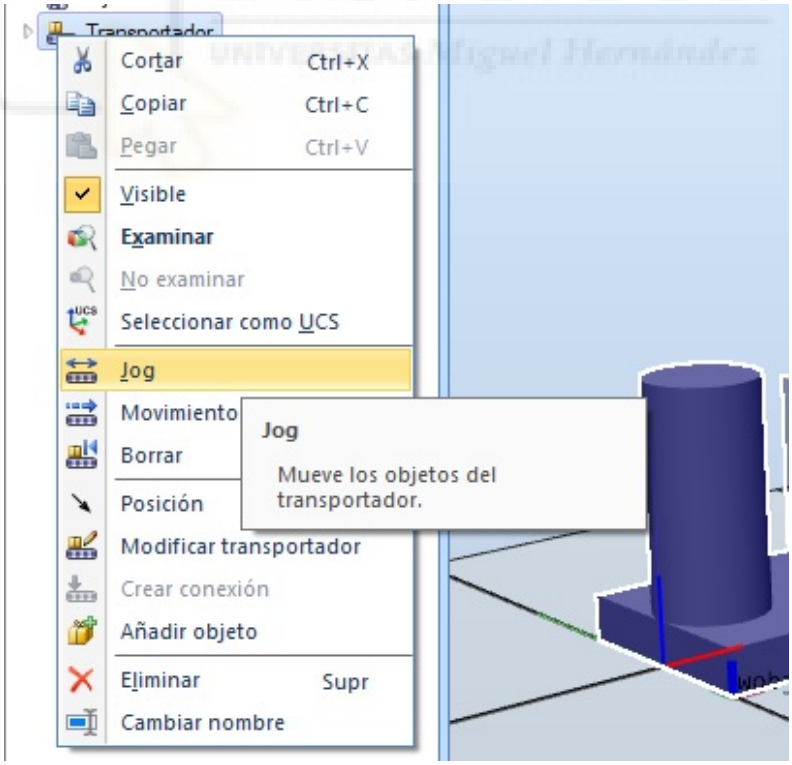


Figura 111: Jog del transportador.

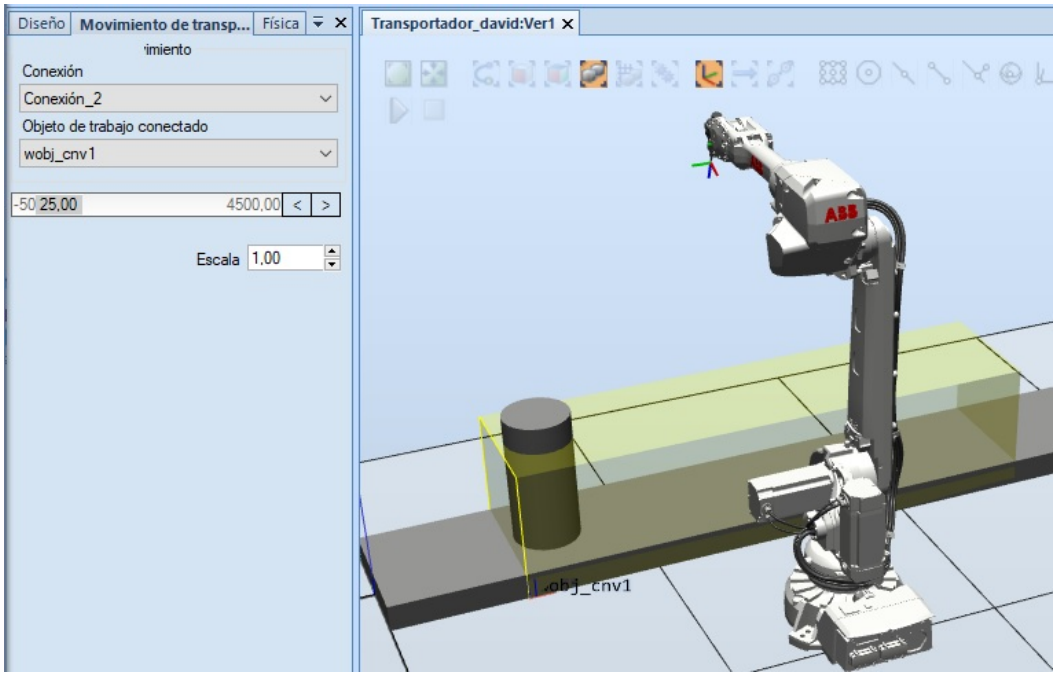


Figura 112: Posición de trabajo de la pieza sobre el transportador.

- Para crear la trayectoria de trabajo, en primer lugar seleccionamos el workobject del transportador, la herramienta y creamos el path desde "**Ruta**"<<"**Trayectoria automática**". Vamos a crear una trayectoria circular seleccionando en la ventana que aparece la opción "**Circular**". Seleccionamos la arista del borde superior del cilindro teniendo cuidado de que la flecha que nos indica en el borde detectado apunte hacia abajo que es la dirección de trabajo seleccionada. Hacer clic dentro de la casilla "**Superficie de referencia**" y luego sobre la cara superior del cilindro para seleccionar la superficie. Podemos configurar más opciones en la trayectoria como un offset de inicio y de fin, la distancia mínima entre los puntos objetivo de la trayectoria, el radio máximo de la curva, la tolerancia y las distancias de partida y aproximación a la trayectoria. Solo modificaremos, en este caso, las distancias de partida y aproximación a la trayectoria con 50 mm. Como podemos comprobar se crea el path con las instrucciones de movimiento al punto de aproximación, el punto de inicio, dos trayectorias semicirculares (**no se pueden crear trayectorias de 360º**) y la trayectoria de partida.

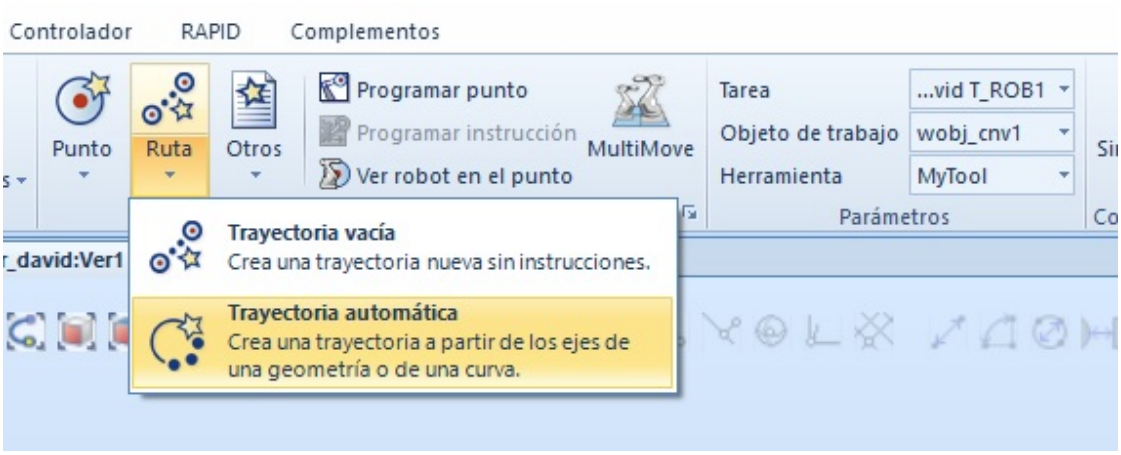


Figura 113: Creación de una trayectoria automática I.

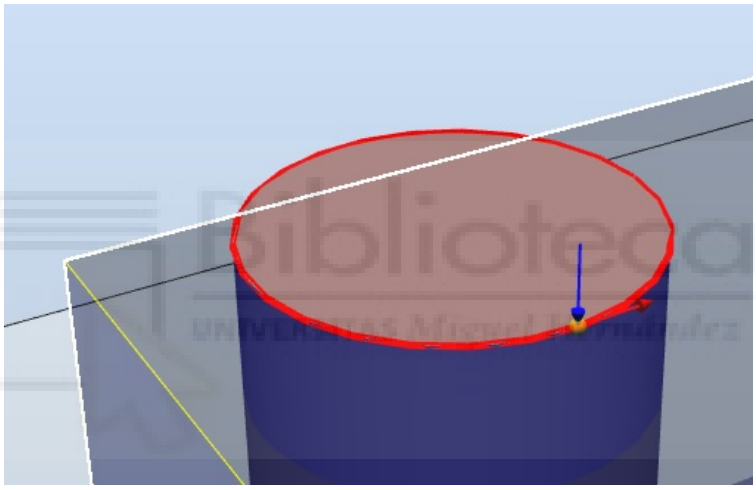


Figura 114: Creación de una trayectoria automática II.

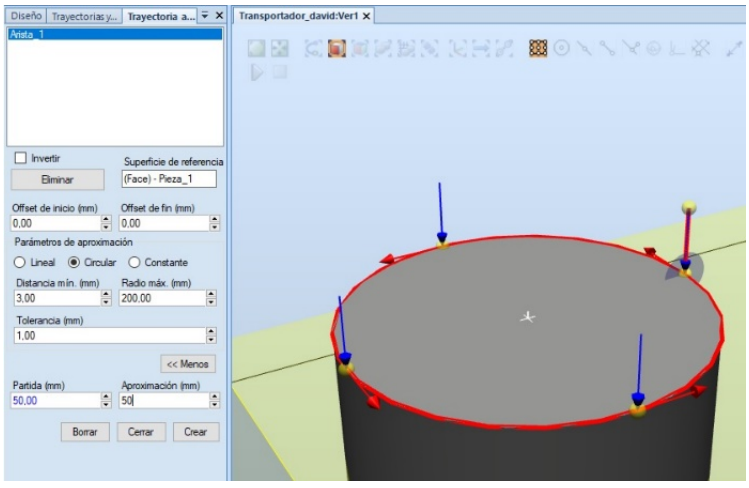


Figura 115: Creación de una trayectoria automática III.

Finalmente comprobaremos las alertas de RobotStudio y modificaremos si es necesario la orientación de la herramienta, alcances y configuraciones para que el robot pueda efectuar dicho path.

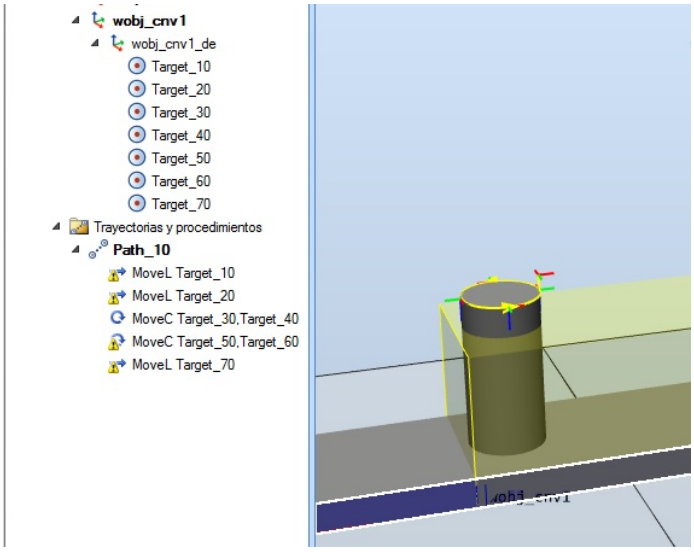


Figura 116: Alertas de RobotStudio en la trayectoria.

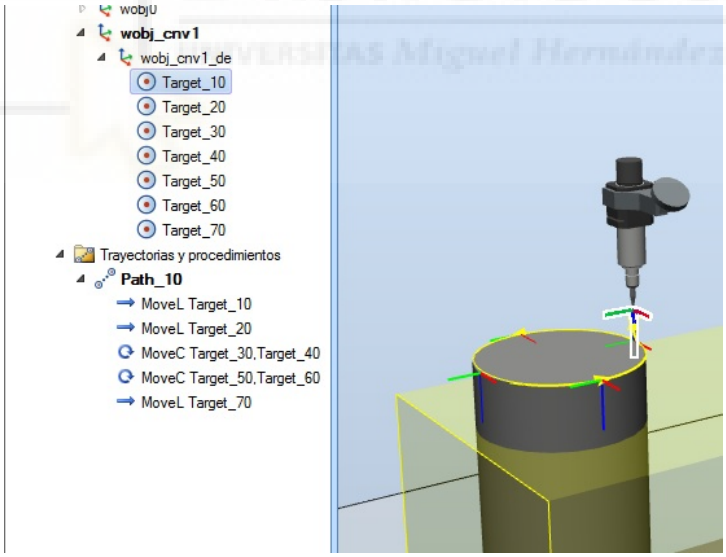


Figura 117: Orientación de la herramienta en la trayectoria.

Crearemos una posición “home” desde la que partirá y a la retornará el robot después de realizar la trayectoria sobre la pieza. Para ello vamos a la pestaña “**Posición inicial**” y en la barra inferior de la pantalla seleccionamos la plantilla de movimiento “MoveAbsj”, v1000,

z100, MyTool y \WObj:=wobj0. Programamos el punto con la instrucción “**Programar instrucción**” del grupo “**Programación de trayectorias**”. En la carpeta “**Posición de ejes**” aparecerá el punto programado. Le cambiamos el nombre a “**home**”.

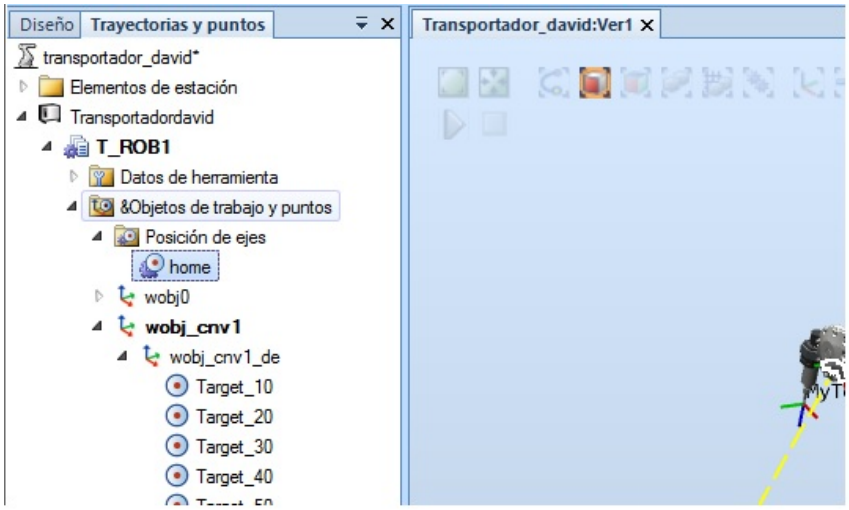


Figura 118: Creación de una posición "home".

Haciendo clic con el botón derecho del ratón sobre la carpeta “**main**” seleccionamos “Insertar instrucción de acción y creamos las siguientes instrucciones”:

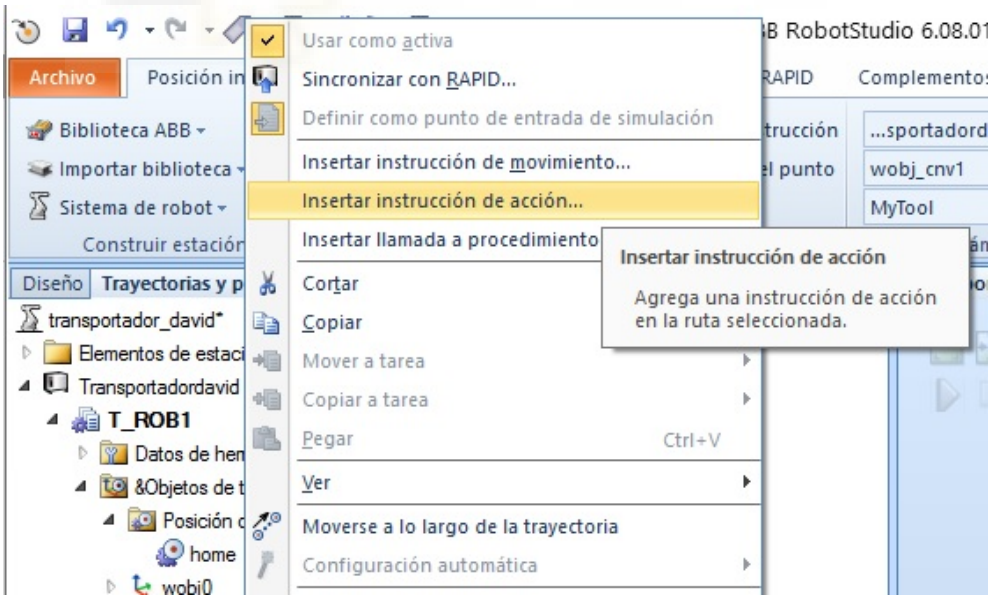


Figura 119: Insertar instrucción de acción.

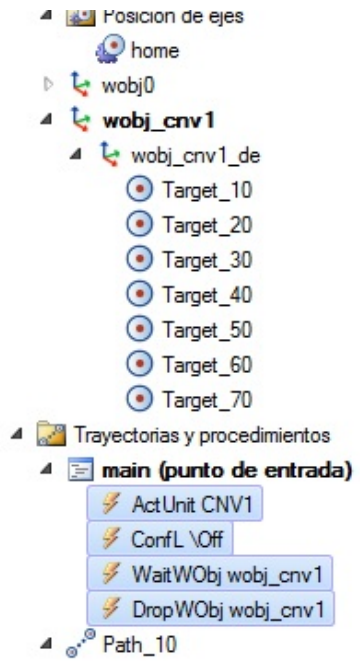


Figura 120: Instrucciones de acción en un trasnportador.

Hacemos clic sobre la posición de ejes "**home**" creada con el botón derecho del ratón y seleccionamos "**Añadir a trayectoria**"<<"**main**"<<"**Primera**" para añadir la trayectoria a "**home**". Hacemos clic sobre el "**main**", seleccionando ahora "**Insertar llamada a procedimiento**"<<"**path_10**" para añadir el path a "**main**".

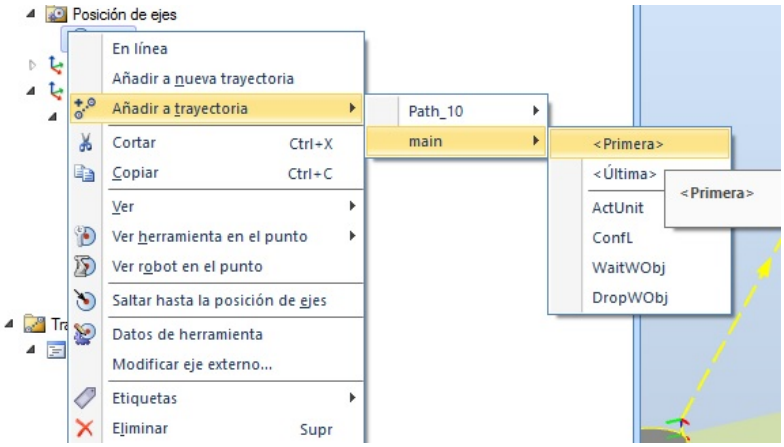


Figura 121: Insertar posición "home" en main.

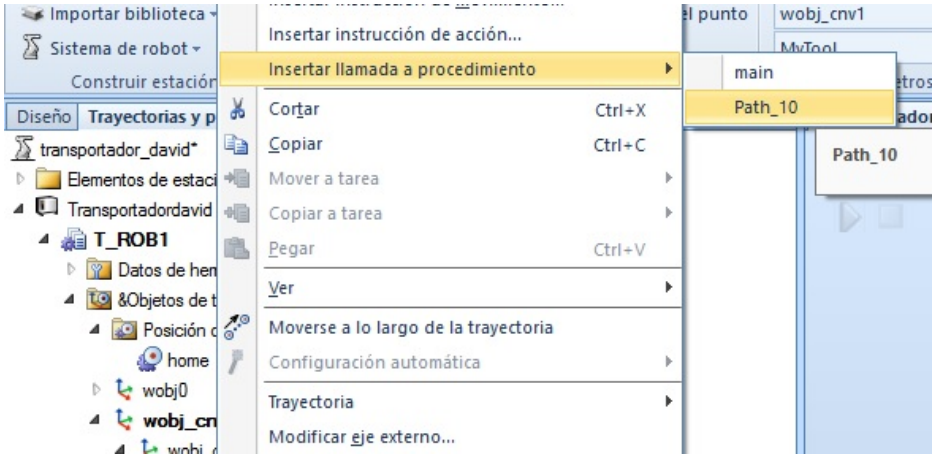


Figura 122: Insertar path en main.

Finalmente reordenamos arrastrando las instrucciones del "**main**" para que quede así:

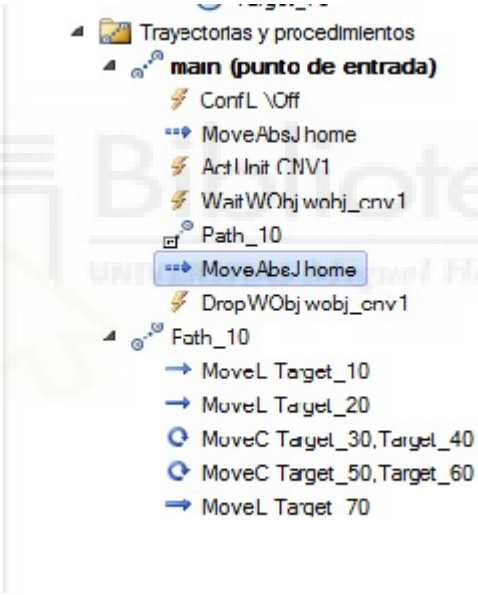


Figura 123: Reordenar instrucciones en main.

Guardamos y sincronizamos con **RAPID**.

4.1.3.11 SmartComponents.

RobotStudio permite crear los denominados "**Componentes inteligentes**" que son objetos con o sin representación gráfica que permiten representar el comportamiento de objetos de una estación tales como sensores, movimiento de piezas, señales de entrada/salida, etc.

Podemos añadir componentes inteligentes de dos formas, la primera desde la pestaña "**Modelado**" que nos permite crear un contenedor al cual podemos añadir diferentes componentes inteligentes y establecer las propiedades y realizar las conexiones entre ellos para implementar la lógica del comportamiento necesario. Su uso estaría destinado a crear objetos con comportamientos deseados, tales como herramientas, etc. La segunda forma de añadir un componente inteligente es desde la pestaña "**Simulación**"<<"**Lógica de la estación**" en donde podemos añadir componentes inteligentes fuera de un contenedor de componentes inteligentes y configurar las propiedades del componente así como implementar la lógica de la estación.

Para crear un contenedor de componentes inteligentes, desde pestaña "**Modelado**" de la cinta superior hacemos clic sobre "**Componente inteligente**" y aparece la venta de creación de componentes inteligentes.

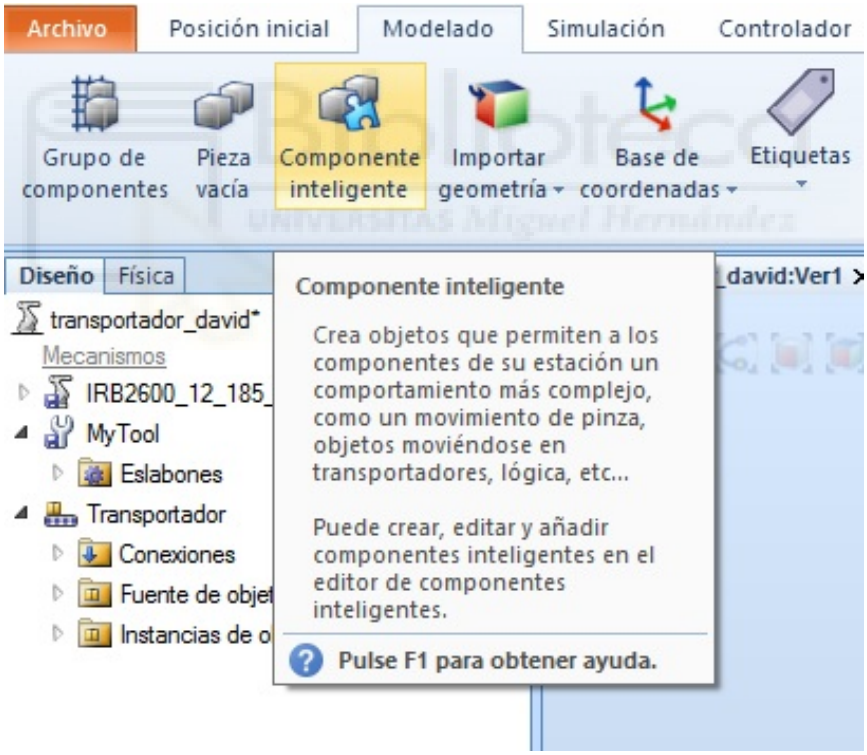


Figura 124: Creación de un componente inteligente I.

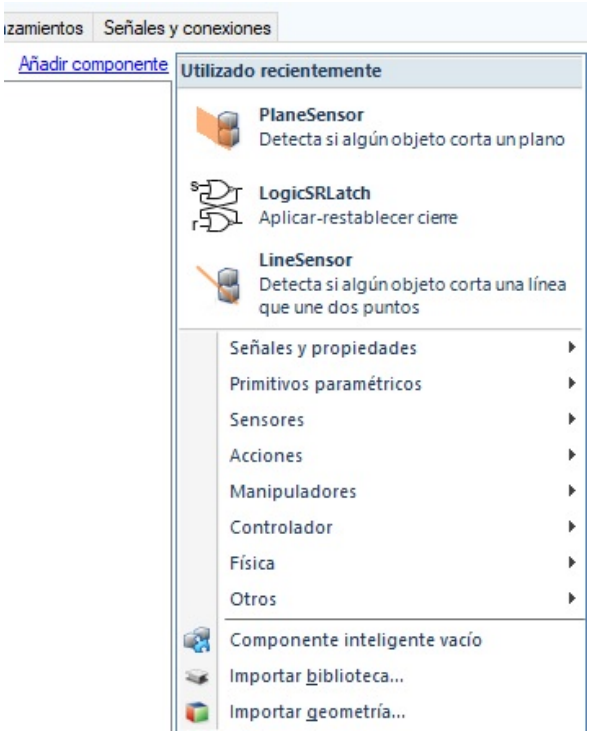


Figura 125: Creación de un componente inteligente II.

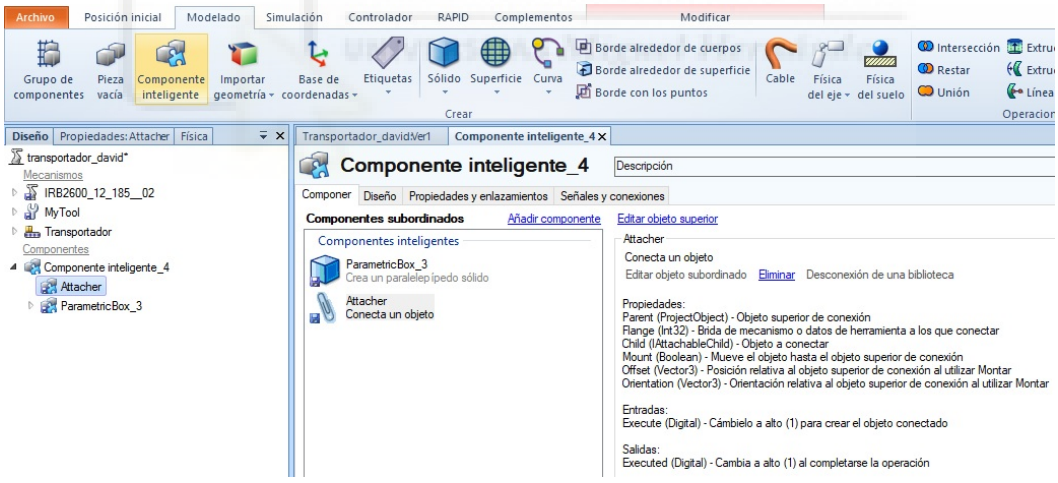


Figura 126: Creación de un componente inteligente III.

Para añadir un componente inteligente a la lógica de la estación, desde pestaña "**Simulación**" de la cinta superior hacemos clic sobre "**Lógica de la estación**" y aparece la venta desde donde podemos añadir componentes inteligentes e implementar la lógica necesaria.

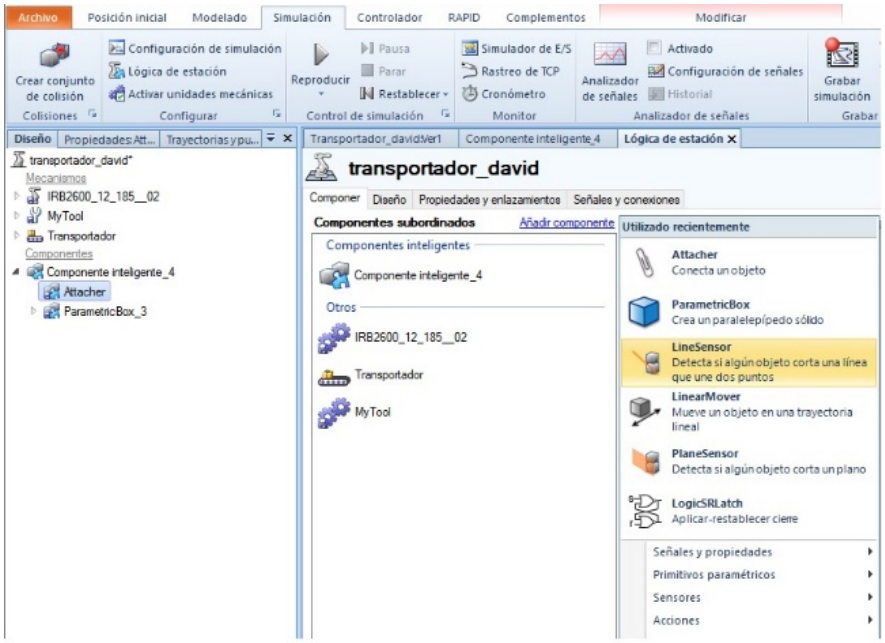


Figura 127: Creación de un componente inteligente IV.

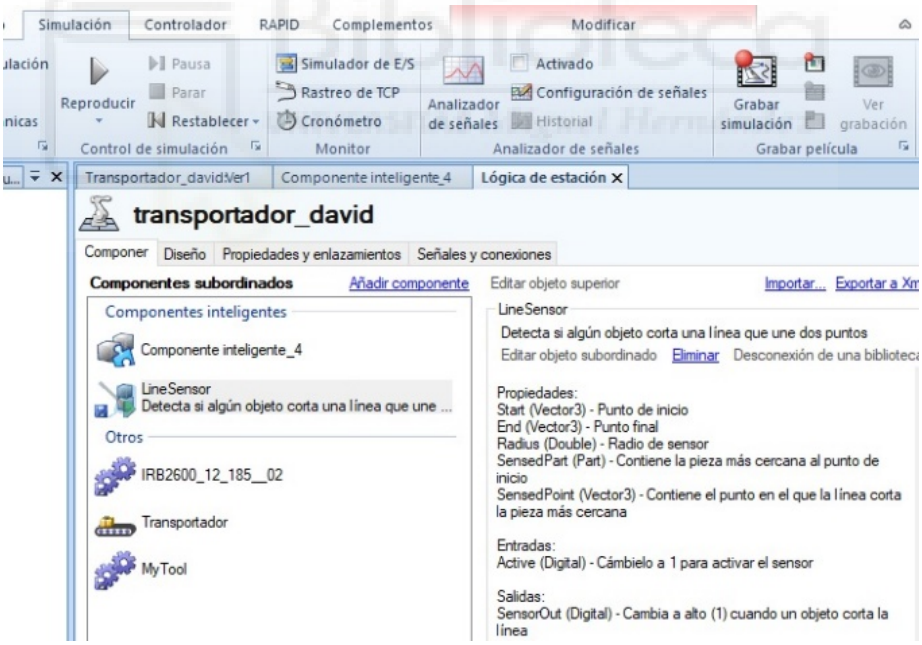


Figura 128: Creación de un componente inteligente V.

Haciendo clic sobre "**Añadir componente**" nos aparece una ventana flotante con todas las categorías de componentes inteligentes incluidos en la biblioteca de RobotStudio:

- **Señales y propiedades**: Podemos encontrar puertas lógicas, comparadores, temporizadores, etc.

Figura 129: CI Señales y propiedades I.

Figura 130: CI Señales y propiedades II.

- **Primitivos paramétricos**: Disponemos de prismas, cilindros, repetidores, etc. que pueden variar su tamaño/acción en función de algún parámetro.

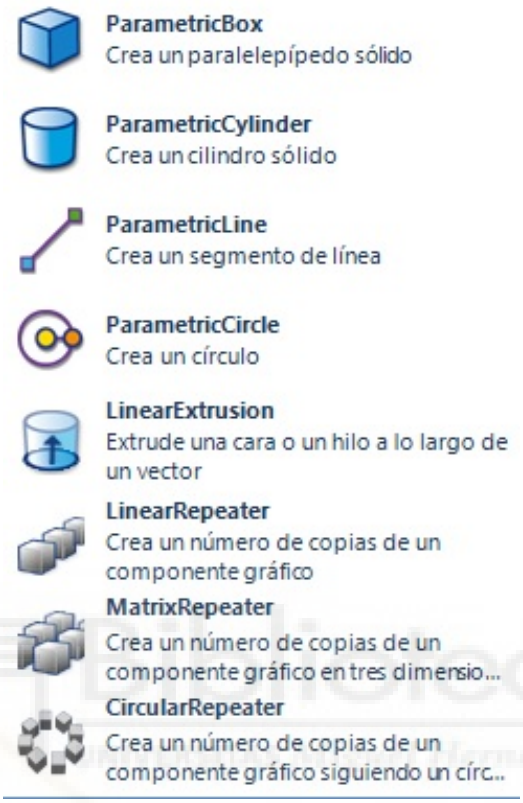


Figura 131: CI Primitivos y paramétricos..

- **Sensores**: Podemos detectar objetos, o posiciones de objetos para simular sensores en la estación.

Figura 132: CI Sensores.

- **Acciones**: Podemos conectar/desconectar objetos a **workobjects** u otros objetos para simular movimientos, crear copias, borrar, etc.

Figura 133: CI Acciones.

- **Manipuladores**: Permiten realizar distinto tipo de movimientos sobre los objetos.

Figura 134: CI Manipuladores.

- **Controlador**: Podemos obtener el valor de una variable de RAPID.

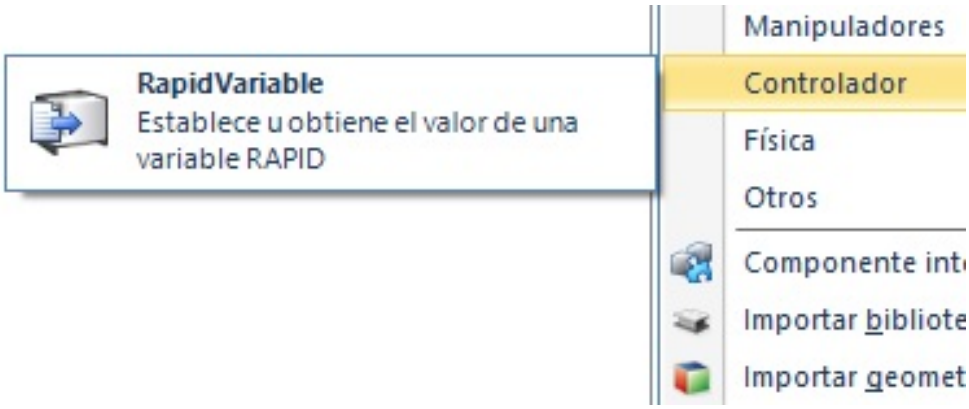


Figura 135: CI Controlador.

- **Física**: Permite controlar algunos parámetros del módulo de simulación de RobotStudio

Figura 136: CI Física.

- Otros: Aquí encontramos SmartComponents para simular colas de objetos, generar números aleatorios, etc.

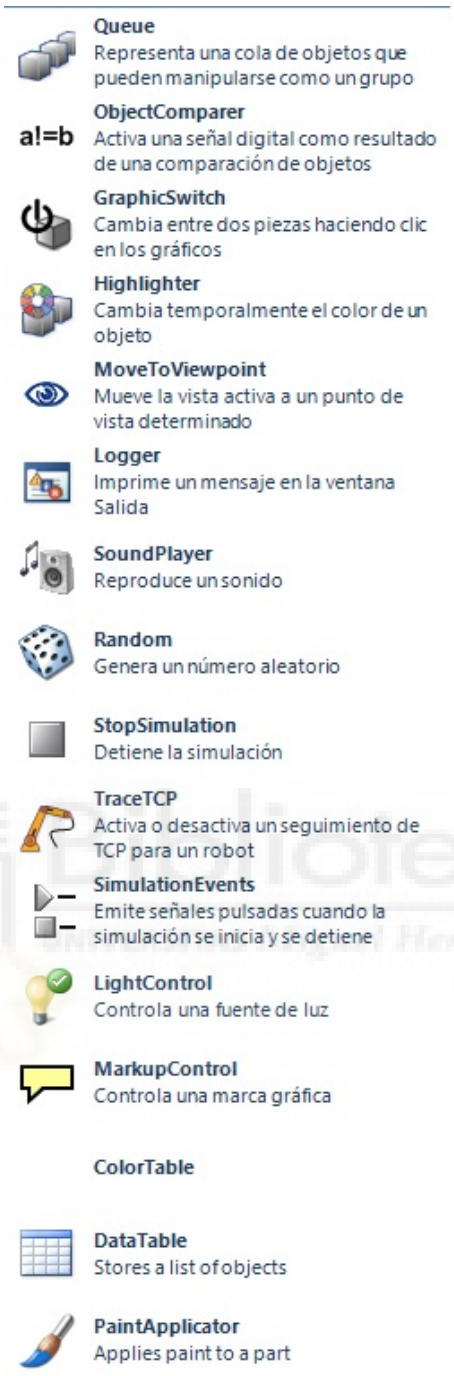


Figura 137: CI Otros.

En el desarrollo de la estación se implementa un ejemplo de componente inteligente para simular las barreras inmateriales de seguridad de la estación.

4.1.3.12 Señales de E/S.

En RobotStudio podemos crear señales de entrada o salida desde la pestaña “Controlador”<<”Configuración”<<”I/O System”, en la ventana que aparece, haciendo clic sobre “Signal” podemos crear una nueva señal de entrada o salida de tipo digital o analógico. Es posible asignarlas un módulo real de entradas/salidas del robot o. si no se especifica nada, crearlas para uso exclusivo de simulación.

Además RAPID dispone de múltiples instrucciones que permiten activar o desactivar salidas y leer el estado de las entradas.

A modo de ejemplo la siguiente orden activa una salida digital en una trayectoria circular: El TCP de la herramienta, tool2, se mueve en círculo hacia la posición p2 con los datos de velocidad v500 y los datos de zona z30. El círculo se define a partir de la posición inicial, el punto de círculo p1 y el punto de destino p2. La salida do1 se activa en el centro de la trayectoria de esquina de p2.

MoveCDO p1, p2, v500, z30, tool2, do1,1;

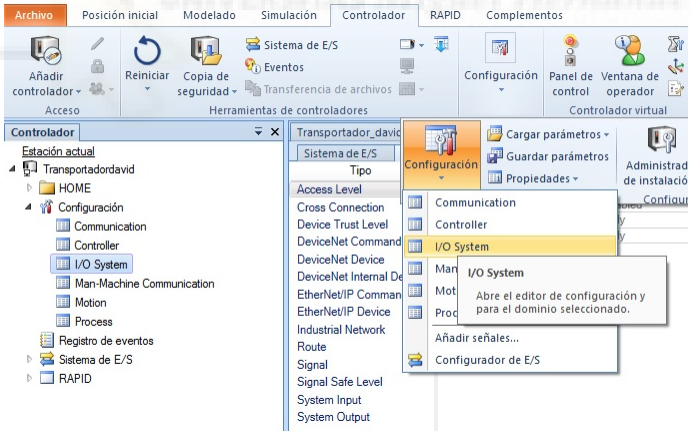


Figura 138: Señales de E/S I.

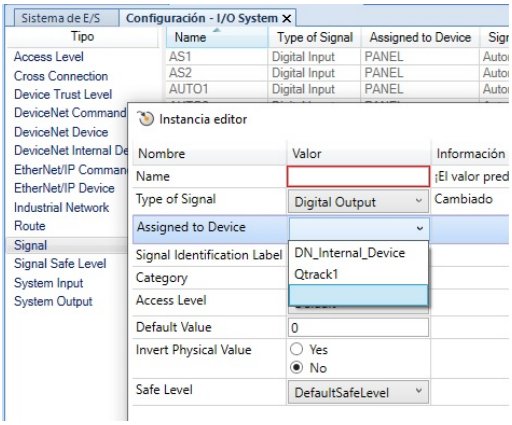


Figura 139: Señales de E/S II.

También es posible crear los denominados “Cross Connectión” que permiten crear conexiones entre entradas y salida para activar una en función de la señal de otra.

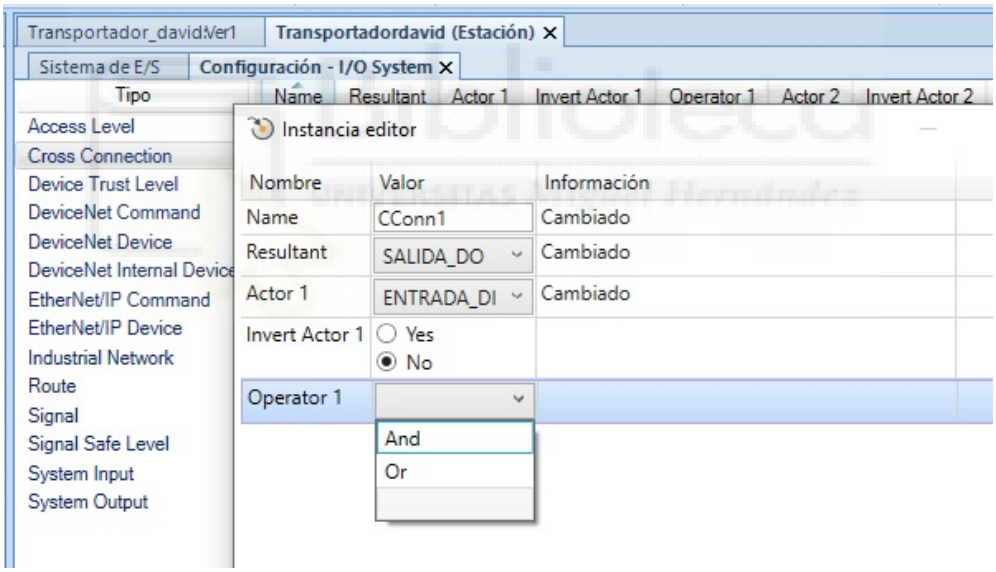


Figura 140: Cross Connection.

Una vez creadas las señales, es necesario reiniciar el sistema desde el icono “Reiniciar” de la pestaña “Controlador” para incorporarlas al sistema.

Figura 141: Reiniciar controlador.

A la hora de simular señales de E/S, puede crear eventos que establecen los valores de las señales cuando se cumplen determinadas condiciones de disparo, o bien puede establecer los valores de las señales manualmente.

Podemos acceder al “Gestor de eventos” desde la pestaña “Simulación”.

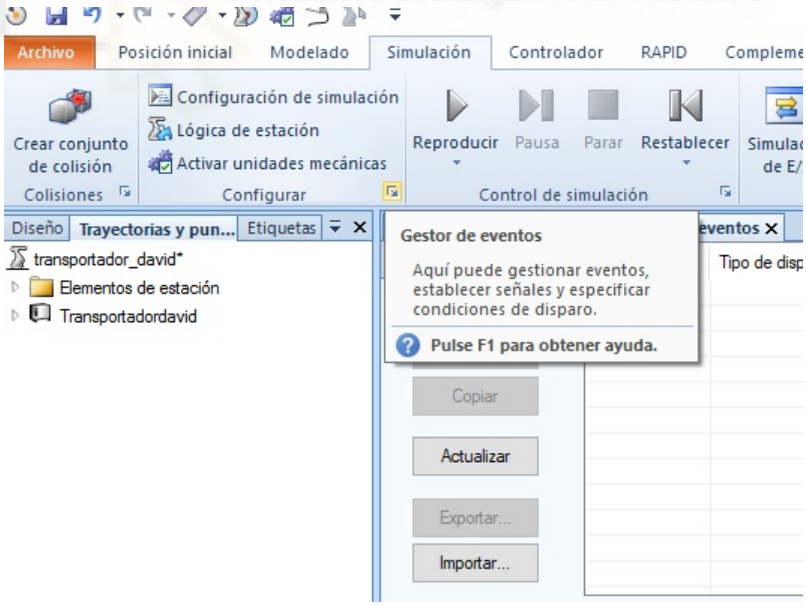


Figura 142: Gestor de eventos.

Para abrir el panel Simulador de E/S, desde la pestaña “Simulación”. Podemos, en este panel, visualizar o activar el estado de diferentes señales. Es posible crear listas para visualizar solo aquellas que nos interesan.

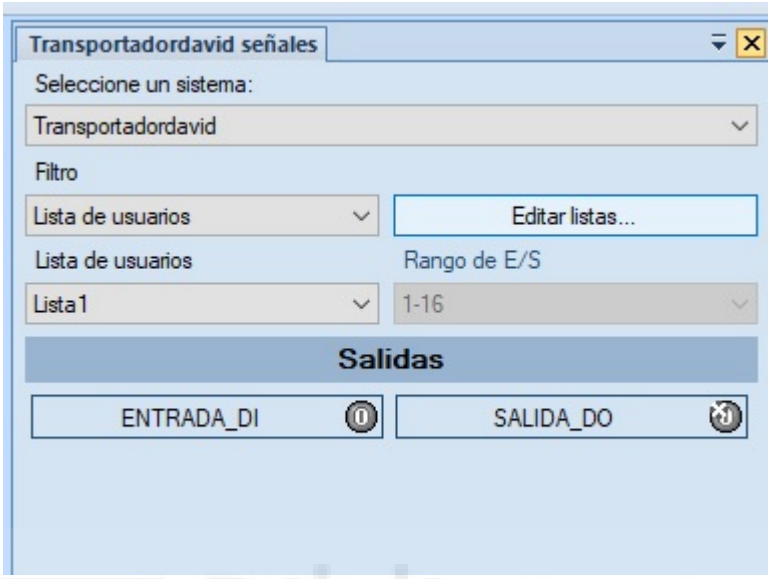


Figura 143: Simulador de E/S.

Hay que recordar de nuevo que para que todos estos cambios se reflejen en el sistema hay que reiniciar el controlador y/o sincronizar con RAPID.

4.1.3.13 Lógica de la estación.

En la pestaña “**Simulación**” de la cinta superior encontramos el icono “**Lógica de la estación**” donde podemos establecer las conexiones entre entradas y salidas de la estación, sistema y componentes inteligentes para implementar la lógica que debe ejecutarse en la estación para su simulación. Consta de cuatro pestañas:

- **Componer**: podemos añadir componentes inteligentes y/o editar sus propiedades.
- **Diseño**: podemos realizar las conexiones de forma gráfica entre los distintos componentes de la estación y sus entradas y salidas.
- **Propiedades y enlazamientos**: es posible establecer conexiones con las propiedades dinámicas de los objetos de la estación.

- **Señales y conexiones**: podemos realizar las conexiones de forma manual entre los distintos componentes de la estación y sus entradas y salidas.

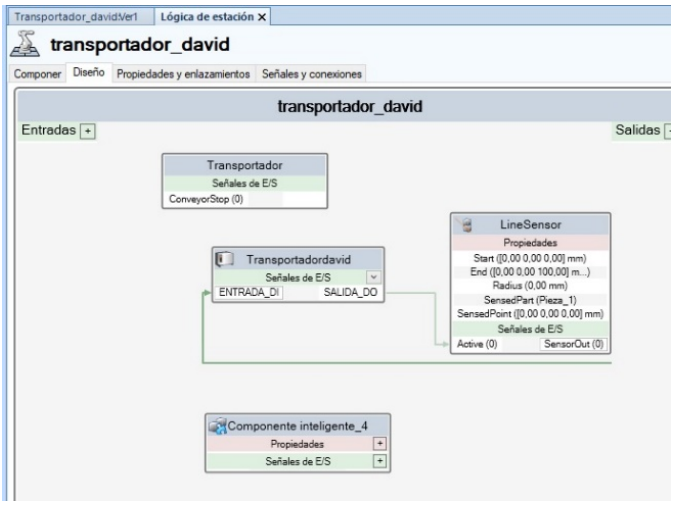


Figura 144: Lógica de la estación (Diseño).

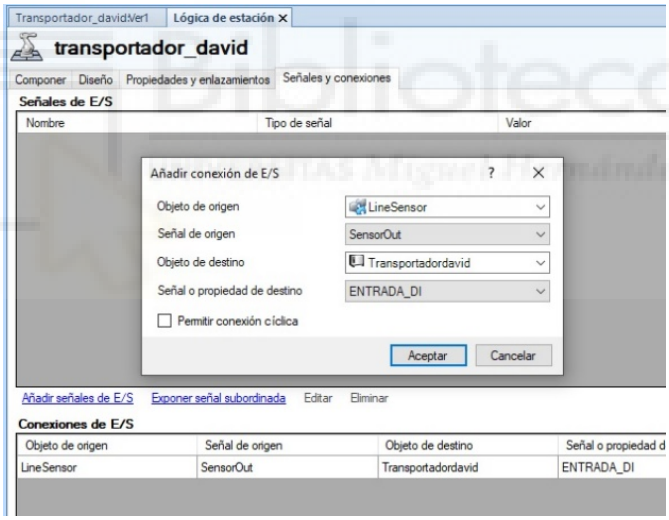


Figura 145: Lógica de la estación (Señales y conexiones).

4.1.3.14 Simulación.

El módulo de simulación de RobotStudio permite:

- La ejecución de programas de robot completos en el controlador virtual.
- La detección de colisiones entre mecanismos (robot, track, posicionador) o entre mecanismos y objetos (piezas, útiles, vallas, etc) mediante la creación de conjuntos de colisión.

- La gestión de eventos para conectar una acción a un disparador (conectar una objeto a otro cuando colisionan o cuando se activa una señal)
- La simulación de E/S: Las señales de entrada y salida del sistema pueden activarse manualmente u observar su estado.
- El seguimiento de la trayectoria del TCP mediante rastreo.
- La medición del tiempo del proceso.

Una vez completado un programa, para simularlo, en la cinta superior hacemos clic sobre la pestaña "**Simulación**"<<"**Configuración de la simulación**". En la ventana que aparece podemos seleccionar que componentes inteligentes y sistema queremos simular y si queremos que ejecute la simulación del programa una sola vez o de forma continua. Si hacemos clic sobre la tarea (en el ejemplo de la figura "**T_ROB1**"), podemos seleccionar el punto de entrada de la simulación (normalmente "**main**" para simular todo el programa).

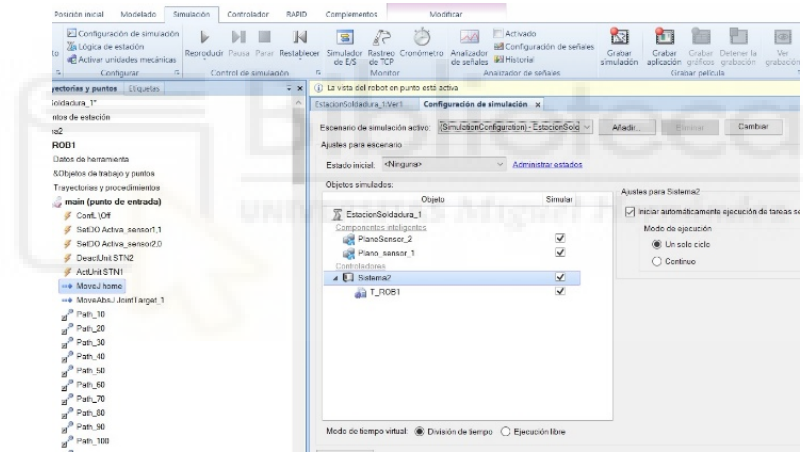


Figura 146: Configuración de simulación I.

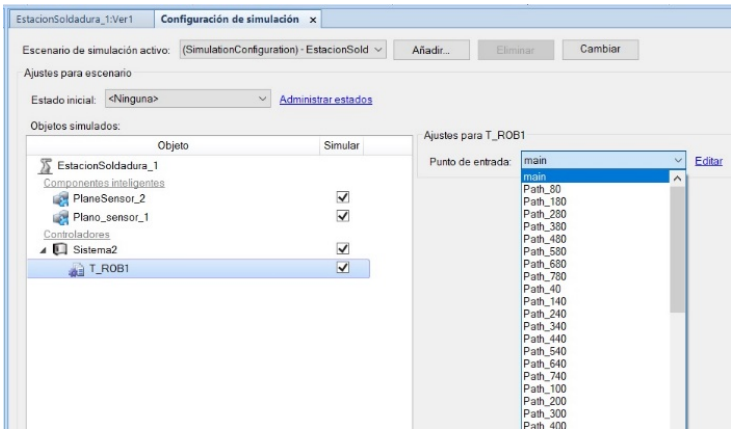


Figura 147: Configuración de simulación II.

Para comenzar la simulación en la cinta superior hacemos clic sobre “Reproducir”, que nos ofrece tres opciones:

- Reproducir la simulación.
- Sincronizar los últimos cambios con RAPID y reproducir la simulación.
- Reproducir la simulación y grabarla a un archivo de video.

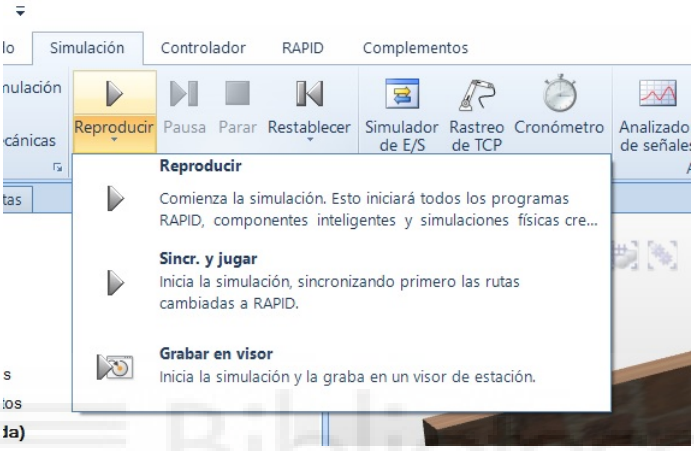


Figura 148: Opciones de reproduccián en Simulación

Para crear un “**conjunto de colisión**” y detectar si se producen durante la ejecución del programa crearemos dos grupos de objetos, Objetos A y Objetos B, en los que puede situar objetos para detectar las colisiones existentes entre ellos. En la pestaña “**Simulación**” haciendo clic sobre “**Crear conjunto de colisión**” y nos aparece en la pestaña “**Diseño**” del árbol de la izquierda el conjunto de colisión creado al que arrastraremos los objetos que queramos en cada carpeta A o B. Cuando cualquier objeto de Objetos A colisiona con cualquier objeto de Objetos B, la colisión se representa en la vista gráfica y se registra en la ventana de salida. Es posible tener varios conjuntos de colisión en la estación, pero cada conjunto de colisión sólo puede contener dos grupos.

Figura 149: Conjunto de colisión.

Para activar el rastreo del TCP, en la pestaña “simulación” de la cinta superior hacemos clic en sobre “Rastreo TCP” y en la ventana que aparece podemos, entre otras opciones, activar o desactivar el rastreo, cambiar el color de la trayectoria del rastreo y borrar los rastreos anteriores.

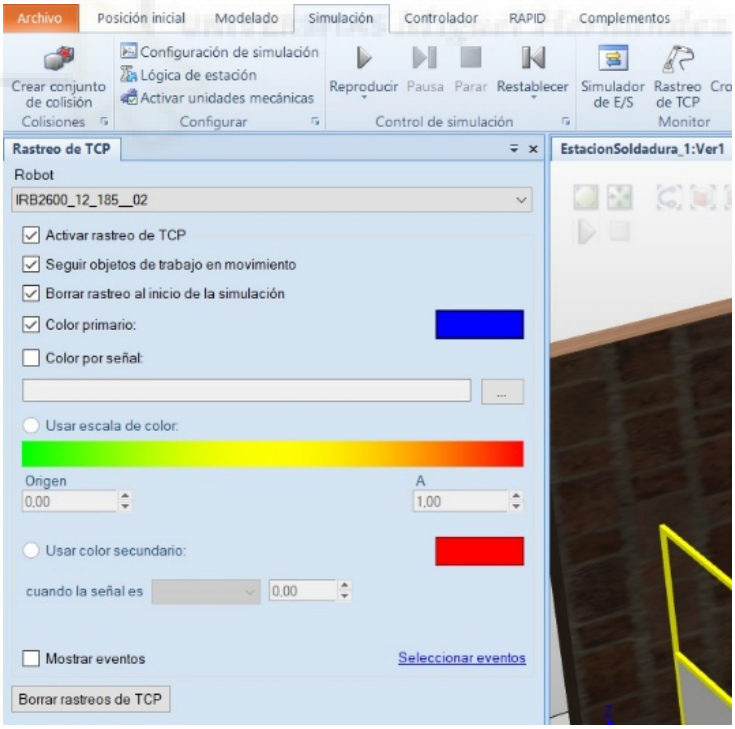


Figura 150: Rastreo del TCP I.

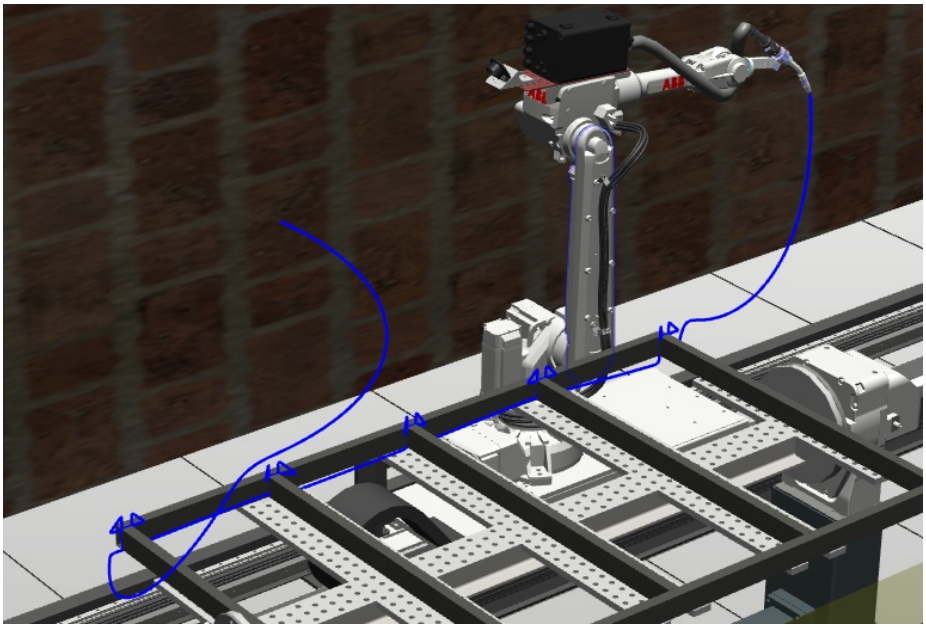
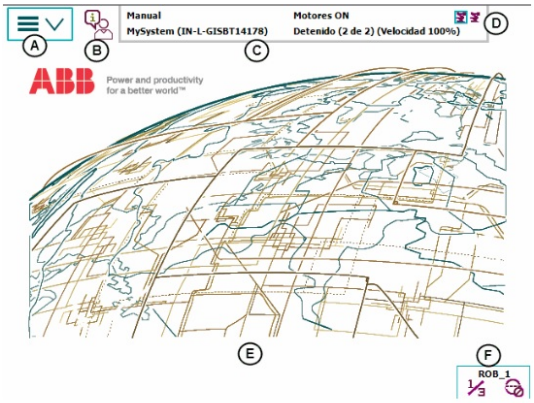


Figura 151: Rastreo del TCP II.

4.1.3.15 FlexPendant.

RobotStudio permite simular el FlexPendant del robot, que es la consola de mano del operador del robot. Con el FlexPendant el operador realiza todas las tareas necesarias para su manejo, desde la carga de programas, la programación sencilla, modificación de programas, movimiento de ejes, calibración etc. El FlexPendant dispone de una pantalla táctil, botones de acceso rápido y un joystick.



A	Menú principal
B	Ventana de operador
C	Barra de estado
D	Botón Cerrar
E	Barra de tareas
F	Menú de configuración rápida

Figura 152: Pantalla del FlexPendant I.

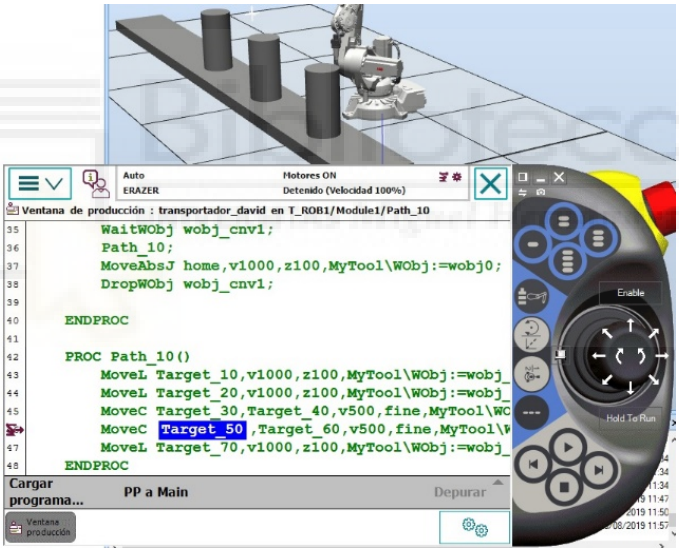


Figura 153: Pantalla del FlexPendant II.

La controladora del robot dispone de tres posiciones de trabajo:

- Modo automático.
- Modo manual a velocidad reducida.
- Modo manual a velocidad máxima.

El modo manual a velocidad reducida es el que siempre debe emplearse por seguridad para las tareas con el FlexPendant dentro del área de trabajo del robot, utilizando el modo

manual a velocidad máxima para comprobaciones del funcionamiento de programas en la fase final de programación y siempre fuera de área de trabajo del robot.

En el modo manual es necesario accionar el botón de habilitación de tres posiciones, en el cual solo permite el movimiento cuando este está pulsado en su posición intermedia.

En el menú principal del FlexPendant se pueden seleccionar los elementos siguientes:

- **Edición en marcha**: se usa para ajustar las posiciones programadas. Puede hacerse en todos los modos de funcionamiento, e incluso mientras el programa se está ejecutando. Es posible ajustar tanto las coordenadas como la orientación. Sólo puede usarse con las posiciones con nombre del tipo robtarget.

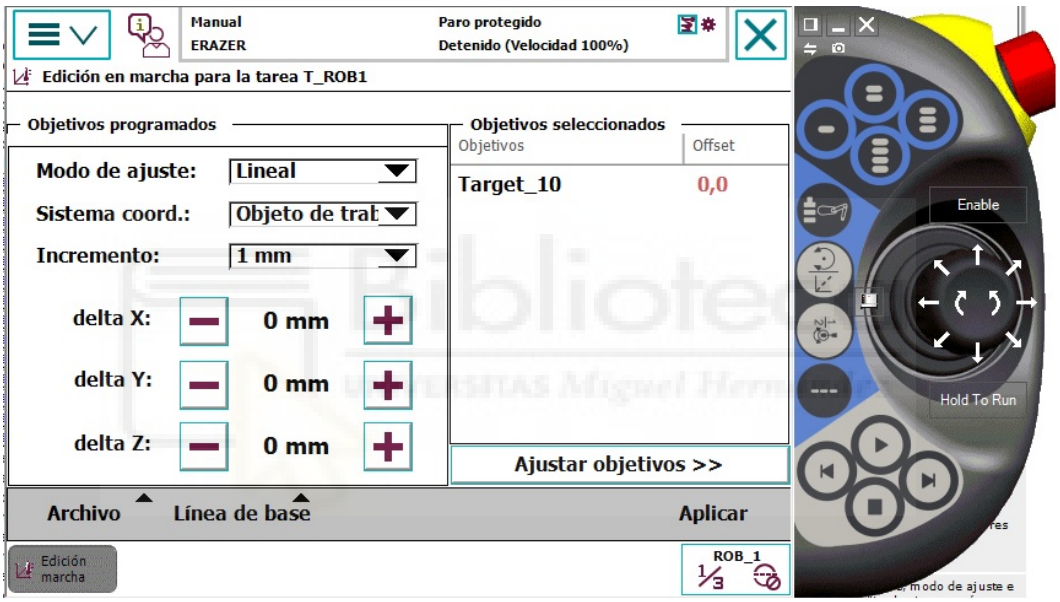


Figura 154: Edición en marcha en FlexPendant.

- **Entradas y salidas**: Las entradas y salidas, E/S, son señales utilizadas en el sistema de robot. La señales se configuran utilizando parámetros del sistema.

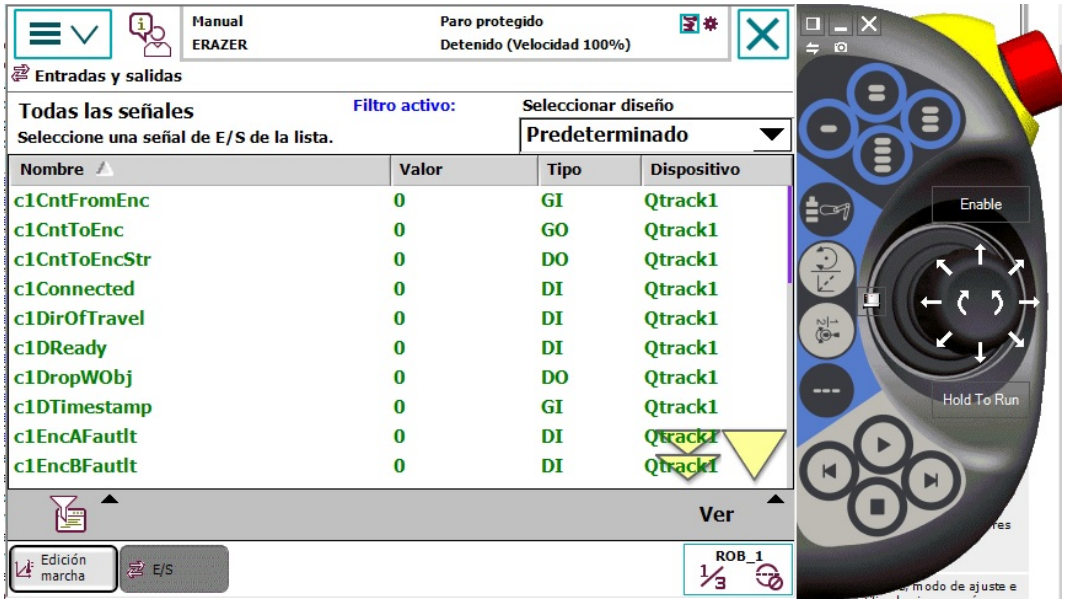


Figura 155: Entradas y salidas en FlexPendant.

- **Movimiento**: Las funciones de movimiento se encuentran en la ventana Movimiento. Las opciones más utilizadas están también disponibles en el menú de configuración rápida.

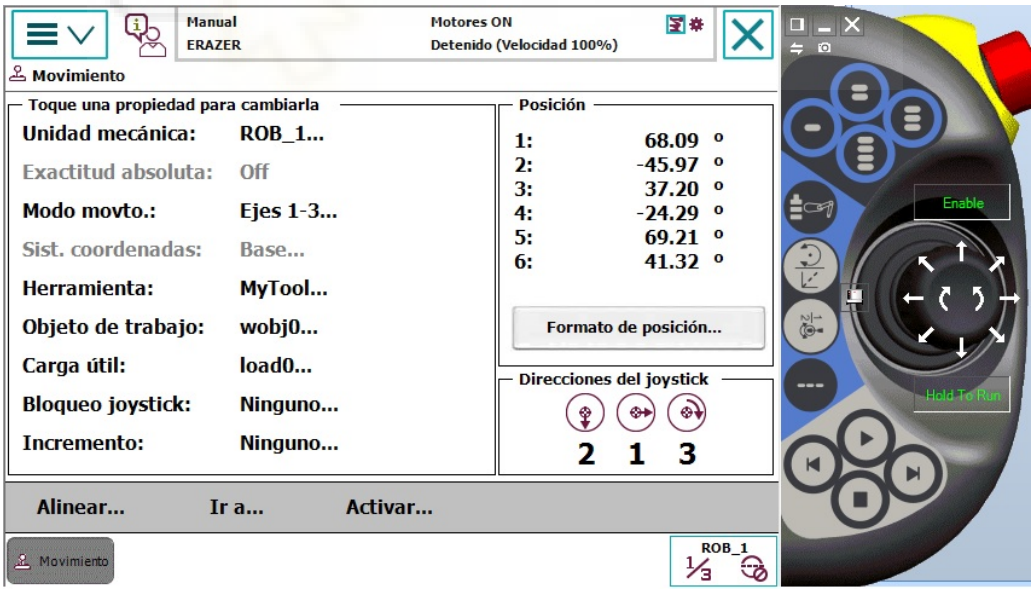


Figura 156: Movimiento en FlexPendant.

- **Ventana de producción**: La ventana de producción se utiliza para el código del programa mientras éste se está ejecutando. Podemos cargar un nuevo programa, mover el puntero a la rutina Main.

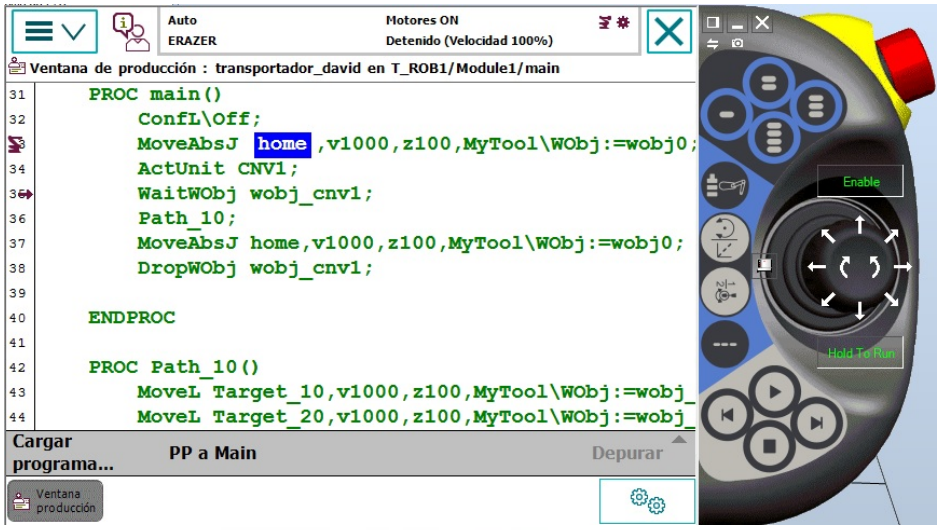


Figura 157: Ventana producción en FlexPendant.

- **Editor de programas**: El Editor de programas es donde se crean o modifican los programas. Puede abrir más de una ventana del Editor de programas, lo cual puede resultar útil cuando se tiene instalada la opción Multitasking. El botón Editor de programas de la barra de tareas muestra el nombre de la tarea.

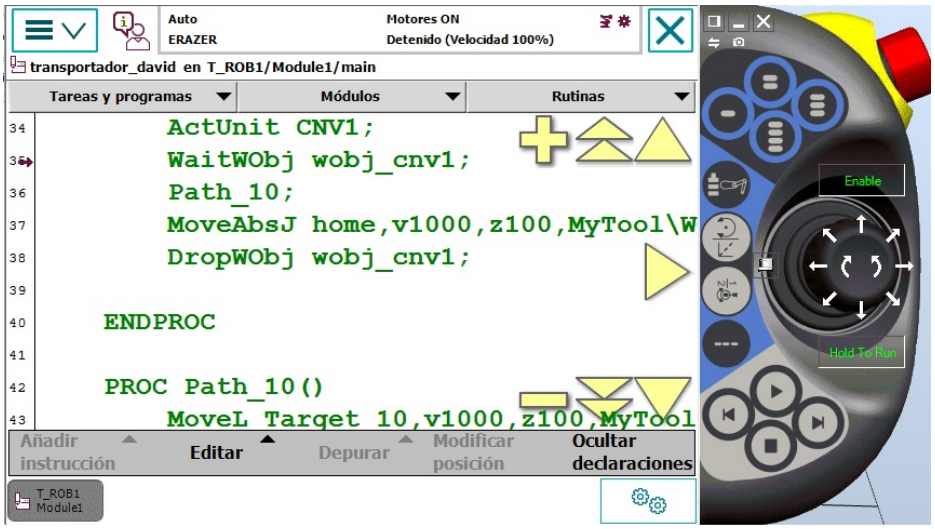


Figura 158: Editor de programas en FlexPendant.

- **Datos de programa**: contiene funciones de visualización y utilización de tipos de datos e instancias. Puede abrir más de una ventana Datos de programa, algo que puede resultar útil cuando se trabaja con muchas instancias o tipos de datos.

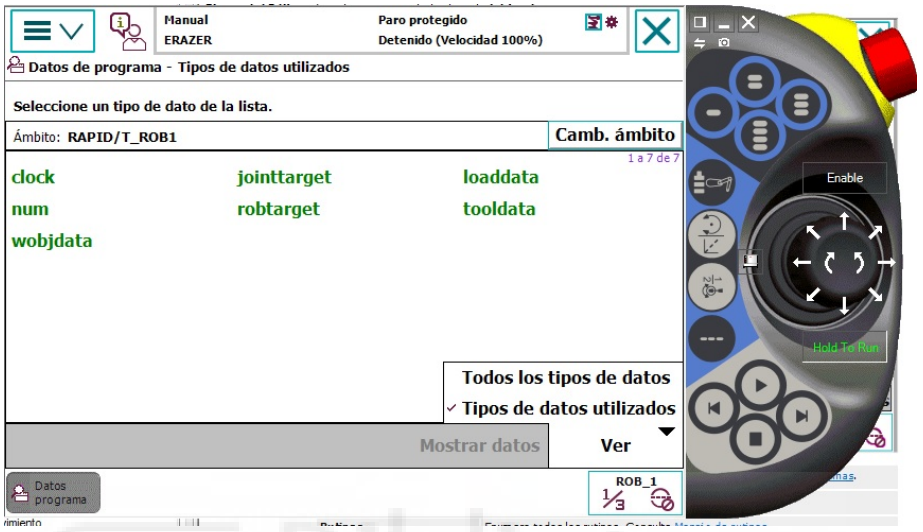


Figura 159: Datos de programa en FlexPendant.

- **Copia de seguridad y restauración**: se usa para realizar copias de seguridad y restaurar el sistema.

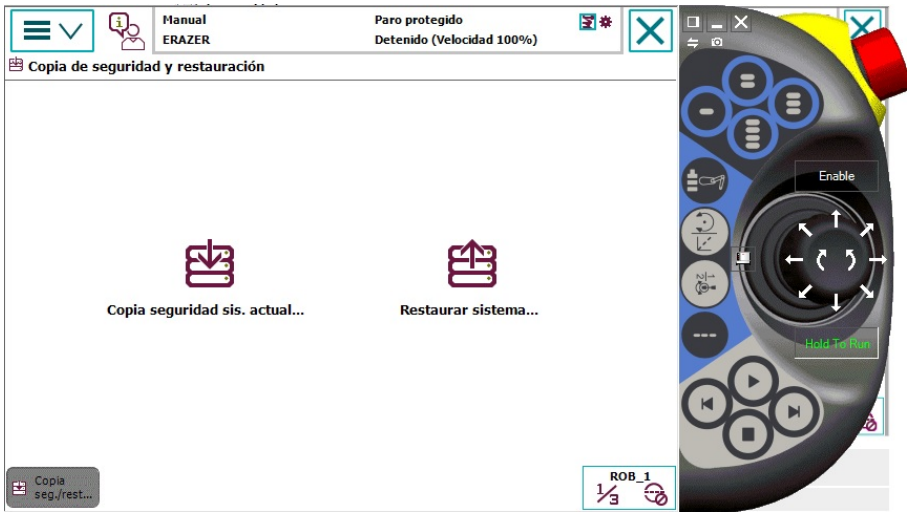


Figura 160: Copia de seguridad y restaraución en FlexPendant.

- **Calibración**: se utiliza para calibrar las unidades mecánicas del sistema de robot.

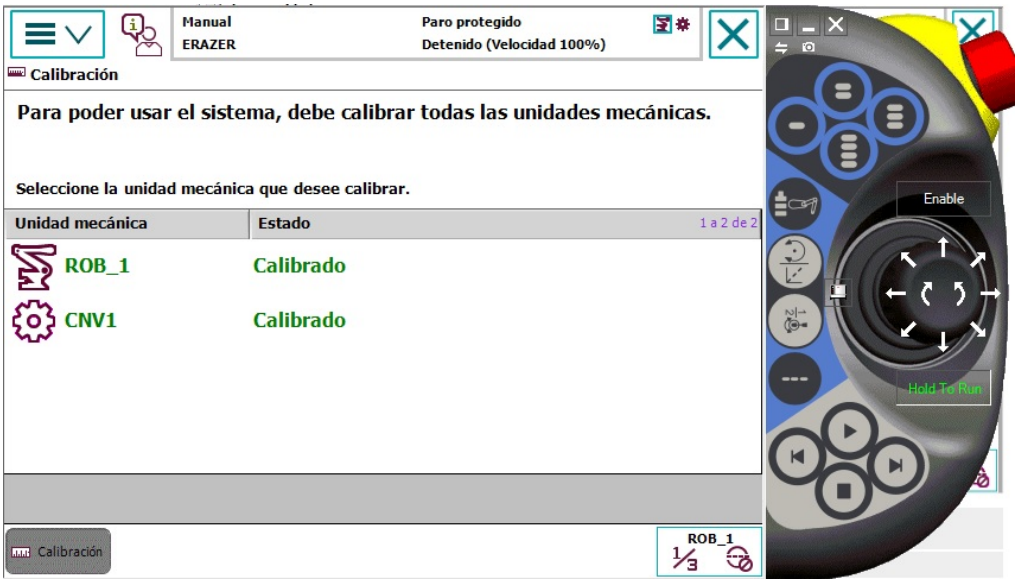


Figura 161: Calibración en FlexPendant.

- **Panel de control**: contiene funciones que permiten personalizar el sistema de robot y el FlexPendant.

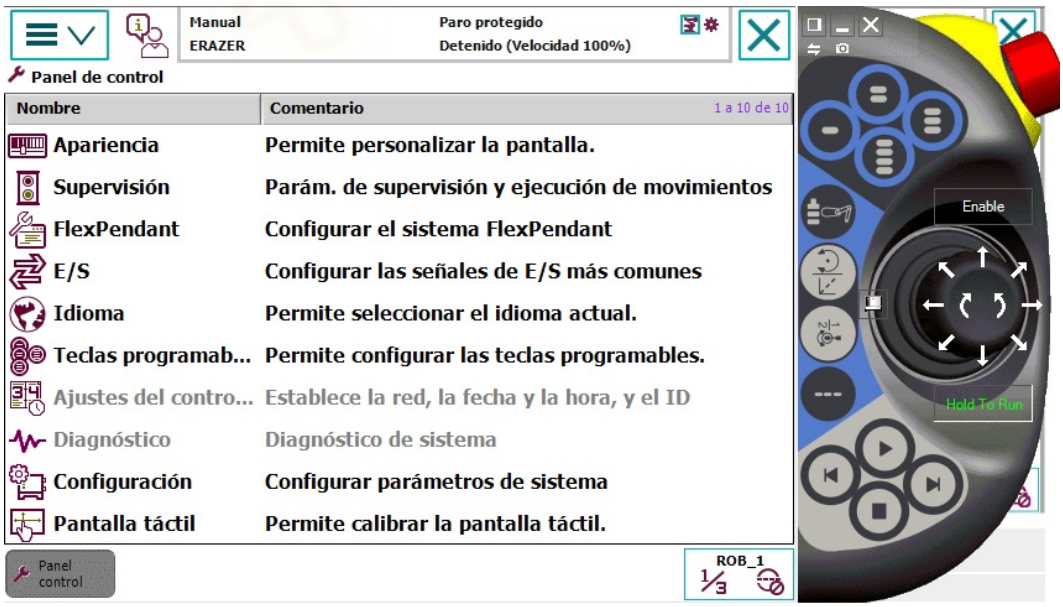


Figura 162. Panel de control en FlexPendant.

- **Registro de eventos**: almacena información acerca de los eventos que han tenido lugar.

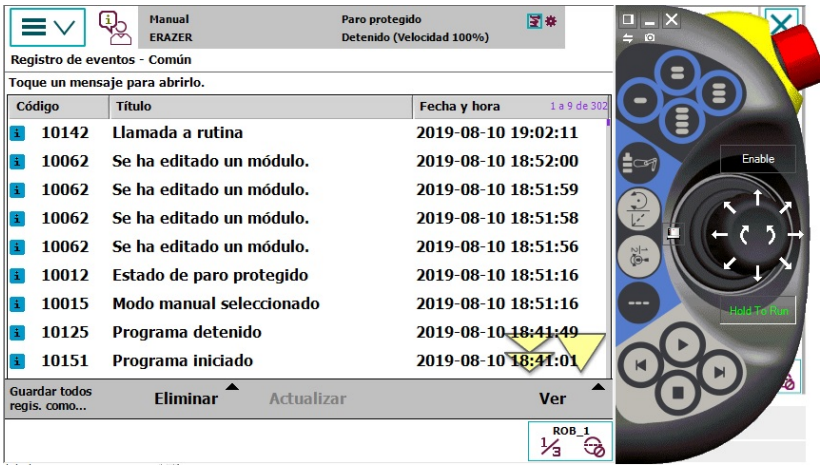


Figura 163: Registro de eventos en FlexPendant.

- **FlexPendant Explorer**: es un administrador de archivos, similar al Explorador de Windows, que permite ver el sistema de archivos del controlador.

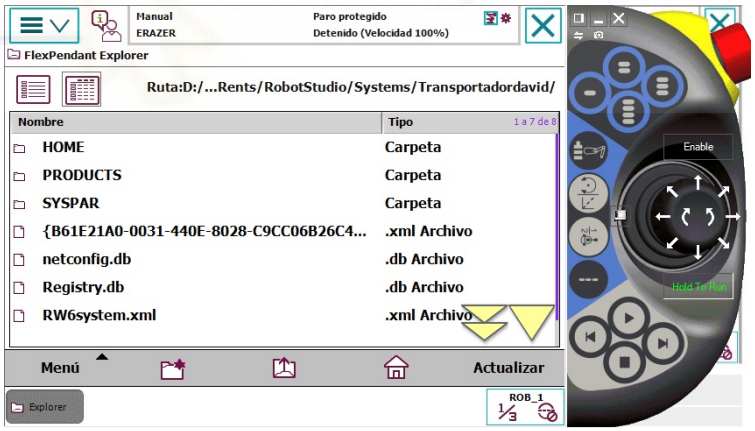


Figura 164: FlexPendan Explorer.

- **Información del sistema**: se muestra información acerca del controlador y el sistema cargado como la versión de RobotWare, las opciones en uso actualmente, las claves actuales de los módulos de control y Drive Modules, las conexiones de red, etc.

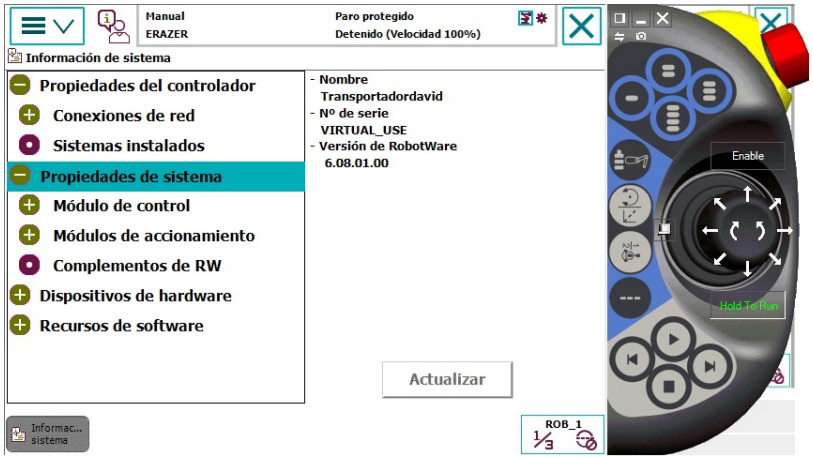


Figura 165: Información del sistema en FlexPendant.

- **Etc**.

El menú de "**Configuración rápida**" ofrece una forma más rápida de cambiar, entre otras cosas, las propiedades del movimiento, en lugar de usar la vista Movimiento.

Cada botón del menú muestra el valor de propiedad o ajuste seleccionado actualmente.

En el modo manual, el botón del menú "**Configuración rápida**" muestra la unidad mecánica, el modo de movimiento y el tamaño de incremento seleccionados actualmente.

Figura 166: Menu configuración rápida en FlexPendant.

5. DISEÑO E IMPLEMENTACIÓN DE LA ESTACIÓN DE SOLDADURA DE CHASIS PARA MAQUINARIA.

5.1. Determinación de las necesidades de la empresa.

La implantación de la estación de soldadura robotizada en la empresa parte de la necesidad de mejorar la productividad y la calidad de la estación de soldadura manual de que dispone actualmente, por lo que el diseño de ésta se ajustará, en principio, al tipo de producto que la empresa fabrica.

La estación se empleará en la soldadura MIG de chasis destinados a maquinaria siendo las dimensiones máximas previstas por la empresa de 2500 mm de largo, 1500 mm de ancho y 500 mm de alto. El peso máximo previsto de los chasis es de 400 kgf.

En la industria existen ya estaciones de soldadura robotizadas destinadas a trabajos similares y suficientemente probadas. La estación de soldadura constará de un robot de seis ejes sobre un track que atenderá a dos posicionadores giratorios donde se dispondrán los chasis a soldar. De esta forma, mientras en un posicionador el robot está soldando, en el otro los operarios están preparando el siguiente chasis a soldar y así sucesivamente.

La alcanzabilidad del robot se ve por tanto muy incrementada gracias al track y los posicionadores.

En cuanto al tipo de soldadura a realizar será de tipo MIG (Metal Inert GAS) también denominada GMAW (Gas Metal Arc Welding), que es el tipo de soldadura que actualmente se emplea en la empresa y en la que tiene una amplia experiencia.

5.2. Equipos de la estación.

5.2.1. Selección de los posicionadores.

Dada la forma de los chasis a soldar y del trabajo a realizar se selecciona el posicionador de ABB IRBP L-600 con las siguientes características:

- **Especificaciones**:

Tabla 1: Especificaciones IRBP-L-600.

Modelo	Capacidad de carga (kg)	Par continua máx. (Nm)	Par máx. de flexión (Nm)	Precisión repetitiva (r=500)	Velocidad máx. de rotación (º/s)
IRBP L-600	600	650	3300	+/-0.05	150

- **Medidas (mm)**:

Tabla 2: Medidas del IRBP-L-600

Modelo	A	B	ØC	D
IRBP L-600	2500	950	1500	3432

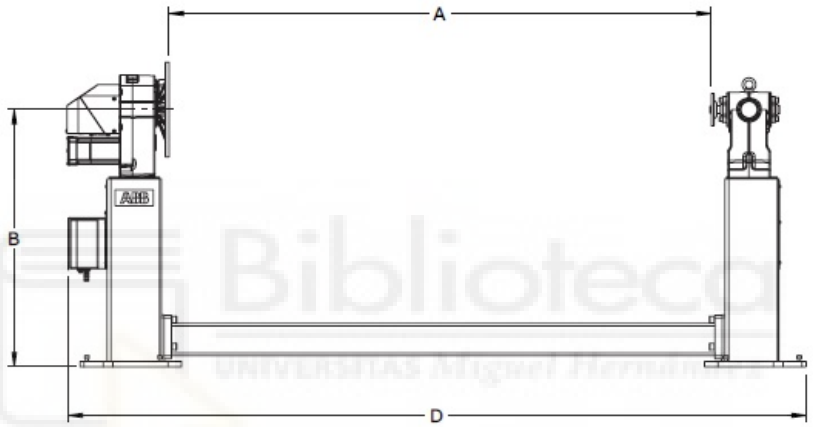


Figura 167: Dimensiones IRBP-L I.

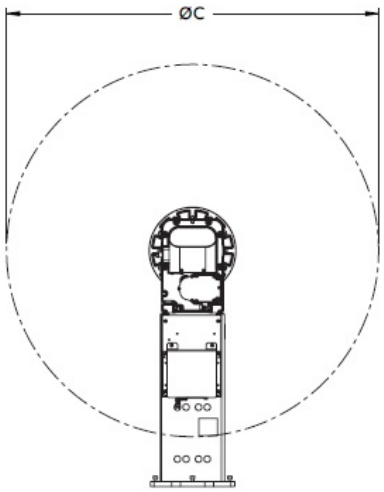


Figura 168: Dimensiones IRBP-L II.

5.2.2. Selección del robot y del track.

La selección del robot y el track se realiza teniendo en cuenta los siguientes condicionantes:

- Espacio disponible.
- Emplazamiento de los posicionadores, track y robot en el espacio de trabajo teniendo en cuenta las distancias de seguridad a otros equipos, paredes, vallas de seguridad, etc.
- Alcanzabilidad del robot.

El robot seleccionado para la estación de trabajo es el modelo de ABB IRB2600 de 12 kg de capacidad de carga y un alcance de 1.85 m.

Figura 169: ABB IRB2600

En la siguiente figura podemos ver la alcanzabilidad del robot en una posición sobre el track.

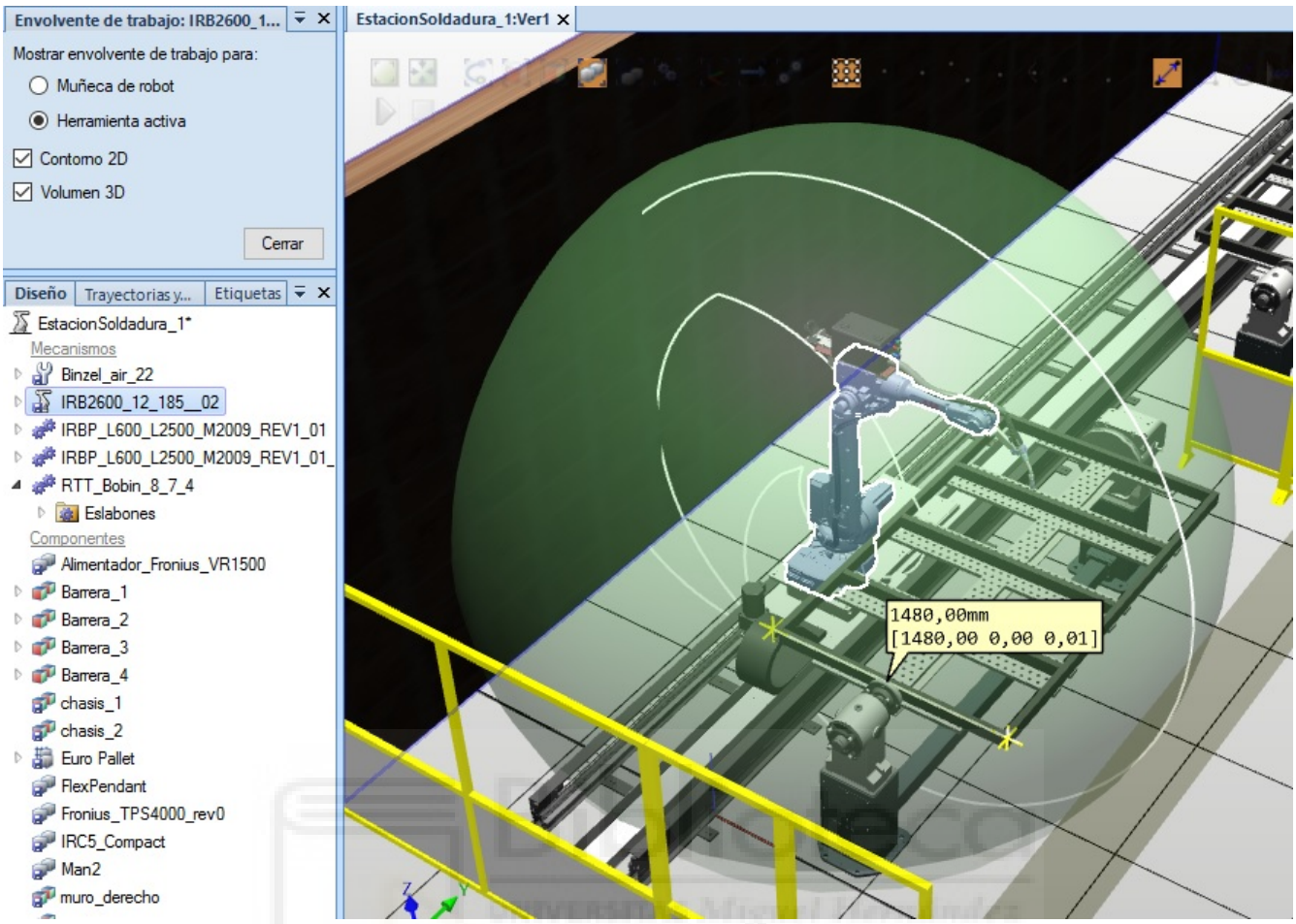


Figura 170: Envolvente de trabajo del IRB2600 sobre el track.

El track seleccionado es el modelo de ABB RTT_Bobin de 8.7 m

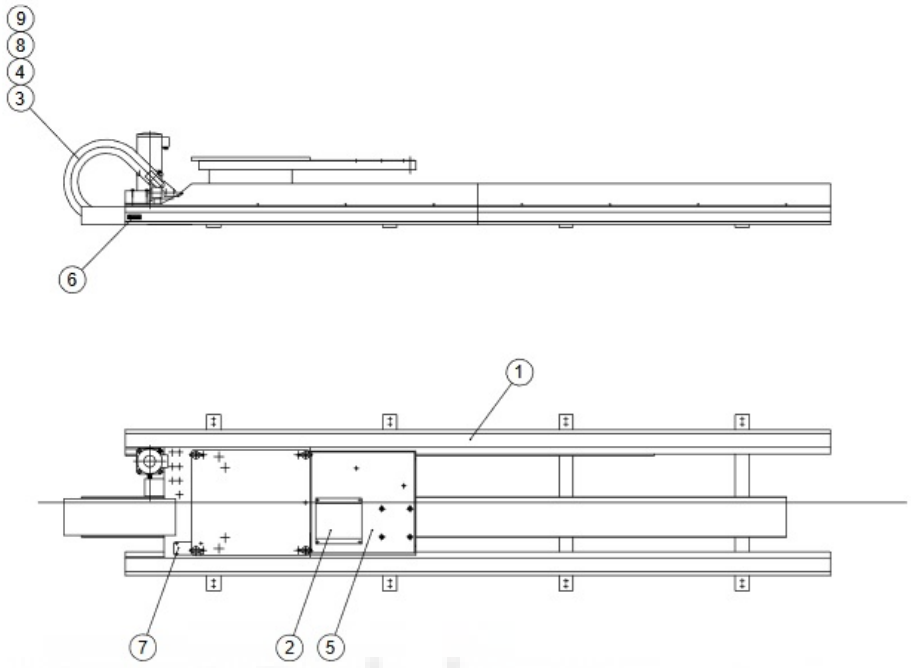


Figura 2 Track de desplazamiento con bobina

Pos.	Descripción	Pos.	Descripción
1	Track de desplazamiento RTT con bobina	6	Placa de datos de servicio
2	Juego de cables internos	7	Indicación de estación
3	Juego de empalmes	8	Cable del motor
4	Juego de cables	9	Cable del resolucionador
5	Soporte para TC92		

Figura 171: Track ABB RTT Bobin.

5.2.3. Selección del equipo de soldadura.

El equipo de soldadura MIG consta de los siguientes elementos:

- Equipo de fuente de energía y control Fronius TPS 4000.

Figura 172: Fronius TPS 4000.

- Alimentador de hilo Fronius VR 1500.

Figura 173: Fronius VR 1500.

- Antorcha Binzel Air-22.

Figura 174: Binzel Air 22.

5.3. Diseño del espacio de trabajo y seguridad.

La disposición de la estación está limitada por el espacio disponible en la empresa, ya que se trata de una automatización robotizada de una estación de soldadura ya existente, por tanto, se adaptará al espacio existente pero manteniendo las medidas de seguridad necesarias.

Para la extracción de los humos de soldadura se dispone ya de una instalación que se aprovechara.

En el diseño de la estación de soldadura se ha tenido en cuenta la siguiente normativa de seguridad

- Ley de prevención de riesgos laborales 31/1995 de 8 de noviembre.
- Real Decreto 485/1997, de 14 de abril, sobre disposiciones mínimas de señalización de seguridad y salud en el trabajo.
- Real Decreto 486/1997, de 14 de abril, por el que se establecen las disposiciones mínimas de seguridad y salud en los puestos de trabajo.
- Real Decreto 487/1997, de 14 de abril, sobre disposiciones mínimas de seguridad y salud relativas a la manipulación manual de cargas que comporten riesgos, en particular dorsolumbares, para los trabajadores.

- Real Decreto 773/1997, de 30 de mayo, sobre disposiciones mínimas de seguridad y salud relativas a la utilización por los trabajadores de equipos de protección individual.
- Real Decreto 1215/1997, de 18 de julio, por el que se establecen las disposiciones mínimas de seguridad y salud para la utilización por los trabajadores de los equipos de trabajo.
- Real Decreto 374/2001, de 6 de abril, sobre la protección de la salud y seguridad de los trabajadores contra los riesgos relacionados con los agentes químicos durante el trabajo.
- Real Decreto 614/2001, de 8 de junio, sobre disposiciones mínimas para la protección de la salud y seguridad de los trabajadores frente al riesgo eléctrico.
- Real Decreto 286/2006, de 10 de marzo, sobre la protección de la salud y la seguridad de los trabajadores contra los riesgos relacionados con la exposición al ruido.
- Real Decreto 486/2010, de 23 de abril, sobre la protección de la salud y la seguridad de los trabajadores contra los riesgos relacionados con la exposición a radiaciones ópticas artificiales.

Se instalan vallas perimetrales de seguridad así como dos barreras de seguridad fotoeléctricas coordinadas con el robot de forma que si se produce una intrusión en la celda durante el trabajo el robot parará inmediatamente.

5.4. Programación de la estación en RobotStudio.

5.4.1 Creación de la estación.

Una vez que ya tenemos diseñada la estación en cuanto al robot, track, posicionadores, elementos de seguridad, equipos auxiliares y la disposición en planta de estos, el primer paso en RobotStudio es la creación de la estación. En este caso, crearemos en primer lugar una estación vacía en RobotStudio con la versión RobotWare 6.08.01, guardaremos la estación y le pondremos nombre.

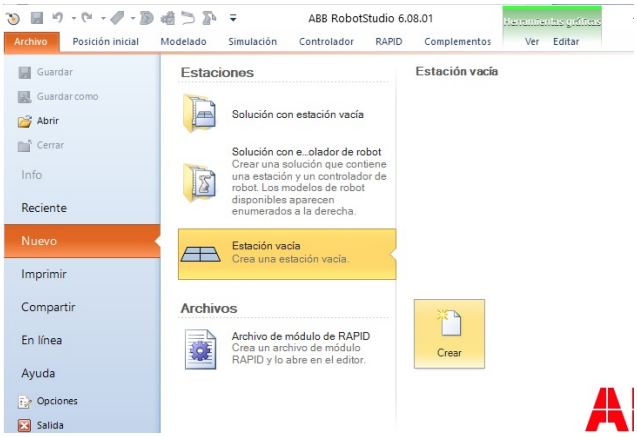


Figura 175: Creación de la estación vacia.

Podemos observar en la siguiente figura el sistema de coordenadas mundo de la estación (RS-WCS).

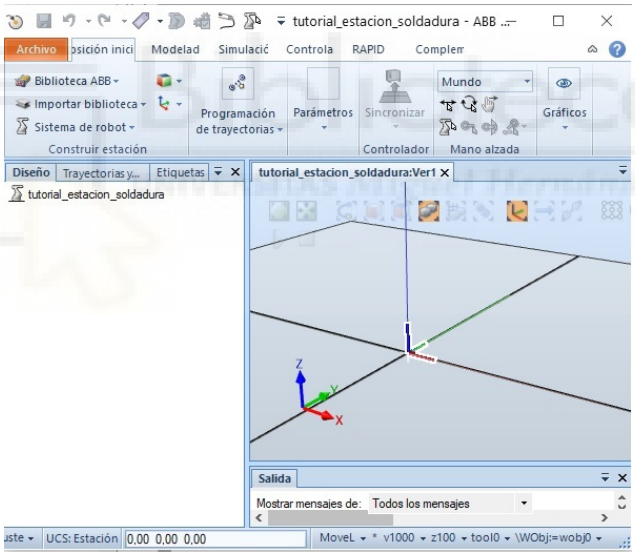


Figura 176: Sistema de coordenadas mundo RS-WCS de RobotStudio

5.4.2 Inserción de los componentes de la estación.

En este paso, insertaremos el robot, el track, los posicionadores y todos aquellos objetos de los que dispongamos en la biblioteca de ABB o disponibles en RobotApps, que nos permitan crear el modelo virtual de la estación. Otros objetos tales como útiles y piezas se crearán en el módulo de modelado de RobotStudio o en un software externo y se importarán a la estación como se explica en apartados posteriores.

Para importar el robot, el track y los posicionadores lo haremos desde “Biblioteca ABB” en la pestaña “Posición inicial” de la cinta superior. En concreto importaremos los siguientes equipos:

- Un Robot IRB 2600 de alcance 1.85 m y 12 kg de carga útil.

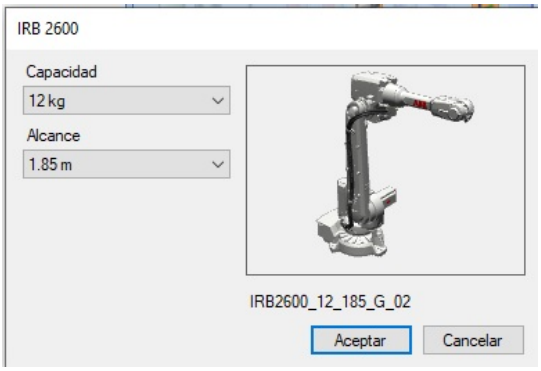


Figura 177: Importación del robot de la estación de soldadura.

- Un Track RTT Bobin de 8.7 m.

Figura 178: Importación del track de la estación de soldadura

- Dos posicionadores IRBP L de 600 kg de capacidad y longitud 2500 mm.

Figura 179: Importación de los posicionadores de la estación de soldadura.

El resto de elementos tales como vallas, mesas, etc. los importaremos desde "**Importar biblioteca**" en la pestaña "**Posición inicial**" de la cinta superior.

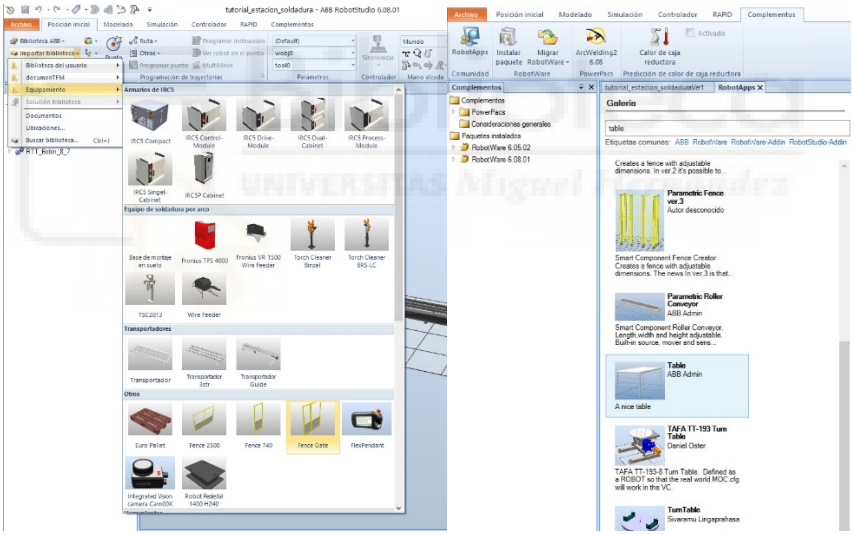


Figura 180: Importación de otros equipos de la estación de soldadura.

Para obtener otros objetos de biblioteca ABB de la comunidad, desde la pestaña "**Complementos**" de la cinta superior podemos descargarnos el archivo de biblioteca a nuestro ordenador y posteriormente desde "**Importar biblioteca**" en la pestaña "**Posición inicial**" de la cinta superior insertarlos en la estación.

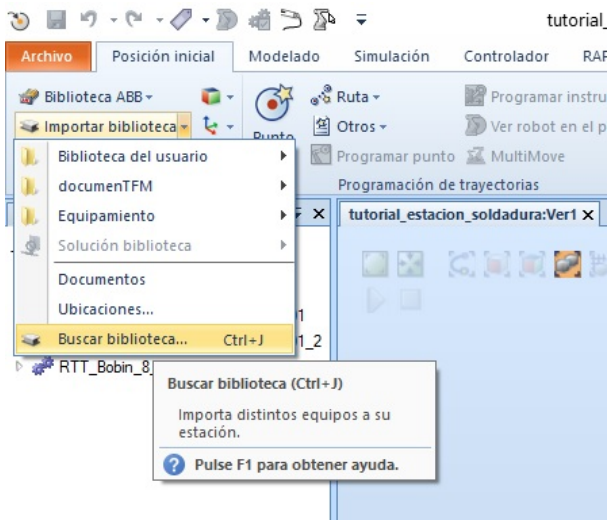


Figura 181: Buscar bibliotecas externas.

Una vez insertados en la estación todos los elementos necesarios, debemos situarlos en su posición. Hacemos clic con el botón derecho del ratón sobre el objeto en el árbol de la izquierda y en la ventana que aparece seleccionamos "**Posición**" que nos permite:

- **Fijar la posición**: Permite situar el objeto respecto del sistema de coordenadas seleccionado. Podemos escribir las coordenadas y orientación a mano o usar las herramientas de ayuda gráfica para seleccionar el punto hasta donde queremos desplazar el objeto (pinchar primero dentro del campo "X" de la ventana y luego marcar el punto de destino para que adquiera las coordenadas.

Figura 182: Fijar la posición de un equipo.

- **Posición de offset**…: Similar al caso anterior, pero en este caso, fijamos una translación y/o rotación respecto del sistema de referencia seleccionado.

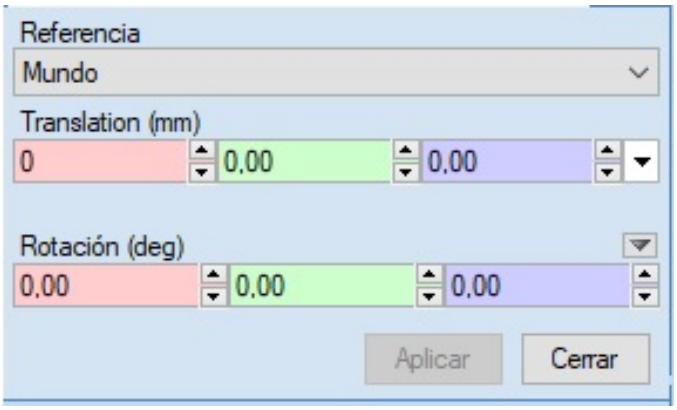


Figura 183: Definir una posición de offset de un equipo.

- **Girar**.: Permite girar el objeto un ángulo determinado respecto a uno o varios ejes (X, Y, Z) en el sistema de referencia elegido.

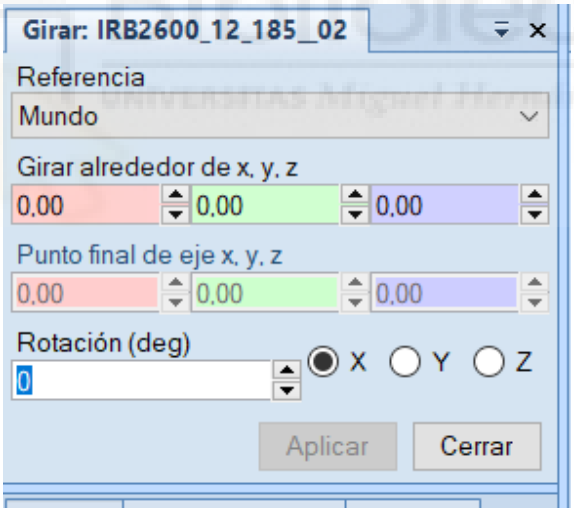


Figura 184: Girar un equipo.

- **Situar**: Dispone de varias opciones, la primera "**Un punto**", permite designar un punto origen y un punto destino y seleccionar los ejes en los que realizar la traslación. La opción "**Dos puntos**" nos permite mover objetos según la relación entre dos planos re. Con "**Tres puntos**" podemos mover el objeto designando tres puntos (el punto origen, un punto en el eje X, un punto en el eje Y) para el punto de

partida y para el punto de destino. “**Base de coordenadas**” permite mover un objeto desde el sistema de referencia actual al de destino manteniendo su posición relativa. ”**Dos bases de coordenadas**” permite designar lel sistema de coordenada origen y el de destino.

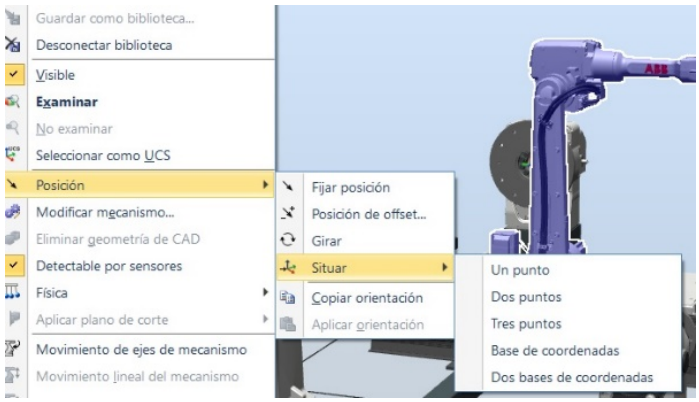


Figura 185: Situar un objeto.

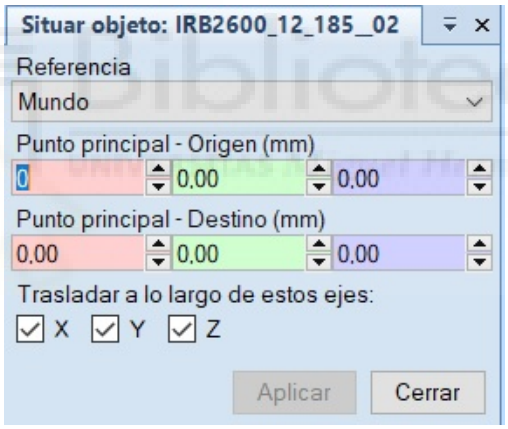


Figura 186: Situar un objeto mediante un punto..

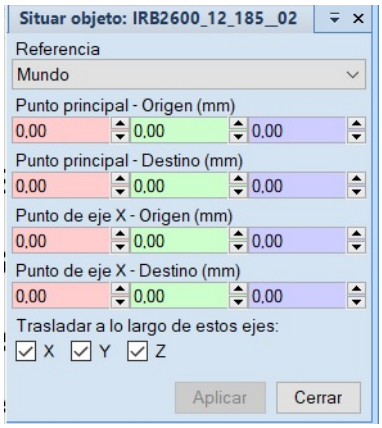


Figura 187: Situar un objeto mediante dos puntos.

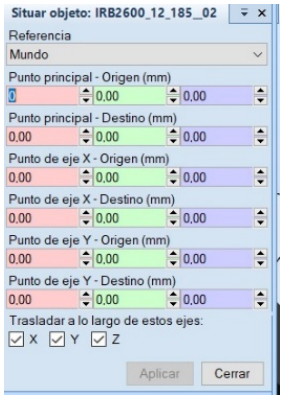


Figura 188: Situar un objeto mediante tres puntos.

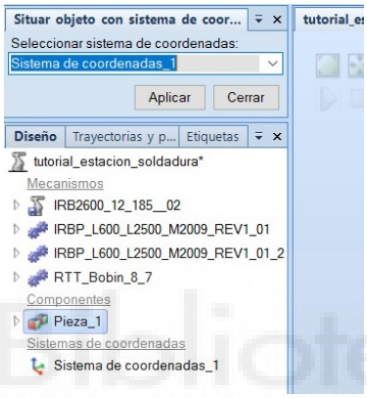


Figura 189: Situar objeto con una base de coordenadas.

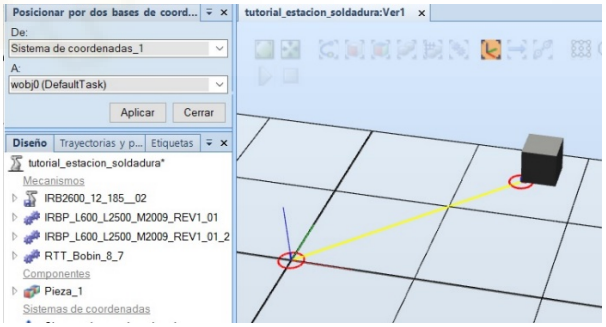


Figura 190: Situar objeto con dos bases de coordenadas

De todas la posibilidades que nos permite RobotStudio para situar objetos, utilizaremos "**Fijar posición**" y luego "**Girar**" si es necesario para situar nuestros objetos en la estación.

Hay que prestar atención a la hora de desplazar un objeto a su sistema de coordenada local, ya que es el que desplazaremos, por lo que debemos basarnos en el para determinar los puntos donde situaremos los objetos. Si nos interesa tener el sistema de coordenadas local del objeto en otra posición y/o orientación, RobotStudio nos permite cambiarlo. Haciendo

clic con el botón derecho del ratón sobre el objeto en el árbol de la izquierda y seleccionado la opción “**Modificar**”<<”**Establecer origen local**” accedemos a la ventana para designar la nueva posición y/u orientación del sistema de coordenadas local del objeto.

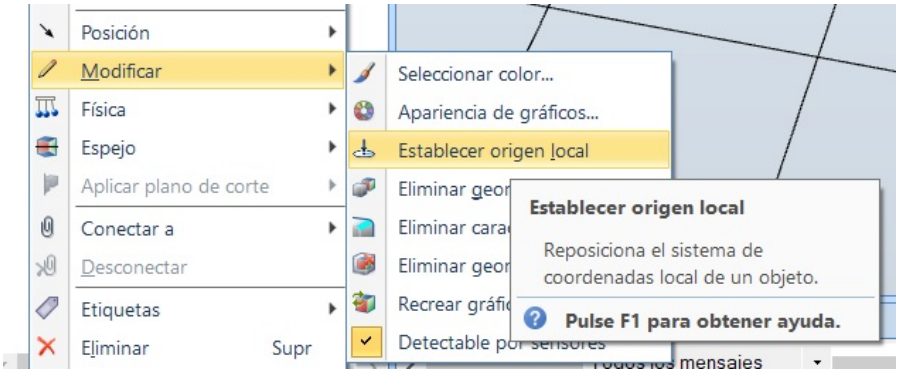


Figura 191: Establecer origen local de una pieza I.

Figura 192: Establecer origen local de una pieza II.

Para simular el alimentador de hilo del equipo de soldadura que está situado sobre el robot y debe moverse solidariamente con el, importamos el objeto a la estación, lo seleccionamos haciendo clic sobre el con el botón derecho del ratón y seleccionamos “Conectar a”, en la ventana flotante vamos explorando los link disponibles hasta que vemos que se selecciona el que nos interesa, en esta caso el “**link 3**” de nuestro robot. Posteriormente podemos mover el objeto para posicionarlo en la forma deseada, pero ya está anclado al sistema de coordenadas local del link del robot y se moverá solidariamente con el.

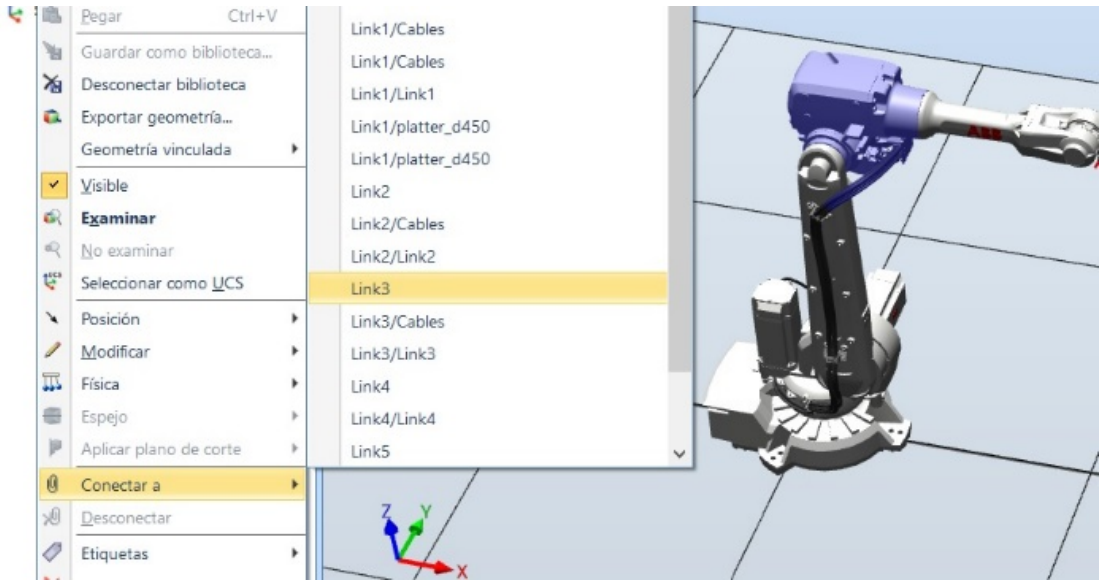


Figura 193: Conectar el alimentador de hilo al robot I.

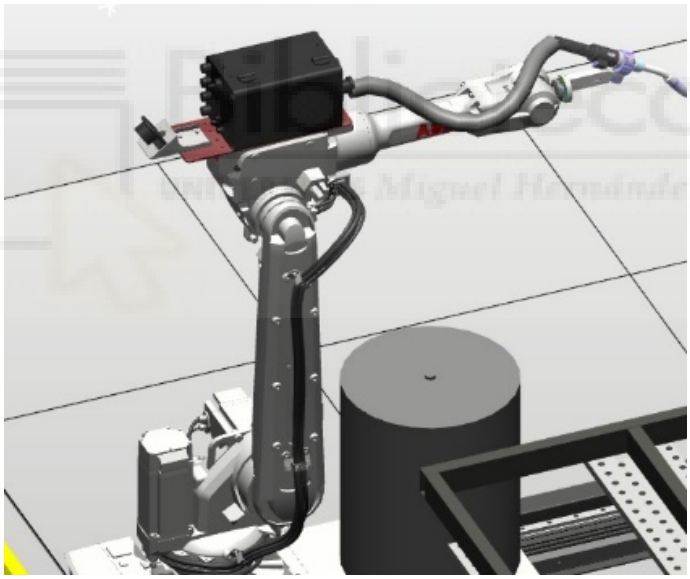


Figura 194: Conectar el alimentador de hilo al robot II.

De forma similar situaremos el bidón de hilo de soldadura sobre el track y los útiles y chasis sobre cada posicionador. En este caso deberemos asegurarnos que el origen local de cada objeto está situado en la posición y orientación necesaria para que al conectarse al posicionador quede colocado en la posición correcta

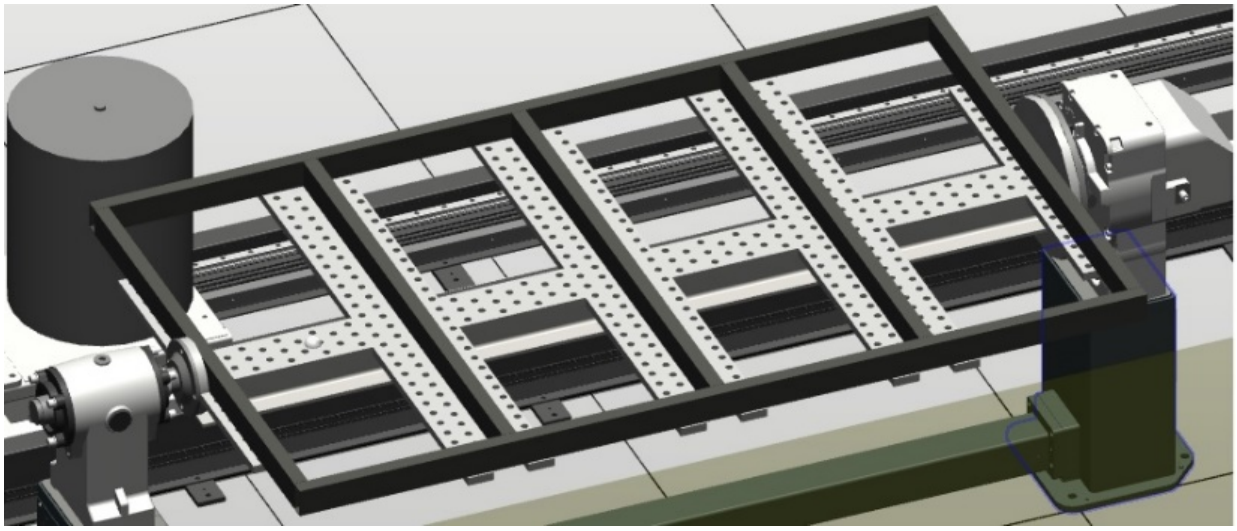


Figura 195: Conectar el útil y el chasis al posicionador I.

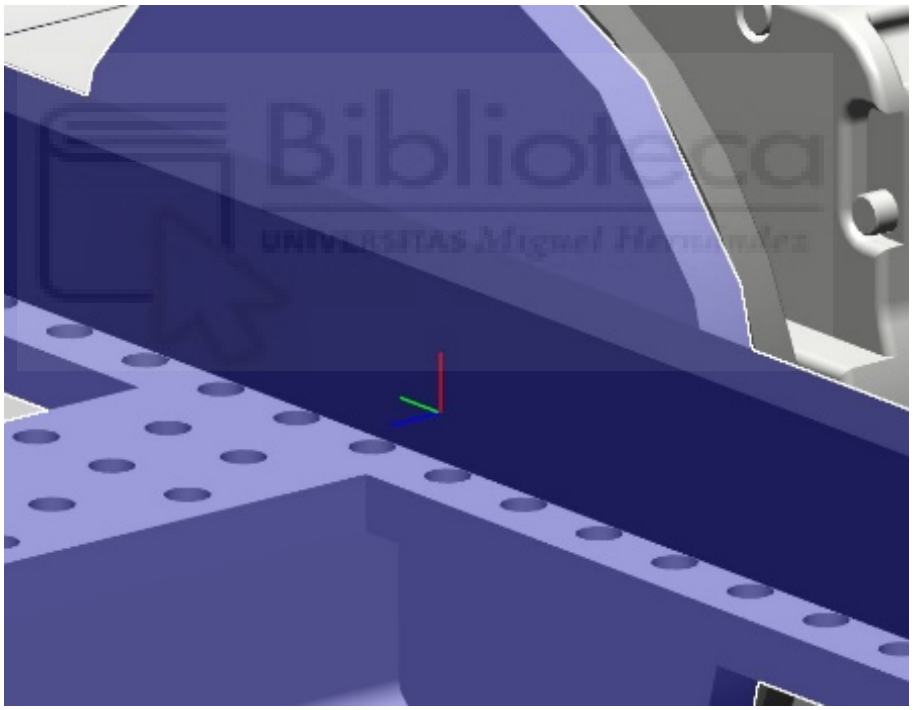


Figura 196: Conectar el útil y el chasis al posicionador II.

Tras insertar todos los objetos, la vista en planta de la estación de soldadura quedaría de la siguiente forma.

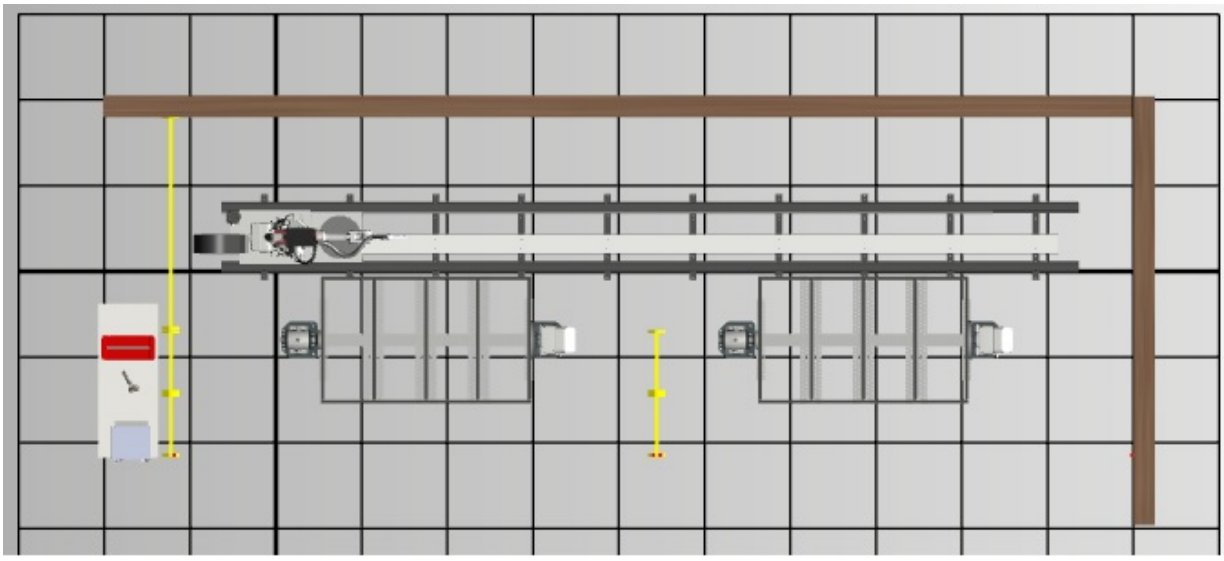


Figura 197: vista en planta de la estación de soldadura.

El posicionamiento del robot sobre el track se explicara en apartados posteriores.

5.4.3 Creación del sistema.

Una vez insertados el robot, el track y los posicionadores en la estación crearemos el sistema, de esta forma, RobotStudio nos seleccionará automáticamente las opciones necesarias para su manejo en el controlador y podremos incluir aquellas otras opciones que deseemos manualmente.

Crearemos el sistema mediante la opción "**Sistema de robot**"<<"**Desde diseño**" en la pestaña "**Posición inicial**" de la cinta superior. Como opción manual añadiremos la opción de "Español" en el idioma del sistema.

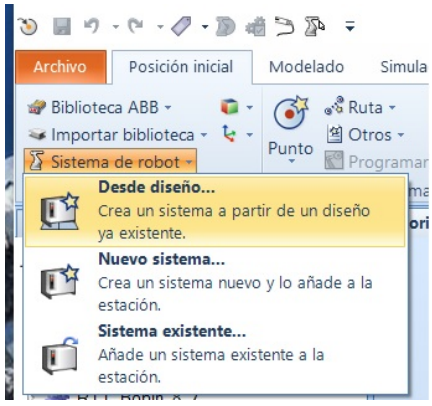


Figura 198: Creación desl sistema de la estación de soldadura I.

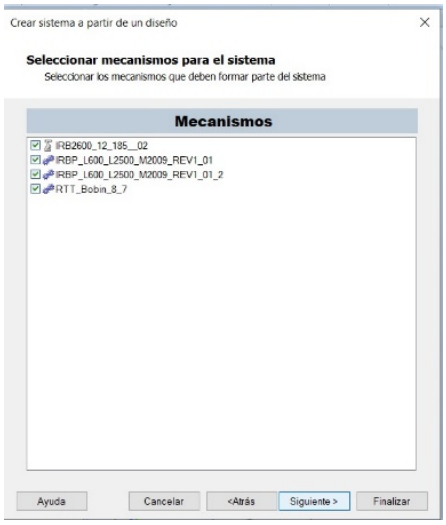


Figura 199: Creación desl sistema de la estación de soldadura II.

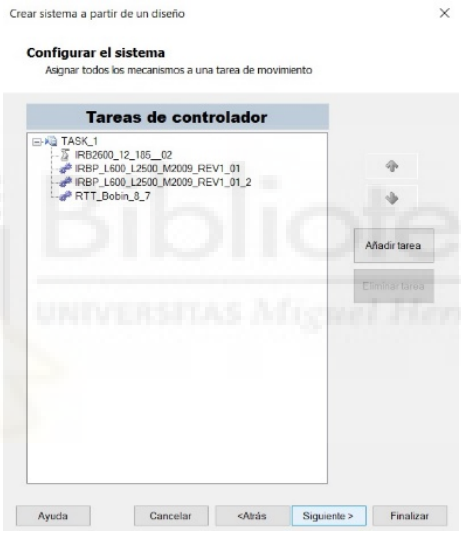


Figura 200: Creación desl sistema de la estación de soldadura III.

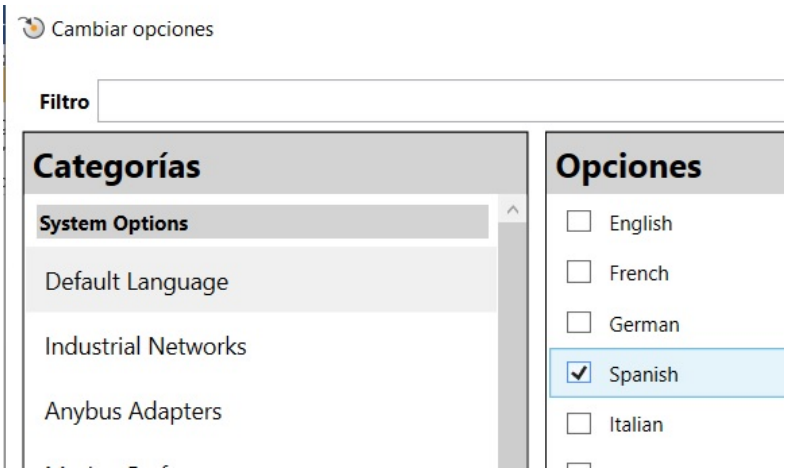


Figura 201: Creación desl sistema de la estación de soldadura IV.

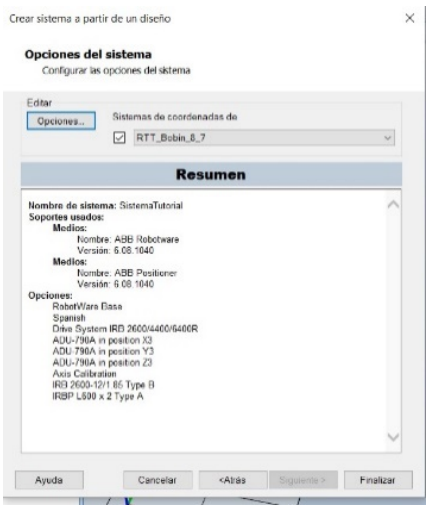


Figura 202: Creación desl sistema de la estación de soldadura V.

5.4.4 Integración del track y los posicionadores.

Al crear el sistema, el track y los posicionadores funcionarán como ejes externos en el controlador del robot, de forma que las posiciones del track y los posicionadores se almacenarán en cada objetivo cuando lo programemos. El controlador moverá tanto los ejes del propio robot como los del track y posicionadores cuando se mueva de un objetivo al siguiente si es necesario para alcanzar la configuración final.

Para colocar el robot sobre el track, seleccionaremos el robot en la pestaña “Diseño” del árbol de la izquierda y lo arrastraremos sobre el track. RobotStudio nos pregunta si queremos actualizar la posición del robot, si debe estar coordinado el robot con el track y si deseamos reiniciar el controlador. Respondemos si a todas.

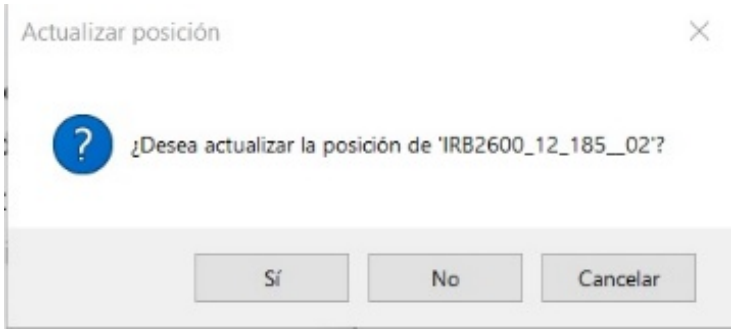


Figura 203: Integración del robot en el track I.

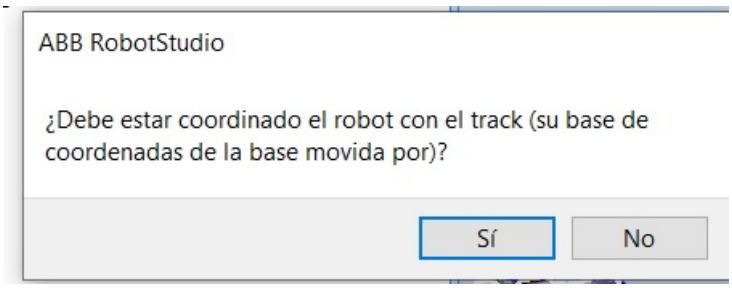


Figura 204: Integración del robot en el track II.

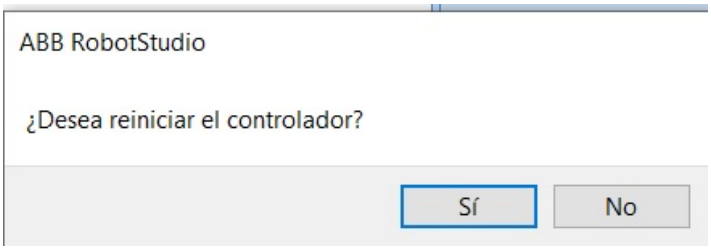


Figura 205: Integración del robot en el track III.

El robot queda posicionado y coordinado con el track.

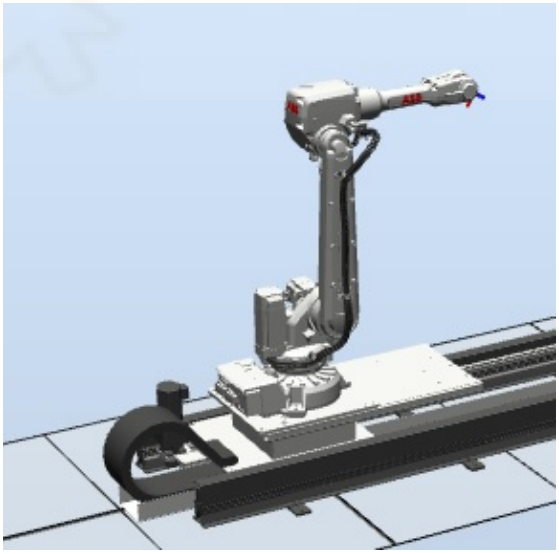


Figura 206: Integración del robot en el track IV.

Para mover los ejes del robot, track o los posicionadores hacemos clic con el botón derecho del ratón sobre el robot en el árbol de la izquierda y seleccionamos “Movimiento de ejes

de mecanismo", en la ventana que aparece podemos cambiar la posición individual de cada eje del robot y la del track y posicionadores de la estación. Al mover el track comprobaremos que el robot se mueve con él.

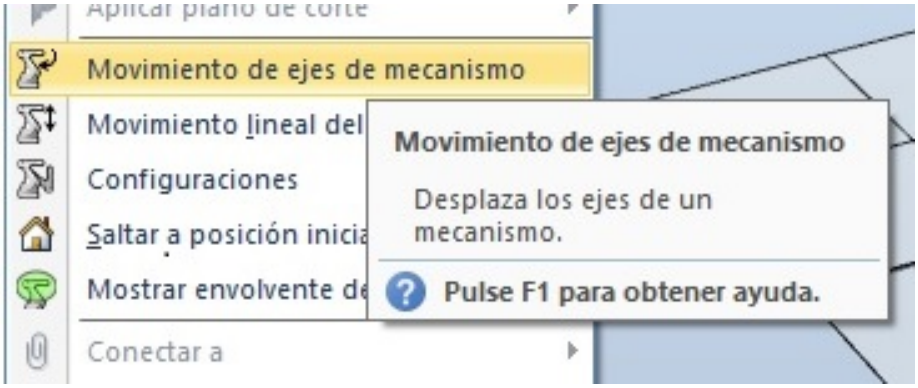


Figura 207: Movimiento de ejes de los mecanismos de la estación I.

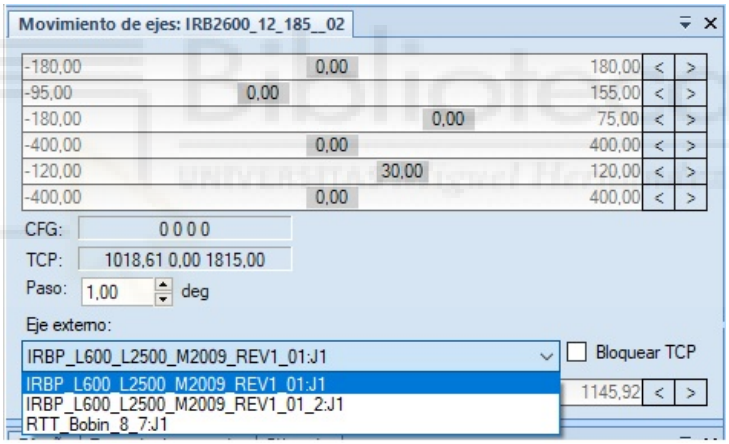


Figura 208: Movimiento de ejes de los mecanismos de la estación II.

Sin embargo, es necesario activar los mecanismos posicionadores para poder moverlos. Para activar el posicionador a mover, hacemos clic sobre "Activar unidades mecánicas" en la pestaña "Simulación" de la cinta superior y en la ventana que aparece seleccionamos el posicionador que queremos mover.

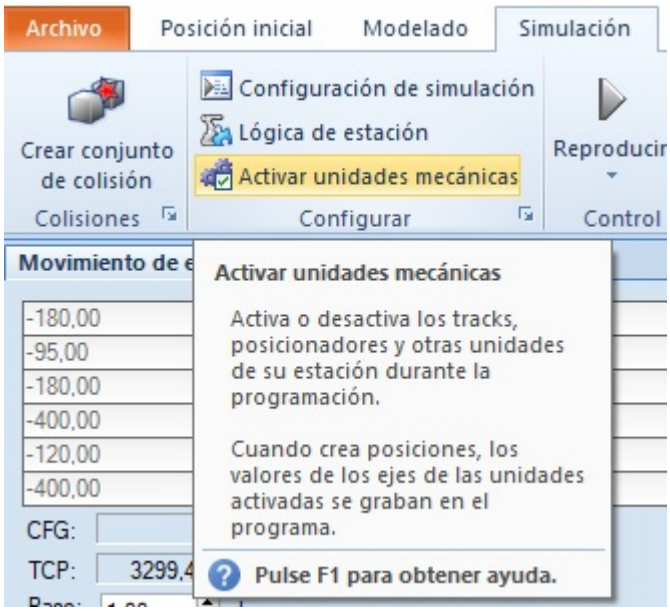


Figura 209: Activar unidades mecánicas.

Las unidades mecánicas de los posicionadores se nombran con STN1, STN2 y así sucesivamente. Podemos saber la relación entre el nombre del posicionador y su STN haciendo clic sobre “Editar sistema” en la pestaña “Controlador”.

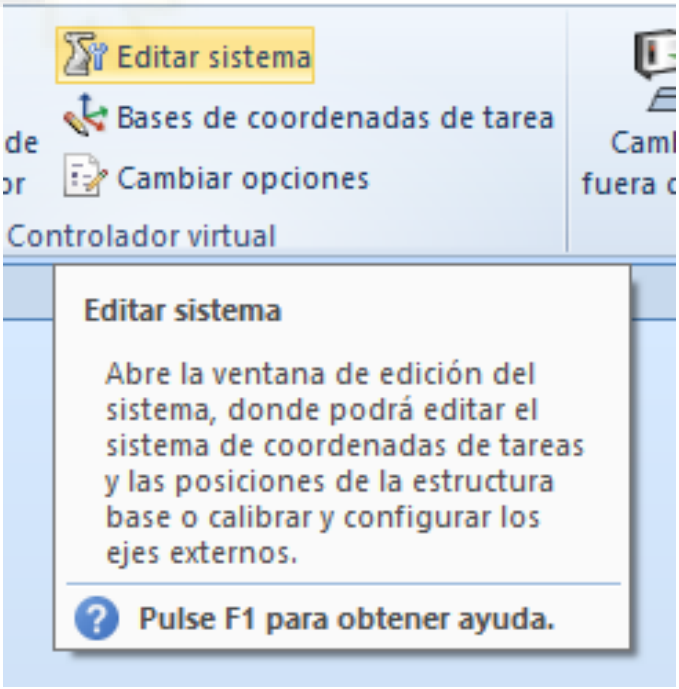


Figura 210: Editar sistema I.

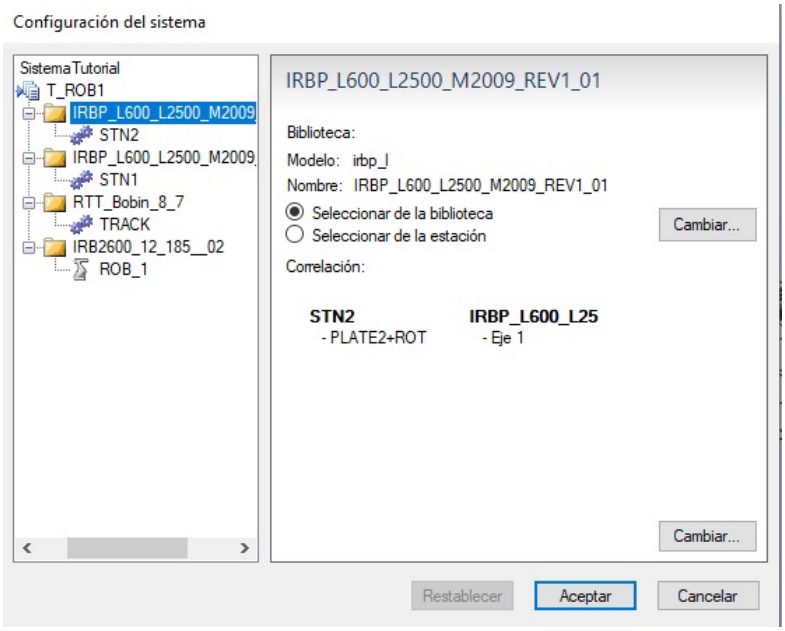


Figura 211: Editar sistema II.

Cuando deseemos que la pose del posicionador quede almacenada en el objetivo programado de forma que el posicionador se mueva a la pose necesaria para que el robot pueda alcanzar el objetivo, es necesario activar la unidad antes.

5.4.5 Creación de los útiles y piezas de trabajo.

Para sujetar las piezas a trabajar en los posicionadores, es necesario el diseño de un útil que sostenga y sujete en una posición determinada. El diseño del útil y las sujeciones puede suponer un tema bastante complejo, ya que la exacta colocación de la pieza va a influir mucho en los tiempos de producción y la calidad del trabajo. El desvió en la posición de todos o de alguno de los elementos de la pieza provocará un trabajo defectuoso y la necesaria reprogramación de los objetivos y consiguiente pérdida de tiempo.

En la mayoría de las ocasiones los útiles se diseñan para un tipo o modelo de pieza concreto.

Para el diseño del útil y las piezas se ha utilizado el programa de diseño de sólidos 3D Autodesk Inventor con licencia de estudiante de UMH. Se ha diseñado por una parte el útil y por otra la pieza (chasis) a soldar. Posteriormente se ha realizado un ensamblaje de las dos pie-

zas para colocar el chasis sobre el útil en su posición. El ensamblaje se ha guardado en formato CAD “sat”. Para importar el ensamblaje a RobotStudio, hacemos clic sobre “Importar geometría”<<”Buscar geometría” en la pestaña “Posición inicial” de la cinta superior y buscamos el archivo CAD a importar. Una vez importado el archivo a RobotStudio podemos cambiarle el nombre y guardarlo como biblioteca haciendo clic sobre el con el botón derecho del ratón en el árbol de la izquierda.

Las geometrías son básicamente archivos de CAD que, al importarlos, se copian a la estación de RobotStudio mientras que las bibliotecas son objetos guardados desde RobotStudio como archivos externos. Guardar el archivo CAD importado como biblioteca nos permite disponer del objeto CAD para otros proyectos, además al importar una biblioteca, se crea un enlace entre la estación y el archivo de biblioteca. Por tanto, el archivo de la estación no aumenta de tamaño como ocurre al importar geometrías.

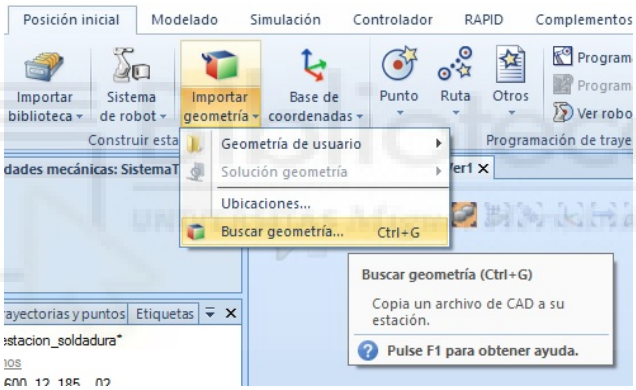


Figura 212: Importar geometría.

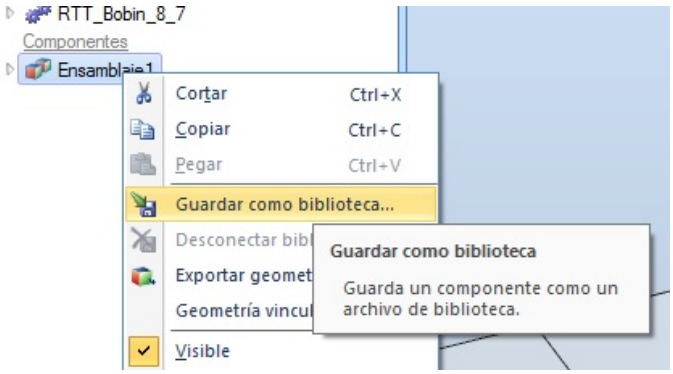


Figura 213: Guardar como biblioteca.

Es necesario que el sistema de referencia local del chasis este situado de forma que cuando se conecte al posicionador quede de la forma correcta, para ello comprobamos la posición y orientación del sistema de referencia local del posicionador seleccionándolo en el árbol de la izquierda podremos ver dicho sistema.

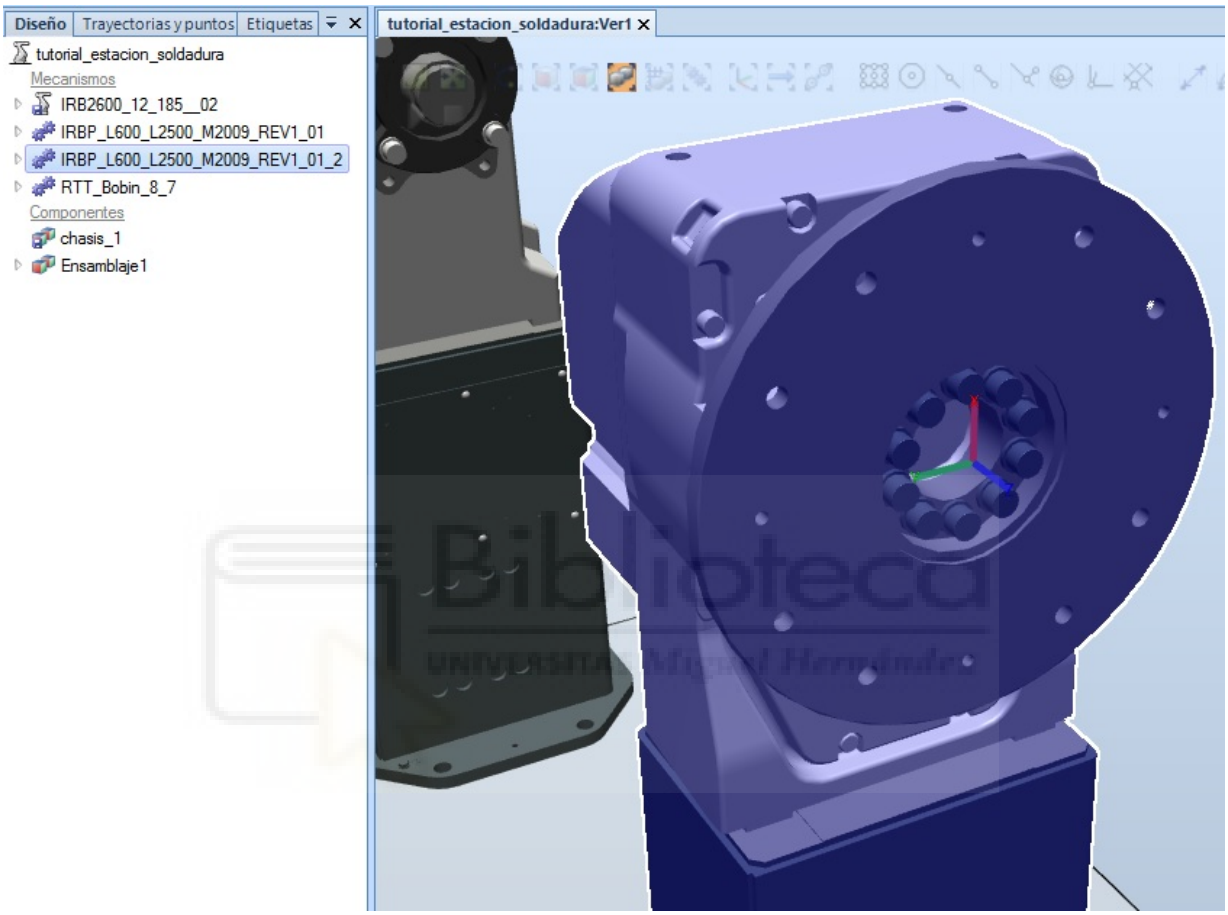


Figura 214: Sistema de coordenadas local del posicionador.

Haciendo lo mismo sobre la geometría importada comprobamos el del chasis. Vemos que el sistema de referencia local de la geometría importada no está en la posición deseada como podemos ver en la siguiente imagen

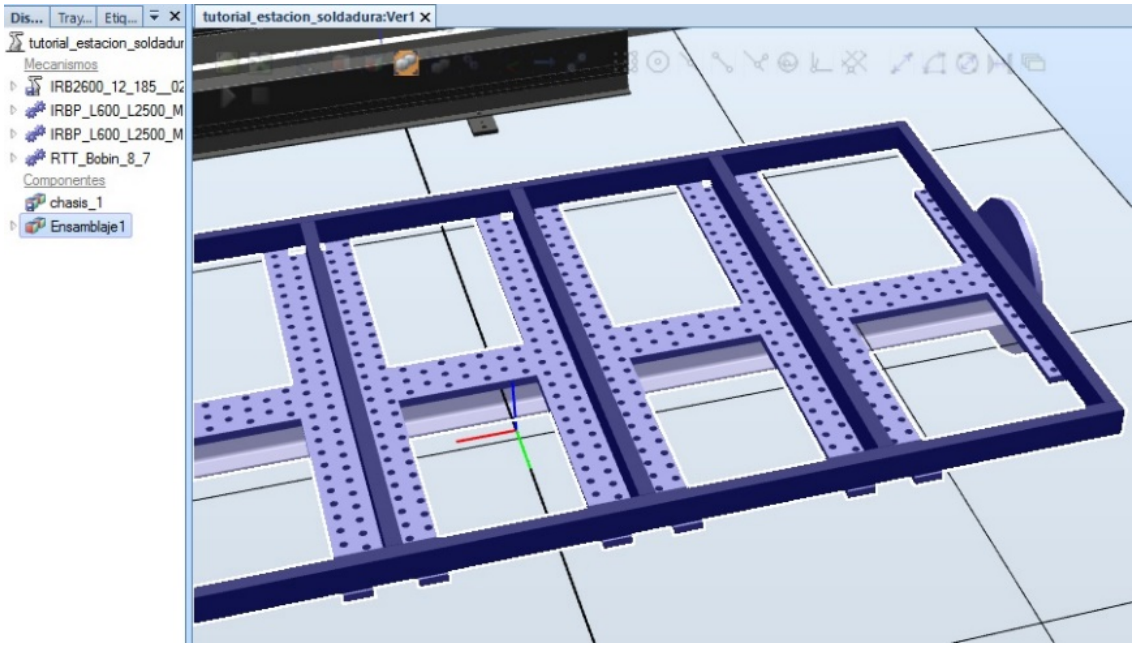


Figura 215; Sistema de coordenadas local del chasis.

En la siguiente figura podemos observar el sistema de referencia local situado en la posición y orientación deseada.

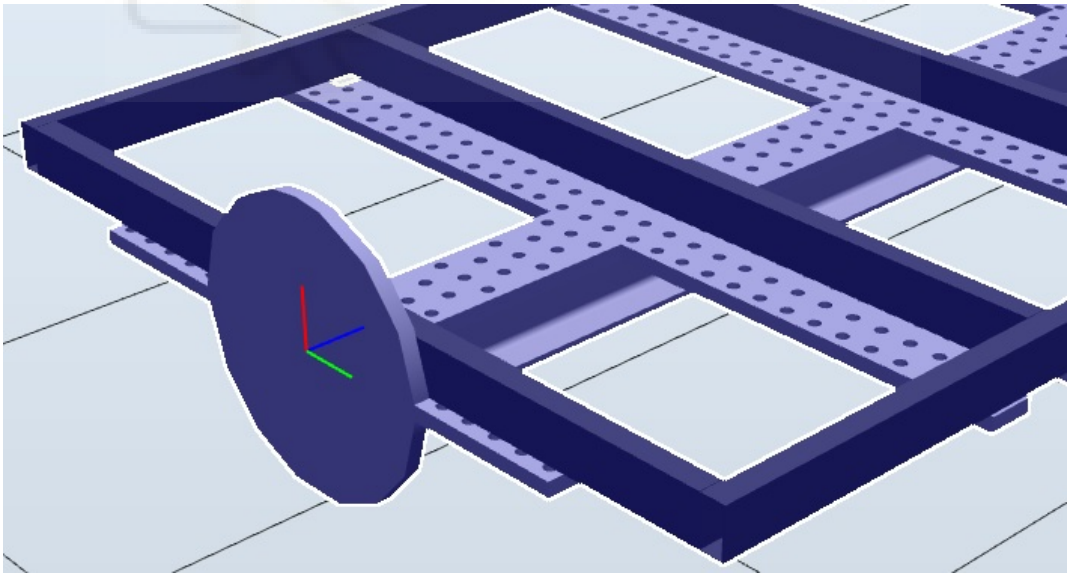


Figura 216: Cambio de posición del sistema de coordenadas local del chasis.

Para modificar el sistema de referencia local hacemos clic con el botón derecho del ratón sobre la geometría y seleccionamos en la ventana que aparece “**Modificar**”<<”**Establecer origen local**”.

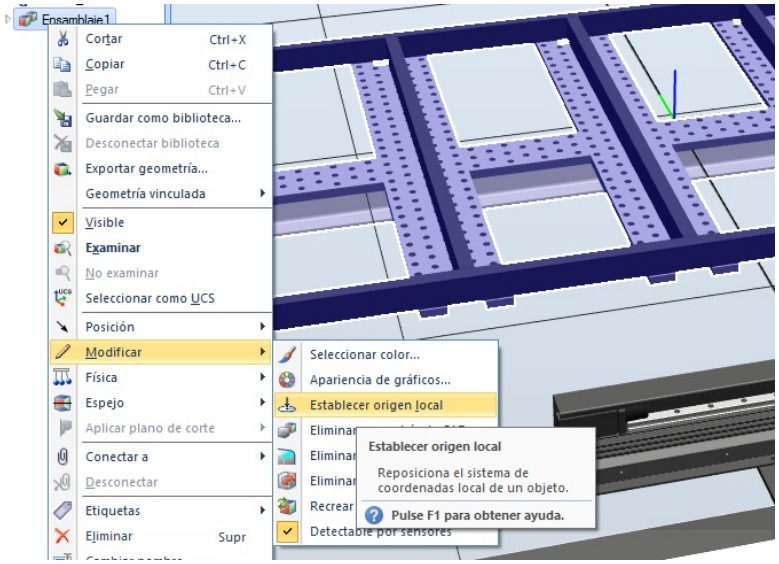


Figura 217: Establecer origen local I.

En la ventana que aparece podemos definir el nuevo punto origen del sistema y su orientación. En referencia seleccionamos “**Local**” ya que nos facilitará ver que ángulo debemos girar cada eje respecto del origen local existente para alcanzar la nueva orientación. Para localizar el nuevo punto origen usaremos las ayudas de localización gráfica RobotStudio y hacemos clic en “Aplicar”, el sistema se mueve al nuevo origen pero conservando la orientación anterior

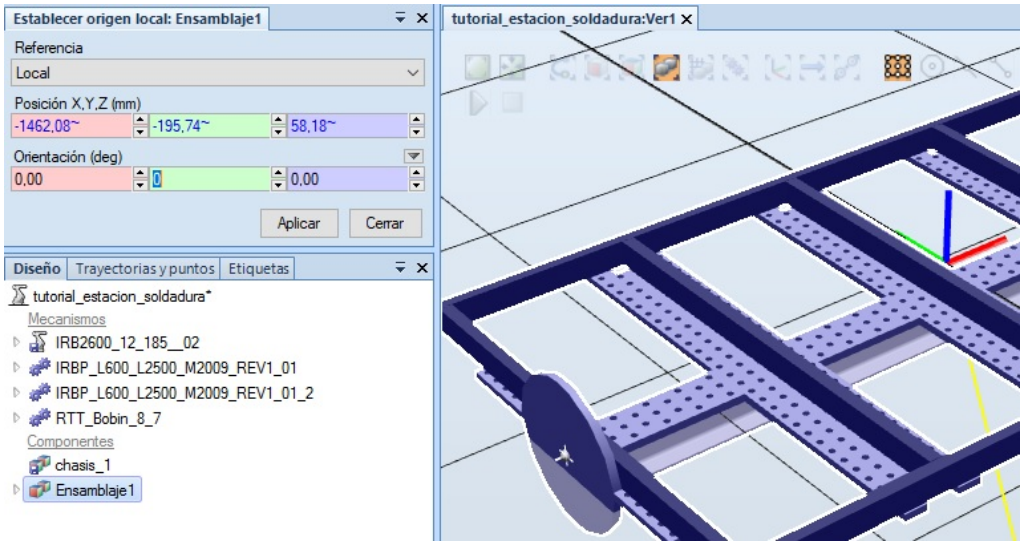


Figura 218: Establecer origen local II.

Volvemos a realizar la misma operación pero ahora cambiamos la orientación de los ejes. Se recomienda girar un eje cada vez para mejor claridad de las rotaciones que vamos realizando hasta obtener la orientación deseada.

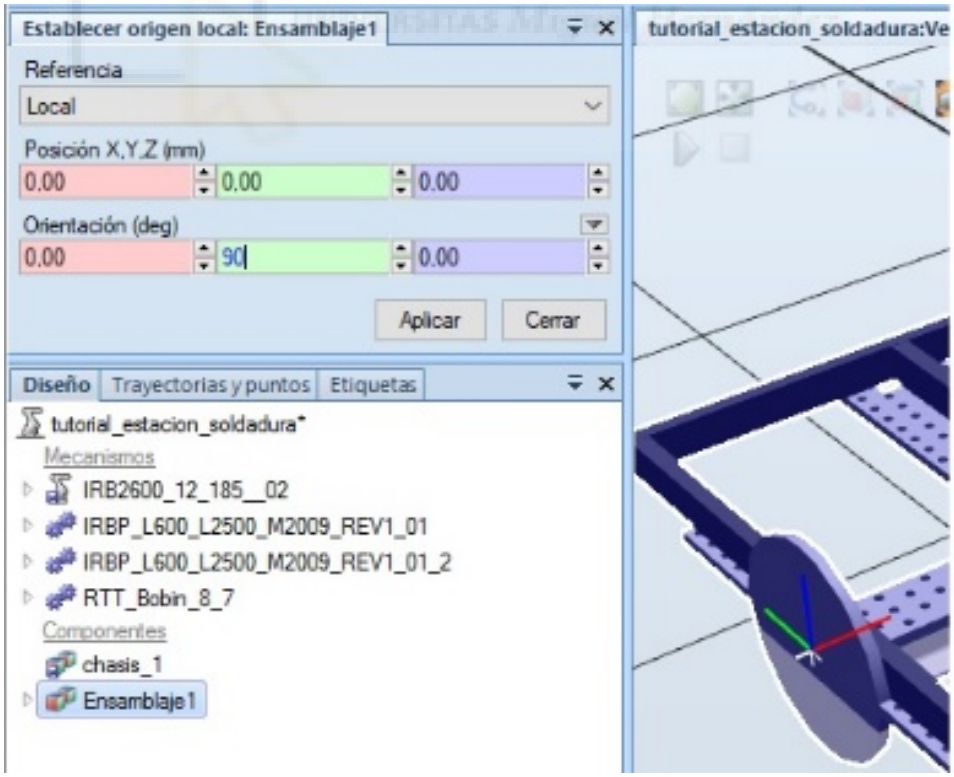


Figura 219: Establecer origen local III.

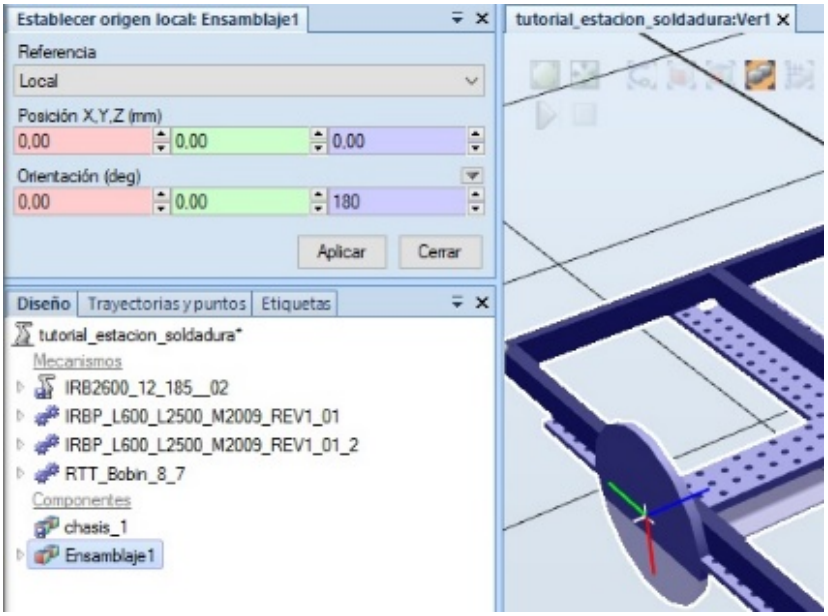


Figura 220: Establecer origen local IV.

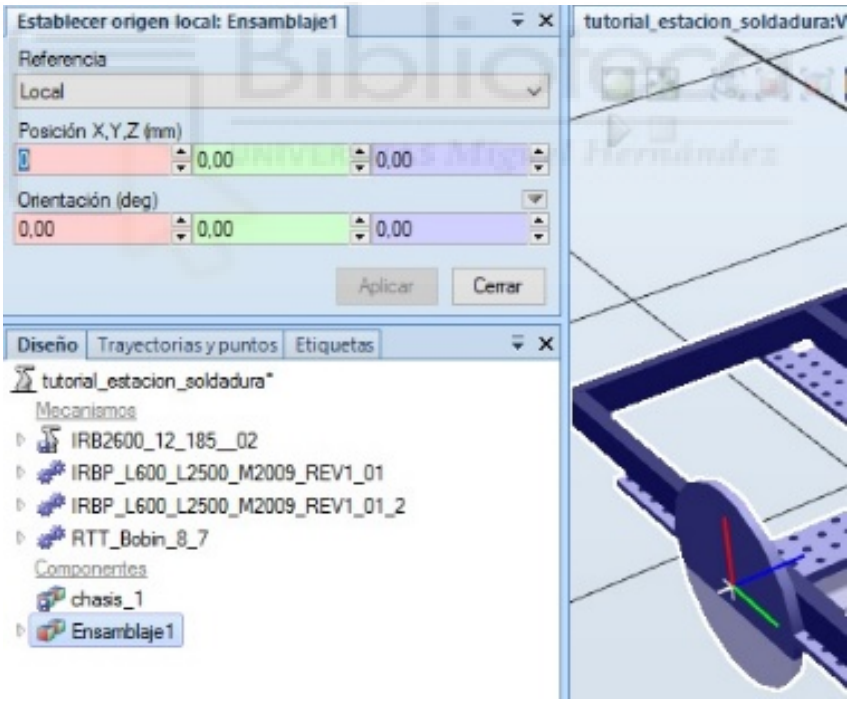


Figura 221: Establecer origen local V.

Una vez guardado como biblioteca conectaremos cada chasis a su posicionador. Haciendo clic con el botón derecho del ratón sobre el chasis seleccionamos "**Conectar a**" y seleccionamos el posicionador deseado. A la pregunta de si deseamos actualizar la posición del chasis contestamos que sí.

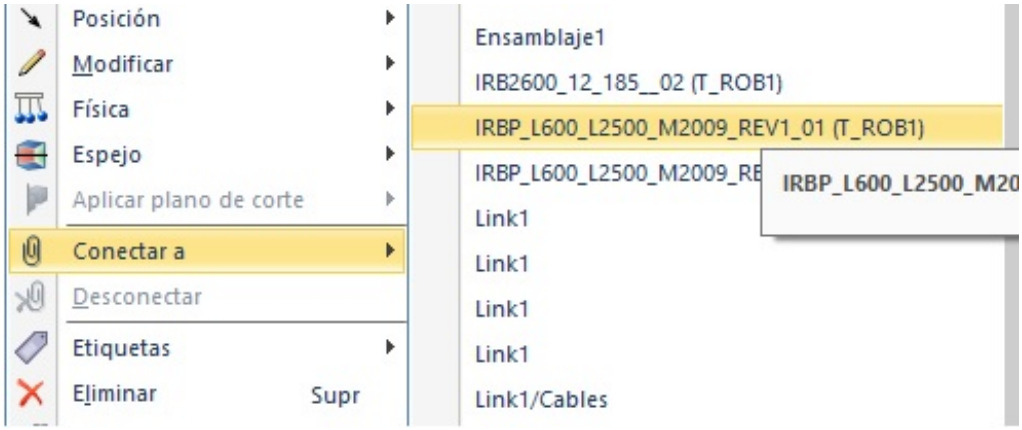


Figura 222: Conectar chasis a posicionador.

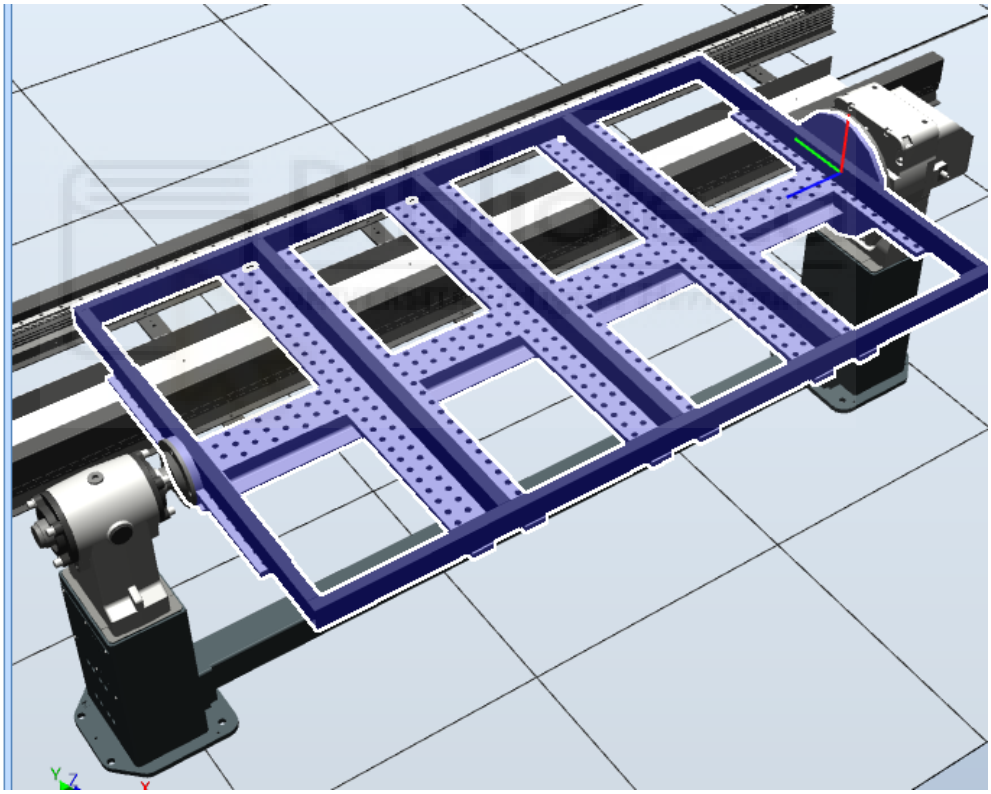


Figura 223: Chasis conectado a posicionador.

Si ahora movemos el eje del mecanismo posicionador, el chasis se moverá con él.

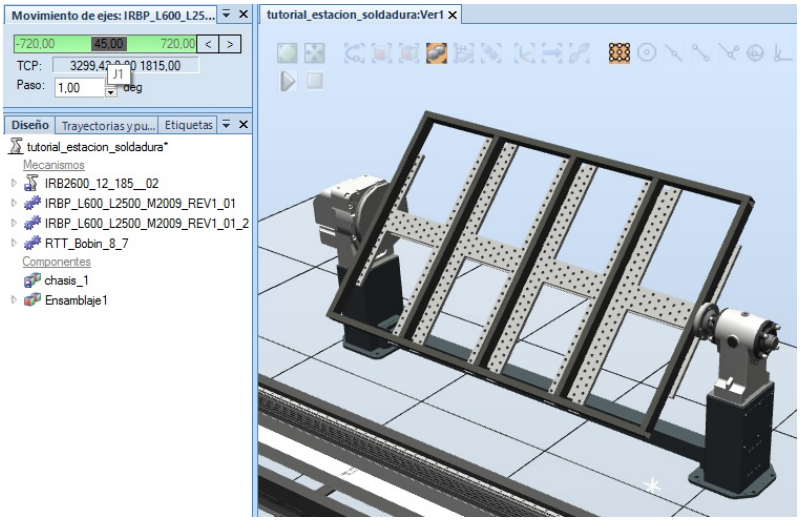


Figura 224: Movimiento del eje del posicionador.

5.4.6 Sistemas de referencia de la estación.

El sistema de coordenadas mundo de RobotStudio (**RS-WCS**) es el sistema de referencia del que dependen todos los demás sistemas de referencia en la estación virtual y lo podemos ver en la siguiente imagen en la que se ha desactivado la vista del suelo para mayor claridad.

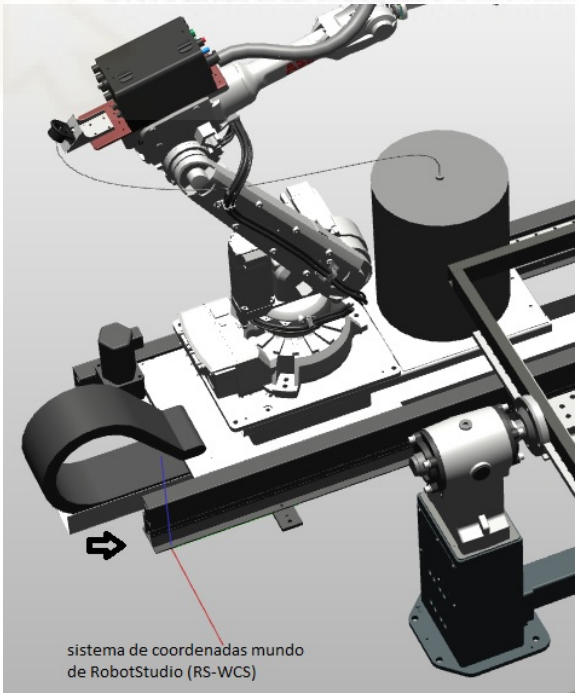


Figura 225: RS-WCS de la estación-

El sistema de coordenadas de la tarea (**TF**) en la estación real es el sistema de coordenadas mundo del controlador del robot (**RC-WCS**) y podemos ver que se ha separado del sistema de coordenadas de la base del robot. Al crear la conexión del robot con el track el **TF** se sitúa en el mismo lugar que el sistema de coordenadas de la base del robot (**BF**), pero desacoplado de este para que quede estacionario en dicha posición si movemos el robot sobre el track. Para conocer donde está situado, hacemos clic en "**Bases de coordenadas de la tarea**" en la pestaña "**Controlador**" de la cinta superior. Aparece una ventana que nos permite posicionarlo en el lugar que nos convenga si lo deseamos y se nos muestran los ejes del sistema de coordenadas mundo de la estación (**RS-WCS**) en la ventana gráfica de RobotStudio. En la siguiente imagen se han creado dos sistemas de coordenadas, uno en la posición del **RS-WCS** y otro en la del **TF** como ayuda para poder visualizarlos. Si movemos el robot en el track, vemos que el **TF** sigue situado en la misma posición que era la de la base del robot cuando se coordinó con el track.

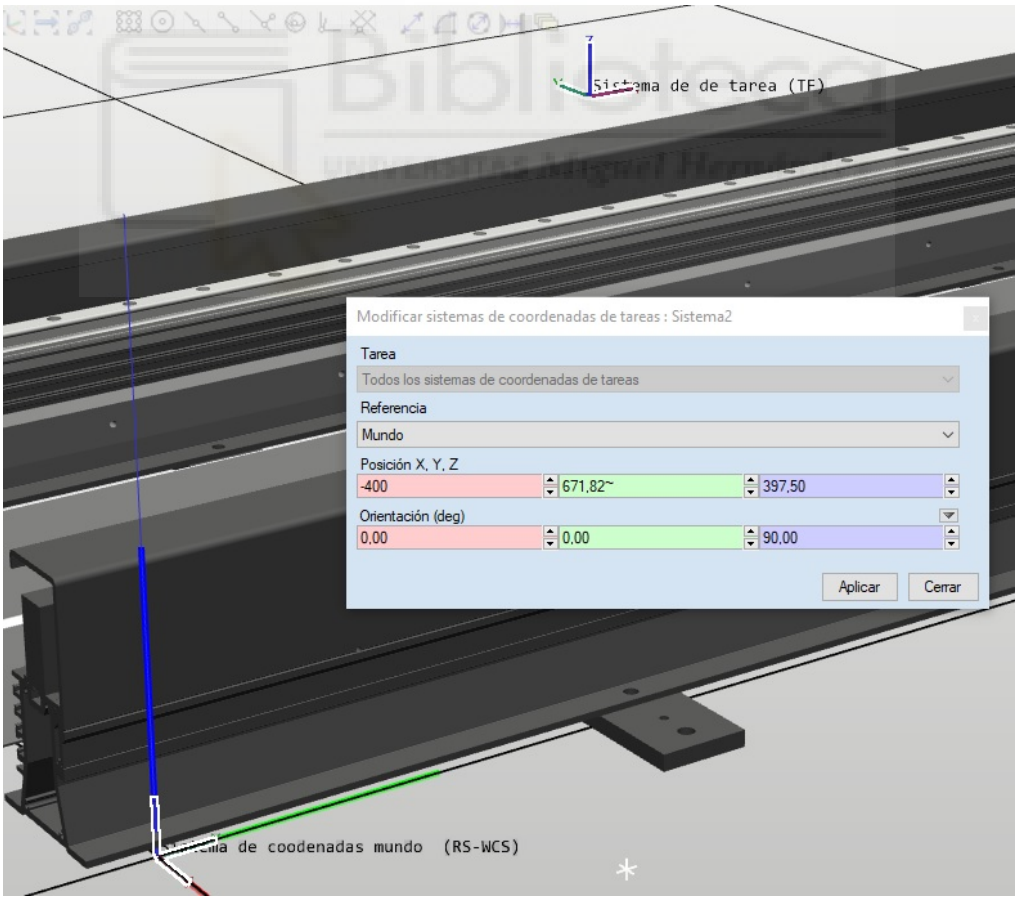


Figura 226: Sistema de coordenadas de la tarea de la estación.

5.4.7 Inserción de la herramienta y conexión al robot.

Ahora importaremos la herramienta del robot y la conectaremos a este tal y como se explicó en el apartado 4.1.3.5. Haremos clic en “Importar biblioteca”<<”Equipamiento” y seleccionaremos la pistola de soldadura “Binzel air 22”.

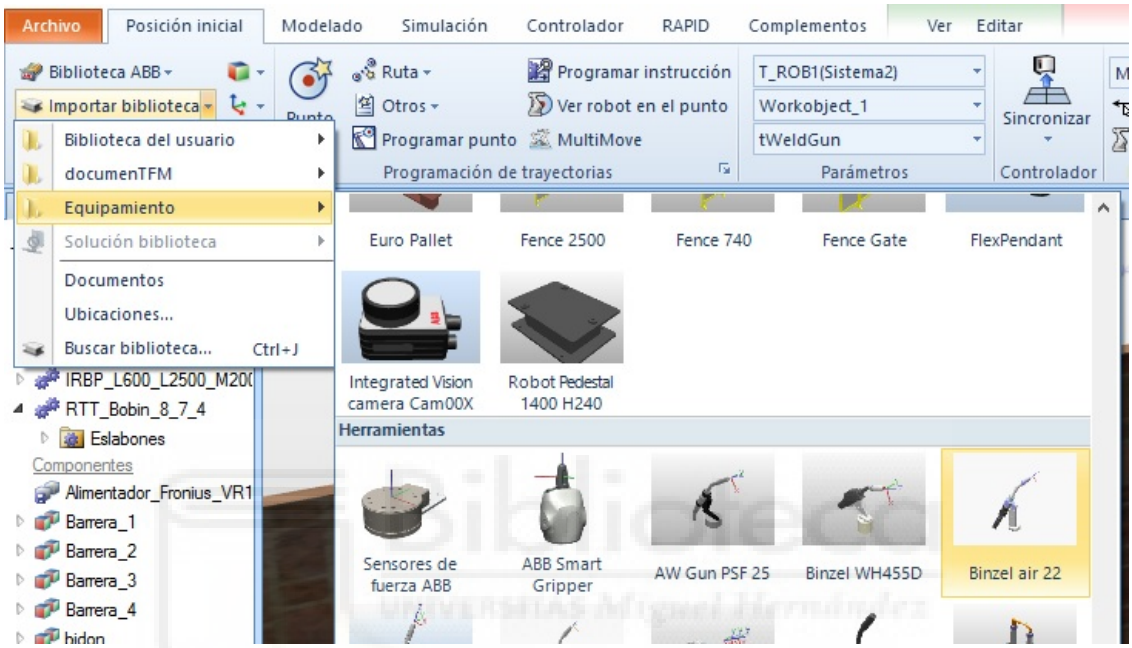


Figura 227: Importar pistola Binzel Air 22

Una vez importada, la seleccionaremos en el árbol de la izquierda y la arrastraremos sobre el robot para conectarla a este. A la pregunta de si queremos actualizar su posición contestamos sí.

Otra forma de conectar la herramienta al robot es haciendo clic con el botón derecho del ratón sobre la herramienta y seleccionar “**Conectar a**” y elegir el robot en la ventana flotante que aparece, luego sincronizar con RAPID.

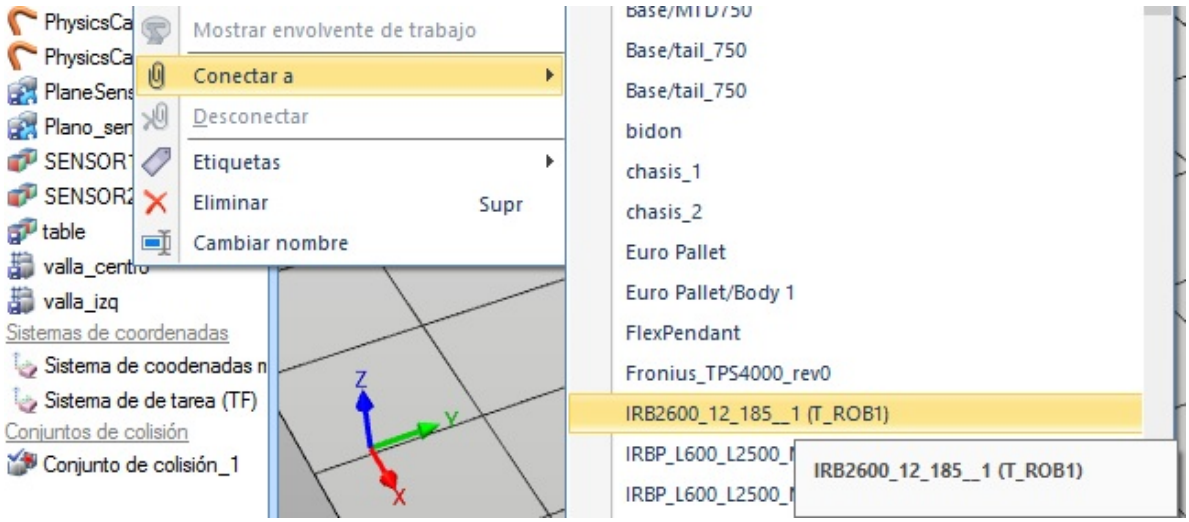


Figura 228: Conectar pistola al IRB2600.

La herramienta queda conectada al TCP del robot.

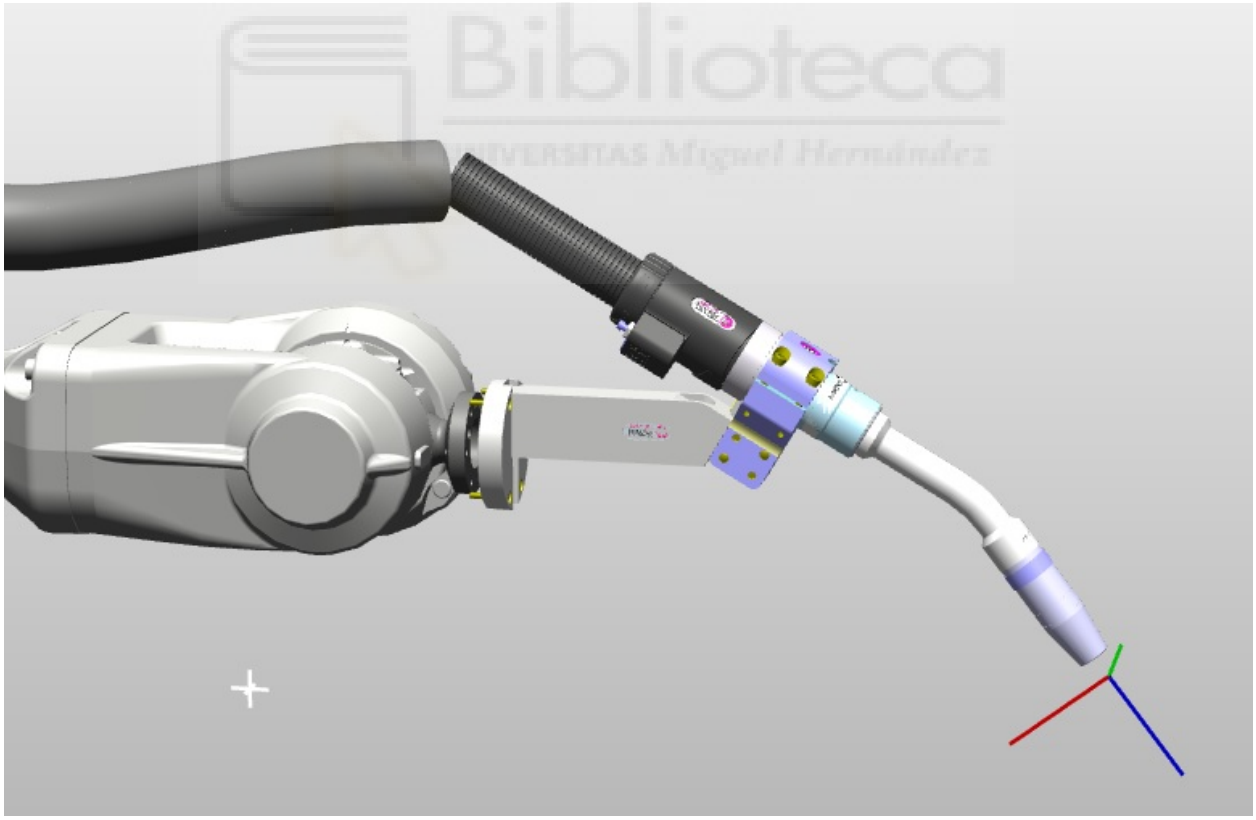


Figura 229: Pistola conectada al TCP del robot.

5.4.8 Creación de los objetos de trabajo y objetivos.

En el apartado **4.1.3.4** se describieron los tipos de sistemas de referencia existentes en RobotStudio y en particular los sistemas de referencia denominados "**objetos de trabajo**". Cuando se programa un objetivo en un "**objeto de trabajo**" este está referenciado a dicho sistema de referencia. Para la programación de los objetivos en la estación se crearán dos "**objetos de trabajo**" y cada uno de ellos lo conectaremos a un posicionador. Esta acción permitirá que cuando el posicionador se mueva, los objetivos se muevan solidariamente con él de forma que el controlador "conozca" su posición en todo momento. Otra ventaja es que si por alguna razón el útil no está en la posición prevista o la pieza de trabajo se mueve respecto de la posición programada es sencillo calibrar las nuevas posiciones introduciendo un offset en el sistema de coordenadas objeto de la pieza.

Para crear un objeto de trabajo, hacemos clic sobre "**Otros**"<<"**Crear objeto de trabajo**" y aparece una ventana en la que podemos especificar el punto de origen y la orientación del objeto de trabajo.

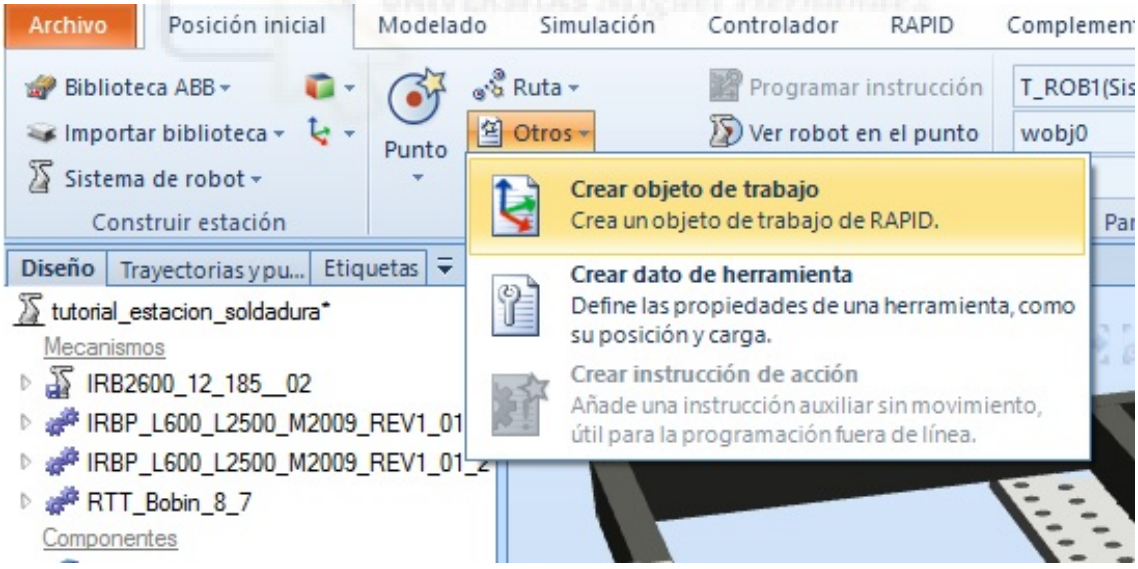


Figura 230: Creación de objetos de trabajo de la estación I.

Recordar que un objeto de trabajo está compuesto por dos sistemas de coordenadas, el del usuario y el del objeto de trabajo. Los puntos objetivo que se programen eligiendo dicho

objeto de trabajo se referenciarán al sistema de coordenadas objeto de este, que a su vez depende del sistema de coordenadas del usuario del objeto de trabajo en la jerarquía de los sistemas de referencia.

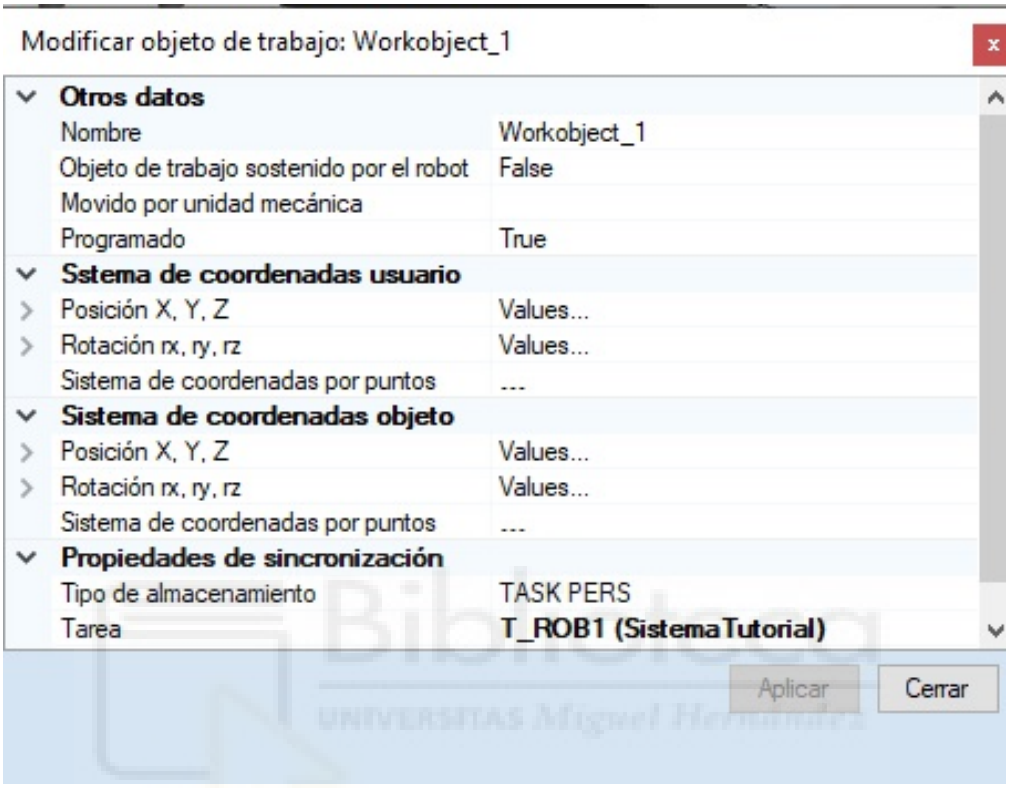


Figura 231: Creación de objetos de trabajo de la estación II.

Situaremos el sistema de coordenadas del usuario sobre una esquina del útil y el sistema de coordenadas objeto sobre una esquina de la pieza de trabajo tal y como podemos observar en la siguiente figura, de esta manera podemos calibrar tanto la posición del útil como la posición de la pieza sobre el útil en la estación real.

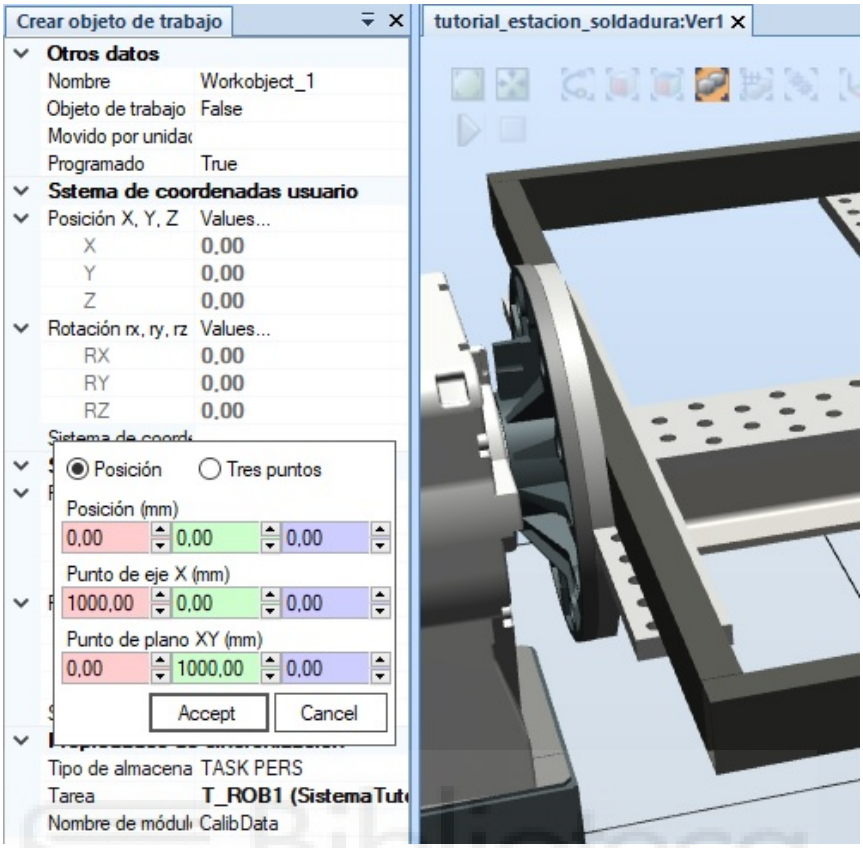


Figura 232: Creación de objetos de trabajo de la estación III.

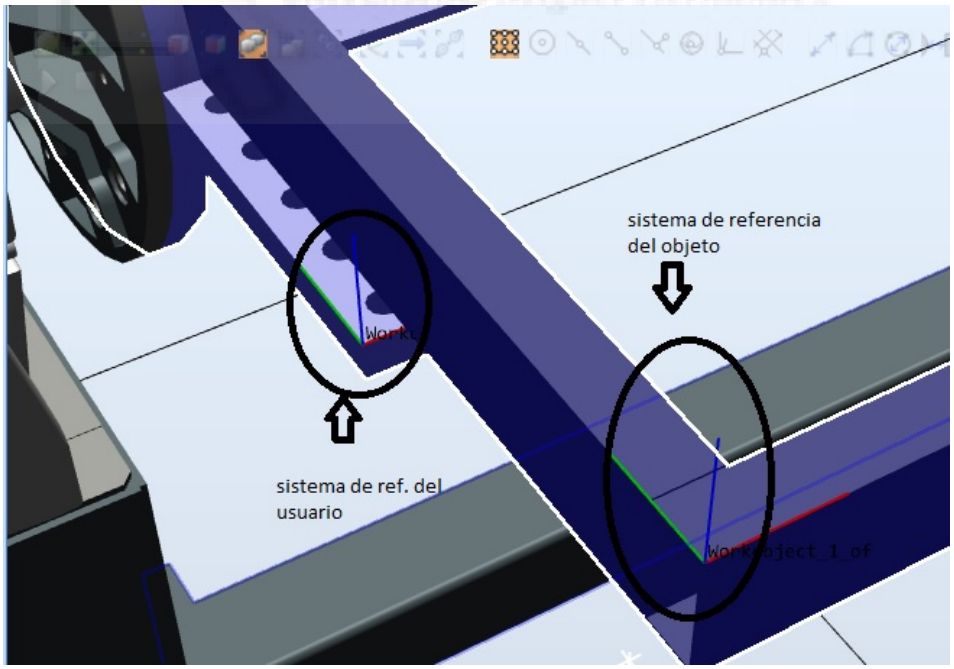


Figura 233: Creación de objetos de trabajo de la estación IV.

5.4.9 Creación de las trayectorias.

Para crear las trayectorias de soldadura en cada uno de los chasis que debe seguir el robot deberemos en primer lugar que realizar las siguientes acciones:

- En la pestaña “**Posición inicial**” de la cinta superior en el grupo “**Parámetros**” seleccionar:
    - La tarea en la que se va a programar el objetivo.
    - El objeto de trabajo en el que se va a programar el objetivo.
    - La herramienta que se va a utilizar.

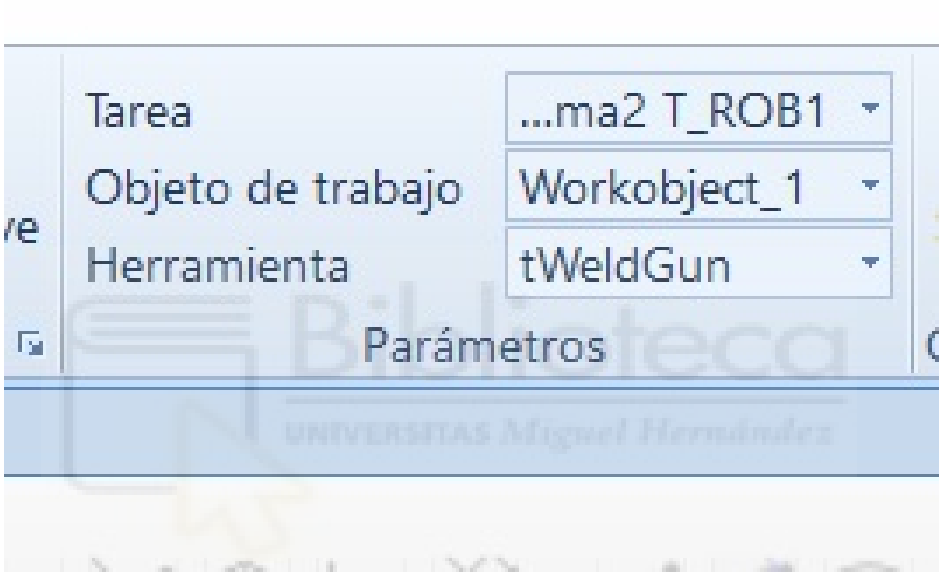


Figura 234: Parámetros de la trayectoria.

- En la pestaña “**Simulación**” de la cinta superior hacer clic sobre “**Activar unidades mecánicas**” y marcar el posicionador sobre el que se están programando los objetivos y trayectorias. Si no se activa la unidad mecánica al programar el objetivo no se almacenara en este la pose del posicionador.

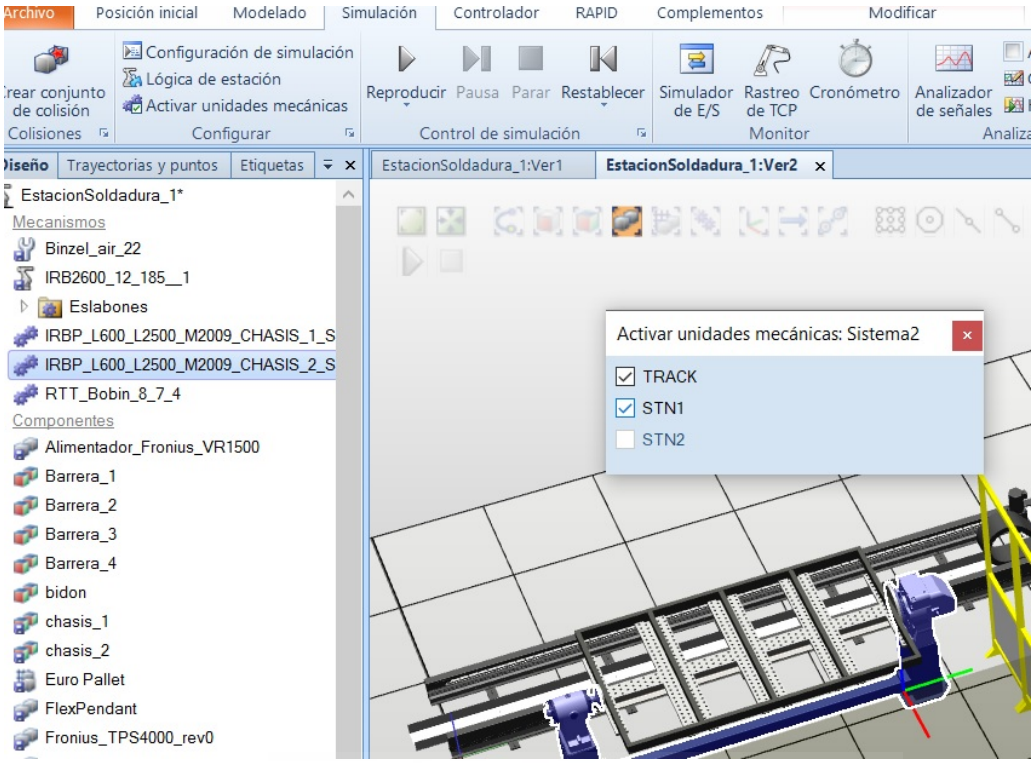


Figura 235: Activación del posicionador.

- Mover de forma manual el eje del track para colocar el robot en la posición deseada y crear los objetivos, la posición del track se almacenará en los objetivos creados. Para mover el track haremos clic sobre el con el botón derecho del ratón en el árbol de la izquierda y seleccionaremos "**Movimiento de ejes de mecanismo**".

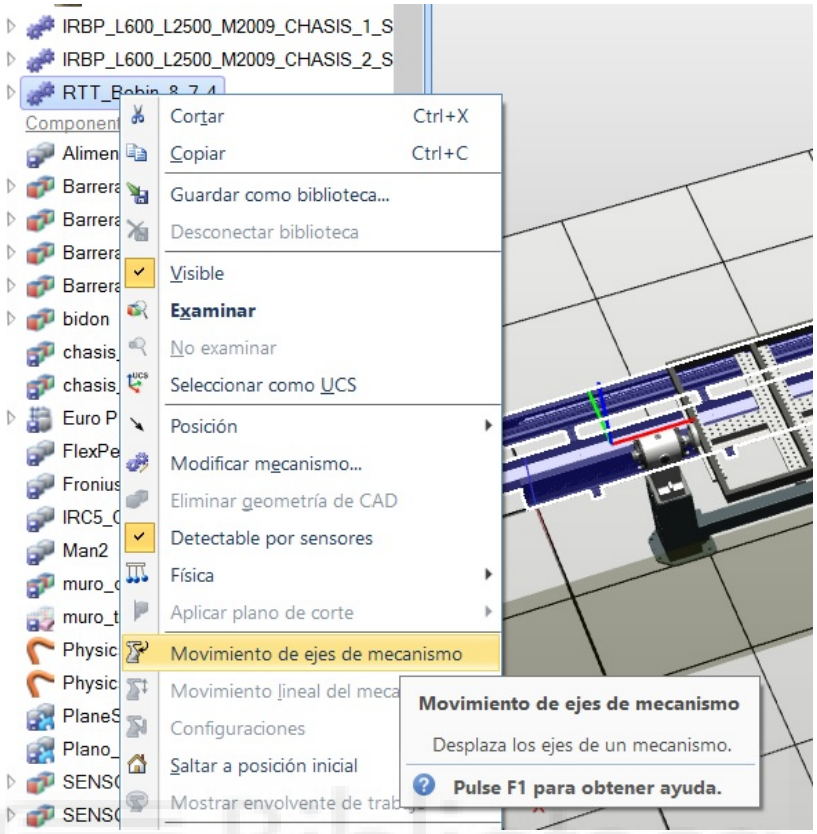


Figura 236: Movimiento del posicionador.

- Crear los objetivos. RobotStudio nos permite diversas formas de crear objetivos. En este caso nos interesa crear las trayectorias directamente a partir de arista con la herramienta "**Trayectoria automática**" ya que nos ofrece, además, la posibilidad de:
    - Elegir el tipo de trayectoria como lineal, circular o constante (genera puntos con una distancia constante.
    - Establecer una distancia mínima entre puntos generados.
    - Establecer una tolerancia con respecto a la descripción geométrica permitida para los puntos generados.
    - Crear un nuevo punto de aproximación y uno de partida a distancia elegida.
    - Cuando seleccionamos la arista de la trayectoria, RobotStudio nos indica gráficamente los puntos de la trayectoria y la dirección que sigue, si deseamos la dirección contraria podemos seleccionar "Invertir" en la ventana de creación de trayectorias.

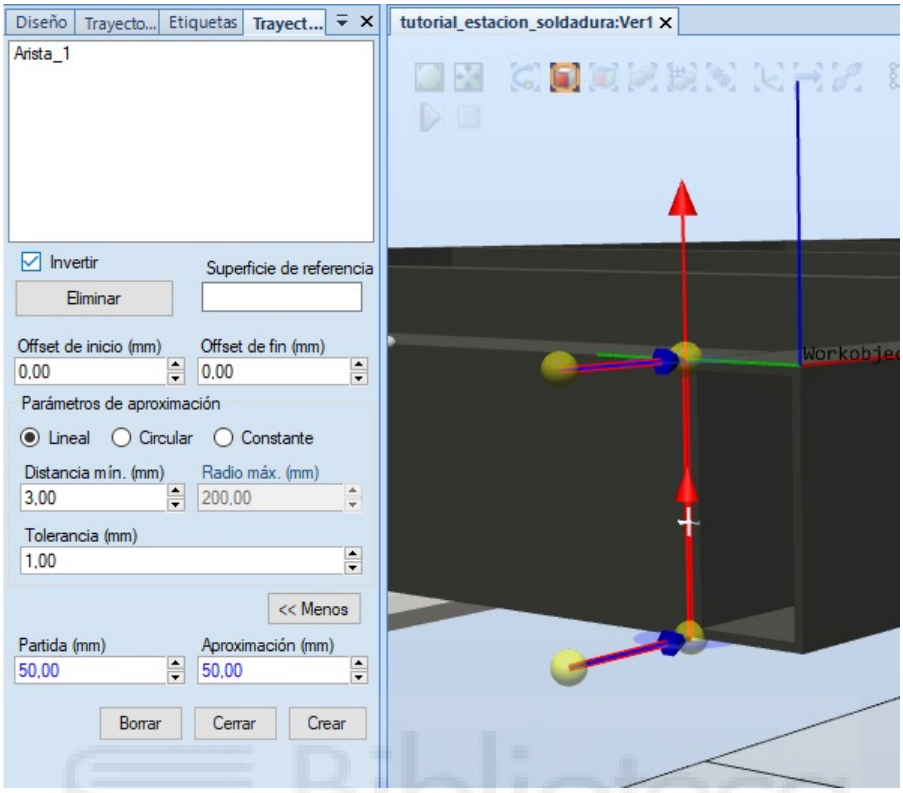


Figura 237: Trayectoria automática.

Antes de comenzar a crear las trayectorias seleccionaremos en la cinta inferior de RobotStudio el tipo de movimiento (línea), la velocidad (v50) y la precisión (fine). Posteriormente podremos cambiar todos esos parámetros de las instrucciones de movimiento editándolas o directamente en RAPID si es necesario.

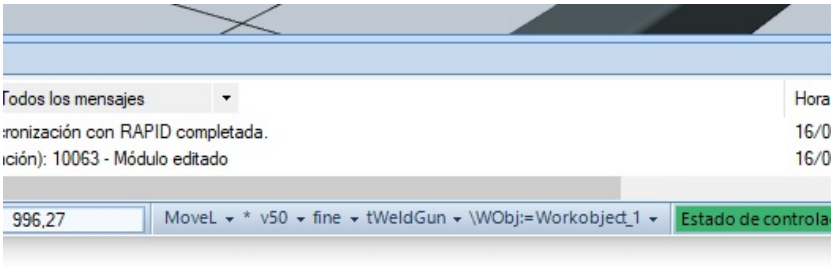


Figura 238: PLantilla de instrucciones de movimiento seleccionada

De esta forma iremos moviendo el robot con el track a la posición adecuada y colocaremos el posicionador en el ángulo necesario para que el robot pueda alcanzar la zona requerida e iremos programando cada trayectoria en el primer chasis.

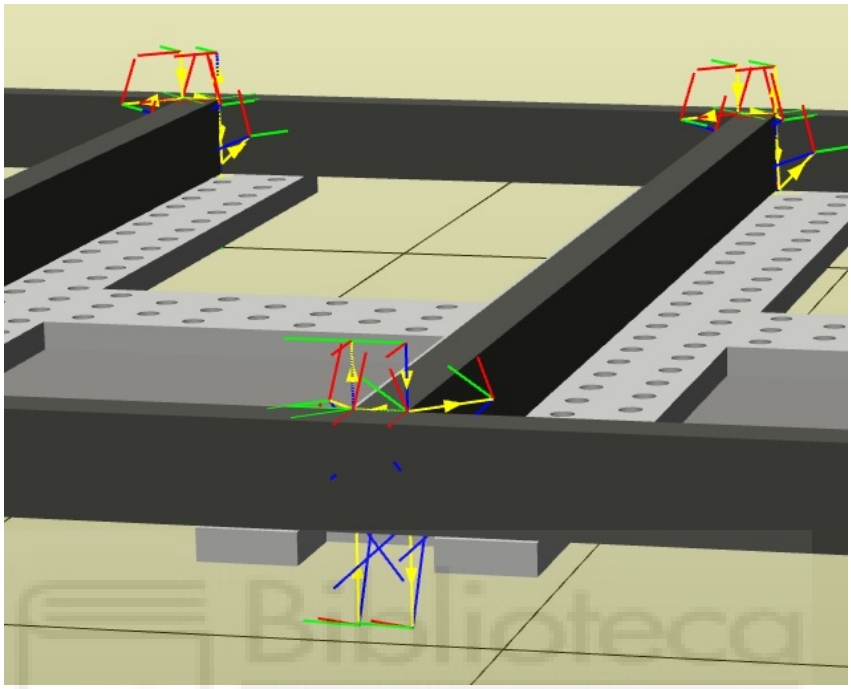


Figura 239: Trayectorias de soldadura.

Una vez programada una la trayectoria, deberemos comprobar la alcanzabilidad del robot para saber si es capaza de realizar la trayectoria. Sincronizamos con RAPID y haciendo clic con el botón derecho del ratón sobre una instrucción de movimiento seleccionaremos "**Ver herramienta en el punto**" y si no está seleccionado ya "**Comprobar alcance**" lo seleccionaremos también.

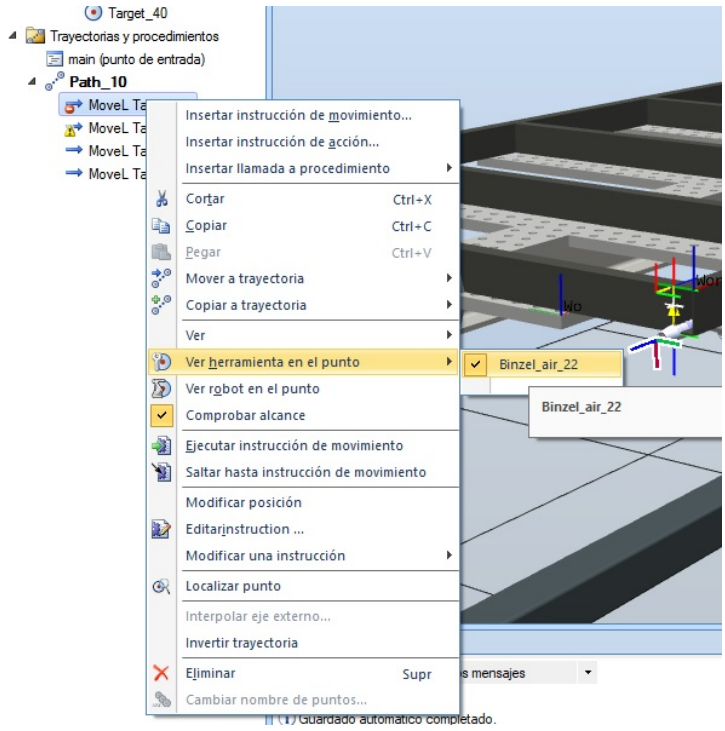


Figura 240: Orientación de la pistola de soldadura en las trayectorias.

RobotStudio nos alerta con un icono rojo en las instrucciones de movimiento del path si no es posible alcanzar esa posición y con uno amarillo si es posible pero en otra configuración de ejes del robot.

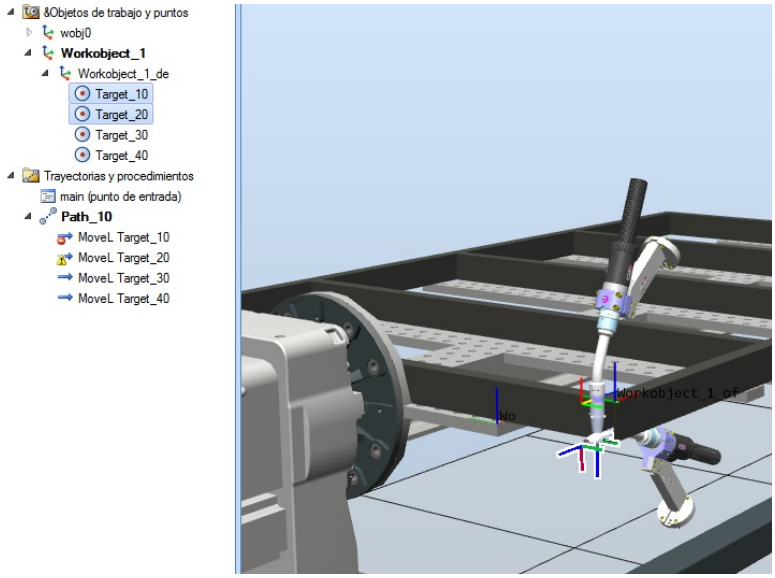


Figura 241: Alertas de alcanzabilidad de RobotStudio.

Si el objetivo es alcanzable, también es posible que RobotStudio nos muestre el robot en el punto haciendo clic con el botón derecho del ratón sobre el objetivo en el árbol de la izquierda y seleccionar “Ver robot en el punto”

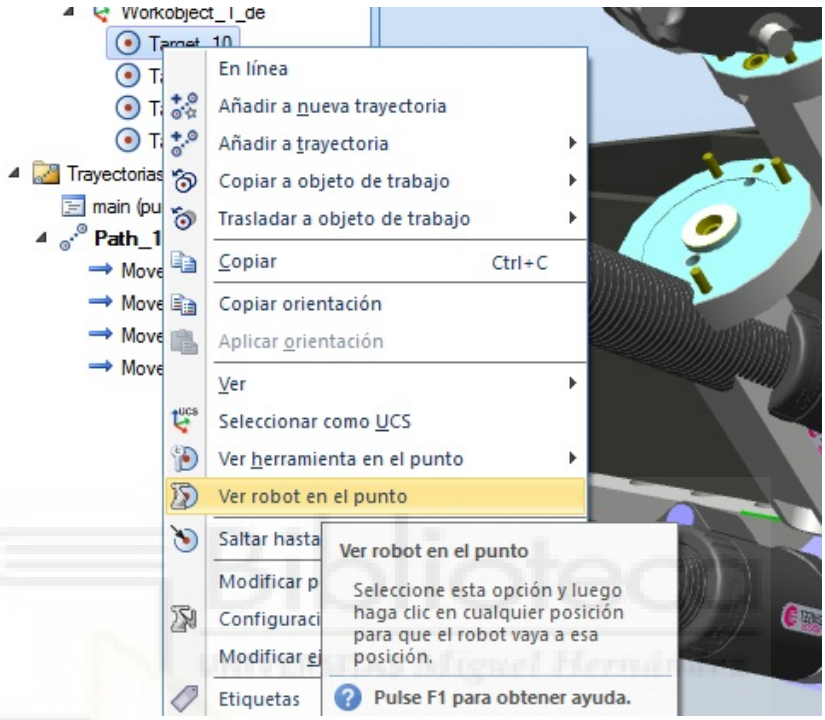


Figura 242: Alcanzabilidad del robot I.

Para cambiar la posición de la herramienta, en árbol de la izquierda hacemos clic con el botón derecho del ratón sobre el objetivo en cuestión y seleccionamos “**Modificar punto**”<<”**Girar**”

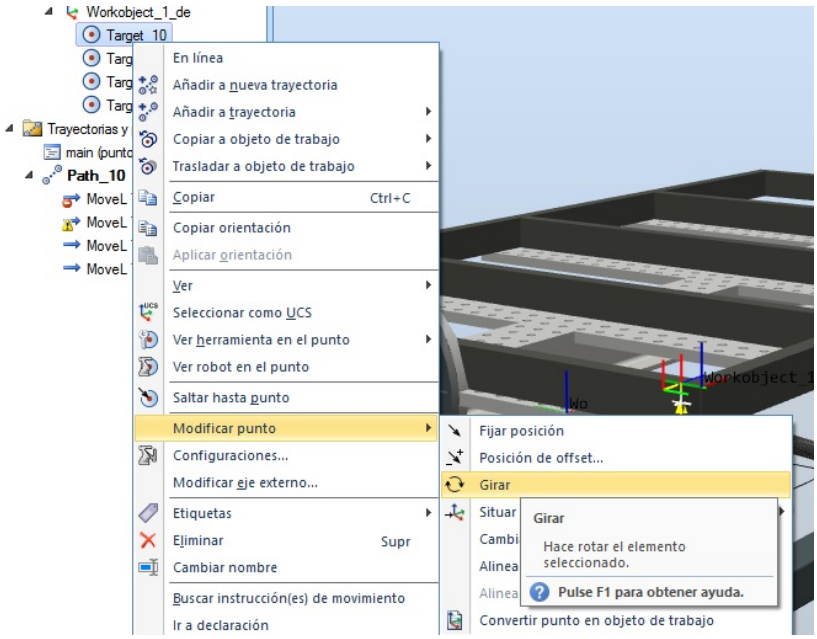


Figura 243: Alcanzabilidad del robot II.

Podemos elegir el eje local sobre el que queremos aplicar el giro y el ángulo. Vamos cambiando la posición de la herramienta hasta que desaparezca, al menos, el aviso rojo en la instrucción de movimiento correspondiente.

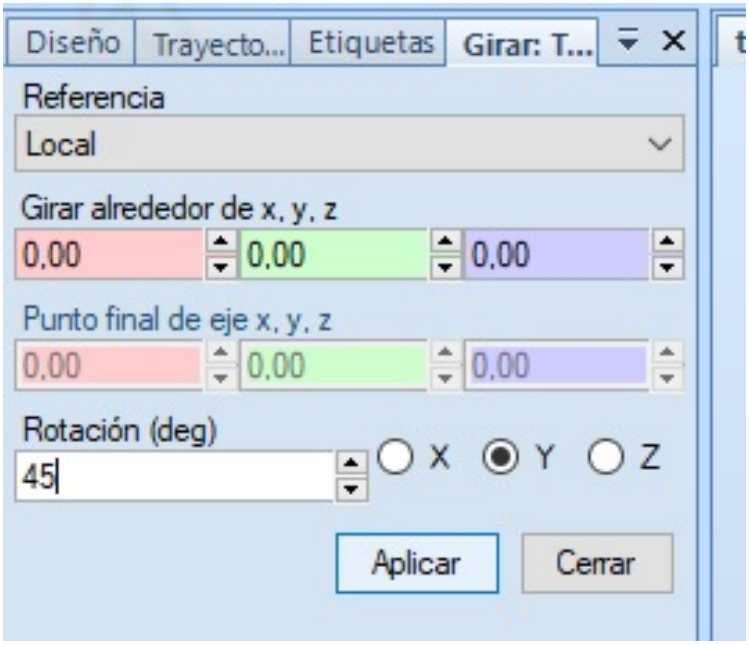


Figura 244: Alcanzabilidad del robot III.

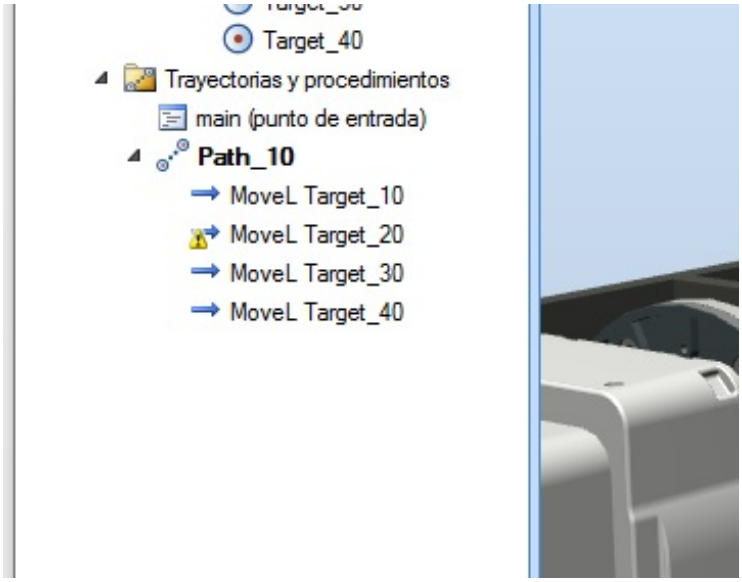


Figura 245: Alcanzabilidad del robot IV.

El aviso amarillo nos indica que es posible el alcance en otra configuración. Para cambiar la configuración actual hacemos clic con el botón derecho del ratón sobre el objetivo y elegimos “Configuraciones…”, en la ventana que aparece podemos seleccionar otra configuración posible.

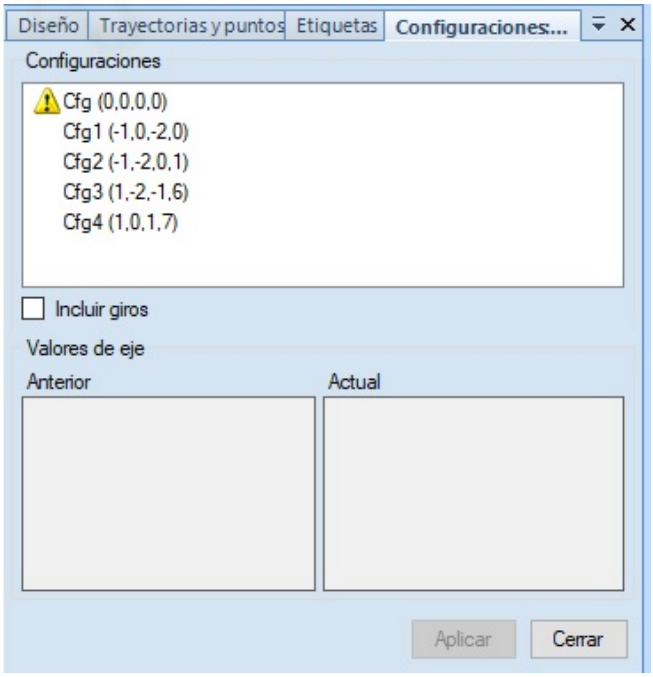


Figura 246: Alcanzabilidad del robot V.

Otra forma de hacerlo es hacer clic con el botón derecho del ratón sobre la trayectoria donde tenemos el aviso y seleccionar "**Configuración automática**"<<"**Todas las instrucciones de movimiento**" que automáticamente seleccionará otra configuración posible para conseguir la alcanzabilidad en todas las instrucciones de la trayectoria.

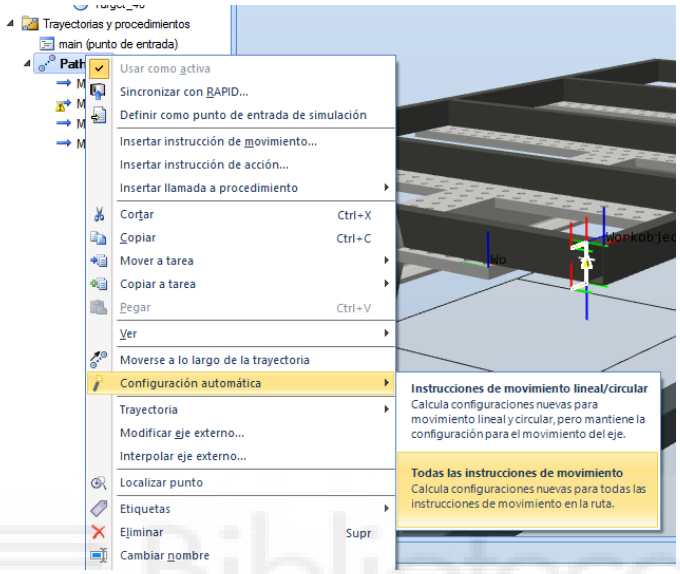


Figura 247: Alcanzabilidad del robot VI.

Si realizando la anteriores acciones no es posible conseguir la alcanzabilidad deberemos mover el robot sobre el track o el posicionador hasta conseguir una posible solución. Finalmente deberemos sincronizar con RAPID.

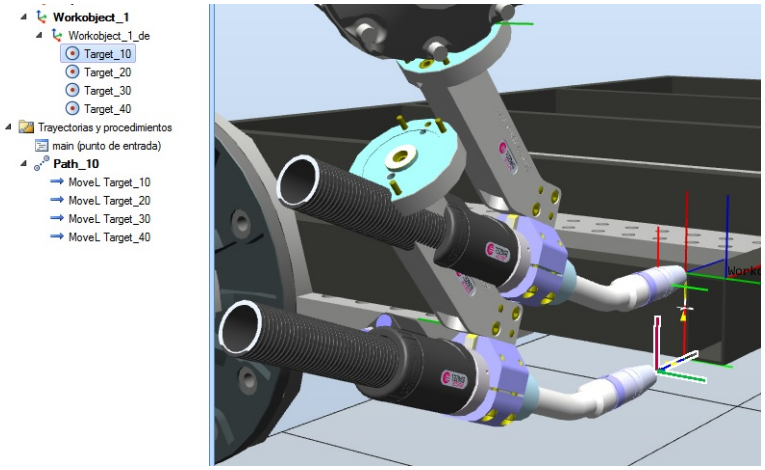


Figura 248: Alcanzabilidad del robot VII.

Si necesitamos desplazar alguno de los objetivos por alguna razón como, por ejemplo, si la herramienta colisiona con algún objeto cercano, podemos introducir un offset. Para ello hacemos clic con el botón derecho del ratón sobre el objetivo y seleccionamos “Modificar punto”<<”Posición de offset”

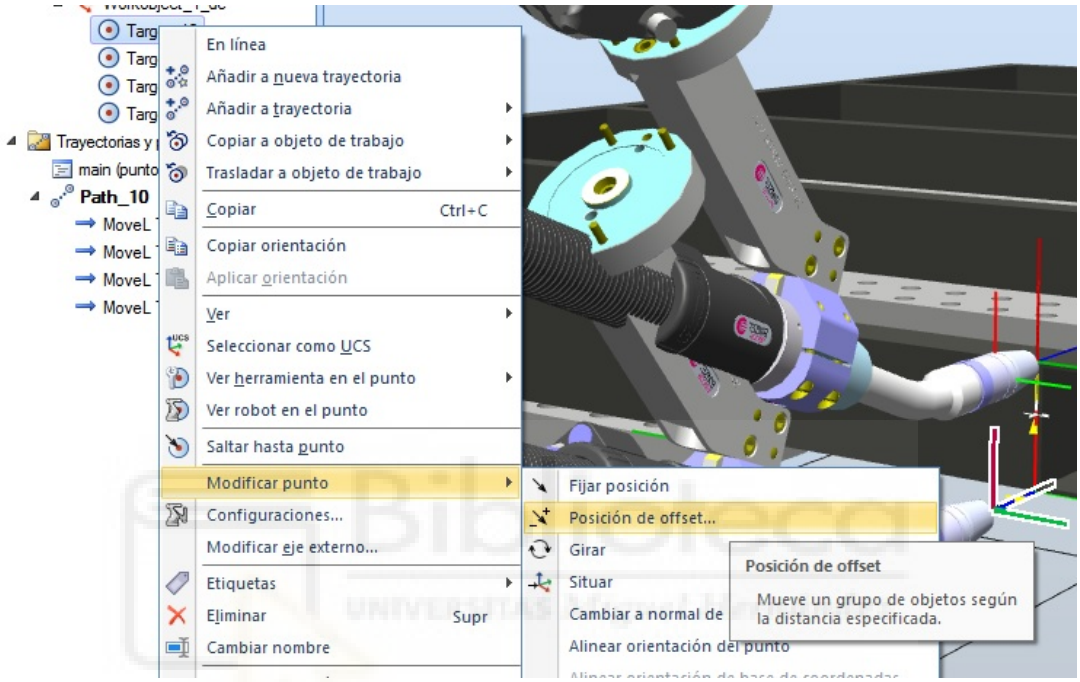


Figura 249: Aplicar un offset a un punto I.

En la ventana que aparece podemos introducir una traslación o rotación respecto del sistema de referencia elegido (local normalmente) y RobotStudio nos indica con un trazo amarillo y un círculo rojo hasta donde se desplazara el punto y su orientación respecto de la posición actual.

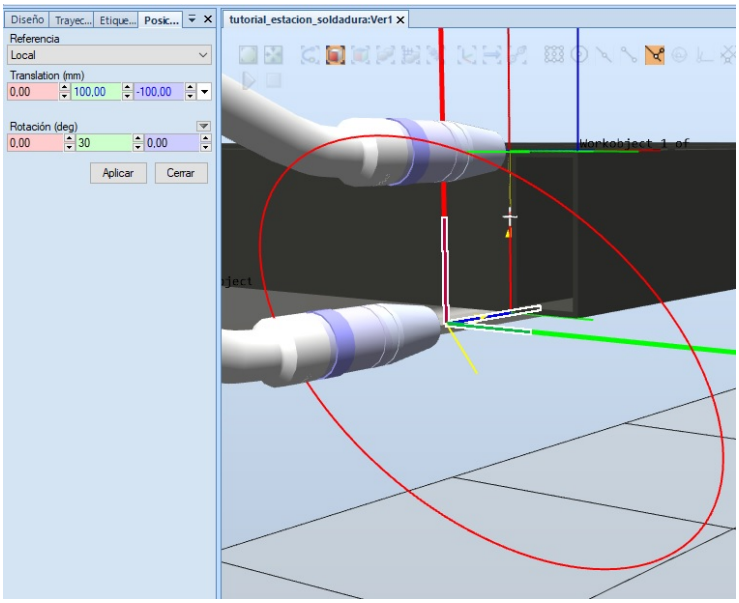


Figura 250: Aplicar un offset a un punto II.

Finalmente podemos comprobar que el robot puede trazar la trayectoria haciendo clic con el botón derecho del ratón sobre la trayectoria y seleccionar "**Moverse a lo largo de la trayectoria**"

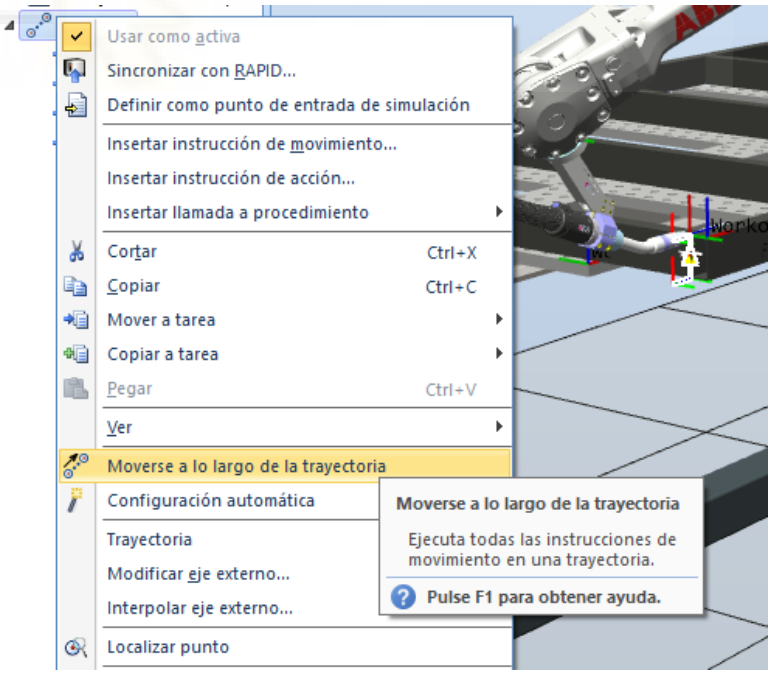


Figura 251: Moverse a lo largo de la trayectoria I.

El robot realizara la trayectoria seleccionada si es posible.

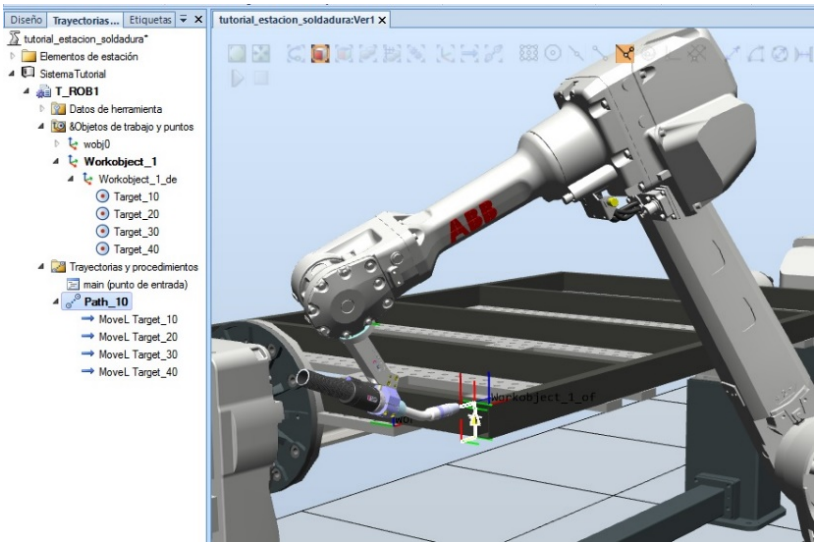


Figura 252: Moverse a lo largo de la trayectoria II.

Si es necesario cambiar alguna instrucción de movimiento o algún parámetro de esta hacemos clic con el botón derecho en el árbol de la izquierda sobre ella y seleccionaremos "**Editar instrucción**".

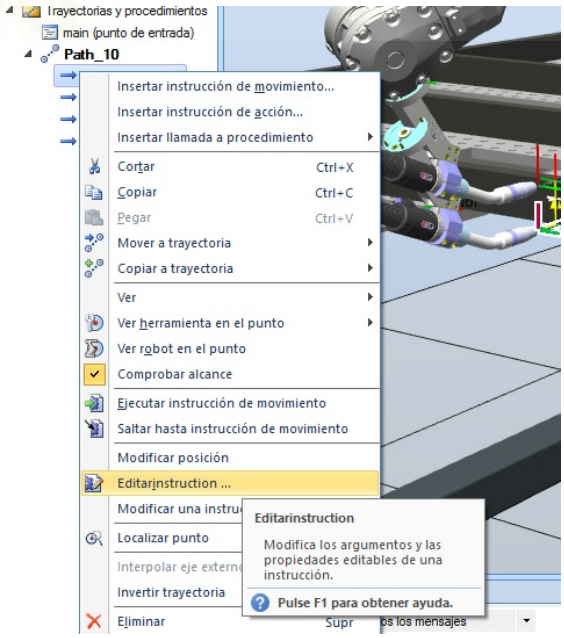


Figura 253: Editar instrucción I.

En la ventana que aparece podemos cambiar el tipo de instrucción (Joint o Linear) y los parámetros que deseemos, aplicamos y sincronizamos. Si seleccionamos varias instrucciones de movimiento al mismo tiempo aplicaremos el cambio a todas ellas.

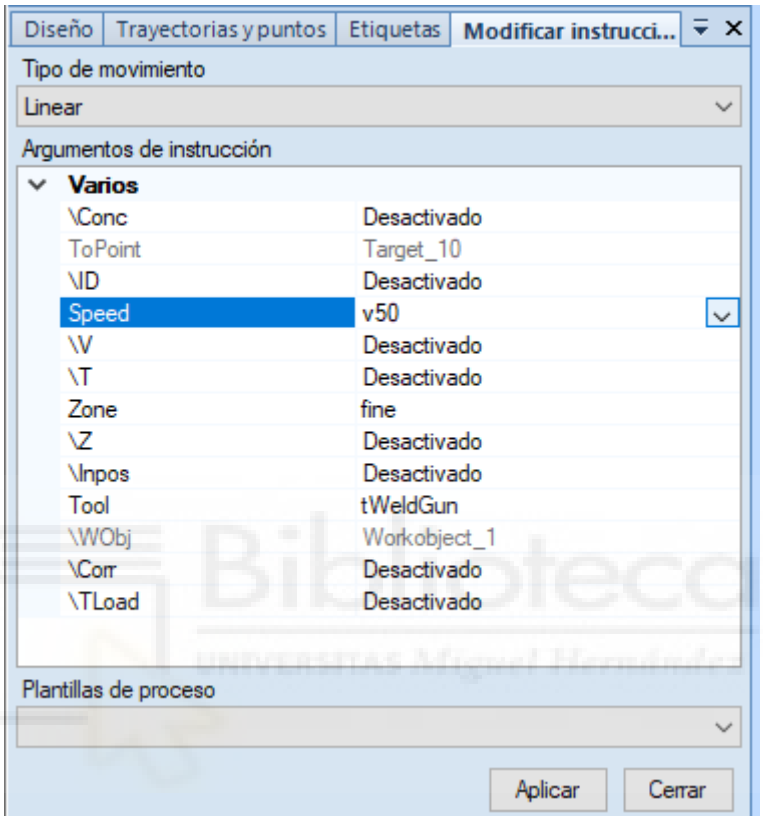


Figura 254: Editar instrucción II.

También es posible editar el código RAPID con los cambios que queramos realizar en las instrucciones o el programa. Para ello hacemos clic sobre la pestaña RAPID de la cinta superior, en el árbol de la izquierda expandimos la carpeta RAPID y hacemos doble clic sobre el módulo a editar para abrirlo. Ahora podemos realizar los cambios en el archivo del módulo RAPID manualmente. Para guardar los cambios realizados haremos clic sobre "**Aplicar**" en la cinta superior.

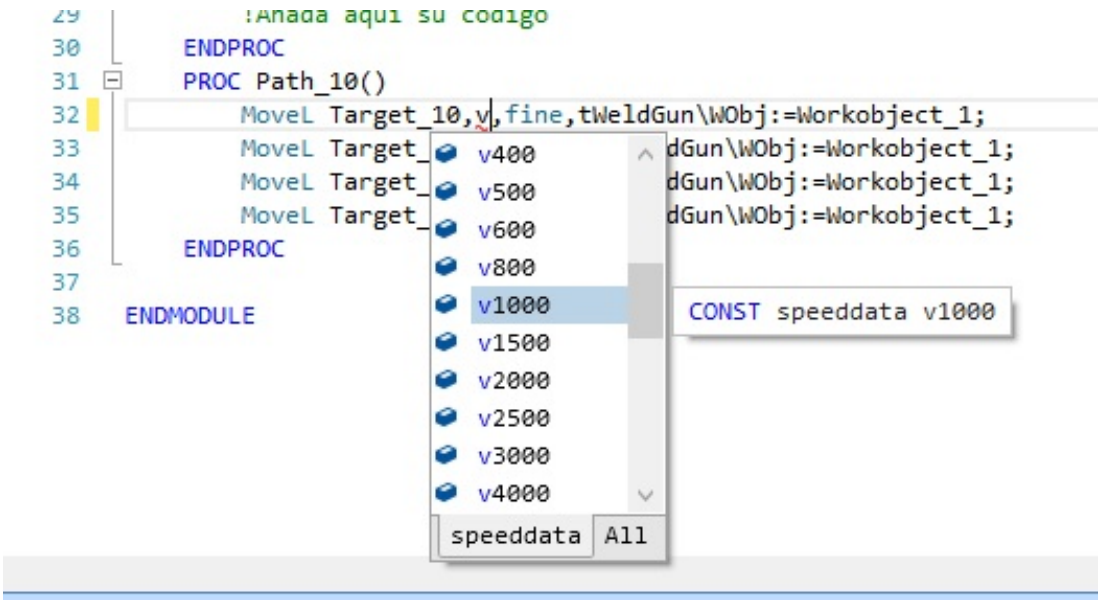


Figura 255: Edición en RAPID I.

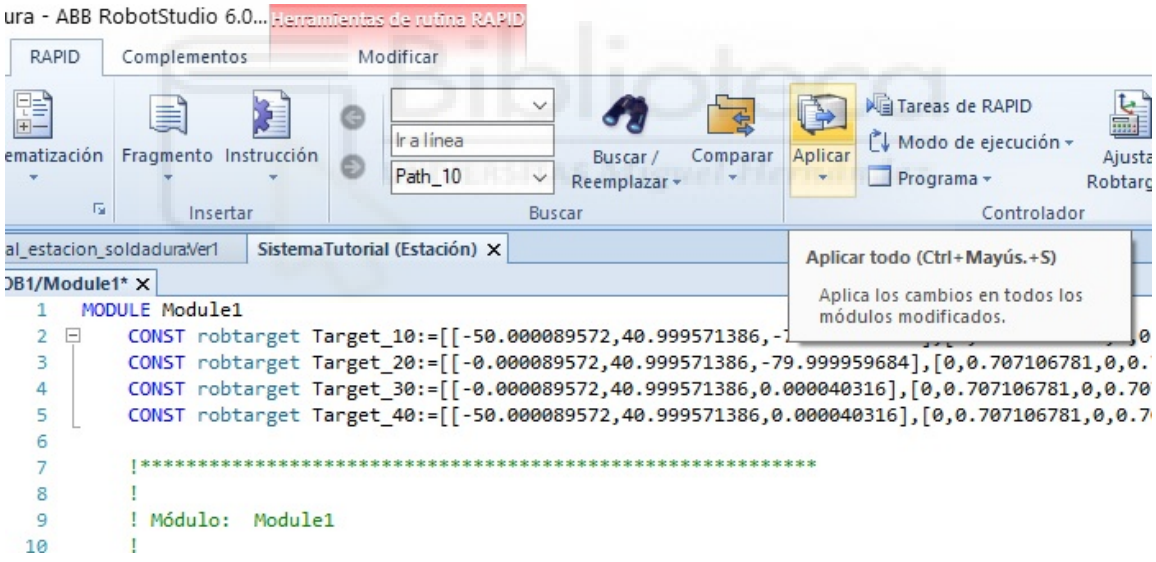


Figura 256: Edición en RAPID II.

Para sincronizar los cambios con la estación haremos clic sobre “**Sincronizar con la estación**”

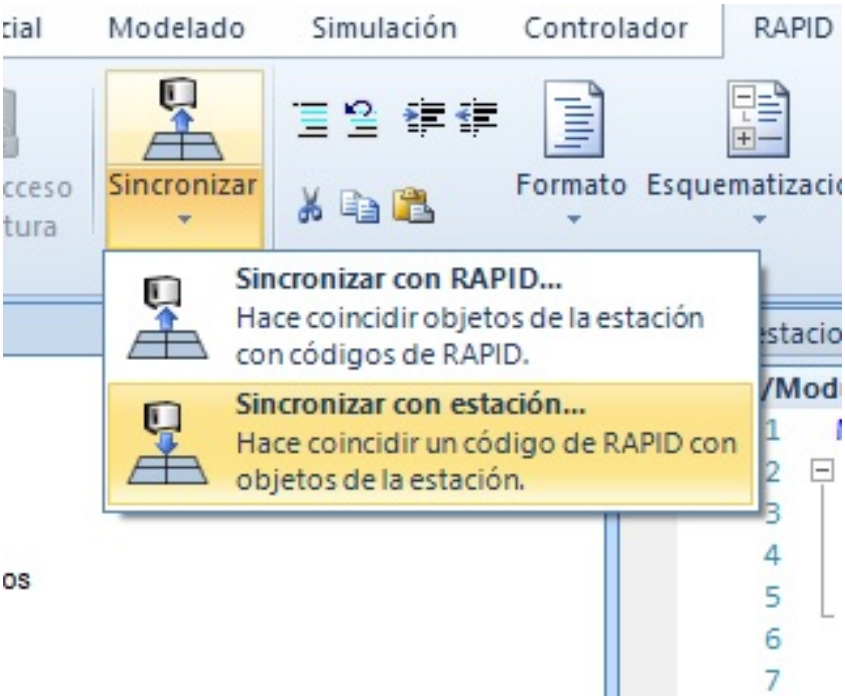


Figura 257: Sinconizar con estación.

En la programación de las trayectorias es muy importante la posición y ángulo de la pistola de soldadura y también tienen mucha importancia los movimientos de un objetivo al siguiente y de una trayectoria a la siguiente, en especial los giros de las tres últimas ejes del robot, ya que pueden provocar que se enrolle el hilo de soldadura que alimenta pistola de soldadura. Todo ello hace que en la tarea de programación de los objetivos apliquemos un proceso iterativo de prueba y error hasta conseguir un resultado válido.

La posibilidad de simular el comportamiento de un cable será de mucha ayuda para visualizar el posible movimiento del hilo de alimentación y se explicará en el apartado 5.4.15.

5.4.10 SmartComponents.

La estación de soldadura dispondrá vallado perimetral y dos barreras inmateriales como medida de seguridad, de forma que cuando el robot esté trabajando sobre uno de los chasis, si se entra en esa zona se parará, mientras que en la zona del otro chasis se podrá entrar para preparar el próximo chasis a soldar y así sucesivamente.

Para simular en RobotStudio la barrera inmaterial usaremos un componente inteligente. En la pestaña “**Modelado**” de la cinta superior hacemos clic sobre “**Componente inteligente**”.

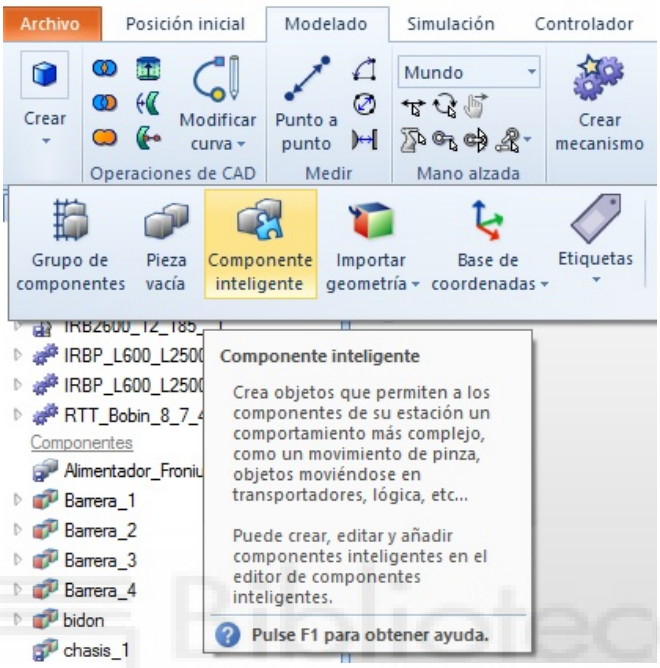


Figura 258: Creacion de CI de la barrera inmaterial I.

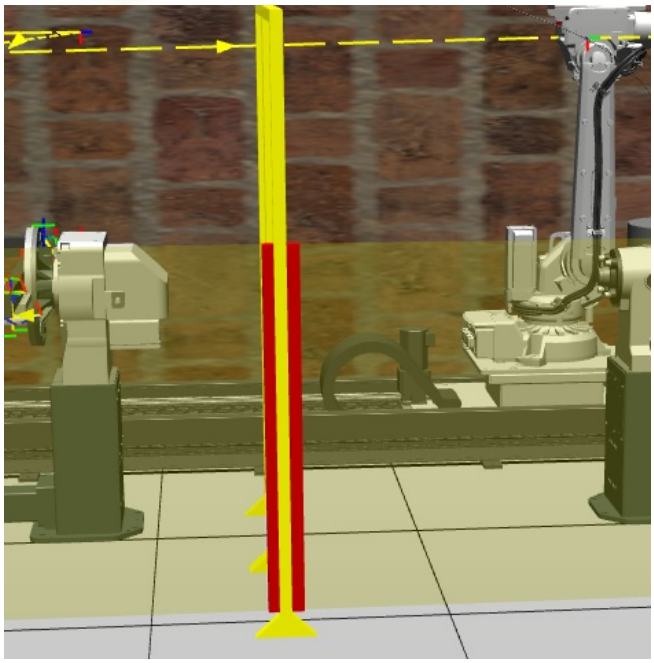


Figura 259: Creacion de CI de la barrera inmaterial II.

En la ventana que aparece hacemos clic sobre “**Añadir componente**” y seleccionamos en el grupo “**Sensores**”<<”**Plane sensor**”. Con este componente inteligente podemos dibujar un plano en la estación de forma que cualquier objeto que lo atraviese provocara que se genere una salida a “1”, manteniéndose a “0” el resto del tiempo. Esta salida la usaremos para detener el robot.

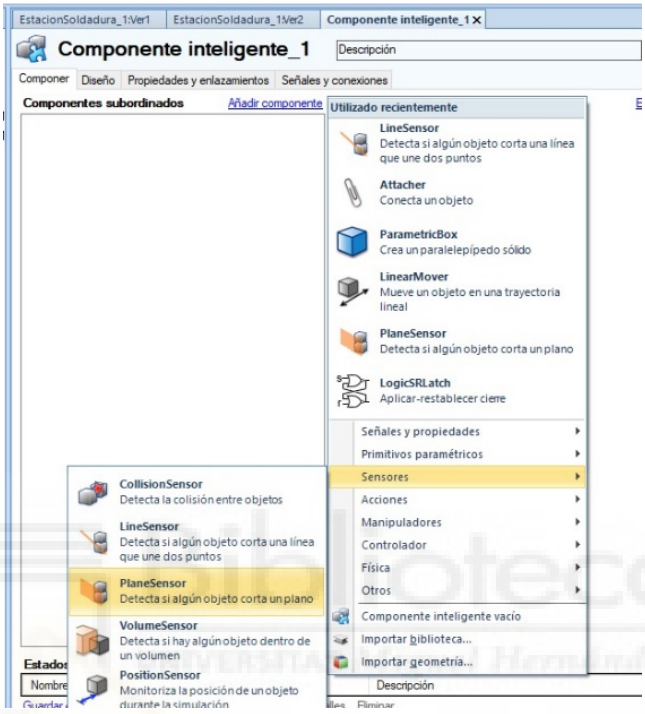


Figura 260: Creacion de CI de la barrera inmaterial III.

Añadiremos dos “**Plane sensor**” a la estación y usando las herramientas de modelado de RobotStudio crearemos cuatro prismas para simular el hardware emisor y detector de la barrera inmaterial. Realizaremos los planos con ayuda de las herramientas graficas de RobotStudio procurando que el plano sensor no toque ningún otro objeto, ya que en este caso, todo el tiempo nos estaría detectando dicho objeto y no serviría para nuestros fines. En las propiedades del componente inteligente podemos especificar el origen del plano y los ejes 1 y 2 que definirán el largo y ancho del plano. Podremos observar si está en contacto con algún objeto de la estación mediante el campo “**SensedPart**”

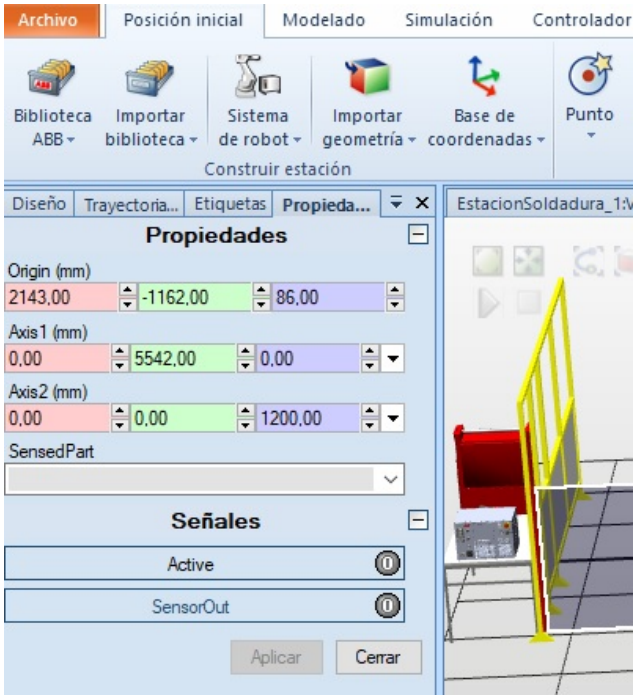


Figura 261: Creacion de CI de la barrera inmaterial IV.

En la siguiente imagen podemos observar la celda con el vallado y los sensores.

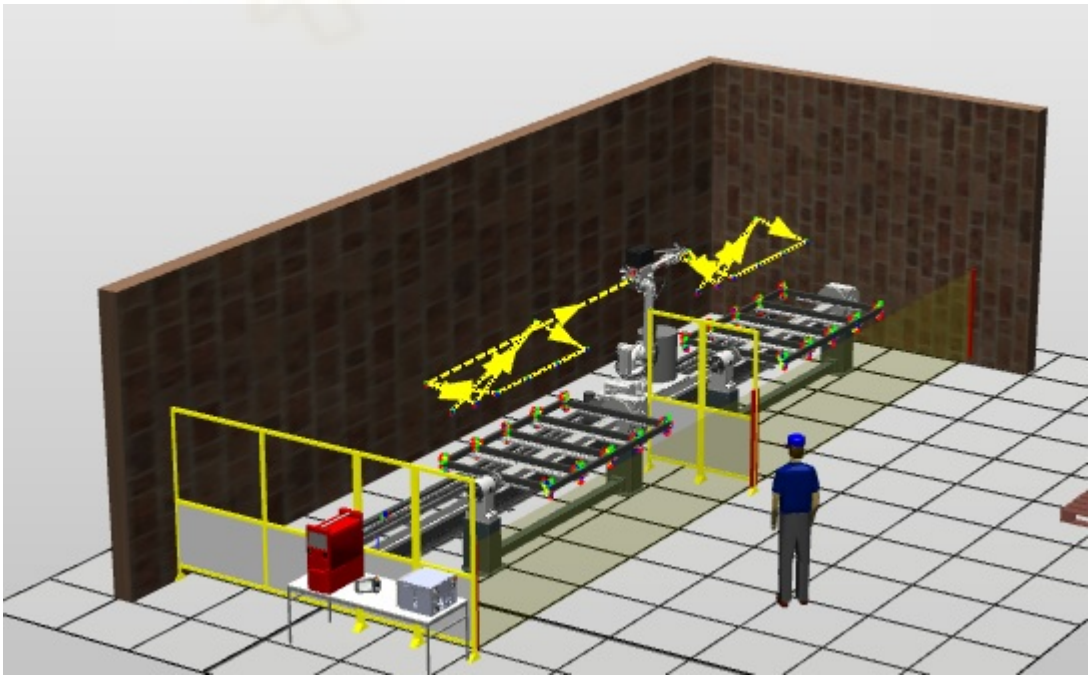


Figura 262: Estación de soldadura.

5.4.11 Señales de E/S.

Las señales necesarias en la estación serán:

Tabla 3: Señales de la estación de soldadura.

TIPO DE SEÑAL	NOMBRE	DESCRIPCIÓN
ENTRADA DIGITAL	Sensor1	Señal de detección del sensor de barrera chasis 1
	Sensor2	Señal de detección del sensor de barrera chasis 2
	Chasis1	Selección de chasis 1 para soldar
	Chasis2	Selección de chasis 2 para soldar
	Continua	Señal de continuar tras una parada por detección del sensor de la barrera
	Limpiar	Señal para llevar la pistola de soldadura a estación de limpieza
SALIDA DIGITAL	ActivaSensor1	Activa la barrera inmaterial de seguridad 1
	ActivaSensor2	Activa la barrera inmaterial de seguridad 2
	Soldar	Activa la soldadura
	Soplar	Activa la sopladora de la estación de limpieza
	Cortar	Activa cortadora hilo soldadura de la estación de limpieza

Crearemos las señales desde la pestaña "**Controlador**" haciendo doble clic sobre "**Configuración**"<<"**I/O System**".

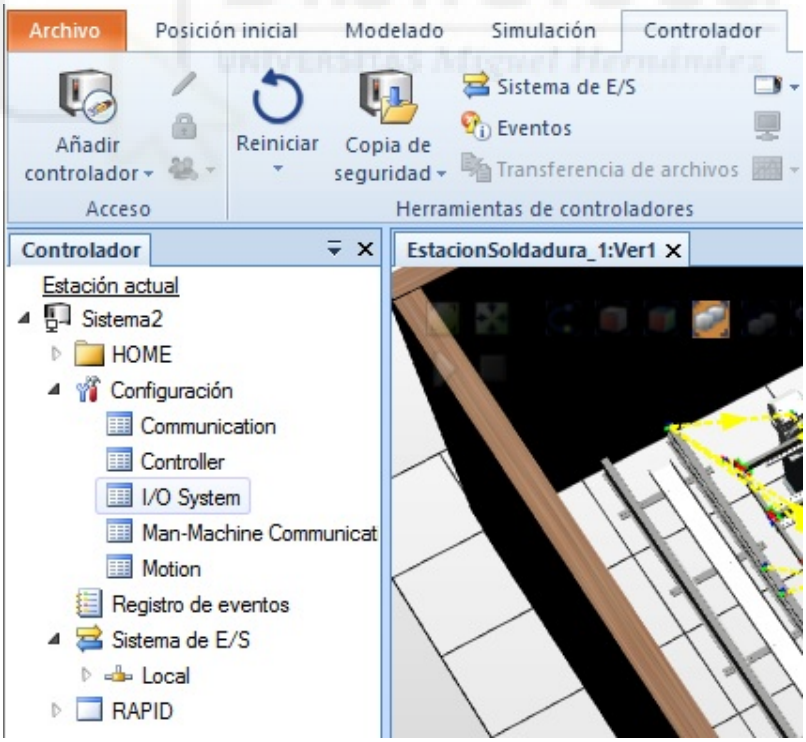


Figura 263: Creación de E/S de la estación I.

En la ventana que aparece hacemos clic con el botón derecho del ratón sobre “**Signal**” y seleccionamos “**Nuevo Signal…”**. En la ventana que aparece definimos el nombre de la señal, el tipo de señal y la tarjeta de señales la dejamos en blanco (RobotStudio permite simular las señales sin necesidad de asignarlas a ningún dispositivo hardware).

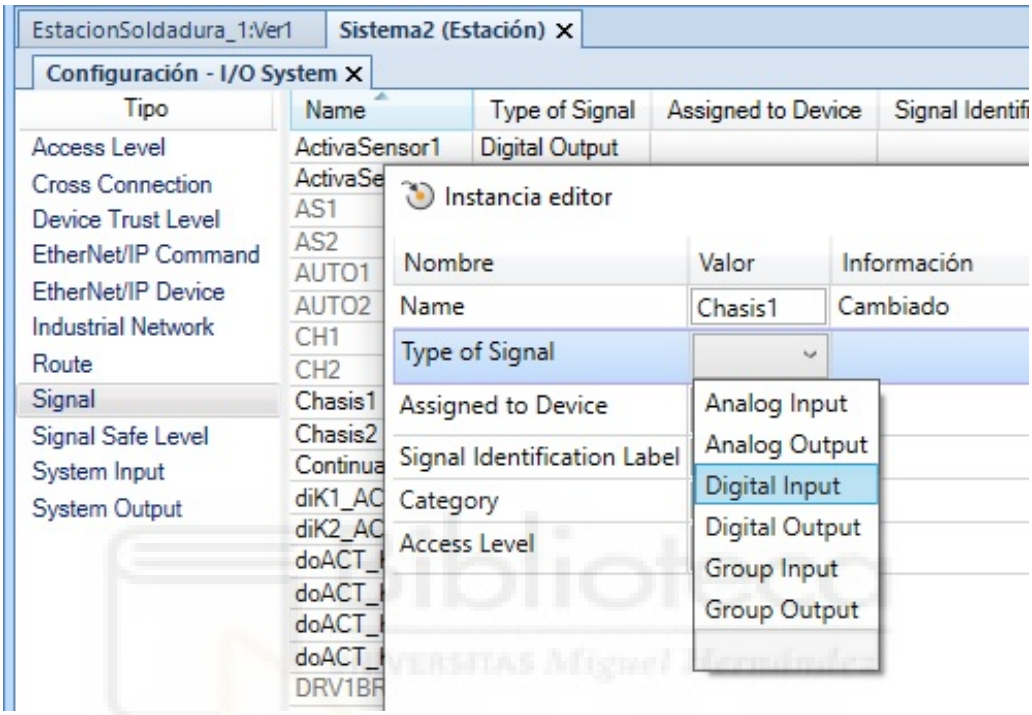


Figura 264: Creación de E/S de la estación II.

De esta forma crearemos todas las señales definidas en la tabla anterior.

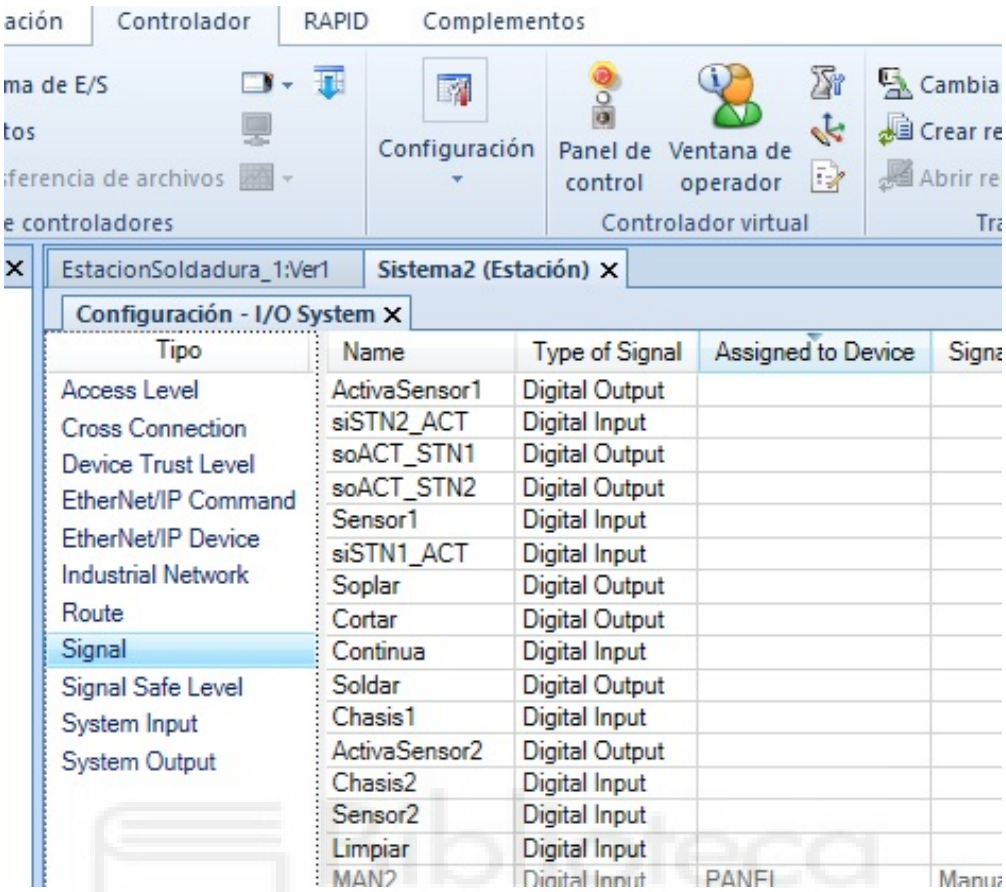


Figura 265: Creación de E/S de la estación III.

5.4.12 Lógica de la estación.

Para implementar la simulación de la estación de soldadura en RobotStudio emplearemos los dos componentes inteligentes que creamos en el apartado **5.4.10** para simular las barreras inmateriales de seguridad y las siguientes entradas y salidas digitales creadas en el apartado **5.4.11**.

Tabla 4: Señales de E/S en Lógica de la estación.

TIPO DE SEÑAL	NOMBRE	DESCRIPCIÓN
ENTRADA DIGITAL	Sensor1	Señal de detección del sensor de barrera chasis 1
	Sensor2	Señal de detección del sensor de barrera chasis 2
SALIDA DIGITAL	ActivaSensor1	Activa la barrera inmaterial de seguridad 1
	ActivaSensor2	Activa la barrera inmaterial de seguridad 2

En la pestaña simulación hacemos clic sobre “**Lógica de la estación**”

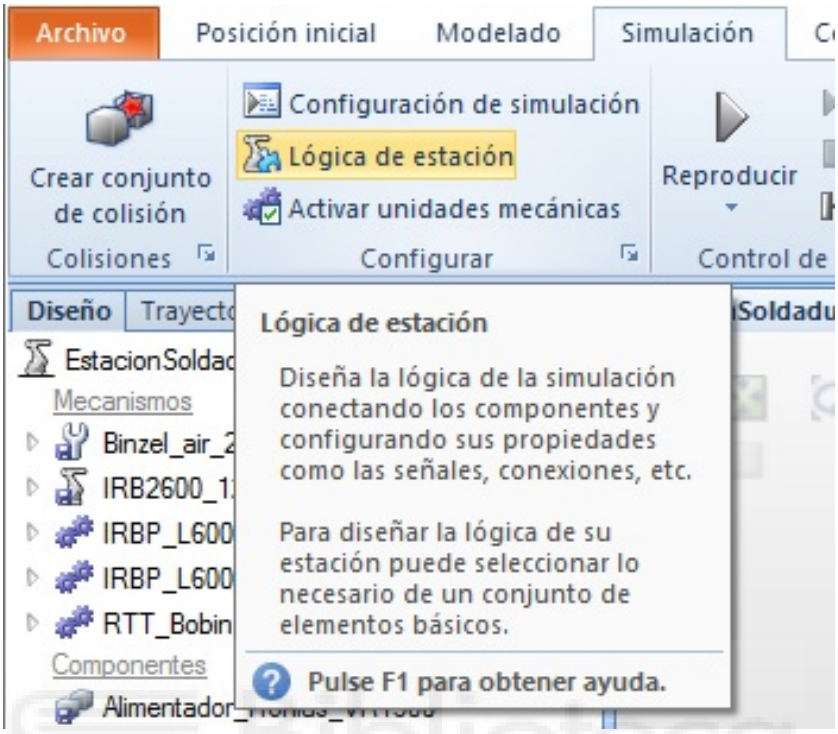


Figura 266: Lógica de la estación de soldadura I.

La ventana que aparece dispones de las siguientes pestañas:

- **Componer**: Podemos ver los componentes inteligentes creados y otros objetos de la estación.
- **Diseño**: En esta pestaña podemos establecer gráficamente la lógica de la estación conectando entradas y salidas entre diferentes componentes, enlazamiento entre propiedades, así como tener acceso a sus propiedades.
- **Propiedades y enlazamientos**: podemos añadir enlazamientos entre las propiedades de los componentes de forma manual.
- **Señales y conexiones**: podemos añadir conexiones entre las entradas y salidas de los componentes de forma manual

En este caso, emplearemos las salidas digitales del sistema denominadas **ActivaSensor1** y **ActivaSensor2** para conectarlas con los componentes inteligentes que simulan las barreras inmateriales de forma que nos permita activarlas o desactivarlas mediante software a través de su entrada “**Active()**”. También emplearemos las entradas digitales

del sistema **Sensor1** y **Sensor2** conectadas a las salidas "**SensorOut()**" de los componentes inteligentes de las barreras inmateriales de forma que cuando tenemos un uno en dichas salidas nos da la señal de detección y el robot debe pararse.

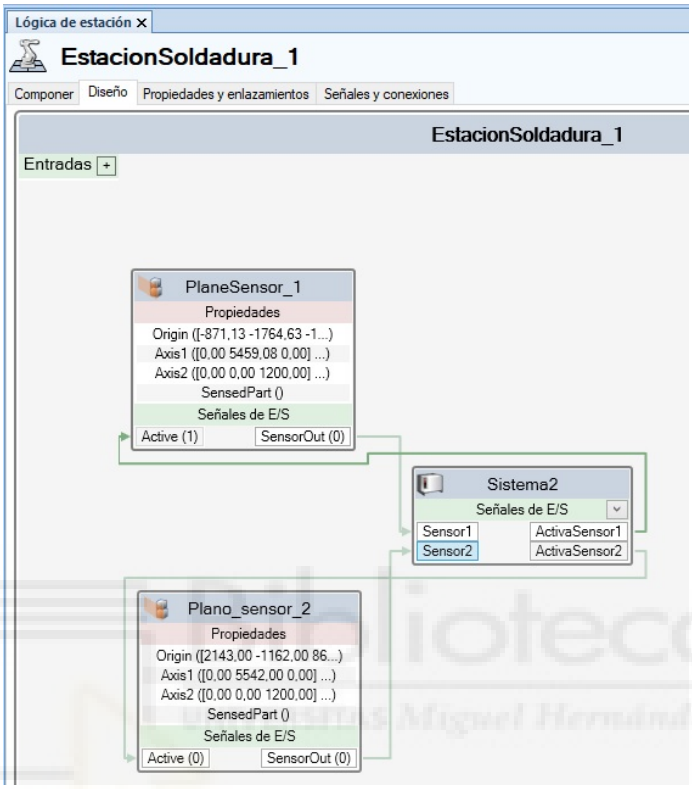


Figura 267: Lógica de la estación de soldadura II.

Para facilitar la tarea al operador del robot en la estación de soldadura se dispondría de una botonera de pulsadores que una vez cargado el programa de soldadura correspondiente le permitirá interaccionar con el programa de soldadura sin tener que usar el FlexPendant de la siguiente forma:

Tabla 5: Pulsadores de la estación de soldadura.

PULSADOR	ACCIÓN
Chasis 1	Pone en marcha la soldadura en el chasis nº 1
Chasis 2	Pone en marcha la soldadura en el chasis nº 2
Continua	Reanuda la soldadura tras una parada por detección de la barrera inmaterial
Limpieza	Leva la pistola de soldadura a la estación de limpieza.

5.4.13 Programa RAPID.

El programa RAPID de la estación de soldadura constará básicamente de un bucle "**main**" que en función de la entrada activada en la botonera de pulsadores del operador, pondrá en marcha el programa de soldadura del chasis 1, del chasis 2 o de la limpieza de la pistola de soldadura. Las entradas de detección de las barreras inmateriales, se atenderán mediante interrupciones TRAP de forma que la parada del robot sea inmediata.

```
PROC main()

!Conectamos rutina interrupción con rutina trap barrera 1
CONNECT barrera1int WITH barrera1Trap;

!Conectamos rutina interrupción con rutina trap barrera 2
CONNECT barrera2int WITH barrera2Trap;

!Solicitamos interrupción cuando pase a 1 la barrera 1
ISignalDI Sensor1,high,barrera1int;

!Solicitamos interrupción cuando pase a 1 la barrera 2
ISignalDI Sensor2,high,barrera2int;

WHILE (TRUE) DO
IF DInput(CHASIS1)=1 THEN
CHASIS_1;
ELSEIF DInput(CHASIS2)=1 THEN
CHASIS_2;
ELSEIF DInput(Limpiar)=1 THEN
LIMPIEZA;
ENDIF
ENDWHILE
```

ENDPROC

La rutina **TRAP** parará de inmediato el robot al activarse la barrera de la zona en que está trabajando el robot sobre uno de los chasis y esperará hasta que el operador restablezca la tarea en proceso mediante el pulsador "**Continua**" de la botonera. Se guardarán los datos necesarios de la trayectoria en el punto de parada y se realizara una limpieza de la pistola de soldadura para volver después al punto de parada y reanudar el trabajo.

```
TRAP barrera1Trap
    VAR robtarget stop_pos;
    !Para robot, guarda trayectoria, herramienta y objeto de trabajo actuales
    StopMove;
    StorePath;
    GetSysData tWeldGun;
    GetSysData Workobject_1;
    stop_pos:=CRobT(\Tool:=tWeldGun\WObj:=Workobject_1);
    WaitUntil CONTINUA=1;
    WaitTime(3);
    ! Realiza limpieza
    MoveJ RelTool(stop_pos,0,0,-200),v50,fine,tWeldGun\WObj:=Workobject_1;
    MoveJ RelTool(stop_pos,300,300,-200),v1000,fine,tWeldGun\WObj:=Workobject_1;
    LIMPIEZA;
    ! Vuelve a la posición de parada, restaura trayectoria y pone en marcha el robot
    ActUnit STN2;
    MoveJ RelTool(stop_pos,300,300,-400),v1000,fine,tWeldGun\WObj:=Workobject_1;
    MoveJ RelTool(stop_pos,0,0,-200),v50,fine,tWeldGun\WObj:=Workobject_1;
    MoveJ stop_pos,v50,fine,tWeldGun,\WObj:=Workobject_1;
    RestoPath;
    StartMove;
ENDTRAP
```

5.4.14 Colisiones.

RobotStudio dispone de una herramienta para la detección de colisiones de forma que durante la simulación podemos detectar movimientos que ocasionarían colisiones con objetos o equipos de la estación. Es importante por tanto realizar un modelado gráfico lo más fiel posible de todos los elementos presentes en la estación y con los que el robot pudiera colisionar por esta dentro de su zona de trabajo.

Podemos crear un conjunto de colisión desde la pestaña “Simulación”<<”Crear conjunto de colisión”. Para configurar el comportamiento durante la simulación de la detección de colisiones podemos en acceder a las Opciones desde la pestaña “Archivo” de RobotStudio. También es posible acceder a las opciones del grupo Colisiones de la cinta superior haciendo clic sobre el pequeño icono de la parte inferior izquierda.

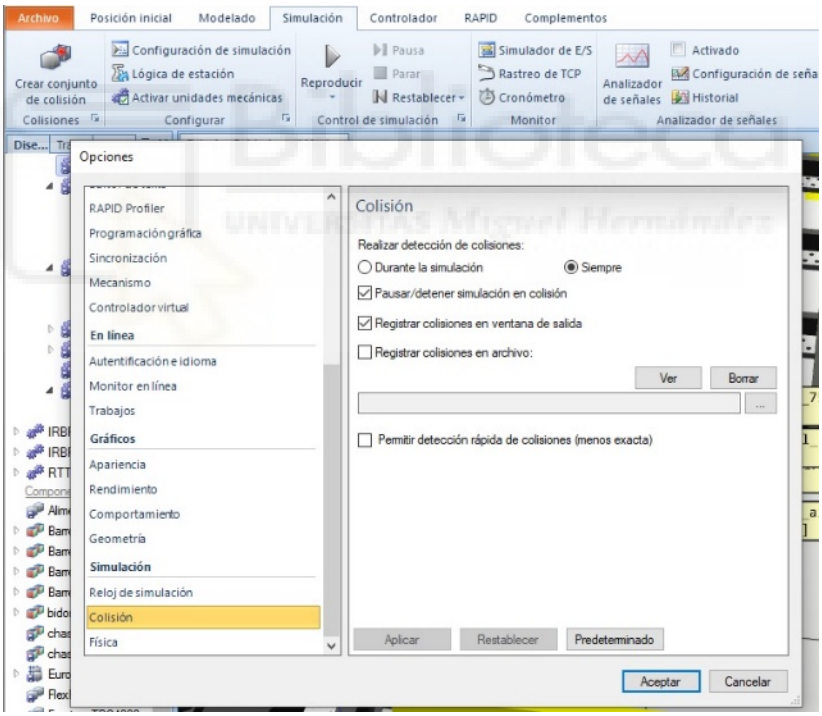


Figura 268: Conjunto de colisiones de la estación de soldadura I.

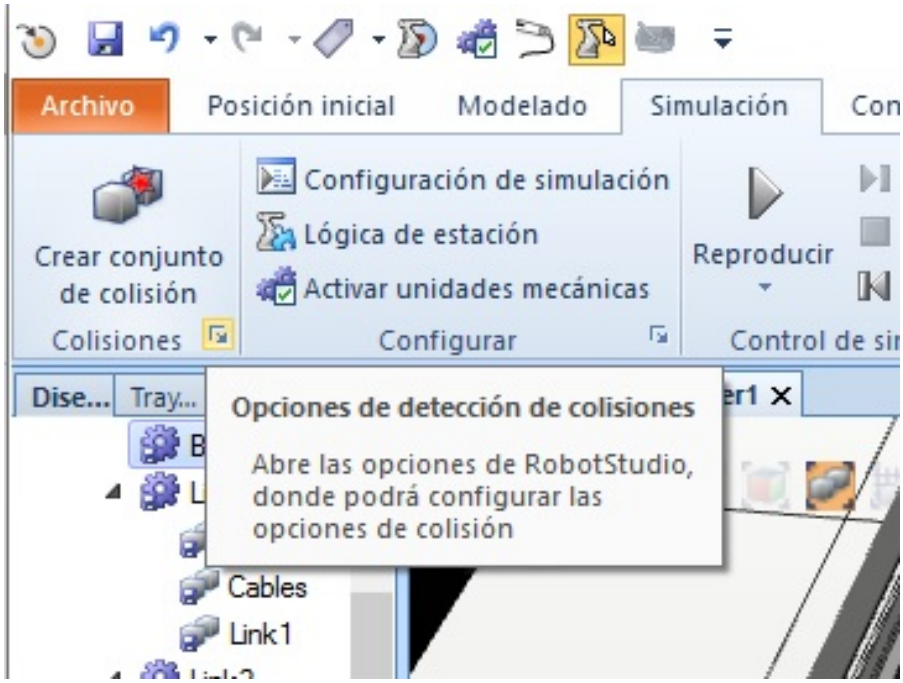


Figura 269: Conjunto de colisiones de la estación de soldadura II.

Para la simulación de la estación se han creado dos conjuntos de colisión. En el primero, en la carpeta **ObjetosA** se ha incluido el robot y la pistola de soldadura y en la carpeta **ObjetosB** el resto de elementos de la estación tales como vallas, posicionadores, muros, etc que están en la zona de trabajo del robot. El track no se ha incluido en este conjunto de colisión ya que la base del robot está en contacto con el y provocaría el aviso de colisión de forma continua, para ello se ha creado un segundo conjunto de colisión en el que en la carpeta **ObjetosA** se ha incluido la pistola y todos los links del robot excepto el **link1** que es la base y en la carpeta **ObjetosB** se ha incluido el track.

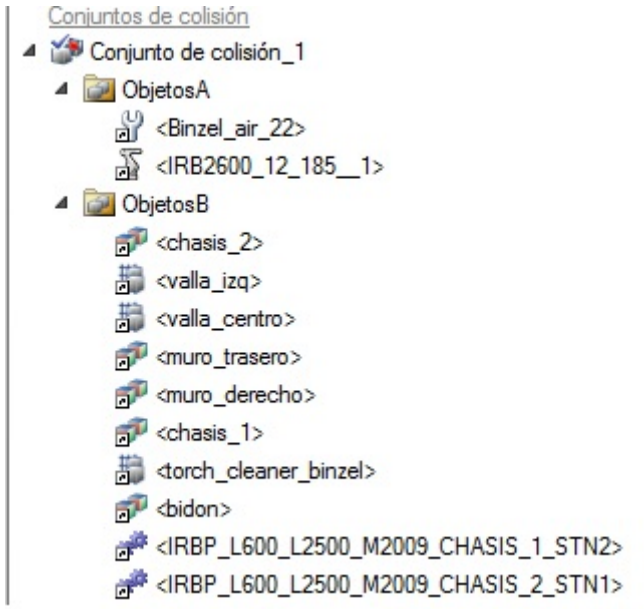


Figura 270: Conjunto de colisiones de la estación de soldadura III.

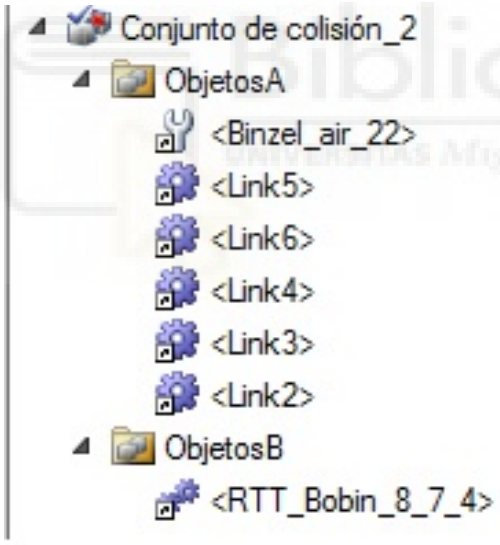


Figura 271: Conjunto de colisiones de la estación de soldadura IV.

De esta forma durante las simulaciones del programa, si hay alguna colisión, el programa se detendrá y marcará en pantalla los objetos que ha colisionado.

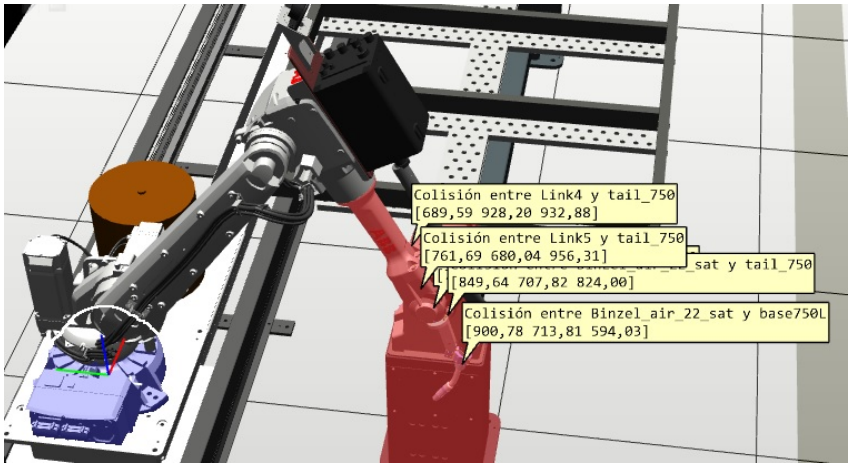


Figura 272: Conjunto de colisiones de la estación de soldadura V.

5.4.15 Simulación.

Antes de simular el programa debemos guardar todos los cambios realizados en el controlador haciendo clic en “Sincronizar” en la pestaña “Posición inicial” y luego hacemos clic en la pestaña “Simulación” de la cinta superior para acceder a todas la herramientas de simulación.

La pestaña simulación dispone de seis grupos de herramientas:

- **Colisiones**: en este grupo podemos crear los conjuntos de colisiones.
- **Configurar**: en este grupo disponemos de herramientas para:
    - Configurar la estación: Podemos elegir entre los objetos a simular, si la simulación va a realizar un solo ciclo o de forma continua, crear diferentes escenarios de simulación con diferentes estados iniciales de los objetos. En este caso seleccionaremos todos los objetos y el modo de ejecución de un solo ciclo.

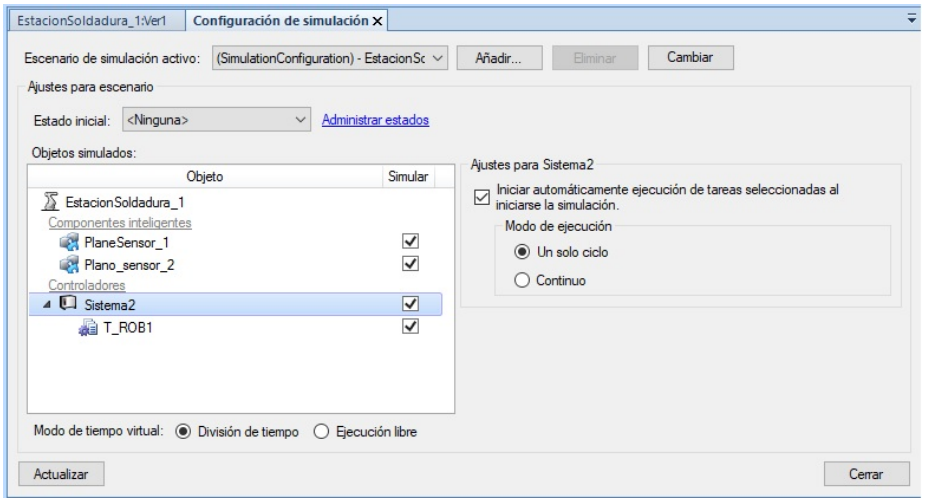


Figura 273: Simulación de la estación I.

- Lógica de la estación: Podemos establecer las señales y conexiones entre los componentes de la estación y el sistema y las propiedades y enlazamientos de los objetos. En este caso se establecen las conexiones entre las señales de los componentes inteligentes que simulan las barreras inmateriales y el sistema.

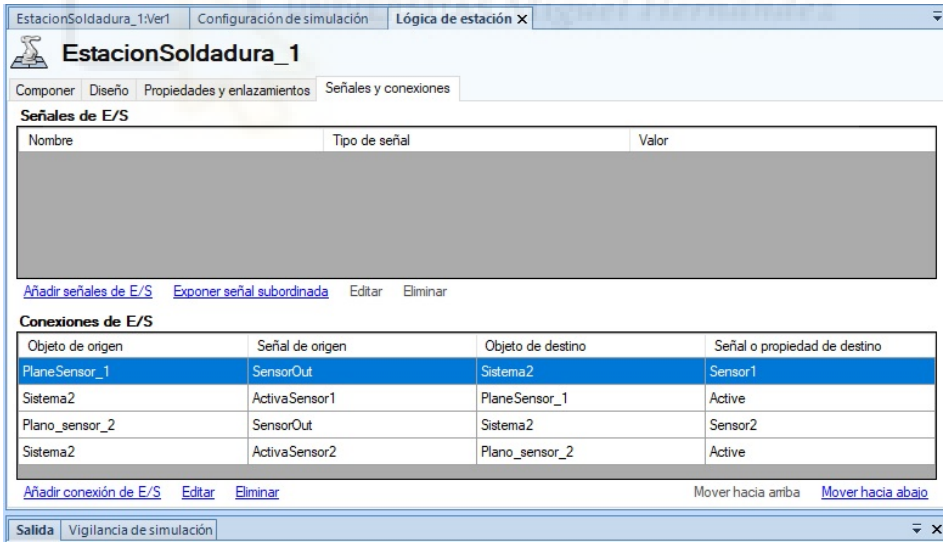


Figura 274. Simulación de la estación II.

- Activar unidades mecánicas: Podemos activar de forma manual las unidades mecánicas para que se puedan mover durante la simulación si no se ha hecho por software mediante las órdenes ActUnit y DeactUnit.

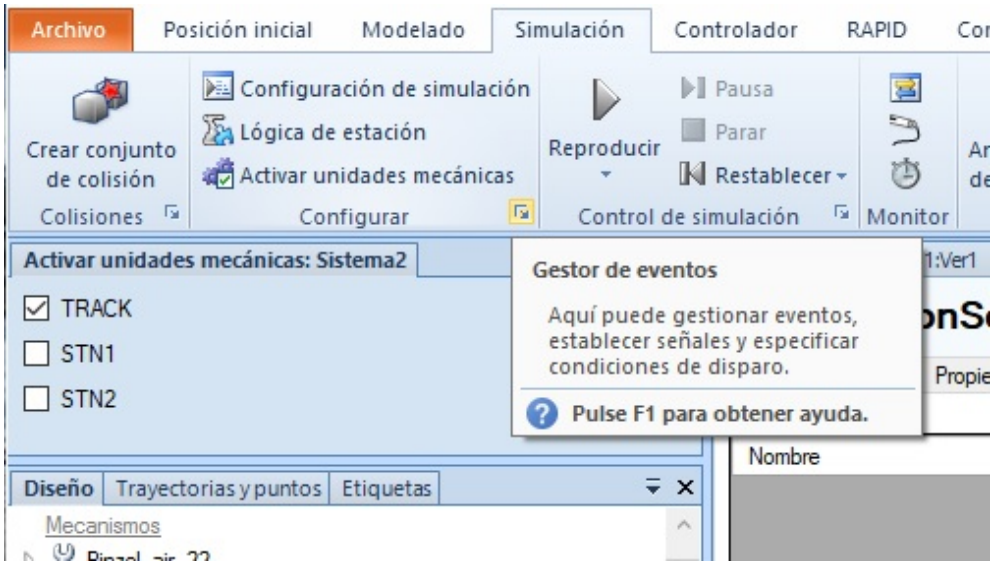


Figura 275: Simulación de la estación III.

- Gestor de eventos: Podemos acceder al gestor de eventos haciendo clic sobre el pequeño icono de la esquina inferior derecha del grupo. En el gestor de eventos podemos crear eventos ligados a señales del sistema o de la estación para activar/desactivar señales, conectar objetos, producir movimientos en mecanismos, etc.. En esta estación no se hace uso de los eventos.

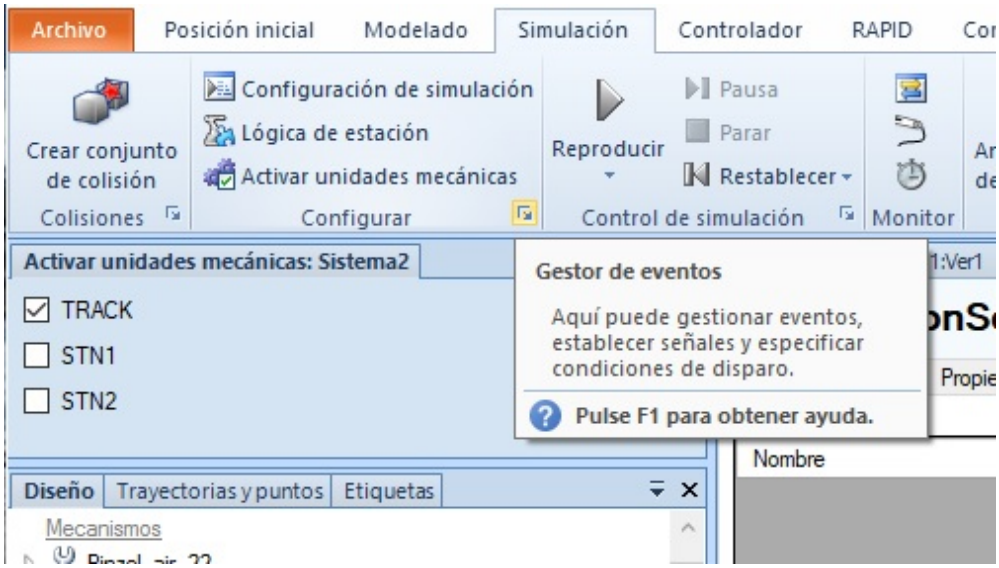


Figura 276: Simulación de la estación IV.

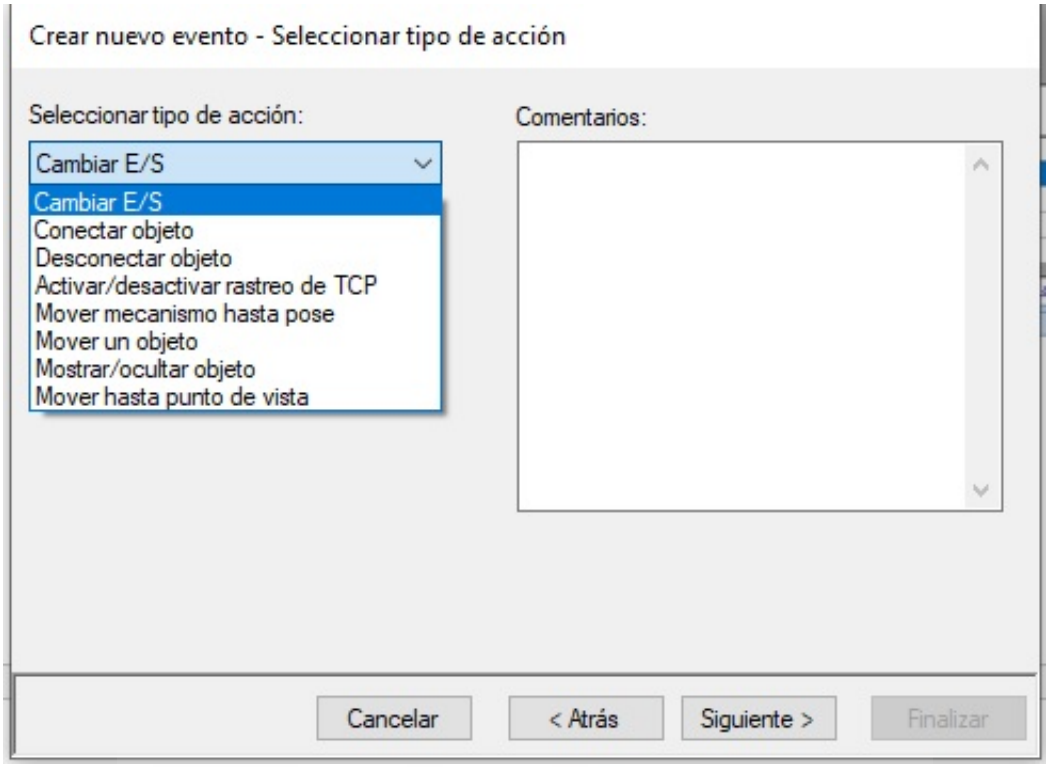


Figura 277: Simulación de la estación V.

- Control de simulación: disponemos de botones para reproducir, sincronizar con RAPID y reproducir, reproducir y grabar la simulación, parar, pausa y restablecer y mediante el pequeño icono de la esquina inferior derecha acceder a las opciones de simulación como la velocidad del reloj de simulación para acelerarla y que tarde menos tiempo.

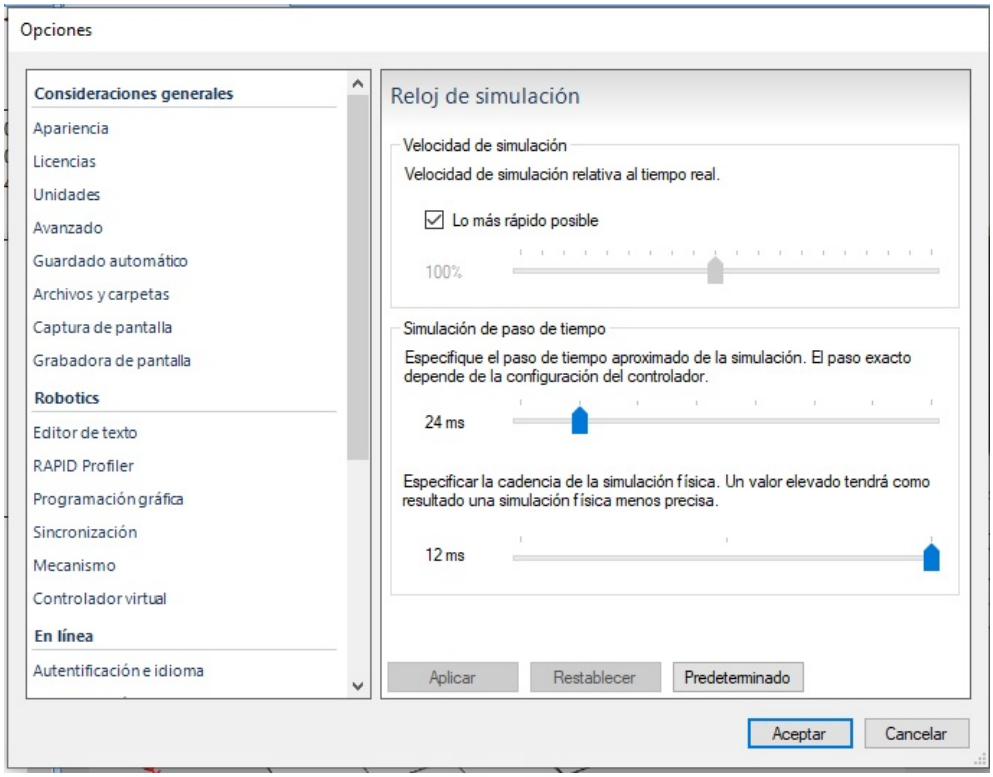


Figura 278: Simulación de la estación VI.

- Monitor: en este grupo disponemos de:
    - Rastreo del TCP: Podemos activar el rastreo, cambiar color de las trayectorias incluso en función de la señal. Con el rastreo del TCP podemos visualizar las trayectorias que describe el TCP activo del robot en sus movimientos.

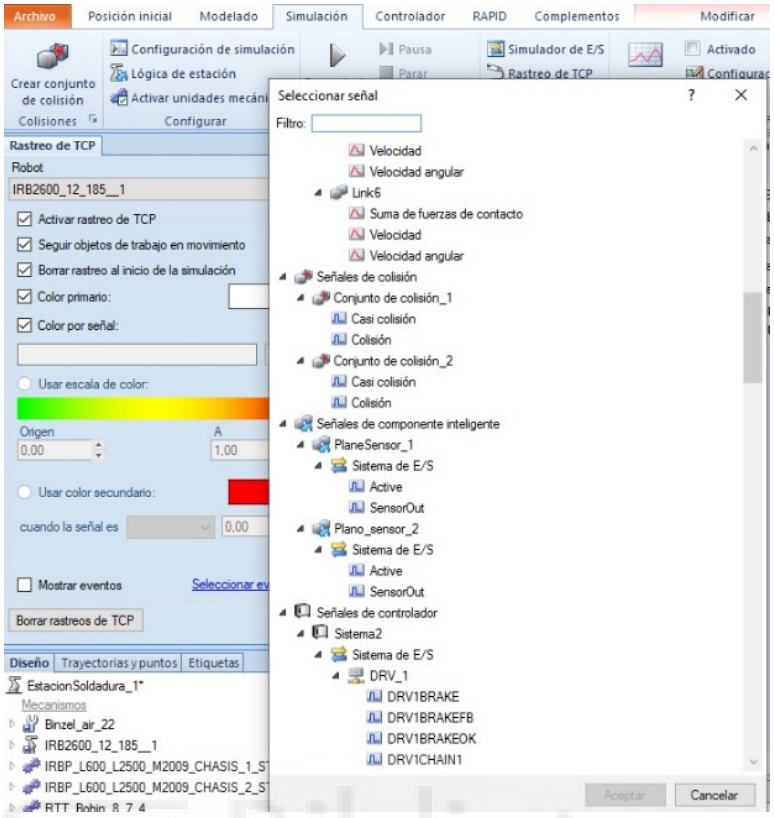


Figura 279: Simulación de la estación VII.

- Simulación de entradas y salidas: Podemos visualizar el estado de las señales de salida y activar/desactivar las señales de entrada del sistema, estación y componentes inteligentes así como crear listas personalizadas de las señales que queremos visualizar durante la simulación. En esta estación se ha creado una lista de usuario denominada SOLDAR en la que se han incluido las señales de la tabla XXX para simular los botones del operador de la estación mediante las señales de entrada y visualizar las señales de salida creadas.

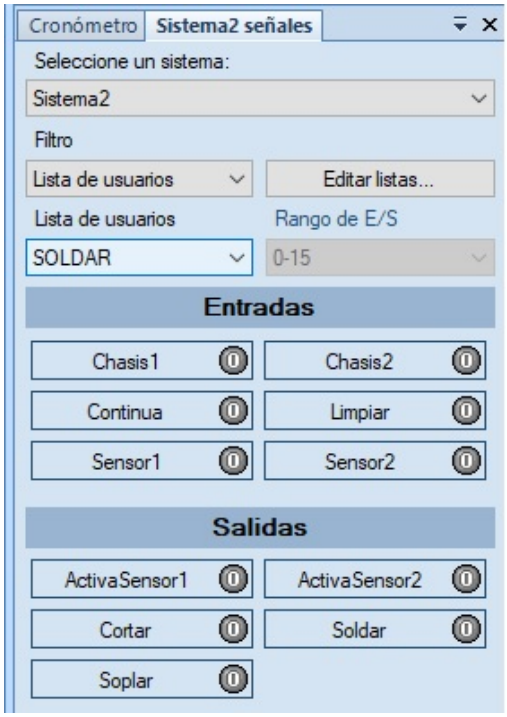


Figura 280: Simulación de la estación VIII.

- Cronómetro: podemos crear diversos cronómetros para medir los tiempos de toda la simulación, entre puntos objetivo seleccionados o entre señales del sistema.

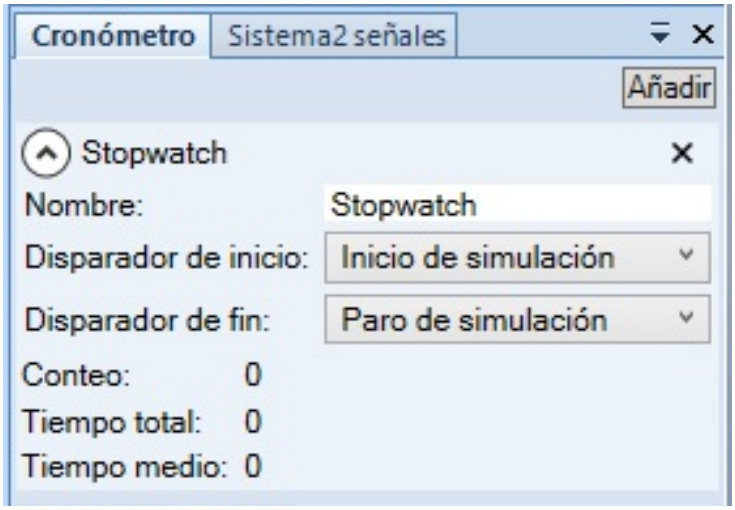


Figura 281: Simulación de la estación IX.

- Analizador de señales: En este grupo podemos elegir las señales a analizar en “Configuración de señales, realizar una simulación y posteriormente en “Analizador de señales” visualizar el comportamiento de las señales durante la simulación.

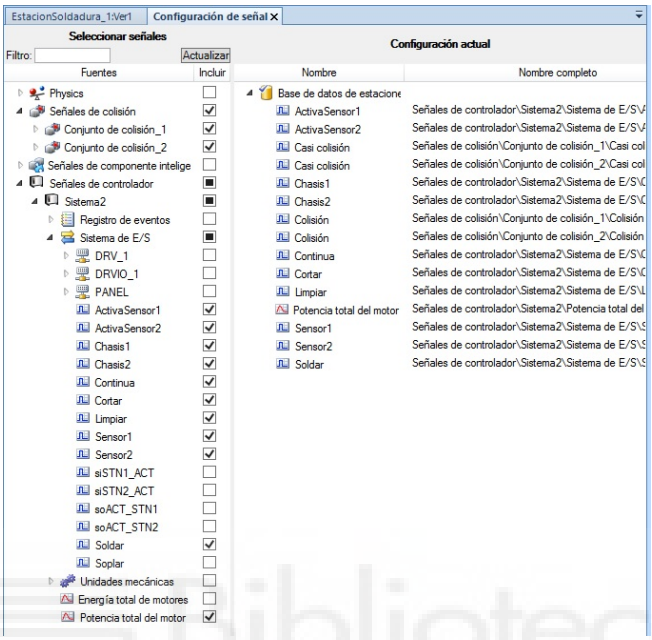


Figura 282: Simulación de la estación X.

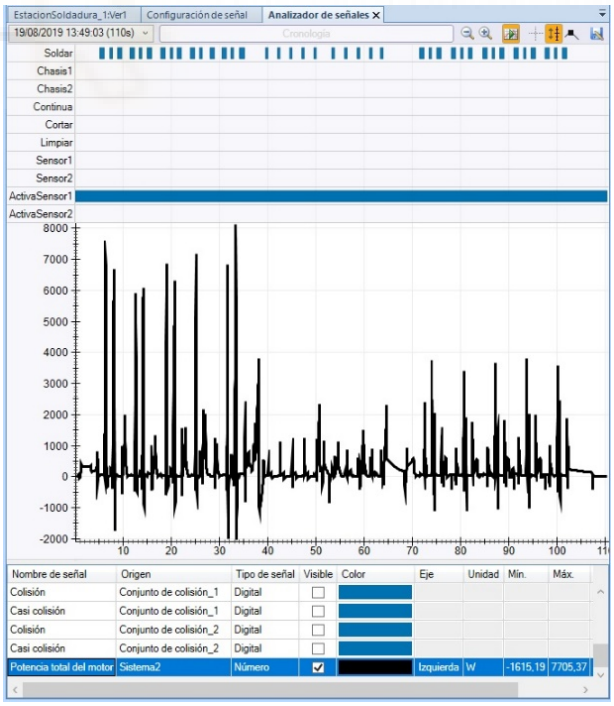


Figura 283: Simulación de la estación XI.

- Grabar película: En este grupo podemos encontrar las herramientas para grabar simulaciones o grabar la ventana de la aplicación y visualizar las grabaciones posteriormente.

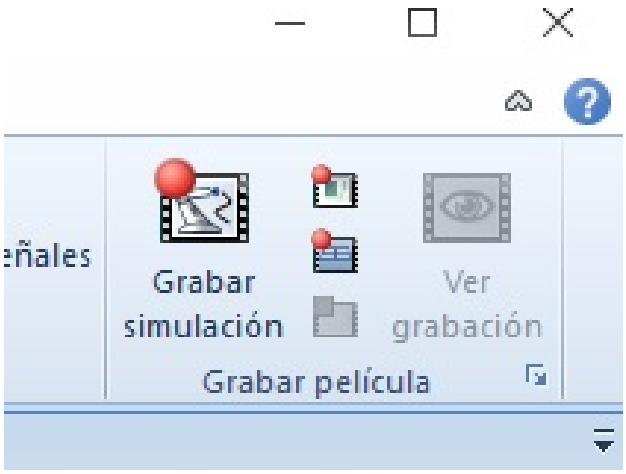


Figura 284: Simulación de la estación XII.

6. RESULTADOS.

6.1 Simulación de la estación de soldadura de chasis para maquinaria.

RobotStudio permite modelar cables y simular la física del movimiento de estos, lo cual sirve de mucha ayuda en la programación de la estación, ya que los movimientos del robot en el trabajo de soldadura deben realizarse de manera que no provoquen que el hilo de soldadura se enrolle.

Para modelar un cable hacemos clic sobre la pestaña "**Modelado**" de la cinta superior

Y después sobre "**Cable**" en el grupo "**Crear**".

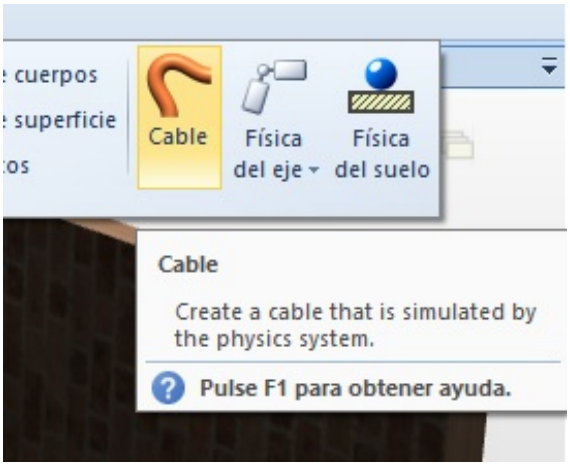


Figura 285: Simulación del cable I.

En la ventana que aparece, podemos especificar el radio del cable, la longitud, el material y su módulo de Young. Nos pide que marquemos un punto de inicio del cable y luego un punto final. Aparece dibujado el cable en pantalla y ahora podemos añadir puntos de control que podemos mover manualmente en el cable pinchando sobre el de manera que le podemos dar la forma deseada. Una vez conseguida la forma deseada hacemos clic sobre "Crea".

Pera realizar la simulación del trabajo de soldadura crearemos un modelo de cable para el hilo de soldadura que desde el alimentador Fronius TPS4000 situado sobre el link3 del robot hasta la pistola de soldadura denominado **PhysicsCable_1**.

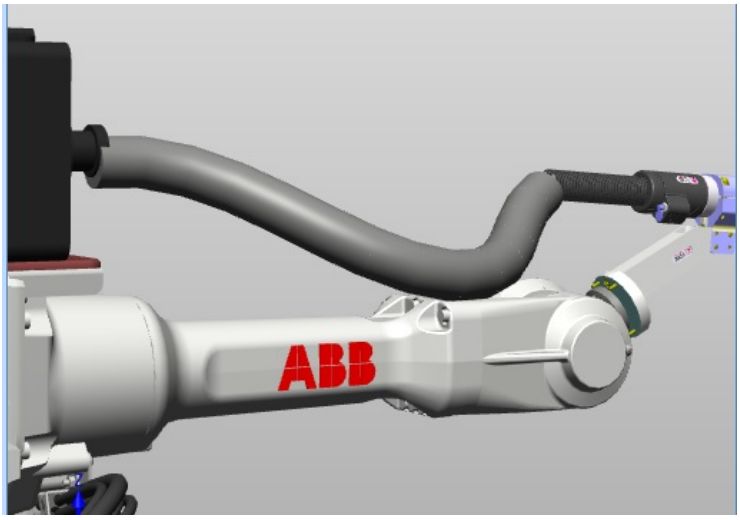


Figura 286: Simulación del cable II.

y otro modelo del hilo que va desde el bidón del hilo situado sobre el track junto al robot hasta el alimentador de hilo denominado **PhysicsCable_2**.

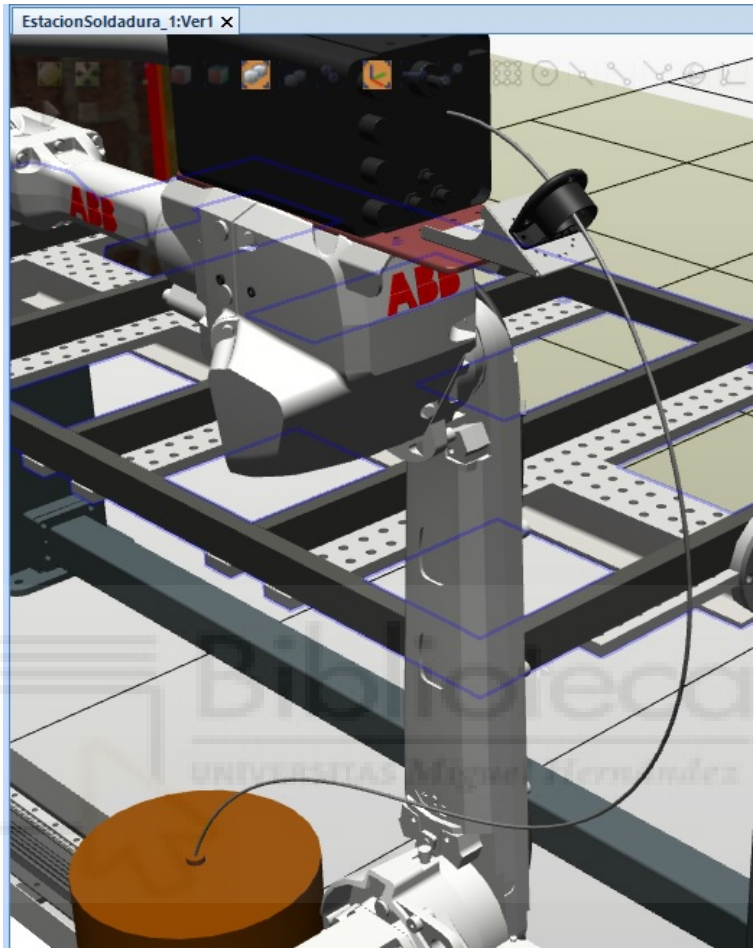


Figura 287: Simulación del cable III.

Los componente inteligentes denominados **PlaneSensor_1** y **PlaneSensor_2** simularán las barreara inmateriales que controlan el acceso a la zona de soldado de cada chasis. Cada componente inteligente dispone de una entrada digital denominada **Active** que permite activarlo o desactivarlo y de una salida digital denominada **SensorOut** que se pone a uno cuando detecta algo.

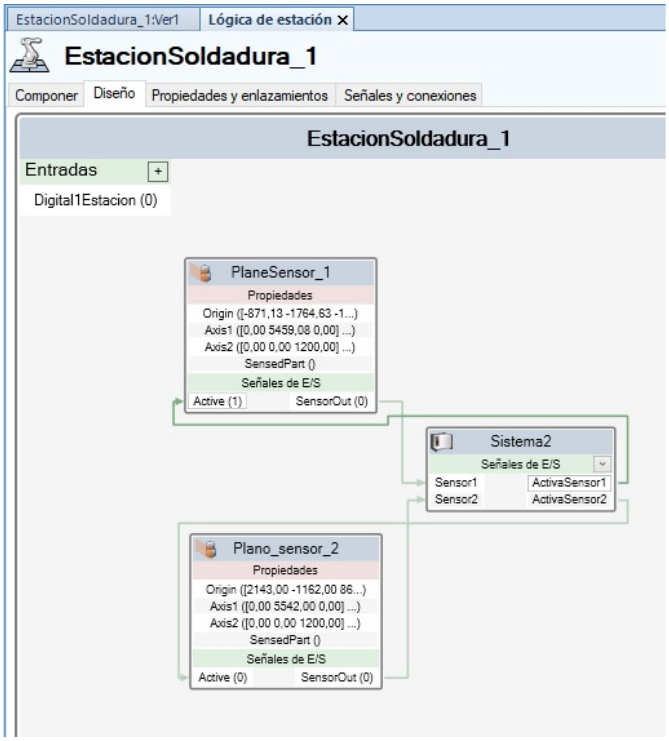


Figura 288: Conexiones de la lógica de la estación.

El programa de soldadura de cada chasis tiene una posición home desde la cual comienza y en la cual termina. Cuando en la pestaña “Simulación” hacemos clic sobre “Reproducir”, el robot permanece parado hasta que el operario pulsa sobre el botón de soldar uno de los dos chasis. Para simular esta acción en simulador de señales hacemos clic sobre los botones “Chasis1” o “Chasis2” para ponerlo a uno y de nuevo para ponerlo a cero, en ese momento comenzará el robot a soldar sobre el chasis

Figura 289: Simulador de las señales de entrada y salida de la estación.

En el programa RAPID, cuando se selecciona un chasis para soldar por parte del operario, se activará el sensor de la barrera de la zona de trabajo de dicho chasis mediante la salida digital del sistema **ActivaSensorX** y quedará desactivado el otro sensor de barrera, de forma que si el sensor activado detecta algún objeto o persona que accede a la zona donde se encuentra trabajando el robo lo parará de forma inmediata mientras que en la zona del otro chasis se permite el acceso para que el operario prepare el siguiente chasis a soldar. Para simular la detección de las barreras inmateriales, podemos mover durante la simulación un objeto para que pase a través del plano del sensor o bien simular la detección con los botones de entrada denominados "**SensorX**" en el simulador de E/S.

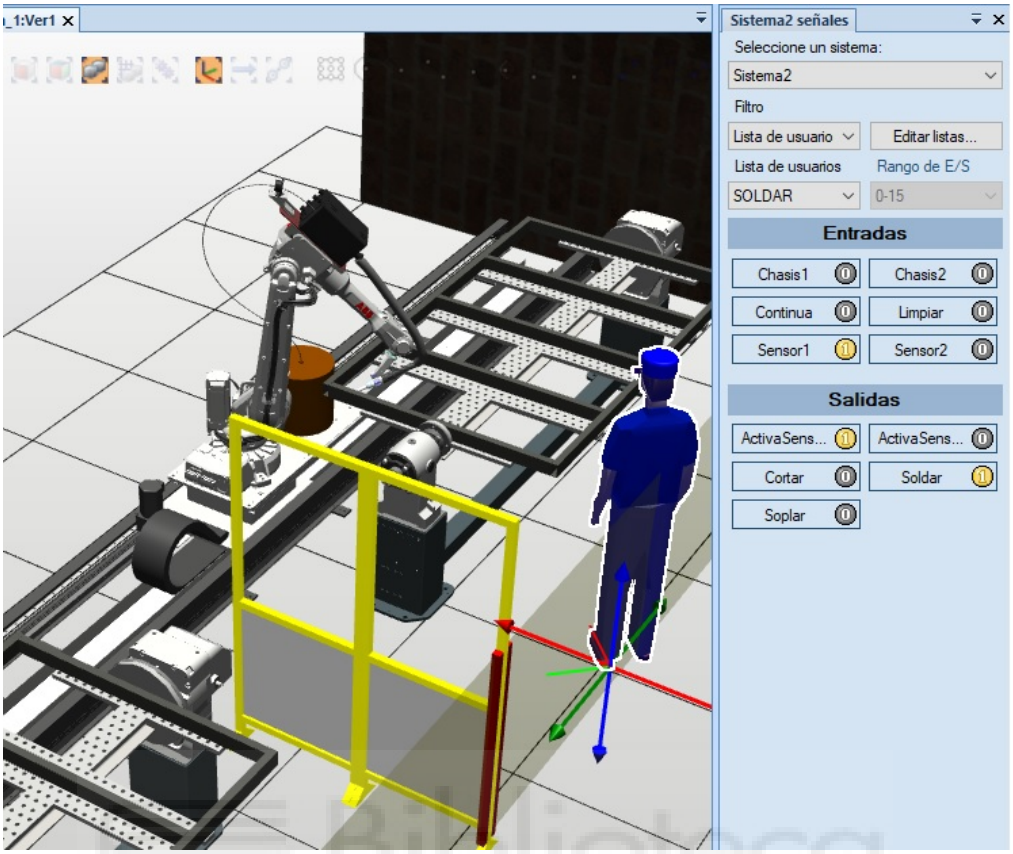


Figura 290: Parada del robot por activación de la barrera inmaterial.

Tras la parada por acceso a la celda y tras comprobar que no hay nadie en la zona de trabajo, el operador del robot debería accionar el pulsador de "**Continua**" de la botonera de la estación para que el robot comience de nuevo a trabajar realizándose, en primer lugar, una rutina de limpieza de la pistola de soldadura en la estación de limpieza y luego vuelve exactamente a la posición donde se produjo el paro y reanuda el trabajo. Para simular esta acción hacemos clic sobre el botón "**Continua**" del simulador de E/S una vez parado el robot.

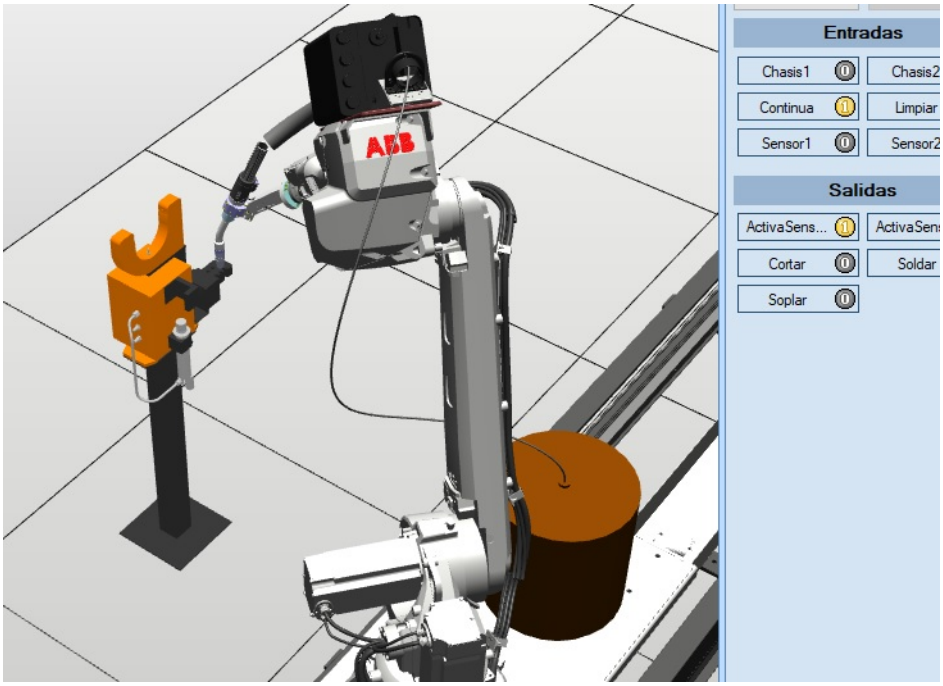


Figura 291: Estación de limpieza de la pistola de soldadura I.

En la operación de limpieza de la pistola de soldadura se activan dos señales de nominadas "**Soplar**" y "**Cortar**" cuando el robot alcanza las posiciones requeridas en la estación de limpieza y que podemos observar en el "**Simulador de E/S**".

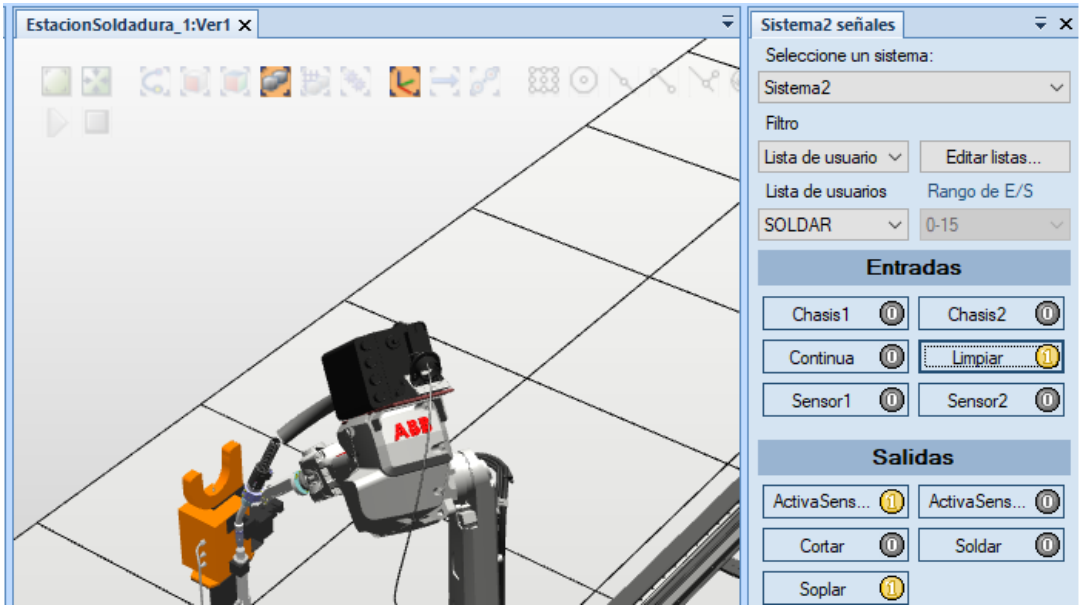


Figura 292: Estación de limpieza de la pistola de soldadura II.

Durante la soldadura sobre un chasis, disponemos de una señal de salida que activa y desactiva la soldadura en los puntos de la trayectoria requeridos y que podemos observar en el "**Simulador de E/S**".

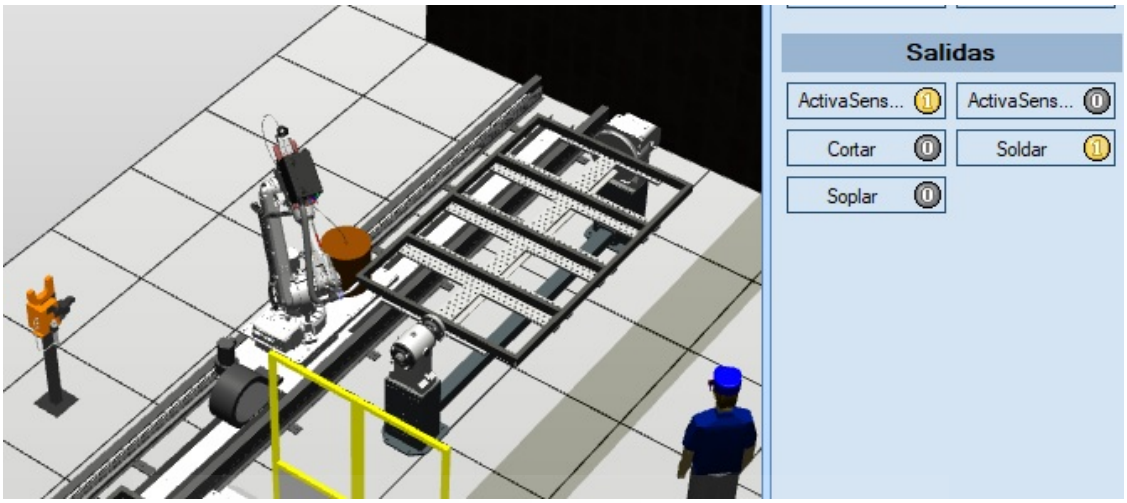


Figura 293: Señal de activación de la soldadura.

Por último, el operador del robot dispone de un pulsador denominado "**Limpieza**" que le permite realizar a voluntad una operación de limpieza de la pistola de soldadura del robot cuando este se encuentra parado **antes** de comenzar a soldar uno de los chasis.

7. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO.

7.1. Conclusiones.

El objetivo principal de este TFM que era conseguir las destrezas y habilidades necesarias para modelar, programar y simular una estación de soldadura robotizada mediante el software de ABB RobotStudio se ha alcanzado pero, ciertamente, las posibilidades de programación y simulación de estaciones robotizadas de este software es muy amplia

La curva de aprendizaje ha sido lenta dado que no he encontrado muchos recursos de aprendizaje y se ha basado en los manuales del programa y algunos video tutoriales que se han encontrado en la web sobre programación de robots ABB.

El lenguaje de programación RAPID es, en sí mismo, un amplísimo y requeriría de un estudio profundo, con los recursos de aprendizaje adecuados, dada la complejidad que puede llegar a alcanzarse en la programación de las estaciones de trabajo robotizadas

Durante la realización de este trabajo se han encontrado muchas dificultades que se han ido solventando a base de prueba y error en muchos casos y en otros estudiando en profundidad los manuales que ofrece ABB de su software y que no son muy prolíficos en ejemplos didácticos.

En cuanto a los resultados conseguidos en la implantación de la robotización de la estación de soldadura han sido satisfactorios ya que el tiempo empleado en la simulación para soldar un chasis es de unos 109 segundos que reduce ampliamente el tiempo empleado en la soldadura manual.

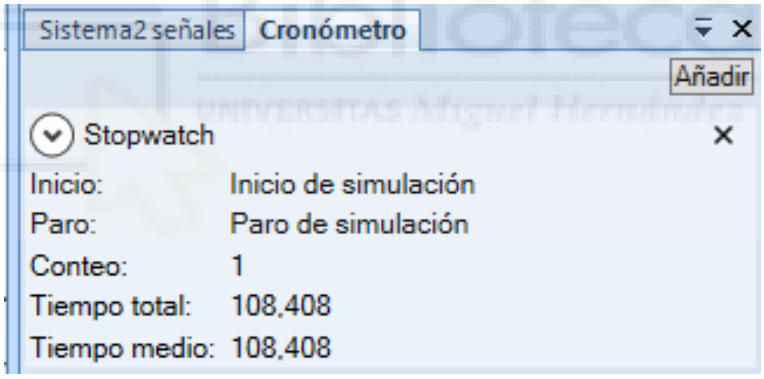


Figura 294: Tiempo de soldadura de un chasis.

Sin embargo, se presentan dificultades en el diseño de los útiles necesarios para sujetar los distintos modelos de chasis ya que se debe garantizar la repetividad en la posición. La rigidez y la exactitud en el posicionamiento tanto del track, el robot y del posicionador introducen un problema que debe solventarse.

El reposicionamiento del objeto de trabajo de forma manual para corregir las desviaciones no es una solución ideal y debería recurrirse a sistemas auto tracking de los cordones de soldadura basados en las características del arco de soldadura u ópticos

7.2. Líneas futuras de trabajo.

Dada la complejidad y posibilidades del software RobotStudio y el lenguaje de programación RAPID se abren múltiples líneas futuras de trabajo de forma que solo comento alguna de las que se pueden plantear en este TFM

Aunque se ha programado una forma de que si se produce una parada durante el trabajo de soldadura sea posible reanudar el trabajo en el punto que se dejó, este es un problema grave ya que puede afectar a la calidad del cordón de soldadura. Sería necesario profundizar en la programación de RAPID para solventar este tipo de situaciones de forma que el robot pueda retroceder sin peligro de colisiones.

Incorporar sensores para detectar colisiones de la pistola de soldadura con objetos y su correspondiente programación RAPID para reaccionar ante estos eventos.

La implantación de sistemas de seguridad como la programación de volúmenes de seguridad en RAPID, el empleo de sistemas de visión con cámaras CCD o sistemas de escáner láser que garanticen la imposibilidad de accidentes y su integración en la estación sería otro campo de estudio a realizar.

El aprendizaje de algún software específico para programación de trabajos de soldadura que permita la interacción con el equipo de soldadura de forma que se puedan seleccionar o variar los parámetros de la soldadura a emplear en cada caso e integrarlo en el programa RAPID.

Otra línea de trabajo posible sería el diseño de útiles que permitan la sujeción de piezas de forma flexible y fiable en los posicionadores.

Por último, y ya se ha comentado en el apartado anterior, la implantación de sistemas que permitan corregir de forma automática las posibles desviaciones de posición de las piezas de trabajo y permitan corregir las trayectorias para garantizar cordones de soldadura correctos.

8. BIBLIOGRAFÍA.

ABB Robotics. (2018) RobotStudio 6.08 Manual del operador. 666 p.

ABB Robotics. (2019) IRB 2600, Especificaciones del producto. 90 p.

ABB Robotics. (2005) Track de desplazamiento RTT. Manual de producto. 62 p.

ABB Robotics. (2019) Robotics product range. 4 p.

ABB Robotics. (2007) Introduction to RAPID. 60 p.

ABB Robotics. (2019) IRBP L Positioner Data Sheet. 2 p.

ABB Robotics. (2019) Descripción general de RAPID, RobotWare 6.08. 70 p.

ABB Robotics. (2017) RAPID Instructions, Functions and Data types. 1760 p.

Querol, C. (2017) Programación de robot ABB para tareas de reparación de defectos en carrocerías de automóvil. Universidad Miguel Hernández. TFM del Master de Robótica UMH. 164 p.

FEMETAL. (2010). Guía de implantación de robots. Sector metal. 127 p.

Neto, P. (2014). A guide for ABB RobotStudio. University of Coimbra. Dept. of Mechanical Engineering. 87 p.

Real Decreto 486/1997, de 14 de abril, por el que se establecen las disposiciones míni-mas de se-guridad y salud en los lugares de trabajo

Páginas web y video tutoriales consultados:

https://library.abb.com/

https://www.infoplc.net/descargas/46-abb-robotica

https://www.youtube.com/channel/UCMjjG2oBntG3j6Jtv5oxkDw/videos

https://www.youtube.com/playlist?list=PLVdvHpsfqw1bX194j7Slup6ri5se2668p

https://www.binzel-abicor.com/ES/spa/home/bienvenido-a-abicor-binzel-espaa.html

https://www.youtube.com/watch?v=zeqUQ1u6QsE

https://www.telwin.com/es/telwin-academy/saldatura/mig-mag-welding/

https://www.ecured.cu/Soldadura_MIG

http://www.boe.es/buscar/act.php?id=BOE-A-1995-24292

9. ANEXOS.

9.1 Lógica de la estación.

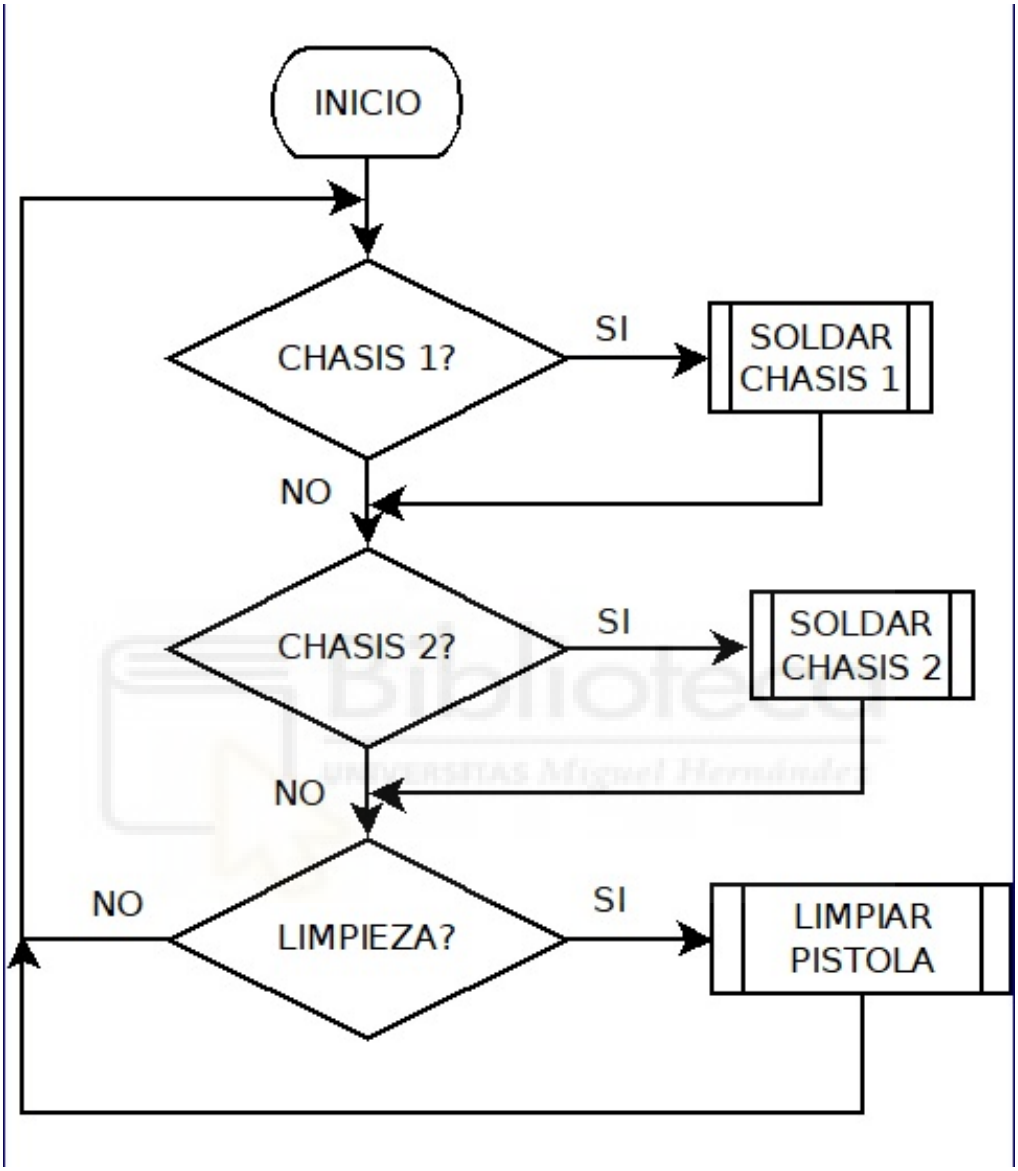


Figura 295: Diagrama de flujo rutina main

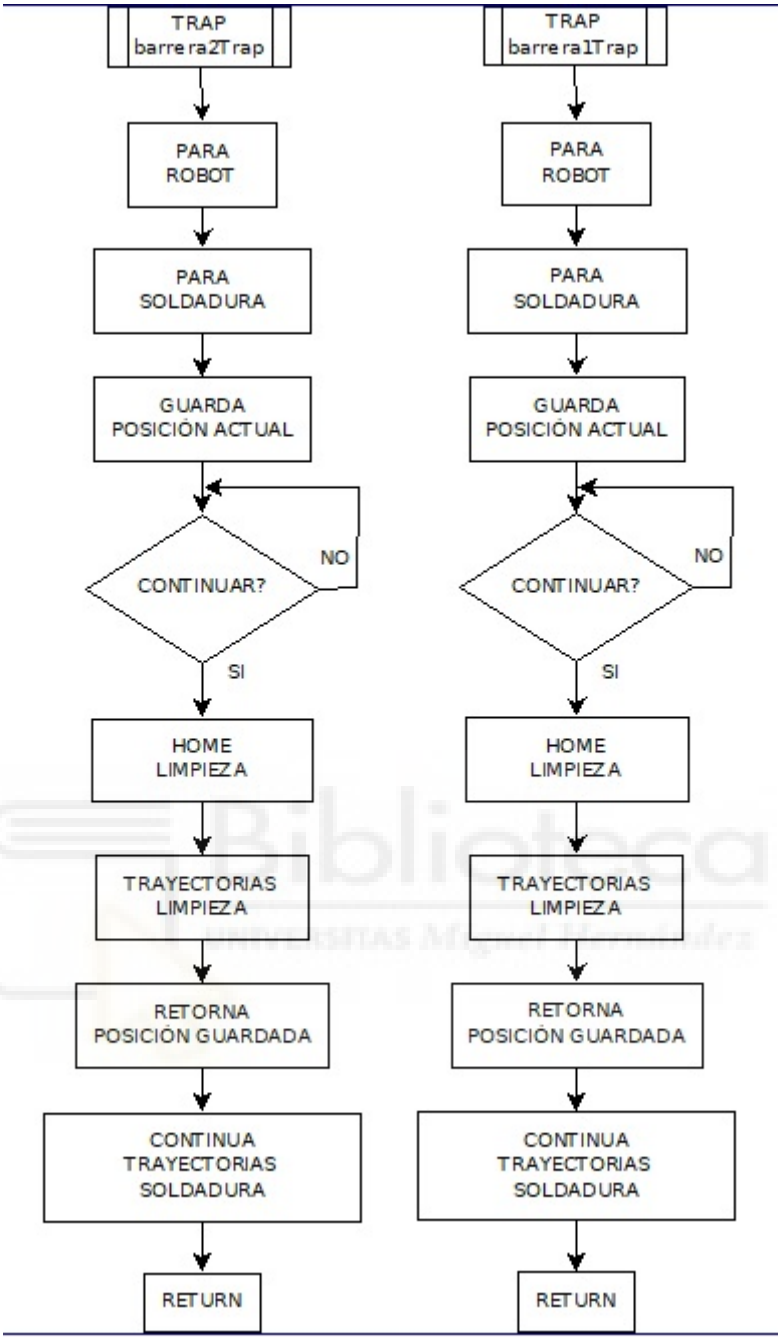


Figura 296: Diagrama de flujo rutinas Trap

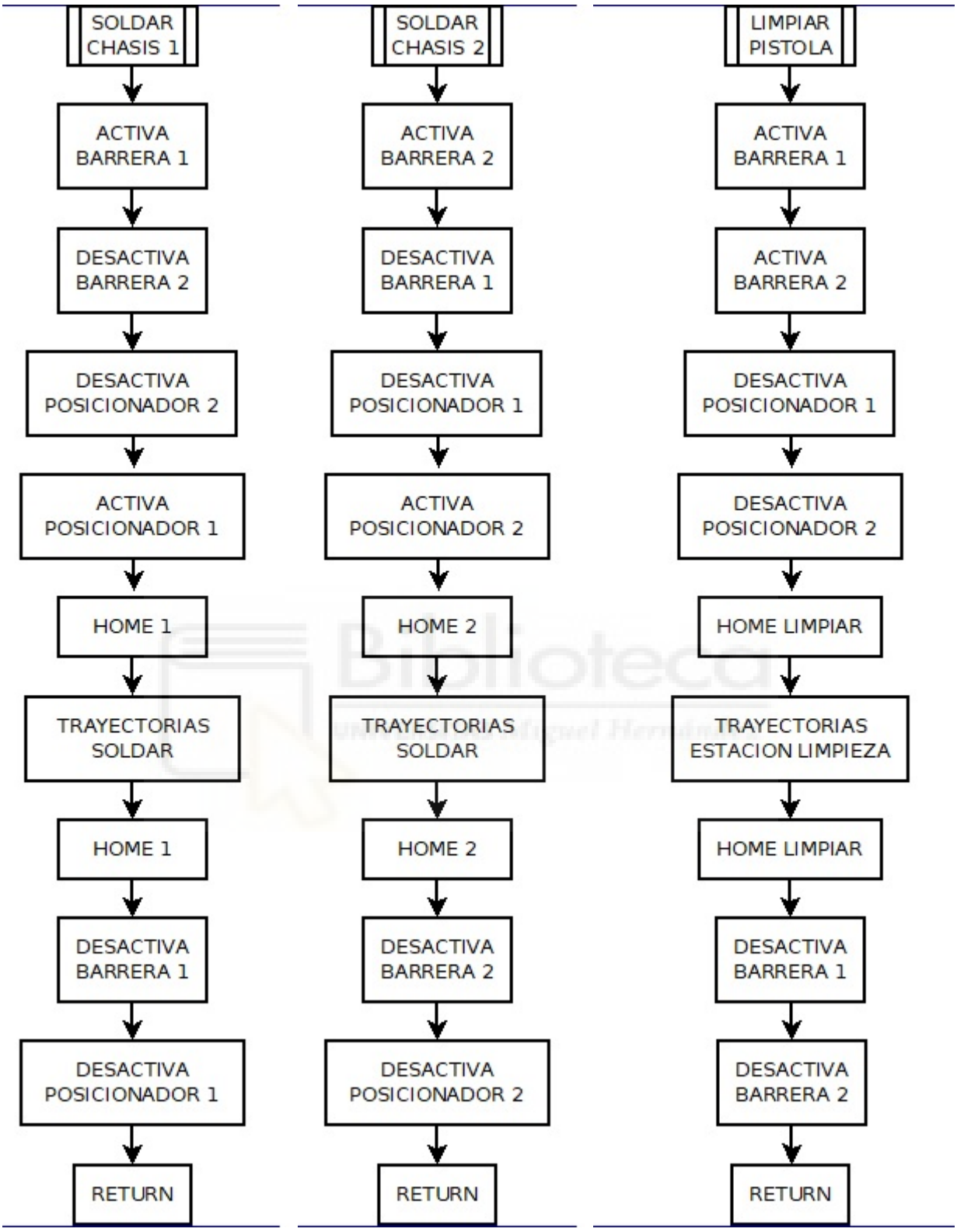


Figura 297: Diagrama de flujo rutinas soldar y limpiar

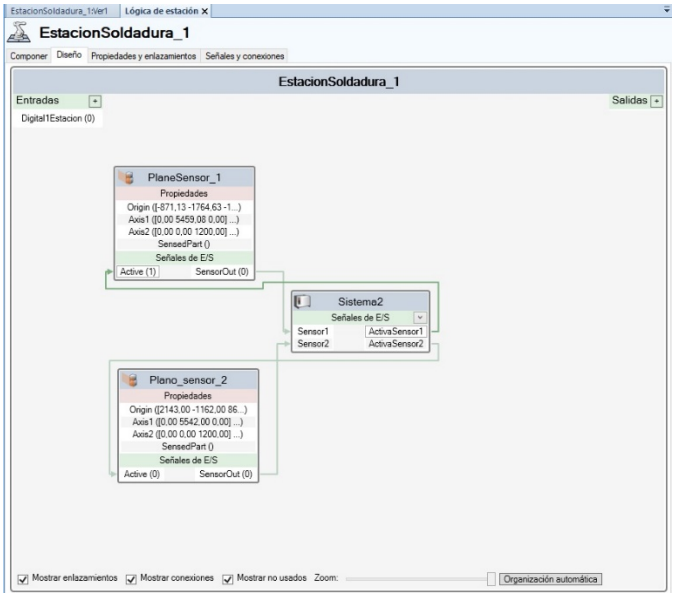


Figura 298: Diseño (Lógica de la estación).

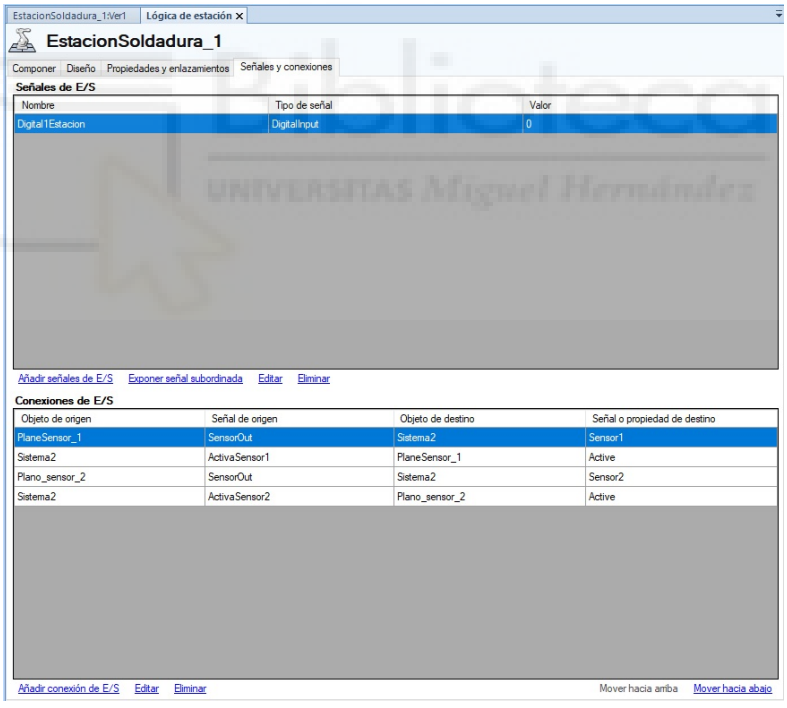


Figura 299: Señales y conexiones (Lógica de la estación).

9.2 Programas.

9.2.1 Código de RAPID de la estación de soldadura de chasis para maquinaria.

```
MODULE Module1
    CONST robtarget
home:=[[849.450585987,0,1334.199408725],[0.325568155,0,0.945518575,0],[0,0,0,0],[0,9E+
09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTarget_1:=[[-90,-
0.000019823,0.000053776,0,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTarget_2:=[[0,0,0,0,0,0],[2000,9E+09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTarget_3:=[[-110,0,0,0,0,0],[300,9E+09,9E+09,9E+09,-
180,9E+09]];
    CONST jointtarget JointTarget_4:=[[0,0,0,0,0,0],[2000,9E+09,9E+09,9E+09,70,9E+09]];
    CONST jointtarget JointTarget_5:=[[0,0,0,0,0,0],[0,9E+09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTarget_6:=[[0,0,0,0,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTarget_7:=[[-90,0,0,0,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTar-
get_2_2:=[[0,0,0,0,0,0],[7100,9E+09,9E+09,9E+09,0,9E+09]];
    CONST jointtarget JointTarget_3_2:=[[-110,0,0,0,0,0],[5400,9E+09,9E+09,9E+09,-
180,9E+09]];
    CONST jointtarget JointTar-
get_4_2:=[[0,0,0,0,0,0],[7100,9E+09,9E+09,9E+09,70,9E+09]];
    CONST jointtarget JointTar-
get_5_2:=[[0,0,0,0,0,0],[5100,9E+09,9E+09,9E+09,0,9E+09]];

    CONST robtarget Target_10:=[[2500.000123282,41.000087025,-
79.999959684],[0.707106781,0,-0.707106781,0],[-
2,0,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
    CONST robtarget Target_20:=[[2450.000123282,41.000087025,-
79.999959684],[0.707106781,0,-0.707106781,0],[-
2,0,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
```

CONST robtarget Tar-
get_30:=[[2450.000123282,41.000087025,0.000040316],[0.707106781,0,-0.707106781,0],[-
2,0,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
CONST robtarget Tar-
get_40:=[[2500.000123282,41.000087025,0.000040316],[0.707106781,0,-0.707106781,0],[-
2,0,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
CONST robtarget Tar-
get_50:=[[2450.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-
2,-1,1,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
CONST robtarget Tar-
get_60:=[[2450.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-
1,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
CONST robtarget Tar-
get_70:=[[2410.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-
1,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
CONST robtarget Tar-
get_80:=[[2410.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-
1,-1,0,0],[300,9E+09,9E+09,9E+09,0,9E+09]];
CONST robtarget Tar-
get_90:=[[2360.000123282,41.000087025,0.000040316],[0.27059805,0.653281482,-
0.27059805,0.653281482],[-2,-1,-1,0],[300,0,0,0,0,0]];
CONST robtarget Tar-
get_100:=[[2410.000123282,41.000087025,0.000040316],[0.27059805,0.653281482,-
0.27059805,0.653281482],[-1,-1,-1,0],[300,0,0,0,0,0]];
CONST robtarget Target_110:=[[2410.000123282,41.000087025,-
79.999959684],[0.27059805,0.653281482,-0.27059805,0.653281482],[-1,-1,-
1,0],[300,0,0,0,0,0]];
CONST robtarget Target_120:=[[2360.000123282,41.000087025,-
79.999959684],[0.27059805,0.653281482,-0.27059805,0.653281482],[-1,-1,-
1,0],[300,0,0,0,0,0]];

CONST robtarget Target_130:=[[1877.061761664,76.705868551,-41.407670363],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_140:=[[1847.500123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-1,0,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_150:=[[1847.500123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-1,0,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_160:=[[1876.286919785,62.213290399,0.00004027],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_170:=[[1847.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[0,0,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_180:=[[1847.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_190:=[[1807.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_200:=[[1807.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_210:=[[1757.500123282,81.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[0,0,0,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_220:=[[1806.925517298,42.988906569,0.84527684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_230:=[[1806.50012328,42.000087023,-78.585746122],[0.146446609,0.853553391,-0.35355339,0.353553391],[-1,0,-1,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_240:=[[1778.713326712,62.213290466,-79.9999597],[0.146446609,0.853553391,-0.35355339,0.35355339],[-1,0,-1,0],[800,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_250:=[[1285.000123252,71.000087027,-65.85782408],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,-1,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_260:=[[1245.000123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_270:=[[1245.000123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_280:=[[1295.000123282,91.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_290:=[[1245.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_300:=[[1245.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_310:=[[1205.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_320:=[[1205.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_330:=[[1179.748860656,65.748824357,0.000040298],[0.205615658,0.779192095,-0.322751933,0.496400111],[-2,-1,-1,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_340:=[[1205.000123282,41.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_350:=[[1205.000123282,41.000087025,-79.999959684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_360:=[[1155.000123282,91.000087025,-49.999959684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[1500,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_370:=[[667.751385934,65.748824359,-49.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_380:=[[642.500123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_390:=[[642.500123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_400:=[[710.876392382,65.231183599,8.669913025],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_430:=[[642.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_440:=[[642.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_450:=[[602.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_460:=[[602.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_470:=[[533.274431803,60.225778463,12.678588178],[0.205615658,0.779192095,-0.322751933,0.496400111],[0,0,0,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_480:=[[602.500123282,41.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_490:=[[602.500123282,41.000087025,-79.999959684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_500:=[[539.68299563,53.81721465,-71.547594442],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[2200,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_530:=[[90.000123282,91.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_540:=[[40.000123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_550:=[[40.000123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_560:=[[61.715852064,69.284358302,0.000040319],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_570:=[[40.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_580:=[[40.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_590:=[[0.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_600:=[[0.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_610:=[[-49.999876718,41.000087025,0.000040316],[0.27059805,-0.653281482,0.27059805,-0.653281482],[0,0,0,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_620:=[[0.000123282,41.000087025,0.000040316],[0.27059805,-0.653281482,0.27059805,-0.653281482],[-1,-1,0,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_630:=[[-49.999876718,41.000087025,-79.999959684],[0.27059805,-0.653281482,0.27059805,-0.653281482],[-1,-1,0,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_640:=[[-149.999876718,-29.710591114,-9.289281585],[0.27059805,-0.653281482,0.27059805,-0.653281482],[-1,-1,0,0],[2900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_660:=[[0.000123282,41.000087025,-129.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,0,0],[2600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_670:=[[0.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[2600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_680:=[[40.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[2600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_690:=[[40.000123282,41.000087025,-129.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[2600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_700:=[[602.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-2,1,0],[2300,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_710:=[[602.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[2300,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_720:=[[642.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[2300,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_730:=[[642.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[2300,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_740:=[[1205.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,2,-3,0],[1700,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_750:=[[1205.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,3,-3,0],[1700,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_760:=[[1245.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,3,-3,0],[1700,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_770:=[[1245.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,3,-3,0],[1700,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_780:=[[1807.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-2,1,0],[1100,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_790:=[[1807.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[1100,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_800:=[[1847.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[1100,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_810:=[[1847.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[1100,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_820:=[[2410.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-2,-1,1,0],[500,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_830:=[[2410.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[500,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_840:=[[2450.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[500,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_850:=[[2450.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[500,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_980:=[[642.500123282,1439.000087025,-159.999959683],[0.382683434,0,0,0.923879532],[0,0,0,0],[2400,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_990:=[[642.500123282,1439.000087025,-79.999959683],[0.382683434,0,0,0.923879532],[-1,-1,0,0],[2400,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1000:=[[602.500123282,1439.000087025,-79.999959683],[0.382683434,0,0,0.923879532],[-1,-1,0,0],[2400,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1010:=[[602.500123282,1439.000087025,-159.999959683],[0.382683434,0,0,0.923879532],[-1,-1,0,0],[2400,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_940:=[[1245.000123282,1439.000087025,-159.999959684],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[1800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_950:=[[1245.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[1800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_960:=[[1205.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[1800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_970:=[[1205.000123282,1439.000087025,-159.999959684],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[1800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_900:=[[1847.500123282,1439.000087025,-159.999959684],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[1200,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_910:=[[1847.500123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[1200,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_920:=[[1807.500123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[1200,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_930:=[[1807.500123282,1439.000087025,-159.999959684],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[1200,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1020:=[[40.000123282,1439.000087025,-149.999959683],[0.707106782,0,0,0.70710678],[0,0,0,0],[2800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1030:=[[40.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-1,-1,0,0],[2800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1040:=[[0.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-1,-1,0,0],[2800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1050:=[[0.000123282,1439.000087025,-129.999959683],[0.707106782,0,0,0.70710678],[-1,-1,0,0],[2800,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_860:=[[2450.000123282,1439.000087025,-129.999959683],[0.707106781,0,0,0.707106782],[-1,-1,0,0],[500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_870:=[[2450.000123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_880:=[[2410.000123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_890:=[[2410.000123282,1439.000087025,-129.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1650:=[[-99.999876718,1439.000087026,-79.999959684],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1640:=[[0.000123282,1439.000087026,-79.999959684],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1630:=[[0.000123282,1439.000087025,0.000040317],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1620:=[[-49.999876718,1439.000087025,0.000040317],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1610:=[[0.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-2,0,0,1],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1600:=[[0.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1590:=[[40.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1580:=[[40.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1570:=[[67.05706582,1406.234005259,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1560:=[[40.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1550:=[[40.000123282,1439.000087026,-79.999959684],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1540:=[[97.367391509,1397.218232605,-49.999959684],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_8:=[[-107.882,0,0,0,26.125,0],[2900,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1530:=[[569.790522789,1400.090494099,-45.719473661],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1520:=[[595.855493034,1432.355456787,-76.579758246],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1510:=[[602.500123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1500:=[[552.500123282,1389.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1490:=[[602.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-2,0,-1,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1480:=[[602.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1470:=[[642.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1460:=[[642.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1450:=[[657.144784242,1403.644747947,0.000040348],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-1,2,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1440:=[[642.500123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-2,2,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1430:=[[643.930431322,1438.72360692,-68.892932755],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-2,2,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1420:=[[692.500123282,1389.000087025,-29.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-2,2,1],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_9:=[[-107.882,0,0,0,26.125,0],[2300,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1410:=[[1185.836124854,1399.125410454,-24.46482508],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1400:=[[1198.355493054,1432.355456744,-76.579758289],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1390:=[[1205.000123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1380:=[[1155.000123211,1389.000087025,0.000040529],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1370:=[[1205.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1360:=[[1205.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1350:=[[1245.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1340:=[[1245.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1330:=[[1259.644784259,1403.644747929,0.000040321],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,0,0,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1320:=[[1245.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1310:=[[1248.84887894,1436.305159303,-78.289858967],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1300:=[[1295.000123282,1389.000087025,-29.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_10:=[[-107.882,0,0,0,26.125,0],[1700,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1290:=[[1792.855462341,1414.000087054,-54.999959654],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1280:=[[1806.513217374,1437.476529498,-79.160442446],[0.191341717,-0.800103145,-0.331413575,-0.461939765],[-1,-2,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1270:=[[1807.500123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1260:=[[1757.500123282,1403.644747956,35.355379366],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1250:=[[1807.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,-2,1,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1240:=[[1807.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-2,-3,2,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1230:=[[1847.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-2,-3,2,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1220:=[[1847.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-2,-3,2,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Tar-
get_1210:=[[1862.144784213,1403.644747976,0.000040317],[0.54702957,-0.24632377,-
0.78123919,-0.172477761],[-2,1,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1200:=[[1847.500123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-
0.78123919,-0.172477761],[-2,1,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1190:=[[1847.500123282,1439.000087025,-
79.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-
2,1,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1180:=[[1897.500123282,1389.000087025,-
29.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-
2,1,0,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_11:=[[-
107.882,0,0,0,26.125,0],[1100,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1170:=[[2395.355462351,1403.644747976,-
79.999959683],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-
1,0],[500,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1160:=[[2410.000123282,1439.000087025,-
79.999959683],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-
1,0],[500,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1150:=[[2410.000123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-
0.313475917,-0.529915529],[-1,-2,-1,0],[500,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1140:=[[2360.000123282,1389.000087025,0.000040317],[0.2194982,-0.756797808,-
0.313475917,-0.529915529],[-1,-2,-1,0],[500,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1130:=[[2410.000123282,1439.000087025,50.000040317],[0,-
0.382683434,0.923879532,0],[-1,-2,-3,0],[500,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1120:=[[2410.000123282,1439.000087025,0.000040317],[0,-
0.382683434,0.923879532,0],[-1,-1,-3,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1110:=[[2450.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,-1,-3,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1100:=[[2450.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,-1,-3,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1090:=[[2500.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,0,0,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1080:=[[2450.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,0,0,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1070:=[[2450.000123282,1439.000087025,-79.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,0,0,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1060:=[[2550.000123282,1439.000087025,70.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,0,0,0],[500,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_10_2:=[[2500.000123282,41.000087025,-79.999959684],[0.707106781,0,-0.707106781,0],[-2,0,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_20_2:=[[2450.000123282,41.000087025,-79.999959684],[0.707106781,0,-0.707106781,0],[-2,0,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_30_2:=[[2450.000123282,41.000087025,0.000040316],[0.707106781,0,-0.707106781,0],[-2,0,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_40_2:=[[2500.000123282,41.000087025,0.000040316],[0.707106781,0,-0.707106781,0],[-2,0,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_50_2:=[[2450.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-2,-1,1,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_60_2:=[[2450.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_70_2:=[[2410.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_80_2:=[[2410.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[5400,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_90_2:=[[2360.000123282,41.000087025,0.000040316],[0.27059805,0.653281482,-0.27059805,0.653281482],[-2,-1,-1,0],[5400,0,0,0,0,0]];

CONST robtarget Target_100_2:=[[2410.000123282,41.000087025,0.000040316],[0.27059805,0.653281482,-0.27059805,0.653281482],[-1,-1,-1,0],[5400,0,0,0,0,0]];

CONST robtarget Target_110_2:=[[2410.000123282,41.000087025,-79.999959684],[0.27059805,0.653281482,-0.27059805,0.653281482],[-1,-1,-1,0],[5400,0,0,0,0,0]];

CONST robtarget Target_120_2:=[[2360.000123282,41.000087025,-79.999959684],[0.27059805,0.653281482,-0.27059805,0.653281482],[-1,-1,-1,0],[5400,0,0,0,0,0]];

CONST robtarget Target_130_2:=[[1877.061761664,76.705868551,-41.407670363],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_140_2:=[[1847.500123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-1,0,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_150_2:=[[1847.500123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-1,0,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_160_2:=[[1876.286919785,62.213290399,0.00004027],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_170_2:=[[1847.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[0,0,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_180_2:=[[1847.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_190_2:=[[1807.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_200_2:=[[1807.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_210_2:=[[1757.500123282,81.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[0,0,0,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_220_2:=[[1806.925517298,42.988906569,0.84527684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_230_2:=[[1806.50012328,42.000087023,-78.585746122],[0.146446609,0.853553391,-0.35355339,0.353553391],[-1,0,-1,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_240_2:=[[1778.713326712,62.213290466,-79.9999597],[0.146446609,0.853553391,-0.35355339,0.35355339],[-1,0,-1,0],[5900,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_250_2:=[[1285.000123252,71.000087027,-65.85782408],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,-1,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_260_2:=[[1245.000123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_270_2:=[[1245.000123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_280_2:=[[1295.000123282,91.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_290_2:=[[1245.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_300_2:=[[1245.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_310_2:=[[1205.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_320_2:=[[1205.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_330_2:=[[1179.748860656,65.748824357,0.000040298],[0.205615658,0.779192095,-0.322751933,0.496400111],[-2,-1,-1,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_340_2:=[[1205.000123282,41.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_350_2:=[[1205.000123282,41.000087025,-79.999959684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_360_2:=[[1155.000123282,91.000087025,-49.999959684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[6600,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_370_2:=[[667.751385934,65.748824359,-49.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_380_2:=[[642.500123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_390_2:=[[642.500123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_400_2:=[[692.500123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_430_2:=[[642.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_440_2:=[[642.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_450_2:=[[602.500123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_460_2:=[[602.500123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-1,-1,0,0],[7300,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_470_2:=[[552.500123282,41.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[0,0,0,0],[7700,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_480_2:=[[602.500123282,41.000087025,0.000040316],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[7700,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_490_2:=[[602.500123282,41.000087025,-79.999959684],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[7700,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_500_2:=[[539.68299563,53.81721465,-71.547594442],[0.205615658,0.779192095,-0.322751933,0.496400111],[-1,-1,-1,0],[7700,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_530_2:=[[90.000123282,91.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,0,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_540_2:=[[40.000123282,41.000087025,-79.999959684],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_550_2:=[[40.000123282,41.000087025,0.000040316],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_560_2:=[[61.715852064,69.284358302,0.000040319],[0.470810326,0.502277573,-0.720540246,0.082925453],[-2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_570_2:=[[40.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_580_2:=[[40.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_590_2:=[[0.000123282,41.000087025,0.000040316],[0,0.707106782,-0.70710678,0],[-2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Tar-
get_600_2:=[[0.000123282,41.000087025,50.000040316],[0,0.707106782,-0.70710678,0],[-
2,0,-1,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_610_2:=[[-
49.999876718,41.000087025,0.000040316],[0.27059805,-0.653281482,0.27059805,-
0.653281482],[0,0,0,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Tar-
get_620_2:=[[0.000123282,41.000087025,0.000040316],[0.27059805,-
0.653281482,0.27059805,-0.653281482],[-1,-1,0,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_630_2:=[[-49.999876718,41.000087025,-
79.999959684],[0.27059805,-0.653281482,0.27059805,-0.653281482],[-1,-
1,0,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_640_2:=[[-149.999876718,-29.710591114,-
9.289281585],[0.27059805,-0.653281482,0.27059805,-0.653281482],[-1,-
1,0,0],[8000,9E+09,9E+09,9E+09,0,9E+09]];

CONST robtarget Target_660_2:=[[0.000123282,41.000087025,-
129.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-
1,0,0],[8000,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_670_2:=[[0.000123282,41.000087025,-
79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-
1,1,0],[8000,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_680_2:=[[40.000123282,41.000087025,-
79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-
1,1,0],[8000,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_690_2:=[[40.000123282,41.000087025,-
129.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-
1,1,0],[8000,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_700_2:=[[602.500123282,41.000087025,-
159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-
2,1,0],[7400,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_710_2:=[[602.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[7400,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_720_2:=[[642.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[7400,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_730_2:=[[642.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[7400,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_740_2:=[[1205.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,2,-3,0],[6800,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_750_2:=[[1205.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,3,-3,0],[6800,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_760_2:=[[1245.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,3,-3,0],[6800,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_770_2:=[[1245.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,3,-3,0],[6800,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_780_2:=[[1807.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-2,1,0],[6200,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_790_2:=[[1807.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[6200,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_800_2:=[[1847.500123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[6200,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_810_2:=[[1847.500123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[6200,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_820_2:=[[2410.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-2,-1,1,0],[5600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_830_2:=[[2410.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[5600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_840_2:=[[2450.000123282,41.000087025,-79.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[5600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_850_2:=[[2450.000123282,41.000087025,-159.999959684],[0.315611509,-0.004090854,0.138907965,0.938657135],[-1,-1,1,0],[5600,9E+09,9E+09,9E+09,-130,9E+09]];

CONST robtarget Target_860_2:=[[2450.000123282,1439.000087025,-129.999959683],[0.707106781,0,0,0.707106782],[-1,-1,0,0],[5600,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_870_2:=[[2450.000123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[5600,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_880_2:=[[2410.000123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[5600,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_890_2:=[[2410.000123282,1439.000087025,-129.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[5600,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_900_2:=[[1847.500123282,1439.000087025,-159.999959684],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[6300,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_910_2:=[[1847.500123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[6300,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_920_2:=[[1807.500123282,1439.000087025,-79.999959683],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[6300,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_930_2:=[[1807.500123282,1439.000087025,-159.999959684],[0.707106781,0,0,0.707106782],[-2,0,-1,0],[6300,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_940_2:=[[1245.000123282,1439.000087025,-159.999959684],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[6900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_950_2:=[[1245.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[6900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_960_2:=[[1205.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[6900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_970_2:=[[1205.000123282,1439.000087025,-159.999959684],[0.707106782,0,0,0.70710678],[-2,0,-1,0],[6900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_980_2:=[[642.500123282,1439.000087025,-159.999959683],[0.382683434,0,0,0.923879532],[0,0,0,0],[7500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_990_2:=[[642.500123282,1439.000087025,-79.999959683],[0.382683434,0,0,0.923879532],[-1,-1,0,0],[7500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1000_2:=[[602.500123282,1439.000087025,-79.999959683],[0.382683434,0,0,0.923879532],[-1,-1,0,0],[7500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1010_2:=[[602.500123282,1439.000087025,-159.999959683],[0.382683434,0,0,0.923879532],[-1,-1,0,0],[7500,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1020_2:=[[40.000123282,1439.000087025,-149.999959683],[0.707106782,0,0,0.70710678],[0,0,0,0],[7900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1030_2:=[[40.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-1,-1,0,0],[7900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1040_2:=[[0.000123282,1439.000087025,-79.999959683],[0.707106782,0,0,0.70710678],[-1,-1,0,0],[7900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1050_2:=[[0.000123282,1439.000087025,-129.999959683],[0.707106782,0,0,0.70710678],[-1,-1,0,0],[7900,9E+09,9E+09,9E+09,-180,9E+09]];

CONST robtarget Target_1650_2:=[[-99.999876718,1439.000087026,-79.999959684],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1640_2:=[[0.000123282,1439.000087026,-79.999959684],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1630_2:=[[0.000123282,1439.000087025,0.000040317],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1620_2:=[[-49.999876718,1439.000087025,0.000040317],[0,0.819152044,0,0.573576437],[-1,-1,-1,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1610_2:=[[0.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-2,0,0,1],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1600_2:=[[0.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1590_2:=[[40.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1580_2:=[[40.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1570_2:=[[67.05706582,1406.234005259,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1560_2:=[[40.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1550_2:=[[40.000123282,1439.000087026,-79.999959684],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1540_2:=[[97.367391509,1397.218232605,-49.999959684],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1530_2:=[[569.790522789,1400.090494099,-45.719473661],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1520_2:=[[595.855493034,1432.355456787,-76.579758246],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1510_2:=[[602.500123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1500_2:=[[552.500123282,1389.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_8_2:=[[-107.882,0,0,0,26.125,0],[8000,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1490_2:=[[602.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-2,0,-1,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1480_2:=[[602.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1470_2:=[[642.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1460_2:=[[642.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,0,-1,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1450_2:=[[657.144784242,1403.644747947,0.000040348],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-1,2,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1440_2:=[[642.500123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-2,2,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1430_2:=[[643.930431322,1438.72360692,-68.892932755],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-2,2,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1420_2:=[[692.500123282,1389.000087025,-29.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,-2,2,1],[7400,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1410_2:=[[1185.836124854,1399.125410454,-24.46482508],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1400_2:=[[1198.355493054,1432.355456744,-76.579758289],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1390_2:=[[1205.000123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-1,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Tar-
get_1380_2:=[[1155.000123211,1389.000087025,0.000040529],[0.2194982,-0.756797808,-
0.313475917,-0.529915529],[-1,-1,-1,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST jointtarget JointTarget_9_2:=[[-
107.882,0,0,0,26.125,0],[7400,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1370_2:=[[1205.000123282,1439.000087025,50.000040317],[0,-
0.382683434,0.923879532,0],[-1,0,-1,1],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1360_2:=[[1205.000123282,1439.000087025,0.000040317],[0,-
0.382683434,0.923879532,0],[-1,0,-1,1],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1350_2:=[[1245.000123282,1439.000087025,0.000040317],[0,-
0.382683434,0.923879532,0],[-1,0,-1,1],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1340_2:=[[1245.000123282,1439.000087025,50.000040317],[0,-
0.382683434,0.923879532,0],[-1,0,-1,1],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1330_2:=[[1259.644784259,1403.644747929,0.000040321],[0.54702957,-0.24632377,-
0.78123919,-0.172477761],[-2,0,0,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Tar-
get_1320_2:=[[1245.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-
0.78123919,-0.172477761],[-2,1,0,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1310_2:=[[1248.84887894,1436.305159303,-
78.289858967],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-
2,1,0,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1300_2:=[[1295.000123282,1389.000087025,-
29.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-
2,1,0,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];
CONST robtarget Target_1290_2:=[[1792.855462341,1414.000087054,-
54.999959654],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-1,-
1,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1280_2:=[[1806.513217374,1437.476529498,-79.160442446],[0.191341717,-0.800103145,-0.331413575,-0.461939765],[-1,-2,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1270_2:=[[1807.500123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1260_2:=[[1757.500123282,1403.644747956,35.355379366],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_10_2:=[[-107.882,0,0,0,26.125,0],[6800,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1250_2:=[[1807.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,-2,1,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1240_2:=[[1807.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-2,-3,2,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1230_2:=[[1847.500123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-2,-3,2,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1220_2:=[[1847.500123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-2,-3,2,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1210_2:=[[1862.144784213,1403.644747976,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1200_2:=[[1847.500123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1190_2:=[[1847.500123282,1439.000087025,-79.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1180_2:=[[1897.500123282,1389.000087025,-29.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,1,0,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1170_2:=[[2395.355462351,1403.644747976,-79.999959683],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1160_2:=[[2410.000123282,1439.000087025,-79.999959683],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1150_2:=[[2410.000123282,1439.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1140_2:=[[2360.000123282,1389.000087025,0.000040317],[0.2194982,-0.756797808,-0.313475917,-0.529915529],[-1,-2,-1,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget JointTarget_11_2:=[[-107.882,0,0,0,26.125,0],[6200,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1130_2:=[[2410.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,-2,-3,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1120_2:=[[2410.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,-1,-3,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1110_2:=[[2450.000123282,1439.000087025,0.000040317],[0,-0.382683434,0.923879532,0],[-1,-1,-3,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1100_2:=[[2450.000123282,1439.000087025,50.000040317],[0,-0.382683434,0.923879532,0],[-1,-1,-3,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1090_2:=[[2500.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-2,0,0,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Tar-
get_1080_2:=[[2450.000123282,1439.000087025,0.000040317],[0.54702957,-0.24632377,-
0.78123919,-0.172477761],[-2,0,0,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Target_1070_2:=[[2450.000123282,1439.000087025,-
79.999959683],[0.54702957,-0.24632377,-0.78123919,-0.172477761],[-
2,0,0,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST robtarget Tar-
get_1060_2:=[[2550.000123282,1439.000087025,70.000040317],[0.54702957,-0.24632377,-
0.78123919,-0.172477761],[-2,0,0,0],[5600,9E+09,9E+09,9E+09,60,9E+09]];

CONST jointtarget Soplador1:=[[75.3,-
3,31.865671642,0,10.149253731,0],[3592.537313433,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Soplador_2:=[[3877.479424672,1117.659,515],[0.002702028,-
0.817654776,0.575702178,0.00060717],[0,-
2,0,1],[3500,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Sopla-
dor_1:=[[3875.36515491,1117.884072492,663.119389],[0.002702028,-
0.817654776,0.575702178,0.00060717],[0,-
2,0,1],[3500,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Cor-
tar1:=[[4150.416337245,1170.495979451,475.073667483],[0.002702028,-
0.817654776,0.575702178,0.00060717],[0,0,-
2,0],[3500,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Cor-
tar2:=[[4150.416246708,1201.779867832,475.073657777],[0.002702028,-
0.817654776,0.575702178,0.00060717],[0,0,-
2,0],[3500,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST jointtarget homeLim-
pieza:=[[0,0,0,0,0,0],[3500,9E+09,9E+09,9E+09,9E+09,9E+09]];

!declaramos interrupción para barrera inmaterial 1

```
VAR intnum barrera1int;

!declaramos interrupción para barrera inmaterial 2
VAR intnum barrera2int;


PROC main()

    !Conectamos rutina interrupción con rutina trap barrera 1
    CONNECT barrera1t WITH barrera1Trap;

    !Conectamos rutina interrupción con rutina trap barrera 2
    CONNECT barrera2int WITH barrera2Trap;

    !Solicitamos interrupción cuando pase a 1 la barrera 1
    ISignalDI Sensor1,high,barrera1int;

    !Solicitamos interrupción cuando pase a 1 la barrera 2
    ISignalDI Sensor2,high,barrera2int;


    WHILE (TRUE) DO
        IF DInput(CHASIS1)=1 THEN
            CHASIS_1;
        ELSEIF DInput(CHASIS2)=1 THEN
            CHASIS_2;
        ELSEIF DInput(Limpiar)=1 THEN

            LIMPIEZA;


        ENDIF
```

```
    ENDWHILE

ENDPROC

!Rutina trap para la barrera inmaterial 1


TRAP barrera1Trap

    VAR robtarget stop_pos;
    !Para robot, guarda trayectoria, herramienta y objeto de trabajo actuales
    StopMove;
    SetDO SOLDAR,0;
    StorePath;
    GetSysData tWeldGun;
    GetSysData Workobject_1;
    stop_pos:=CRobT(\Tool:=tWeldGun\WObj:=Workobject_1);

    WaitUntil CONTINUA=1;
    WaitTime(3);

    ! Realiza limpieza
    MoveJ RelTool(stop_pos,0,0,-200),v50,fine,tWeldGun\WObj:=Workobject_1;
    MoveJ RelTool(stop_pos,300,300,-200),v1000,fine,tWeldGun\WObj:=Workobject_1;
    LIMPIEZA;

    ! Vuelve a la posición de parada, restaura trayectoria y pone en marcha el robot
    ActUnit STN2;
    MoveJ RelTool(stop_pos,300,300,-400),v1000,fine,tWeldGun\WObj:=Workobject_1;
    MoveJ RelTool(stop_pos,0,0,-200),v50,fine,tWeldGun\WObj:=Workobject_1;

    MoveJ stop_pos,v50,fine,tWeldGun,\WObj:=Workobject_1;
```

```
    RestoPath;
    StartMove;

ENDTRAP



!Rutina trap para la barrera inmaterial 2
TRAP barrera2Trap

    VAR robtarget stop_pos;
    !Para robot, guarda trayectoria, herramienta y objeto de trabajo actuales
    StopMove;
    SetDO SOLDAR,0;
    StorePath;
    GetSysData tWeldGun;
    GetSysData Workobject_2;
    stop_pos:=CRobT(\Tool:=tWeldGun\WObj:=Workobject_2);

    WaitUntil CONTINUA=1;
    WaitTime(3);

    ! Realiza limpieza
    MoveJ RelTool(stop_pos,0,0,-200),v50,fine,tWeldGun\WObj:=Workobject_2;
    MoveJ RelTool(stop_pos,300,300,-200),v300,fine,tWeldGun\WObj:=Workobject_2;
    LIMPIEZA;

    ! Vuelve a la posición de parada, restaura trayectoria y pone en marcha el robot
    ActUnit STN1;
    MoveJ RelTool(stop_pos,300,300,-400),v300,fine,tWeldGun\WObj:=Workobject_2;
    MoveJ RelTool(stop_pos,0,0,-200),v50,fine,tWeldGun\WObj:=Workobject_2;
```

```
    MoveJ stop_pos,v50,fine,tWeldGun,\WObj:=Workobject_2;
    RestoPath;
    StartMove;


ENDTRAP


!Programa limpieza de la pistola de soldadura
PROC LIMPIEZA()
    SetDO SOLDAR,0;
    ConfL\Off;
    DeactUnit STN1;
    DeactUnit STN2;
    MoveAbsJ homeLimpieza,v500,z100,tWeldGun\WObj:=wobj0;
    MoveJ Soplador_1,v500,fine,tWeldGun\WObj:=wobj0;
    MoveL Soplador_2,v50,fine,tWeldGun\WObj:=wobj0;
    SetDO Soplar,1;
    WaitTime(3);
    SetDO Soplar,0;
    MoveL Soplador_1,v50,fine,tWeldGun\WObj:=wobj0;
    MoveJ Offs(Soplador_1,200,0,0),v50,fine,tWeldGun\WObj:=wobj0;
    MoveJ Cortar1,v200,fine,tWeldGun\WObj:=wobj0;
    MoveL Cortar2,v50,fine,tWeldGun\WObj:=wobj0;
    SetDO Cortar,1;
    WaitTime(3);
    SetDO Cortar,0;
    MoveL Cortar1,v50,fine,tWeldGun\WObj:=wobj0;
    MoveAbsJ homeLimpieza,v500,z100,tWeldGun\WObj:=wobj0;
ENDPROC
```

```
!Programa soldadura para el Chasis 1 (derecha)
PROC CHASIS_1()

    ConfL\Off;
    SetDO ActivaSensor1,1;
    SetDO ActivaSensor2,0;
    DeactUnit STN1;
    ActUnit STN2;
    MoveAbsJ JointTarget_6,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveAbsJ JointTarget_7,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_410;
    Path_420;
    Path_430;
    Path_440;
    Path_450;
    Path_460;
    Path_470;
    Path_480;
    Path_490;
    Path_500;
    Path_510;
    Path_520;
    Path_530;
    Path_540;
    Path_550;
    MoveAbsJ JointTarget_2_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_560;
    Path_570;
    Path_580;
    Path_590;
    Path_600;
    MoveAbsJ JointTarget_3_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
```

```
    Path_610;
    Path_620;
    Path_630;
    Path_640;
    Path_650;
    MoveAbsJ JointTarget_4_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_660;
    Path_670;
    Path_680;
    MoveAbsJ JointTarget_8_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_690;
    Path_700;
    Path_710;
    MoveAbsJ JointTarget_9_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_720;
    Path_730;
    Path_740;
    MoveAbsJ JointTarget_10_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_750;
    Path_760;
    Path_770;
    MoveAbsJ JointTarget_11_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    Path_780;
    Path_790;
    Path_800;
    MoveAbsJ JointTarget_5_2,v500,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

!Programa soldadura para el Chasis 2 (izquierda)
PROC CHASIS_2()

    ConfL\Off;
```

```
SetDO ActivaSensor1,0;
SetDO ActivaSensor2,1;
DeactUnit STN2;
ActUnit STN1;
MoveJ home,v1000,z100,tWeldGun\WObj:=wobj0;
MoveAbsJ JointTarget_1,v1000,z100,tWeldGun\WObj:=Workobject_2;
Path_10;
Path_20;
Path_30;
Path_40;
Path_50;
Path_60;
Path_70;
Path_80;
Path_90;
Path_100;
Path_110;
Path_120;
Path_130;
Path_140;
Path_150;
MoveAbsJ JointTarget_2,v1000,z100,tWeldGun\WObj:=Workobject_2;
Path_160;
Path_170;
Path_180;
Path_190;
Path_200;
MoveAbsJ JointTarget_3,v1000,z100,tWeldGun\WObj:=Workobject_2;
Path_210;
Path_220;
Path_230;
Path_240;
```

```
    Path_250;
    MoveAbsJ JointTarget_4,v1000,z100,tWeldGun\WObj:=Workobject_2;
    Path_260;
    Path_270;
    Path_280;
    MoveAbsJ JointTarget_8,v1000,z100,tWeldGun\WObj:=Workobject_2;
    Path_290;
    Path_300;
    Path_310;
    MoveAbsJ JointTarget_9,v1000,z100,tWeldGun\WObj:=Workobject_2;
    Path_320;
    Path_330;
    Path_340;
    MoveAbsJ JointTarget_10,v1000,z100,tWeldGun\WObj:=Workobject_2;
    Path_350;
    Path_360;
    Path_370;
    MoveAbsJ JointTarget_11,v1000,z100,tWeldGun\WObj:=Workobject_2;
    Path_380;
    Path_390;
    Path_400;
    MoveJ home,v1000,z100,tWeldGun\WObj:=wobj0;
ENDPROC


!Trayectorias de soldadura
PROC Path_10()
    MoveJ Target_10,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_20,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_30,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
```

```
    MoveL Target_40,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_20()
    MoveL Target_50,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_60,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_70,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_80,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_30()
    MoveL Target_90,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_100,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_110,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_120,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_40()
    MoveL Target_130,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_140,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_150,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_160,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_50()
    MoveL Target_170,v1000,z100,tWeldGun\WObj:=Workobject_2;
```

```
    MoveL Target_180,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_190,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_200,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_60()
    MoveL Target_210,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_220,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_230,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_240,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_70()
    MoveL Target_250,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_260,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_270,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_280,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_80()
    MoveL Target_290,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_300,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_310,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_320,v1000,z100,tWeldGun\WObj:=Workobject_2;
```

```
ENDPROC

PROC Path_90()
    MoveL Target_330,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_340,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_350,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_360,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_100()
    MoveL Target_370,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_380,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_390,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_400,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_110()
    MoveL Target_430,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_440,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_450,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_460,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_120()
    MoveL Target_470,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_480,v100,fine,tWeldGun\WObj:=Workobject_2;
```

```
    SetDO SOLDAR,1;
    MoveL Target_490,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_500,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_130()
    MoveL Target_530,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_540,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_550,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_560,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_140()
    MoveL Target_570,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_580,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_590,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_600,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_150()
    MoveL Target_610,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_620,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_630,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_640,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC
```

```
PROC Path_160()
    MoveJ Target_660,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_670,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_680,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_690,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_170()
    MoveL Target_700,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_710,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_720,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_730,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_180()
    MoveL Target_740,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_750,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_760,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_770,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_190()
    MoveL Target_780,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_790,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
```

```
    MoveL Target_800,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_810,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_200()
    MoveL Target_820,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_830,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_840,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_850,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_240()
    MoveL Target_980,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_990,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1000,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1010,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_230()
    MoveL Target_940,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_950,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_960,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_970,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC
```

```
PROC Path_220()
    MoveL Target_900,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_910,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_920,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_930,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_250()
    MoveL Target_1020,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1030,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1040,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1050,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_210()
    MoveJ Target_860,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_870,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_880,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_890,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_260()
    MoveJ Target_1650,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1640,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1630,v100,fine,tWeldGun\WObj:=Workobject_2;
```

```
    SetDO SOLDAR,0;
    MoveL Target_1620,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_270()
    MoveL Target_1610,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1600,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1590,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1580,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_280()
    MoveL Target_1570,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1560,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1550,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1540,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_290()
    MoveL Target_1530,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1520,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1510,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1500,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_300()
```

```
    MoveL Target_1490,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1480,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1470,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1460,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_310()
    MoveL Target_1450,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1440,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1430,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1420,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_320()
    MoveL Target_1410,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1400,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1390,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1380,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_330()
    MoveL Target_1370,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1360,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1350,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
```

```
    MoveL Target_1340,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_340()
    MoveL Target_1330,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1320,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1310,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1300,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_350()
    MoveL Target_1290,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1280,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1270,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1260,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_360()
    MoveL Target_1250,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1240,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1230,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1220,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_370()
    MoveL Target_1210,v1000,z100,tWeldGun\WObj:=Workobject_2;
```

```
    MoveL Target_1200,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1190,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1180,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_380()
    MoveL Target_1170,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1160,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1150,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1140,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_390()
    MoveL Target_1130,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1120,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1110,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1100,v1000,z100,tWeldGun\WObj:=Workobject_2;
ENDPROC

PROC Path_400()
    MoveL Target_1090,v1000,z100,tWeldGun\WObj:=Workobject_2;
    MoveL Target_1080,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,1;
    MoveL Target_1070,v100,fine,tWeldGun\WObj:=Workobject_2;
    SetDO SOLDAR,0;
    MoveL Target_1060,v1000,z100,tWeldGun\WObj:=Workobject_2;
```

```
ENDPROC

PROC Path_410()
    MoveJ Target_10_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_20_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_30_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_40_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_420()
    MoveL Target_50_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_60_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_70_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_80_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_430()
    MoveL Target_90_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_100_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_110_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_120_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_440()
    MoveL Target_130_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_140_2,v100,fine,tWeldGun\WObj:=Workobject_1;
```

```
    SetDO SOLDAR,1;
    MoveL Target_150_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_160_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_450()
    MoveL Target_170_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_180_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_190_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_200_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_460()
    MoveL Target_210_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_220_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_230_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_240_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_470()
    MoveL Target_250_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_260_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_270_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_280_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC
```

```
PROC Path_480()
    MoveL Target_290_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_300_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_310_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_320_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_490()
    MoveL Target_330_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_340_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_350_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_360_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_500()
    MoveL Target_370_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_380_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_390_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_400_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_510()
    MoveL Target_430_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_440_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
```

```
    MoveL Target_450_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_460_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_520()
    MoveL Target_470_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_480_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_490_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_500_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_530()
    MoveL Target_530_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_540_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_550_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_560_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_540()
    MoveL Target_570_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_580_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_590_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_600_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC
```

```
PROC Path_550()
    MoveL Target_610_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_620_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_630_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_640_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_560()
    MoveJ Target_660_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_670_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_680_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_690_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_570()
    MoveL Target_700_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_710_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_720_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_730_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_580()
    MoveL Target_740_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_750_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_760_2,v100,fine,tWeldGun\WObj:=Workobject_1;
```

```
    SetDO SOLDAR,0;
    MoveL Target_770_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_590()
    MoveL Target_780_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_790_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_800_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_810_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_600()
    MoveL Target_820_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_830_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_840_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_850_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_610()
    MoveJ Target_860_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_870_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_880_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_890_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_620()
```

```
    MoveL Target_900_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_910_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_920_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_930_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_630()
    MoveL Target_940_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_950_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_960_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_970_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_640()
    MoveL Target_980_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_990_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1000_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1010_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_650()
    MoveL Target_1020_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1030_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1040_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
```

```
    MoveL Target_1050_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_660()
    MoveJ Target_1650_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1640_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1630_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1620_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_670()
    MoveL Target_1610_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1600_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1590_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1580_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_680()
    MoveL Target_1570_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1560_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1550_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1540_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_690()
    MoveL Target_1530_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
```

```
    MoveL Target_1520_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1510_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1500_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_700()
    MoveL Target_1490_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1480_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1470_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1460_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_710()
    MoveL Target_1450_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1440_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1430_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1420_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_720()
    MoveL Target_1410_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1400_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1390_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1380_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
```

```
ENDPROC

PROC Path_730()
    MoveL Target_1370_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1360_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1350_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1340_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_740()
    MoveL Target_1330_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1320_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1310_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1300_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_750()
    MoveL Target_1290_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1280_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1270_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1260_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_760()
    MoveL Target_1250_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1240_2,v100,fine,tWeldGun\WObj:=Workobject_1;
```

```
    SetDO SOLDAR,1;
    MoveL Target_1230_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1220_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_770()
    MoveL Target_1210_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1200_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1190_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1180_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_780()
    MoveL Target_1170_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1160_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1150_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1140_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC

PROC Path_790()
    MoveL Target_1130_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    MoveL Target_1120_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,1;
    MoveL Target_1110_2,v100,fine,tWeldGun\WObj:=Workobject_1;
    SetDO SOLDAR,0;
    MoveL Target_1100_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
ENDPROC
```

```
    PROC Path_800()
        MoveL Target_1090_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
        MoveL Target_1080_2,v100,fine,tWeldGun\WObj:=Workobject_1;
        SetDO SOLDAR,1;
        MoveL Target_1070_2,v100,fine,tWeldGun\WObj:=Workobject_1;
        SetDO SOLDAR,0;
        MoveL Target_1060_2,v1000,z100,tWeldGun\WObj:=Workobject_1;
    ENDPROC

ENDMODULE
```

9.2.2 Código de RAPID de la estación de ejemplo con Transportador.

El código **RAPID** de la estación con transportador del apartado **4.1.2.10** es el siguiente:

```
MODULE Module1
   CONST robtarget Target_10:=[[250,250,650],[0,0.923879532,-0.382683433,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST robtarget Target_20:=[[250,250,600],[0,0.923879532,-0.382683433,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST robtarget Target_30:=[[125,375,600],[0,0.923879532,-0.382683433,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST robtarget Target_40:=[[0,250,600],[0,0.923879532,-0.382683433,0],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST robtarget Target_50:=[[125,125,600],[0,0.923879532,-0.382683433,0],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST robtarget Target_60:=[[250,250,600],[0,0.923879532,-0.382683433,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST robtarget Target_70:=[[250,250,650],[0,0.923879532,-0.382683433,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   CONST jointtarget home:=[[0,0,0,0,30,0],[9E+09,9E+09,9E+09,9E+09,9E+09,0]];
   !*************************************************************
   !
   ! Módulo:  Module1
   !
   ! Descripción:
   !   <Introduzca la descripción aquí>
   !
   ! Autor: david
   !
   ! Versión: 1.0
   !
   !*************************************************************


   !*************************************************************
   !
   ! Procedimiento Main
   !
   !   Este es el punto de entrada de su programa
   !
   !*************************************************************
   PROC main()
      ConfL\Off;
      MoveAbsJ home,v1000,z100,MyTool\WObj:=wobj0;
      ActUnit CNV1;
      WaitWObj wobj_cnv1;
      Path_10;
      MoveAbsJ home,v1000,z100,MyTool\WObj:=wobj0;
```

```
    DropWObj wobj_cnv1;

ENDPROC

PROC Path_10()
    MoveL Target_10,v1000,z100,MyTool\WObj:=wobj_cnv1;
    MoveL Target_20,v1000,z100,MyTool\WObj:=wobj_cnv1;
    MoveC Target_30,Target_40,v500,fine,MyTool\WObj:=wobj_cnv1;
    MoveC Target_50,Target_60,v500,fine,MyTool\WObj:=wobj_cnv1;
    MoveL Target_70,v1000,z100,MyTool\WObj:=wobj_cnv1;
ENDPROC
 ENDMODULE
```

9.3. Hojas de Características.

9.3.1 Hoja de Características del robot ABB IRB2600.

Specification

Robot version	Reach (m)	Handling capacity (kg)	Wrist torque (Nm) Axis 4 & 5	Axis 6
IRB 2600-20/1.65	1.65	20	36.3	16.7
IRB 2600-12/1.65	1.65	12	21.8	10
IRB 2600-12/1.85	1.85	12	21.8	10
Number of axes	6+3 external (up to 36 with MultiMove)			
Protection	Standard IP67; optional FoundryPlus 2			
Mounting	Floor, wall, shelf, tilted, inverted			
Controller	IRC5 Single Cabinet			

Performance (according to ISO 9283)

	Position repeatability	Path repeatability
IRB 2600-20/1.65	0.04 mm	0.13 mm
IRB 2600-12/1.65	0.04 mm	0.14 mm
IRB 2600-12/1.85	0.04 mm	0.16 mm

Technical information

Electrical Connections	
Supply voltage	200-600 V, 50/60 Hz
Energy consumption	3.4 kW
Physical	
Robot base	676 x 511 mm
Robot height	
IRB 2600-20/1.65	1382 mm
IRB 2600-12/1.65	1382 mm
IRB 2600-12/1.85	1582 mm
Robot weight	
IRB 2600-20/1.65	272 kg
IRB 2600-12/1.65	272 kg
IRB 2600-12/1.85	284 kg
Environment	
Ambient temperature for mechanical unit	
During operation	+5°C (41°F) up to +50°C (122°F)
During transportation and storage	–25°C (13°F) up to +55°C (131°F)
During short periods (max. 24 h)	up to +70°C (158°F)
Relative humidity	Max. 95%
Noise level	Max. 69 dB (A)
Safety	Double circuits with supervision, emergency stops and safety functions, 3-positions enable device.
Emission	EMC/EMI-shielded
Options	Foundry Plus 2

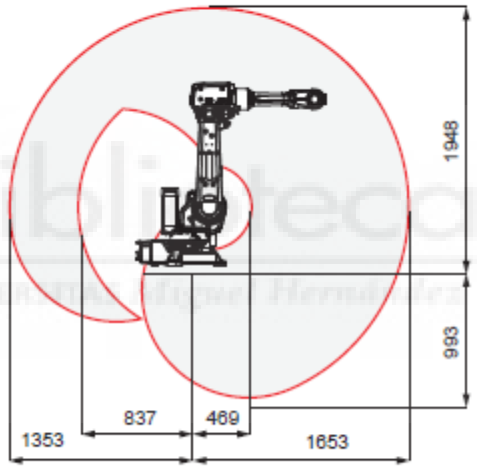
Data and dimensions may be changed without notice.

Movement

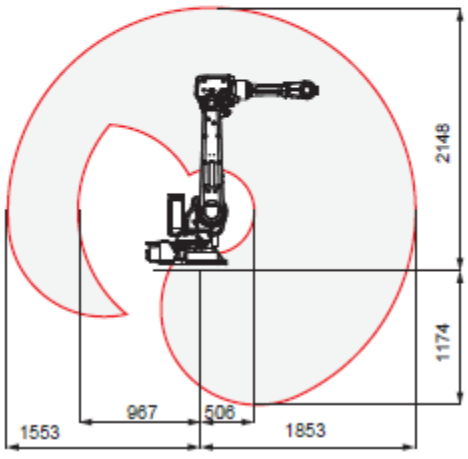
Axis movement	Working range	Axis max speed
Axis 1 rotation	+180° to -180°	175°/s
Axis 2 arm	+155° to -95°	175°/s
Axis 3 arm	+75° to -180°	175°/s
Axis 4 rotation	+400° to -400° Max. rev: +251 to -251	360°/s
Axis 5 bend	+120° to -120°	360°/s
Axis 6 turn	+400° to -400° Max. rev: + 274 to - 274	500°/s

A supervision function prevents overheating in applications with intensive and frequent movements.

Working range, IRB 2600-20/1.65, IRB 2600-12/1.65



Working range, IRB 2600-12/1.85



9.3.2 Hoja de Características del track ABB RTT Bobin.

3 Requisitos y datos técnicos

3.1 Datos técnicos

3.1.1 Rendimiento

En la tabla siguiente se proporcionan datos importantes para el rendimiento del track de desplazamiento.

Función	Rendimiento
Longitud de desplazamiento	1,7-11,7 m
Velocidad max.	1,06 m/s a 3.000 rpm en el motor
Longitud del soporte	3-13 m
Aceleración	2,5 m/s^2, 1,5 m/s^2 para +250 kg
Retardo	2,6 m/s^2, 1,6 m/s^2 para +250 kg
Precisión del repetidor[1]	
Carga máxima	250 kg además del peso del robot
Peso carro	220-430 kg
Grado de protección	IP54
Precisión en el posicionamiento	± 0,05 mm

1. Parada repetida en el sentido de desplazamiento, en el mismo punto.

Tiempo de posicionamiento

A la velocidad máxima con tiempos de aceleración y desaceleración incluídos:

Longitud de transporte	Tiempo
0,5 m	1,2 s
1 m	1,7 s
2 m	2,7 s

9.3.3 Hoja de Características del posicionador ABB IRBP-L.

Specification

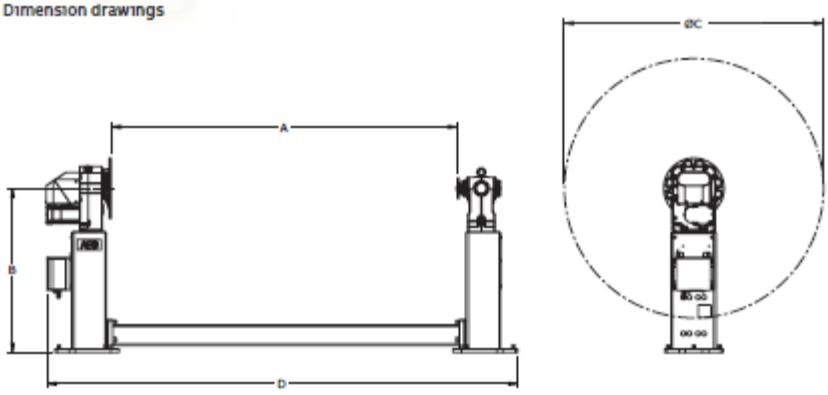
Variants	Handling capacity (kg)	Max continuous torque (Nm)	Max bending moment (Nm)	Repetitive accuracy (r=500)	Max rotation speed (°/s)
IRBP L-300	300	350	600	+/-0.05	180
IRBP L-600	600	650	3300	+/-0.05	150
IRBP L-1000	1000	900	5000	+/-0.05	150
IRBP L-2000	2000	3800	15000	+/-0.05	90
IRBP L-5000	5000	9000	60000	+/-0.05	39

Measurements

Variants	A	B	ø C	D
IRBP L-300	1250	950	1500	1979
	1600	950	1500	2329
	2000	950	1500	2729
	2500	950	1500	3229
	3150	950	1500	3879
	4000	950	1500	4729
IRBP L-600 & IRBP L-1000	1250	950	1500	2182
	1600	950	1500	2532
	2000	950	1500	2932
	2500	950	1500	3432
	3150	950	1500	4082
	4000	950	1500	4932
IRBP L-2000	1250	950	1500	2423
	1600	950	1500	2773
	2000	950	1500	3173
	2500	950	1500	3673
	3150	950	1500	4323
	4000	950	1500	5173
IRBP L-5000	-	1200	2200	-
	-	1200	2200	-
	-	1200	2200	-
	-	1200	2200	-
	-	1200	2200	-
	-	1200	2200	-

For complementary information, please see the product specification. ABB reserves the right to change data without notice.

Dimension drawings

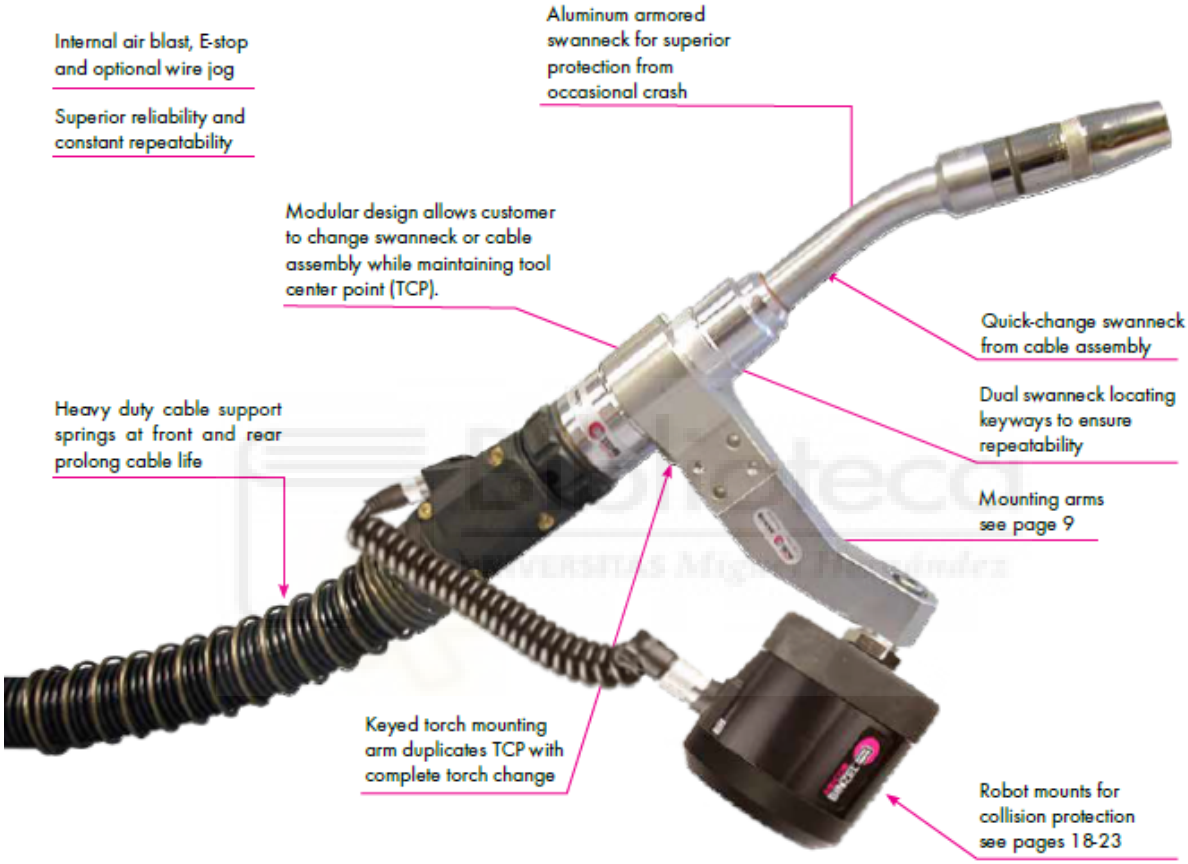


9.3.4 Hoja de Características de la herramienta de soldadura.

ABIROB® AIR COOLED ROBOTIC MIG GUNS
360 and 500 Amp

The modular design of the rugged, yet flexible, ABIROB® torches allow the user to quickly replace the swanneck or cable assembly while maintaining tool center point (TCP), assuring accurate repeatability and continuous precision welding. The keyed torch mounting-arm duplicates TCP with complete torch change. Available in 360 and 500 amperes, ABIROB® torches are second to none with aluminum armored swannecks for crash protection, internal air blast, internal control wire, emergency stop connector, and heavy duty cable support springs at front and rear to prolong cable life. Direct mounts allow ABIROB® torches to connect to most major wire feed models.

RUGGED YET FLEXIBLE.



ABIROB® A360 and A500 Air-Cooled

Type / Rating	*Shielding Gas CO_2	*Shielding Gas Mixed	Duty Cycle (%)	Wire Size Inches	Wire Size mm
A360	360	290	100%	.030 - .062	0.8 - 1.6 mm
A500	500	400	100%	.030 - .062	0.8 - 1.6 mm

Available torch neck geometries: 22, 35, 45, and 180 degrees.

*In case of mixed and/or pulsed arc welding, the rating data may be reduced to 35%

ABIROB® A360 / A500
With Performance Cable

ORDERING INSTRUCTIONS FOR COMPLETE ABIROB® GUNS

Example: R5A-4022EX-B

R5 (TORCH MODEL) **A** (TYPE) **-** **40** (CABLE LENGTH) **22** (SWANNECK ANGLE) **E** (BACK END) **X** (CONTROL CONNECTOR) **-** **B** (OPTIONS)

TORCH MODEL
R5 - 500 amp
R3 - 360 amp

TYPE
A - ABIROB Air-Cooled with Performance Cable

CABLE LENGTH
30 - 3 ft.
40 - 4 ft.
45 - 4.5 ft.
50 - 5 ft.
60 - 6 ft.
70 - 7 ft.
80 - 8 ft.
90 - 9 ft.
10 - 10 ft.
12 - 12 ft.

SWANNECK ANGLE
22 - 22 degree bend
45 - 45 degree bend
18 - 180 degree straight
22L - 22 degree bend long neck
45L - 45 degree bend long neck

BACK END
C - OTC Daihen® CMRE 741 AF4001
D - OTC Daihen® CM 147,231
E - Euro
F - Fronius® VR 1500/4000
K - Lincoln® PF10, 4R90
L - Lincoln® Synergic 7, LN Series
M - Miller®
P - Panasonic®
T - Motoman®/Tweco® #5

CONTROL CONNECTOR
X - None
A - ABICOR
B - Brad Harrison®
C - Burndy®
D - Fanuc®
E - Amp 4 Pin
F - Panasonic®
H - Denki® 2 Pin
K - OTC® 2 Pin

OPTIONS
A - Voltage Sense
B - Nozzle Sense
C - Wire Inch
E - No E-Stop
F - No Airblast
G - Wire Brake 035/045
H - Wire Brake 052/062

OHD. 0008

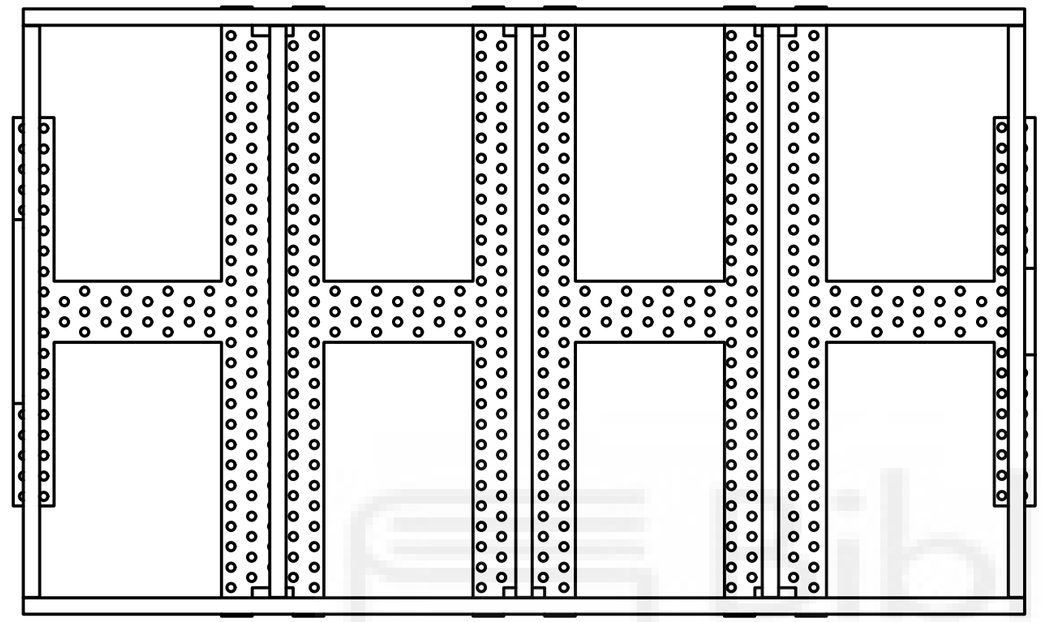
Fronius is a wholly owned company of Fronius International GmbH. OTC DAIHEN, Inc. is a wholly-owned subsidiary of DAIHEN Corporation. Lincoln® is a registered trademark of The Lincoln Electric Company. Panasonic® is a registered trademark of Panasonic Corporation. Miller® is a registered trademark of Miller Electric, Inc. Fanuc® is a registered trademark of Fanuc Ltd. Japan. Motoman® is a registered trademark of Yaskawa Electric Corporation. Tweco® is a registered trademark of Thermadyne Industries, Inc. Brad Harrison® is a registered trademark of Brad Harrison Co. Burndy® is a registered trademark of FCI-Burndy Product. ABB® is a registered trademark of ABB Corporation. Denki® is a registered trademark of American Denki Co. Ltd.

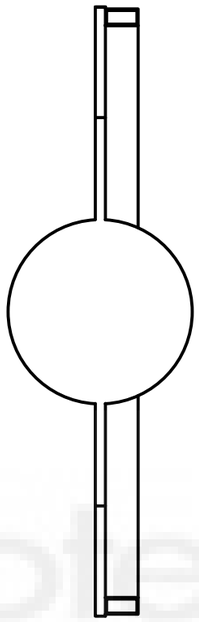
Collision Mounts
See pages 25 - 30

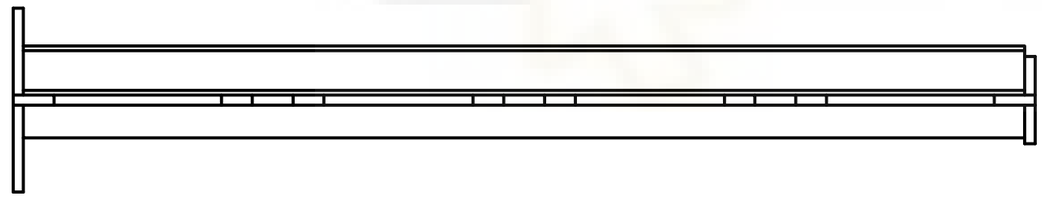
Mounting Arms
See Pages 8

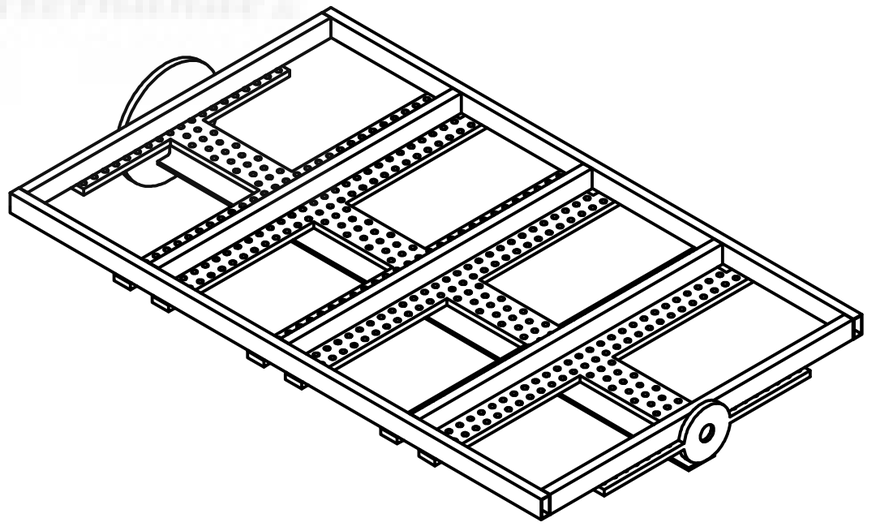
Alignment Jigs
See Page 24

9.4. Planos.





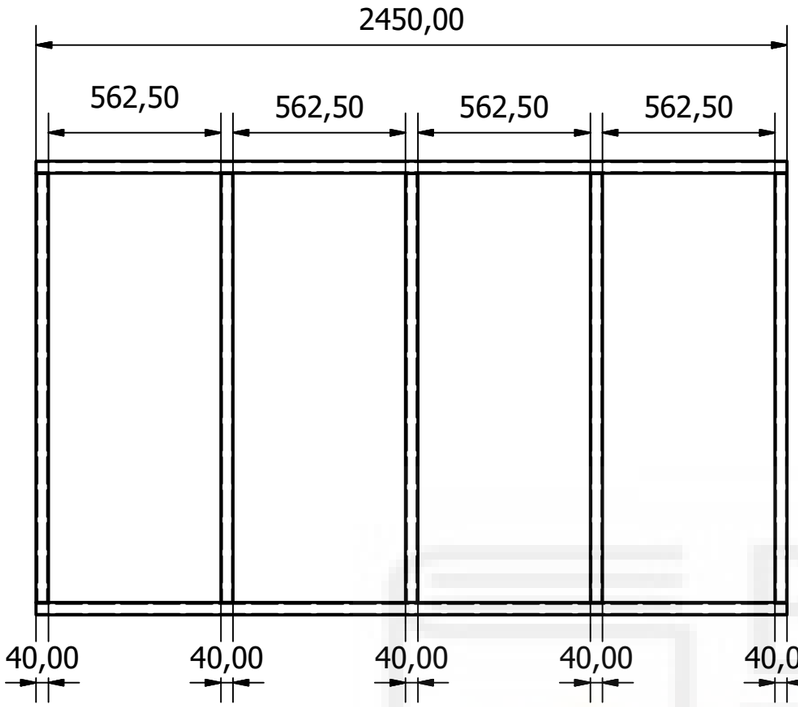


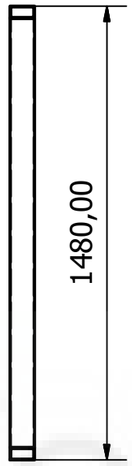


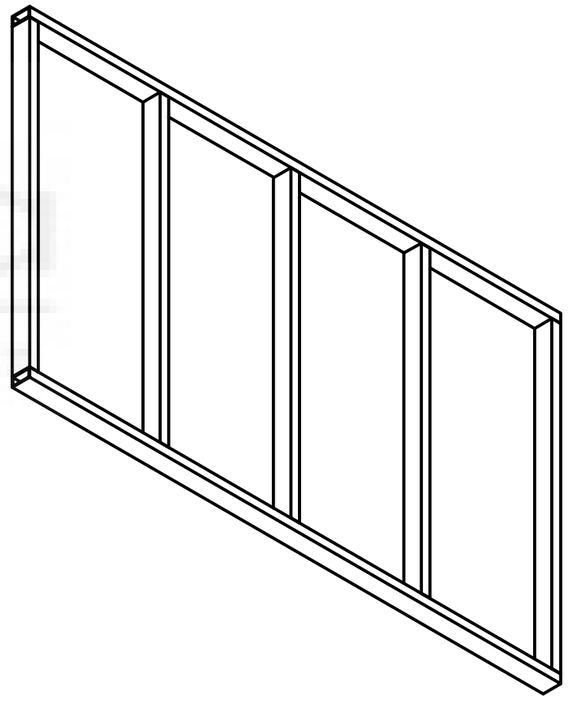
Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
David J. Roldán					01/09/2019	

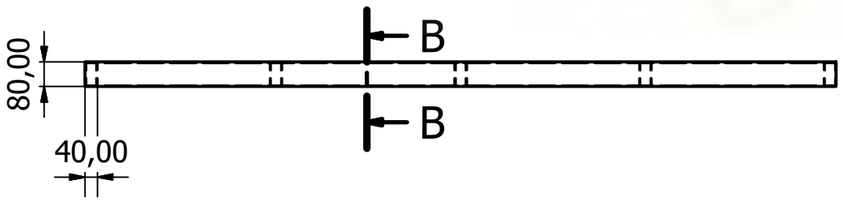
VISTA DE CONJUNTO ÚTIL + CHASIS

	Edición	Hoja
vista_conjunto		1 / 1

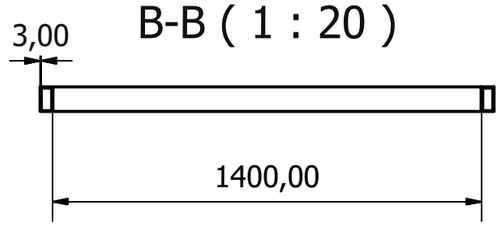








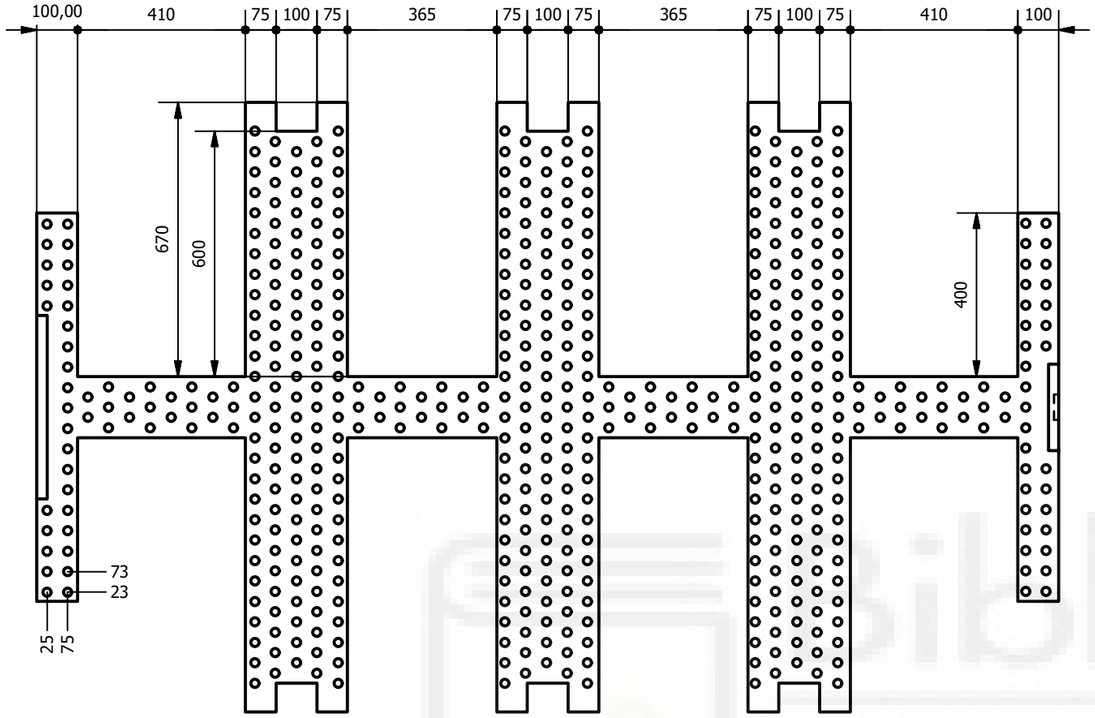
B-B (1 : 20)

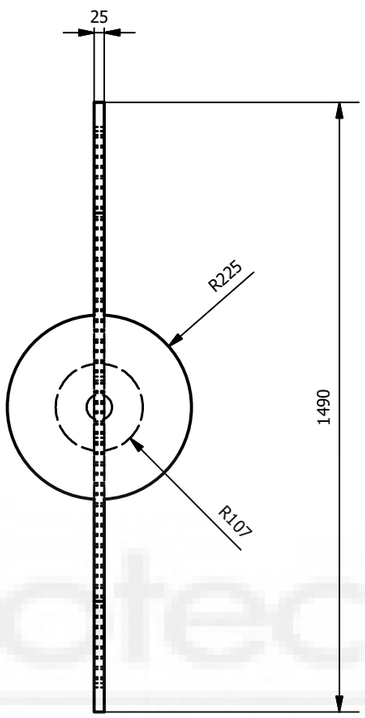


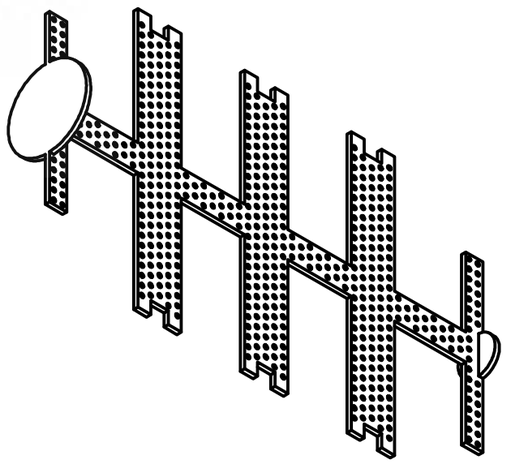
Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
David J. Roldán					01/09/2019	

MODELO DE CHASIS

	Edición	Hoja
		1 / 1



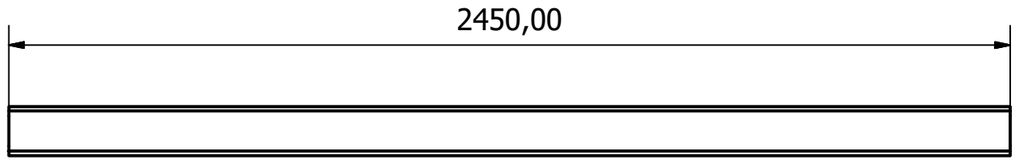


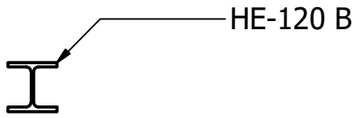


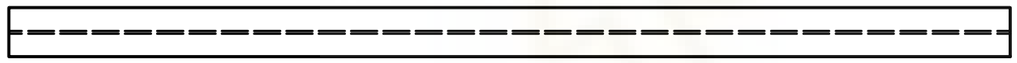
Diseño de	Revisado por	Aprobado por	Fecha	Fecha	
David J. Roldán				31/08/2019	

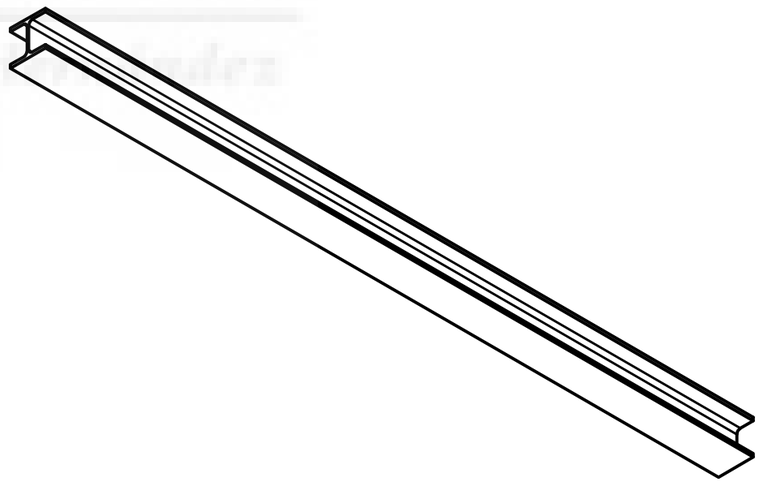
ÚTIL PARA SUJECCIÓN DE CHASIS

	Edición	Hoja
		1 / 1









Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
David J. Roldán					01/09/2019	

VIGA SOPORTE ÚTIL

	Edición	Hoja
		1 / 1