UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del diseño

Modelado de una célula robotizada y desarrollo de pruebas de simulación y experimentales

TRABAJO FINAL DE GRADO

en Ingeniería Eléctrica

REALIZADO POR

José Bonafont Cartagena

TUTORIZADO POR

Luís Ignacio Gracia Caladin

CURSO ACADÉMICO: 2020 / 2021

«*Quiero agradecer a la Universidad Politécnica de Valencia por esta etapa que me ha brindado y, en especial, a todos aquellos profesores que he tenido y han fomentado el desarrollo de mi curiosidad. También agradecer a mis compañeros, ahora amigos, por los inolvidables momentos que me han hecho vivir durante estos años. Y no puedo olvidarme de mi familia, que me ha apoyado en todo momento y ha hecho todo lo posible para que me convierta en la mejor versión de mí mismo.*»

RESUMEN

En el presente Trabajo de Final de Grado se realiza el modelado de una célula robotizada existente en el laboratorio de robótica, que se encuentra en el departamento de sistemas y automática de la Universidad Politécnica de Valencia.

Posteriormente, se realiza la simulación de un “Pick & Place” con el robot industrial IRB-140, en el entorno de programación de RobotStudio de la marca ABB. Este “Pick & Place” tratará de recoger las piezas que vengan desde una cinta transportadora y las llevará a distintos puntos de la mesa donde se encuentra el brazo robot.

Para finalizar, se realiza el mismo “Pick & Place”, anteriormente simulado en RobotStudio, pero esta vez de manera experimental.

En este proyecto se trata de ver la diferencia entre simular y ensayar de manera experimental un “Pick & Place”, y a su vez, analizar las ventajas de realizar una simulación previa a un trabajo real con un robot industrial.

Palabras clave del TFG: programación, modelado, robot industrial, simulación, Pick & Place.

ABSTRACT

In this Final Degree Project, is carried out the modeling of an existing robotic cell in the Robotics department of the Polytechnic University of Valencia.

Subsequently, a “Pick & Place” simulation is performed with the IRB-140 industrial robot, in the RobotStudio programming environment of the ABB brand. This "Pick & Place" will try to collect the pieces that come from a conveyor belt and take them to different points on the table where the robot arm is located.

Finally, the same “Pick & Place” is carried out, previously simulated in RobotStudio, but this time in an experimental way.

This project tries to see the difference between simulating and experimentally testing a “Pick & Place”, and in turn, analyzing the advantages of carrying out a simulation prior to a real job with an industrial robot.

Keywords of the TFG: programming, modeling, industrial robot, simulation, Pick & Place.

DOCUMENTOS DEL TRABAJO DE FIN DE GRADO

- **DOCUMENTO I……..MEMORIA DESCRIPTIVA**
- **DOCUMENTO II……PLIEGO DE CONDICIONES**
- **DOCUMENTO III……PRESUPUESTO**

Tabla de contenido de la memoria descriptiva

Tabla de contenido del pliego de condiciones

Tabla de contenido del presupuesto

Índice de figuras

DOCUMENTO I

MEMORIA DESCRIPTIVA

1. Objeto

En el presente Trabajo de Fin de Grado se va a realizar el estudio de un “Pick & Place”, tarea básica y muy utilizada en la industria, realizando una previa simulación con el entorno de programación RobotStudio de la marca ABB.

El objeto de este documento será analizar y comprobar las ventajas, o por el contrario inconvenientes, que se presentan al realizar simulaciones antes de realizar un trabajo con un robot.

Además, también servirá como guía para la realización tanto de simulaciones con RobotStudio, como para trabajos reales con robots de la marca ABB que, junto a Kuka, es una de las marcas más importantes en el sector de la robótica. Para ello se va a documentar paso a paso cada acción que se realice.

La tarea para realizar está compuesta de varios procesos, desde recrear el entorno de trabajo ayudándonos de las herramientas de modelado 3D que contiene RobotStudio, hasta la realización del “Pick&Place” en el entorno real, pasando por la programación de las trayectorias del robot.

Estos robots constan de 6 grados de libertad, gracias a los cuales son capaces de desplazarse por los recorridos necesarios para situarse en ciertos puntos, con ciertos ángulos, y poder realizar las tareas que les ordenemos.

Finalmente, otro de los objetos de este trabajo será comprobar las aptitudes de los entornos y robots para las siguientes características:

- Facilidad y comodidad en el entorno de programación para poder realizar tareas como desplazamiento de piezas.
- Seguridad a la hora de trabajar directamente con el robot.
- Flexibilidad en el manejo y accesibilidad para modificar cualquier parámetro en la línea de producción por cualquier operario.

2. Antecedentes de la robótica

Desde la antigüedad, el ser humano ha intentado construir máquinas que imitan partes del cuerpo humano. Los egipcios unieron brazos mecánicos a las estatuas de sus dioses, operados por sus sacerdotes quienes clamaban que el movimiento de estos era inspiración de sus dioses. Los griegos construyeron estatuas con sistemas hidráulicos, que utilizaban para fascinar a los adoradores de los templos. Sin embargo, todo esto no eran más que juguetes que no realizaban ningún trabajo útil.

En el siglo XX es cuando realmente su desarrollo comienza a tomar importancia. A principios de siglo se emplea por primera vez el término de robot, que proviene de la palabra checa *robota* cuyo significado es "trabajo forzado".

En 1954 se construye un brazo robot articulado fabricado para realizar movimientos programables, este brazo robot es considerado el primer robot industrial y da comienzo a una carrera para la creación de robots industriales que faciliten la vida en la propia industria, además de en la medicina, agricultura, en el hogar y un sinfín de sectores.

La evolución de los robots industriales se encuentra marcada por distintas generaciones:

- 1ª Generación: robots que repiten una misma tarea programada. Carece de sensores, no toma información del exterior.
- 2ª Generación: los robots ya llevan sensores para adquirir información del entorno. Los movimientos de esta generación son más complejos que la anterior. Su uso principal está enfocado en la industria automotriz
- 3ª Generación: se programan mediante lenguaje natural. Son reprogramables y tienen mayor percepción del entorno. Pueden realizar tareas automáticamente. Son el inicio de la inteligencia artificial.
- 4ª Generación: robots inteligentes, similares a los anteriores. Tienen sensores que envían constantemente información al ordenador de control y esta toma decisiones inteligentes.
- 5ª Generación: considerada como el futuro de los robots relacionado con la inteligencia artificial. Serán autónomos y podrán satisfacer las necesidades de los humanos. Los nuevos robots serán capaces de percibir y razonar en entornos dinámicos. Podrán simular funciones de la mente humana.

3. La robótica

"La robótica es la rama de la ingeniería mecánica, de la ingeniería eléctrica, de la ingeniería electrónica, de la ingeniería biomédica y de las ciencias de la computación, que se ocupa del diseño, construcción, operación, estructura, manufactura y aplicación de los robots."

3.1 La robótica industrial

La robótica industrial, junto con la automatización, es uno de los pilares que ha hecho posible la consolidación de la industria 4.0, consiguiendo una mayor productividad y eficiencia de los recursos de producción.

Los diferentes modelos de automatización industrial de la actualidad consiguen menores márgenes de error y procesos más precisos, al mismo tiempo que liberan a la mano de obra humana de tareas repetitivas o peligrosas.

La definición de robot industrial según la Asociación de Industrias Robóticas es:

"Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas, o dispositivos especiales, según trayectorias variables, programadas Para realizar tareas diversas"

Por tanto, la robótica industrial es una solución orientada a la mejora en la gestión de procesos.

3.2 El robot industrial

Los robots industriales son los más utilizados. Surgen de la alta exigencia y demanda de productividad en la industria. Son por lo general brazos robots grandes articulados destinados a su uso en la industria, para la realización de una tarea repetitiva o peligrosa, automatizando un proceso. Su definición según marca la norma **ISO 8373:2012** es:

''Manipulador multifuncional, controlado automáticamente, reprogramable en tres o más ejes, que puede estar fijo o móvil para uso en aplicaciones de automatización industrial.''

Figura 1 – Robot industrial

La cantidad de aplicaciones que puede tener un robot industrial ha ido aumentando desde su aparición hasta la actualidad, entre las numerosas aplicaciones para lo que se pueden emplear se encuentran las siguientes:

- Manipulación robotizada
- Ingeniería inversa
- Verificación de procesos
- Extracción robotizada
- Pick & place
- Pulido
- Modelado por inyección
- Empaquetado y paletizado
- Control de calidad
- Montaje y ensamblado
- Supervisión de maquinaria
- Mecanizado (Perforado, fresado, etc)
- Pruebas y análisis de laboratorio
- Pegado y soldadura

Debido a la gran importancia en la industria de estos robots, se ha creado una nueva industria que se dedica a la fabricación de estos robots. Entre los fabricantes más importantes destacan ABB Motors, KUKA Robotics, EPSON Robots y Kawasaki Heavy Industries.

4. Modelado de una célula robotizada en RobotStudio

El programa RobotStudio, de la marca ABB, es un entorno de desarrollo basado en el sistema operativo Windows por lo que se hace muy intuitivo en el manejo de sus funciones básicas como crear, cargar y guardar archivos.

En este trabajo de final de grado se utilizará la herramienta de RobotStudio para realizar un "Pick & Place" con el robot IRB-140 de la marca ABB.

RobotStudio es un programa de programación y simulación muy completo, presenta las siguientes características:

- Podemos crear una estación eligiendo el robot ABB que deseemos.
- Permite la importación de objetos 3D con formatos como CaD o la creación de estos objetos en el propio programa.
- Ofrece facilidades para la creación de células robotizadas (robot dentro de su entorno de trabajo).
- Permite programar sobre el robot directamente de una forma más gráfica o, por el contrario, usando lenguaje RAPID.
- Permite simular tareas o trayectorias programadas.
- Facilita la exportación de simulaciones dentro del programa a una estación real.

Esta herramienta presenta la posibilidad de simulación muy realistas, recreando el entorno de trabajo dónde se encuentra el robot.

4.1 Selección del Robot Industrial

El robot seleccionado es el compacto y potente **IRB 140 de la marca ABB**, puesto que es uno de los robots que se encuentran en el departamento de robótica de la UPV y, por tanto, vamos a poder ensayar de manera experimental las simulaciones que realicemos en RobotStudio.

Figura 2 - Robot IRB 140

El fabricante, en su web oficial, nos indica que el robot industrial multiusos de 6 ejes IRB 140 soporta una carga de 6 kg con un alcance de 810 mm (con respecto al eje 5). Se puede instalar en el suelo, de manera invertida o en la pared desde cualquier ángulo.

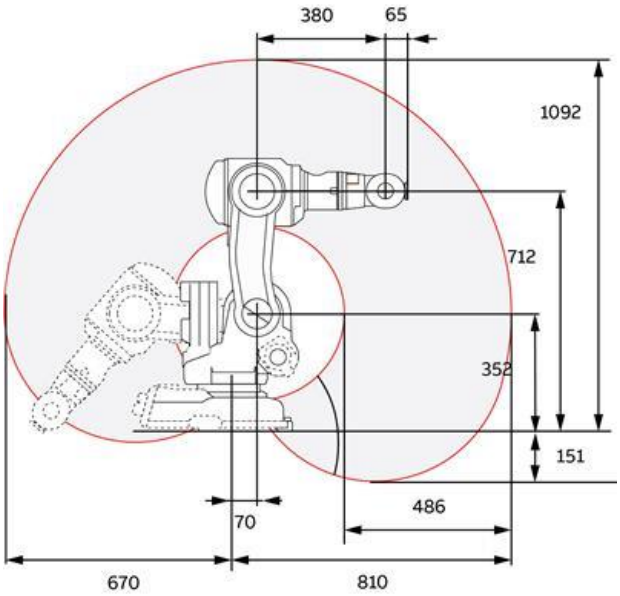


Figura 3 - Área de trabajo IRB 140

Tiene disponibles varias versiones: estándar, FoundryPlus, Clean Room y Hermético, en el que el robot tiene una completa protección IP67 que lo hace apropiado para una extensa gama de aplicaciones.

Su diseño robusto, con cables completamente integrados, además de su completa flexibilidad y su función de detección de colisiones con completa retracción de la trayectoria, aseguran la fiabilidad y seguridad de este robot.

4.2 Descarga del RobotStudio

Antes de empezar a trabajar con el programa deberemos descargarlo desde la pagina oficial de ABB con el siguiente enlace:

https://new.abb.com/products/robotics/es/robotstudio/descargas

Es muy importante solicitar una licencia, de lo contrario podremos utilizar el programa pero en una versión de prueba, en la que tendremos limitadas opciones muy básicas como guardar un archivo, importar bibliotecas, realizar grabaciones, etc.

En nuestro caso, disponemos de una licencia para estudiantes gracias a la Universidad Politécnica de Valencia, que utilizaremos desde casa conectados a la red VPN de la UPV.

4.3 Creación de una nueva estación

Una vez abierto el programa, tras haberlo descargado e instalado con su licencia correspondiente, deberemos crear una nueva estación vacía. Como RobotStudio es un programa basado en Windows este primer paso nos resultará muy intuitivo, solo tendremos que seguir la ruta:

Archivo → Nuevo → Estación vacía → Crear

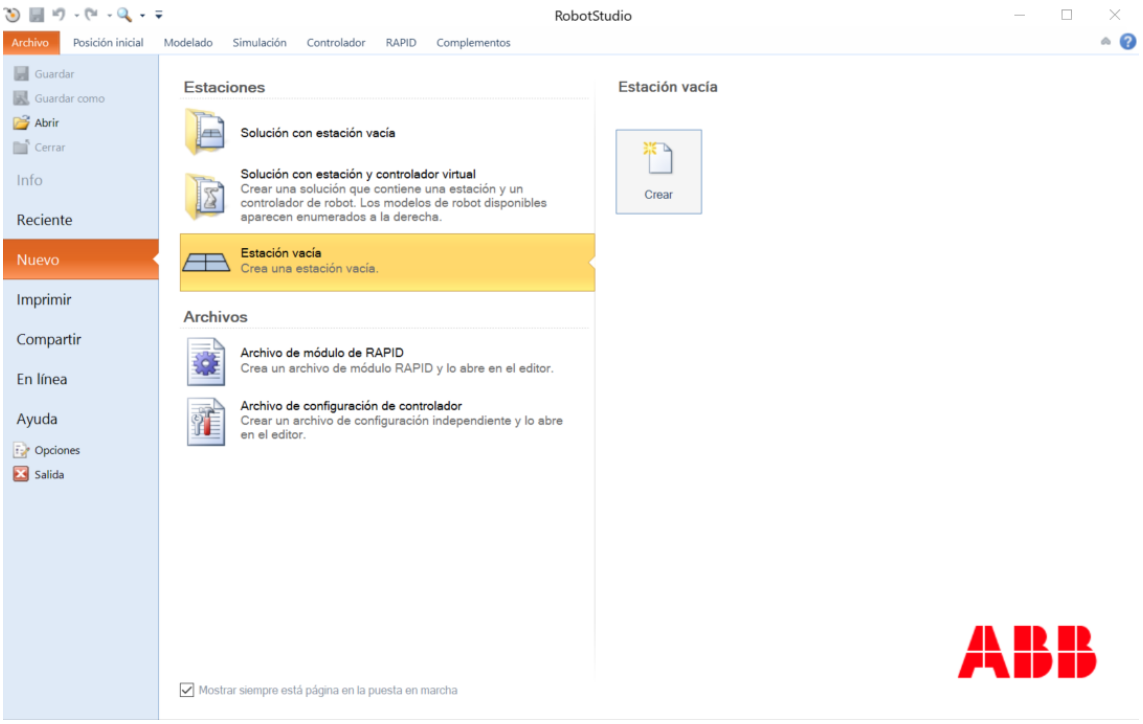


Figura 4 - Creación de una estación vacía

Una vez realizado este sencillo primer paso, se nos abrirá la estación vacía. Deberemos comprobar que la licencia del programa es correcta fijándonos en las notificaciones que aparecen en la parte inferior del programa.

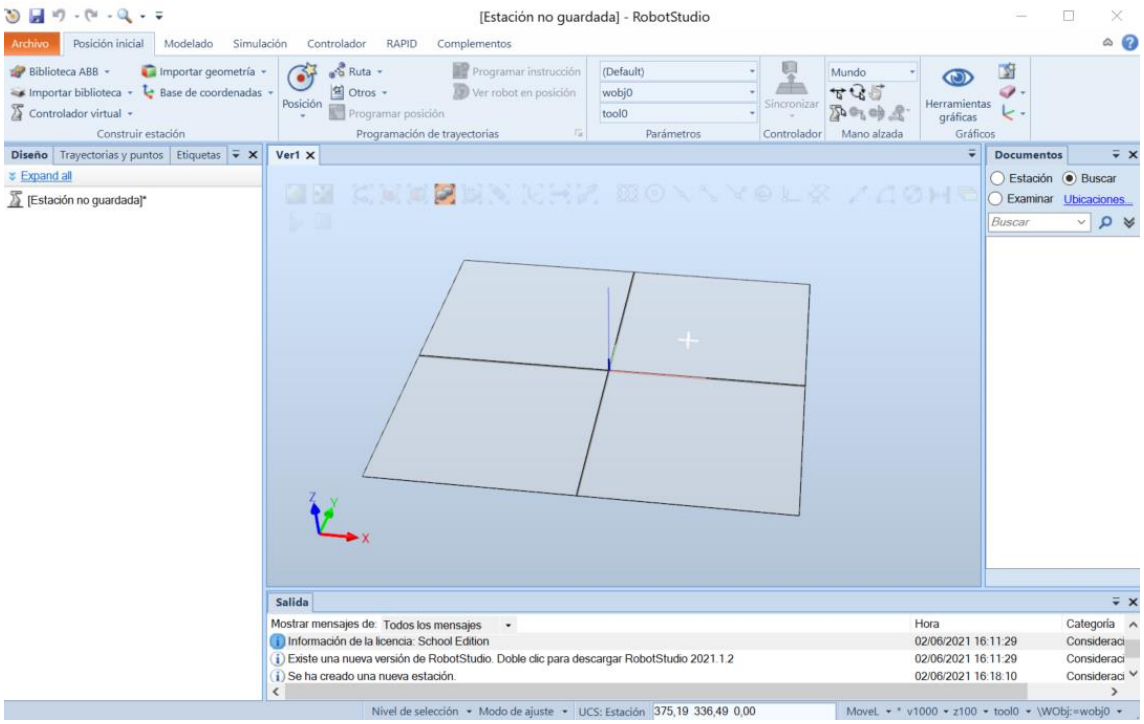


Figura 5 - Estación vacía

Con la estación vacía podemos ver todas las opciones que ofrece el programa, situadas en la parte superior de la ventana:

- Posición inicial
- Modelado
- Simulación
- Controlador
- RAPID
- Complementos

Explicaremos más adelante para que se utilizan cada una de ellas conforme las vayamos usando a lo largo de este proyecto.

4.4 Implementación del IRB-140

Una vez creada la estación, deberemos implementar el robot escogido anteriormente para este proyecto, el IRB-140 de ABB.

Este Robot, junto con otros de la marca ABB, se encuentra dentro de las bibliotecas del programa, por lo que sólo tendremos que seleccionarlo. La ruta que deberemos seguir es la siguiente:

Posición inicial → Biblioteca ABB → IRB 140

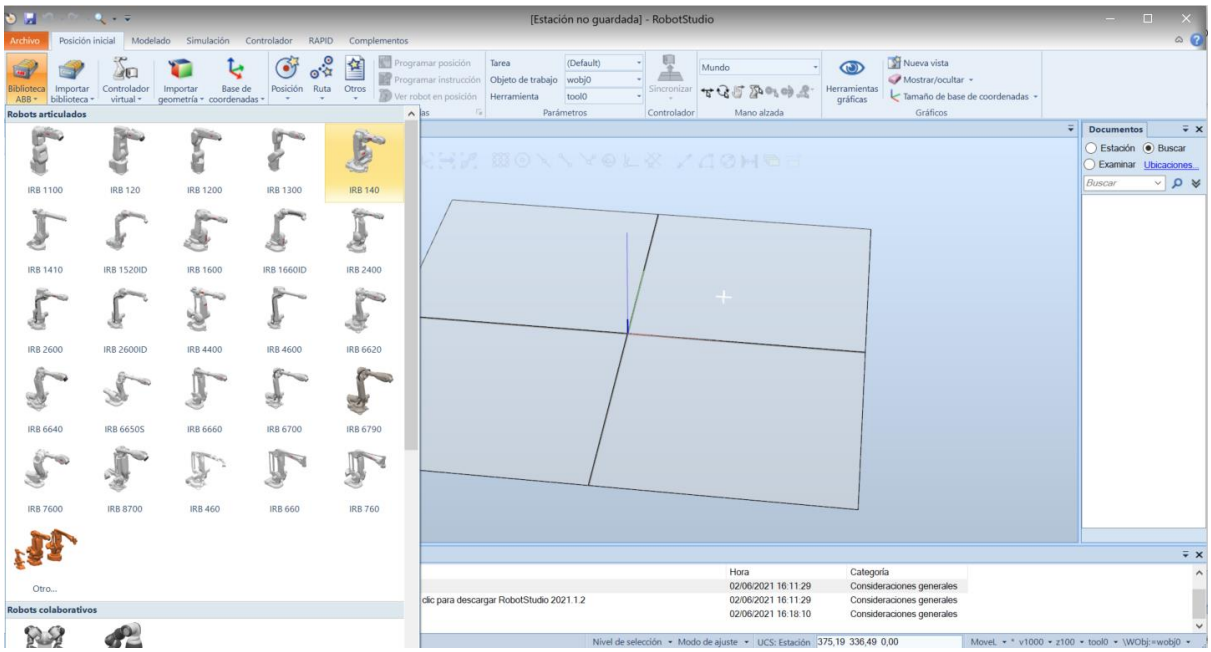


Figura 6 - Implementación IRB 140

Al clicar sobre el modelo que queremos, el IRB-140, se colocará en el sistema de coordenadas predeterminado que genera el programa. También nos aparecerá en la pestaña *Diseño*, dónde podremos hacer clic derecho sobre el para ver las distintas opciones que nos da el programa para realizar sobre el robot.

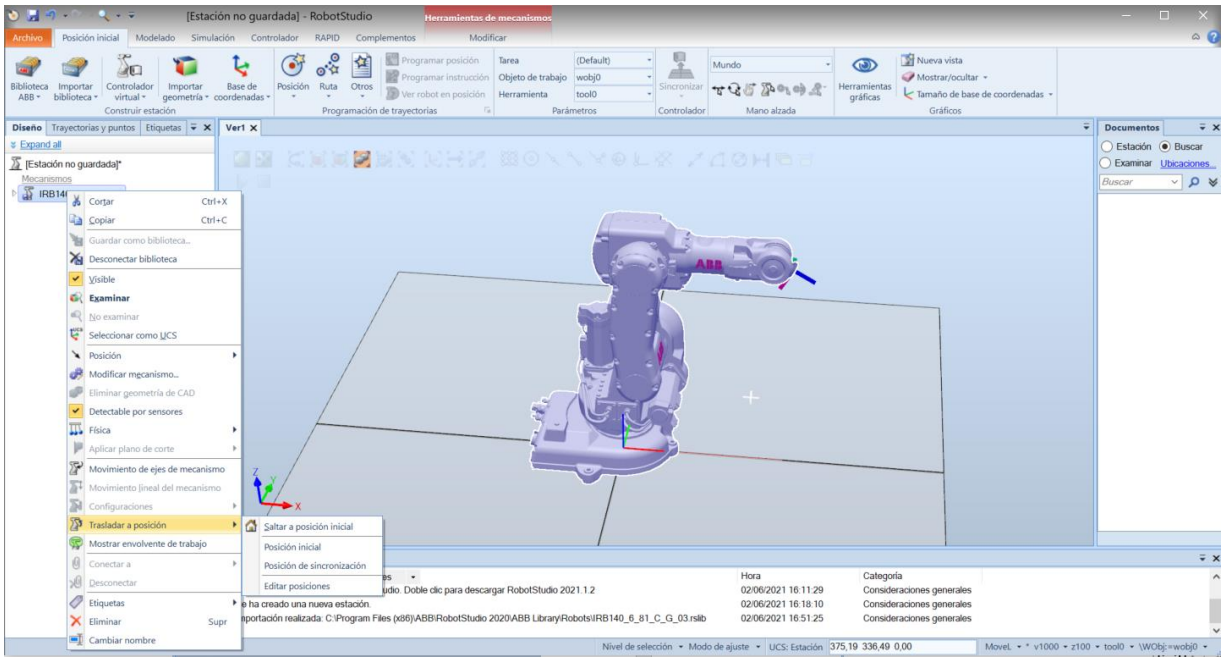


Figura 7 - IRB 140 en la estación

El siguiente paso para poder programar el robot será añadir el controlador a la estación. El controlador se encargará de dejar el robot operativo para así poder programarlo y moverlo a nuestro placer. Para hacer esto deberemos ir al apartado *Complementos* y descargar la última versión del RobotWare para el IRC5, de la siguiente manera:

Complementos → RobotApps → RobotWare para IRC5 → Versión 6.12.01 → Añadir

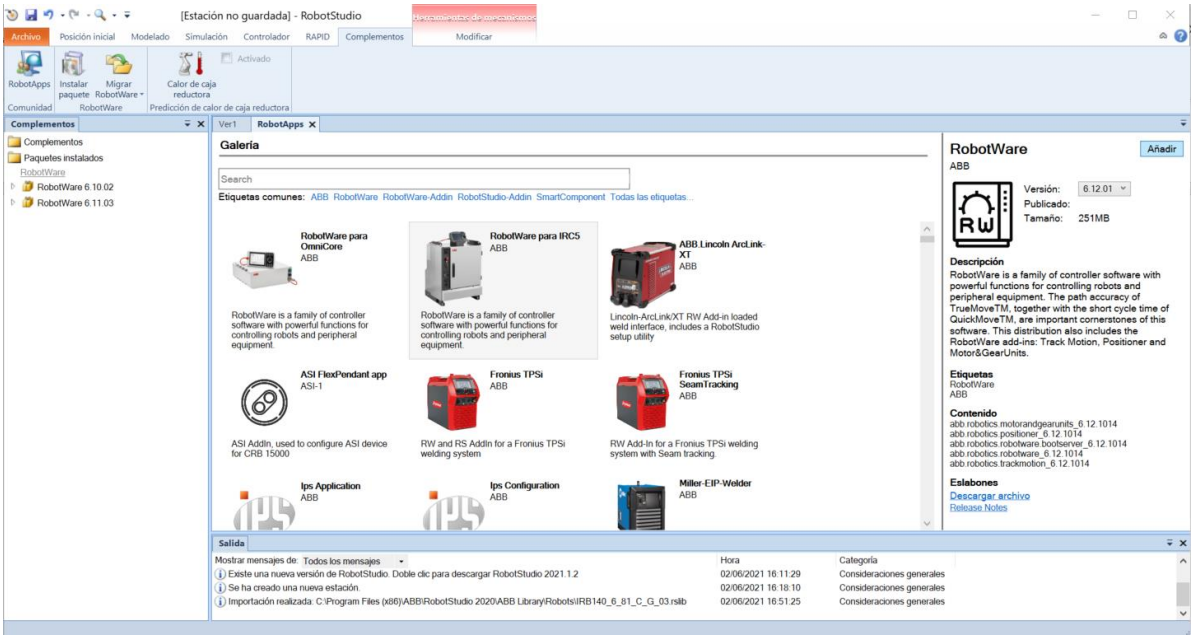


Figura 8 - Añadir controlador

4.5 Creación e implementación de la herramienta

A continuación, deberemos añadir la herramienta que utilizará el robot IRB-140 para poder coger los objetos que vienen por la cinta y transportarlos hasta la mesa.

Para este caso hemos seleccionado una herramienta con una ventosa en su extremo de 310 mm de longitud.

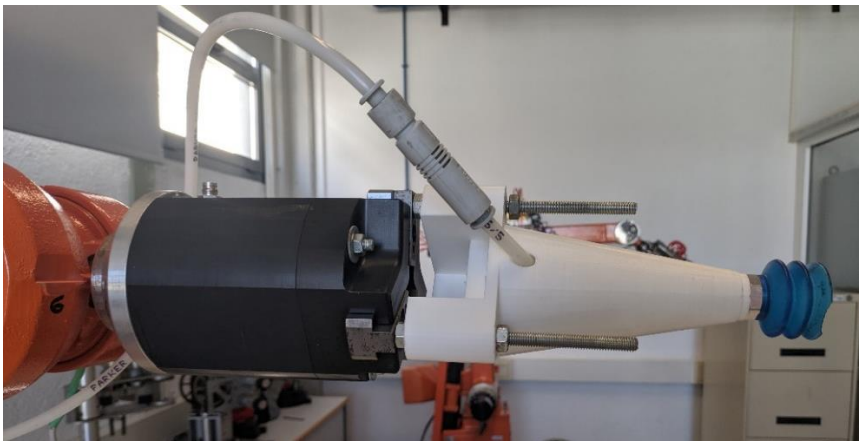


Figura 9 - Herramienta ventosa

RobotStudio permite importar bibliotecas propias con herramientas y objetos que pueden servir para modelar nuestro entorno, incluso ofrece algunas bibliotecas predeterminadas del programa donde aparecen herramientas y objetos que se usan frecuentemente que nos pueden ser muy útiles.

Importar biblioteca → Equipamiento → Herramientas

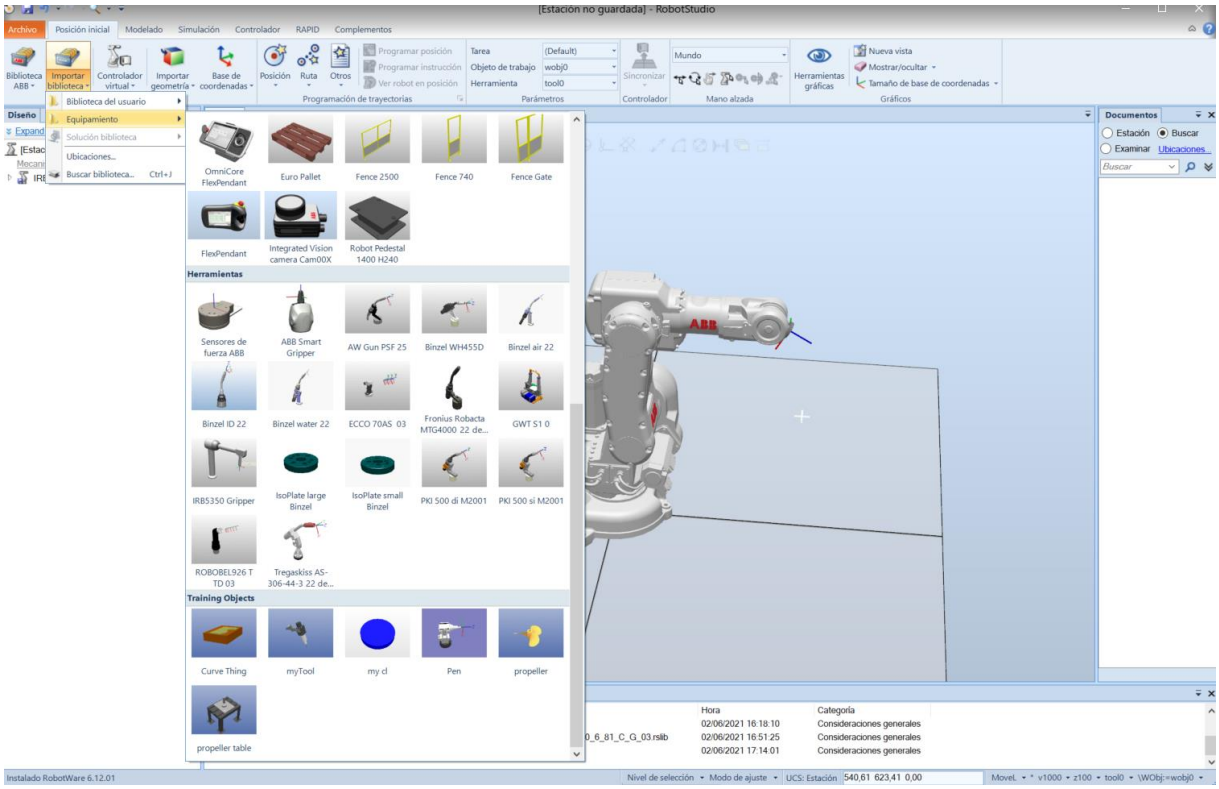


Figura 10 - Importar biblioteca

Como podemos observar, RobotStudio no dispone ninguna herramienta igual o parecida a la que queremos usar en nuestro proyecto, por lo que tendremos que crearla en CaD e importarla como una biblioteca.

También, como alternativa a crearla en CaD, podremos crear la herramienta con el propio RobotStudio, aunque será de una manera menos exacta y sin profundizar en detalles.

En este proyecto vamos a optar por crearla con el RobotStudio ya que no disponemos de ninguna licencia para realizarla con otro programa y no necesitamos que la herramienta sea exactamente igual para la simulación, lo único que nos importará es que presente las mismas dimensiones que la real.

Para que nos sea más cómodo crear la herramienta, ocultaremos el robot, haciendo que no se vea en la estación y nos interfiera a la hora de trabajar.

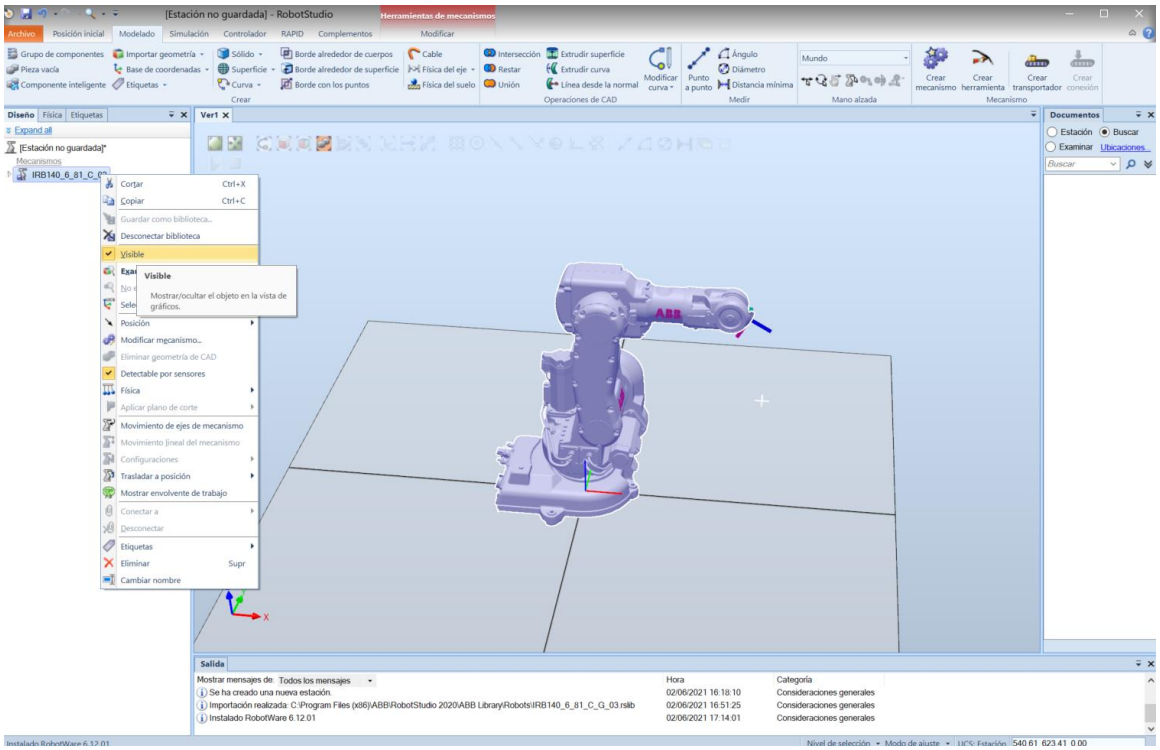


Figura 11- Ocultar robot

El primer paso para crear la herramienta será crear el sistema de coordenadas que irá en su extremo, para ello lo crearemos a una distancia en el eje vertical Y, respecto al eje de coordenadas del entorno, igual a la longitud de la herramienta. Para crear el eje de coordenadas de la herramienta haremos:

Posición inicial → Base de coordenadas → Creación de un sistema de coordenadas

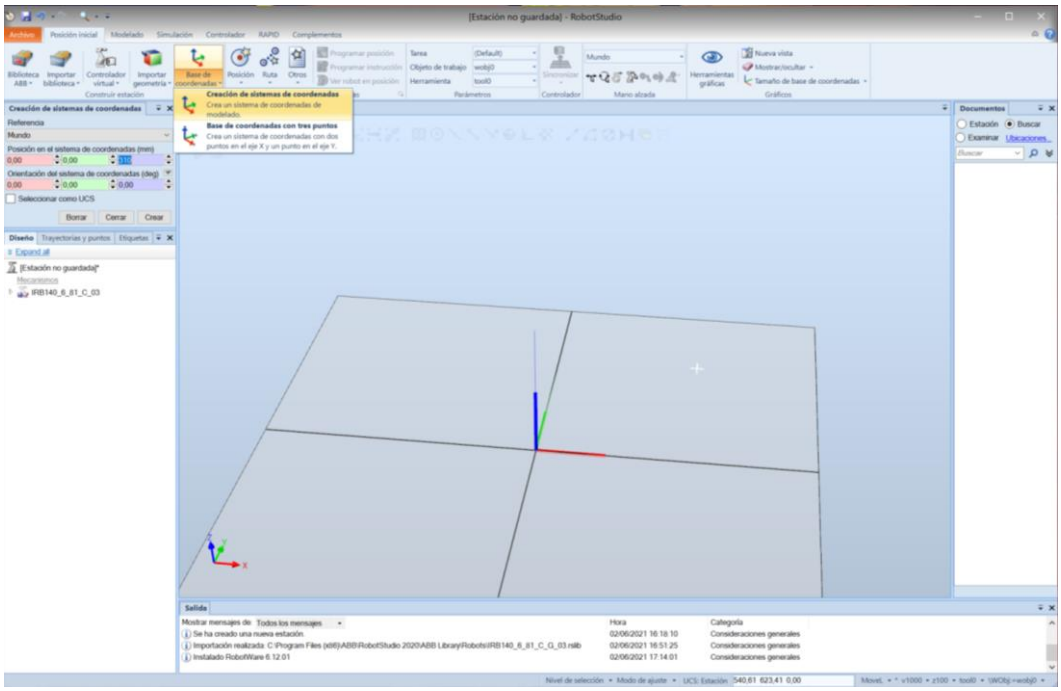


Figura 12 - Creación sistema de coordenadas de la herramienta

Una vez creado el sistema de coordenadas para la herramienta que queremos crear ya podemos generar la herramienta con el RobotStudio siguiendo estos pasos:

Modelado → Crear herramienta

Se nos abrirán la siguiente pestaña donde sólo tendremos que ponerle nombre a la nueva herramienta y seleccionar el sistema de coordenadas de la herramienta que hemos creado anteriormente.

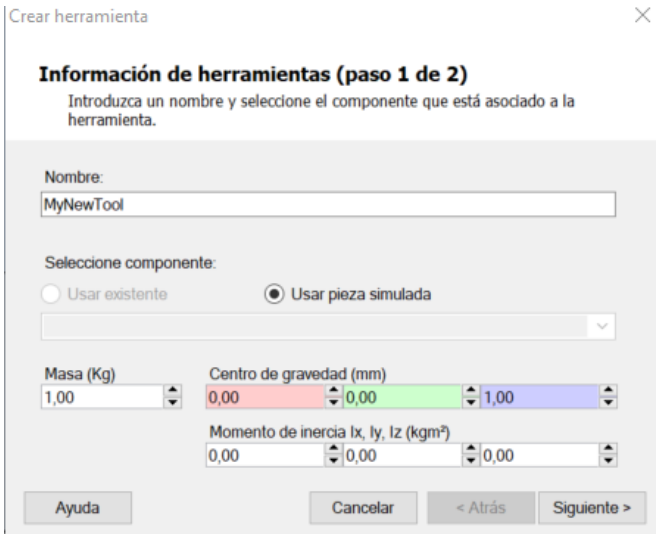


Figura 13 - Crear herramienta (paso 1)

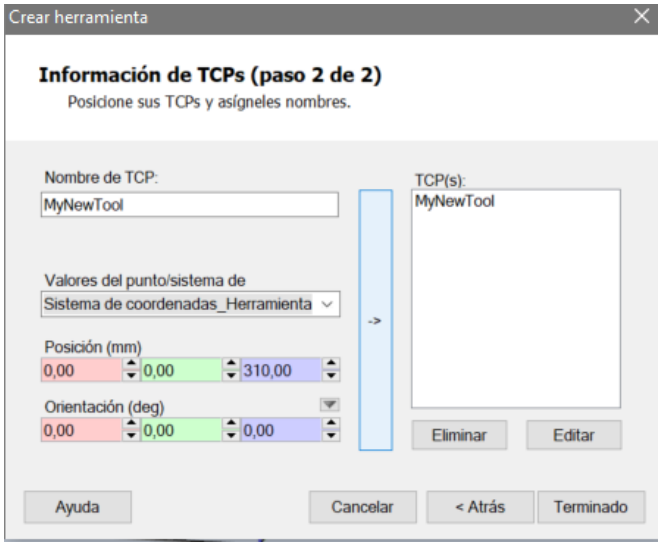


Figura 14 - Crear herramienta (paso 2)

Una vez creada la herramienta, aparecerá directamente situada sobre el eje de coordenadas del entorno con su extremo situado en el eje de coordenadas de la herramienta que habíamos creado anteriormente.

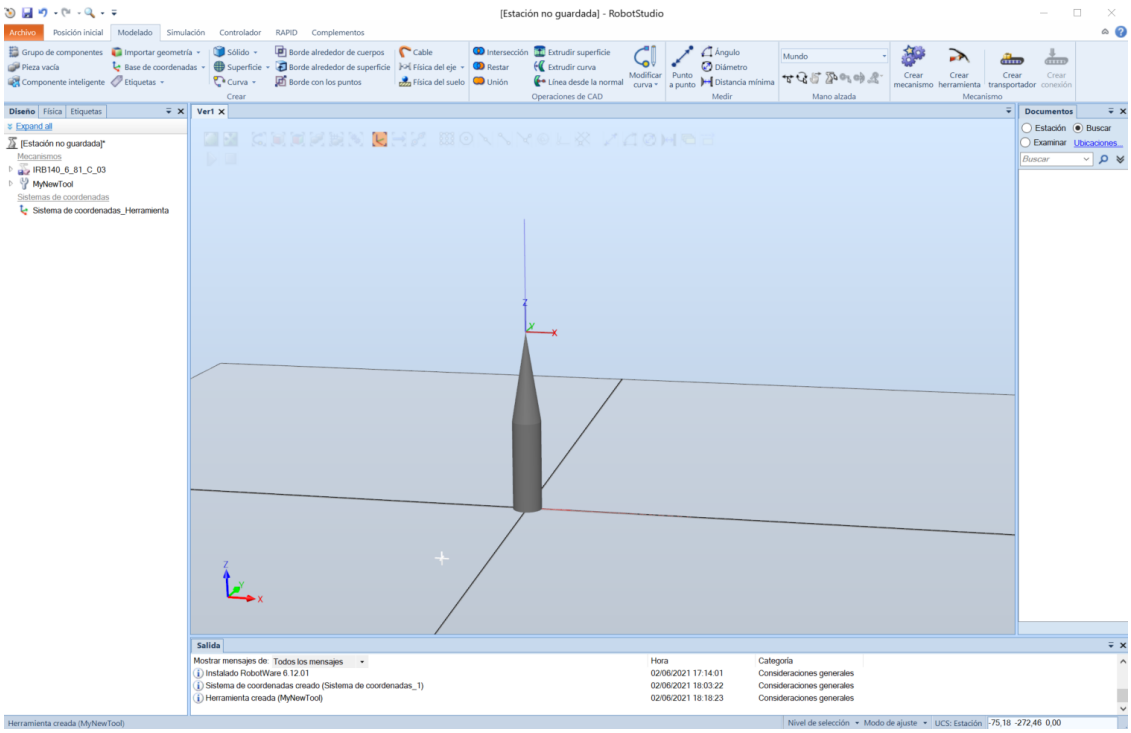


Figura 15 - Herramienta sobre el entorno

Ahora que ya está creada la herramienta, sólo nos queda implementarla en el extremo del robot. Para ello volveremos a hacer visible el robot y arrastraremos la herramienta hasta el robot dentro de la pestaña de *Diseño*. Podemos observar que el eje vinculado a la herramienta también se desplaza junto a ella.

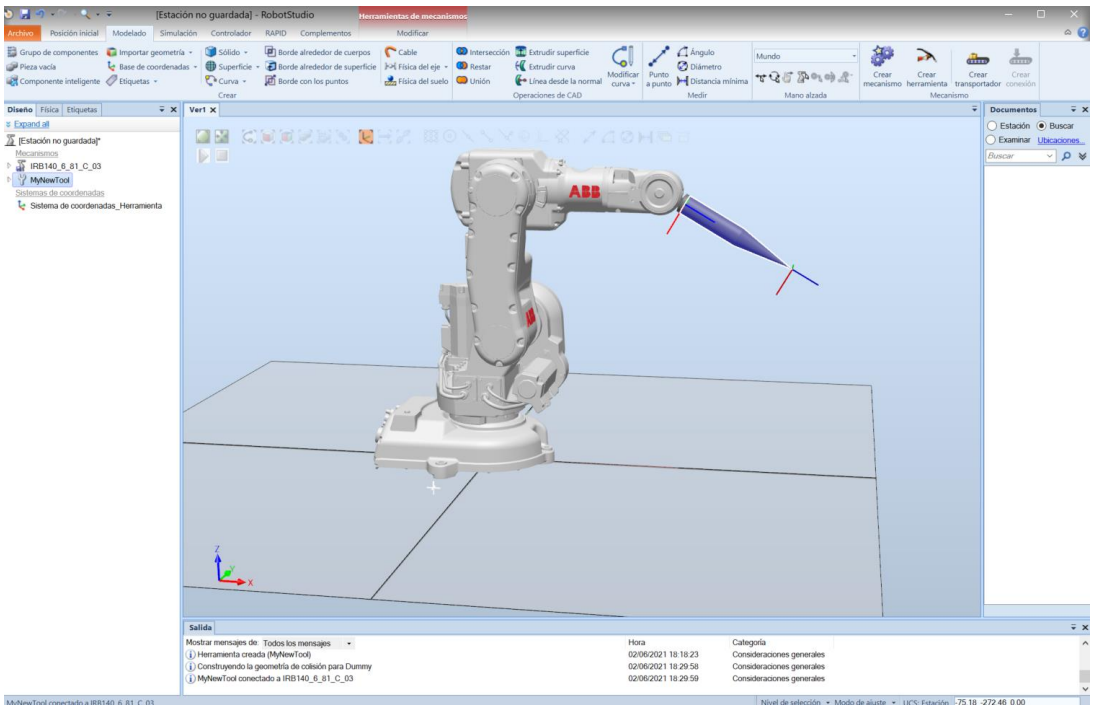


Figura 16 - Situar herramienta en el robot

4.6 Modelación del entorno

Una vez tenemos el robot IRB-140 a punto para trabajar con él solo nos faltará replicar el entorno donde este se encuentra.

Además de para hacer la simulación los más parecida a la realidad posible, es importante realizar un modelado del entorno para asegurarse de que los movimientos y trayectorias que le ordenemos al robot son correctas y no tienen riesgo de colisión con ningún elemento del entorno.

Como hemos mencionado antes, el robot IRB-140 se encuentra en el laboratorio de que se encuentra en el departamento de ingeniería de sistemas y automática, situado en la segunda planta del edificio 5D (DISA – ETSII).

Figura 17 - Ubicación del laboratorio de robótica

Para empezar con el proyecto tuvimos que ir para fotografiar y tomar medidas del laboratorio pudiéndolo así modelar en RobotStudio.

Figura 18 - Fotografía del laboratorio

Se puede observar que en el entorno aparecen 2 pequeños robots más encima de la mesa, estos no los vamos a modelar ya que el que está a la izquierda no interfiere en el área de trabajo de nuestro robot y el otro se encuentra sobre una bandeja que posibilita moverlo de sitio.

Una vez conocemos como es el entorno del robot y sus medidas podemos empezar a modelarlo.

Lo primero que vamos a hacer va a ser ocultar en el RobotStudio el robot, para que no interfiera a la hora de modelar y nos sea más cómodo.

Lo siguiente será ir al apartado *Modelado*, y crearemos un *Grupo de componentes*, que llamaremos, por ejemplo, *Mesa1.*

Modelado → Grupo de componentes → Cambiar nombre

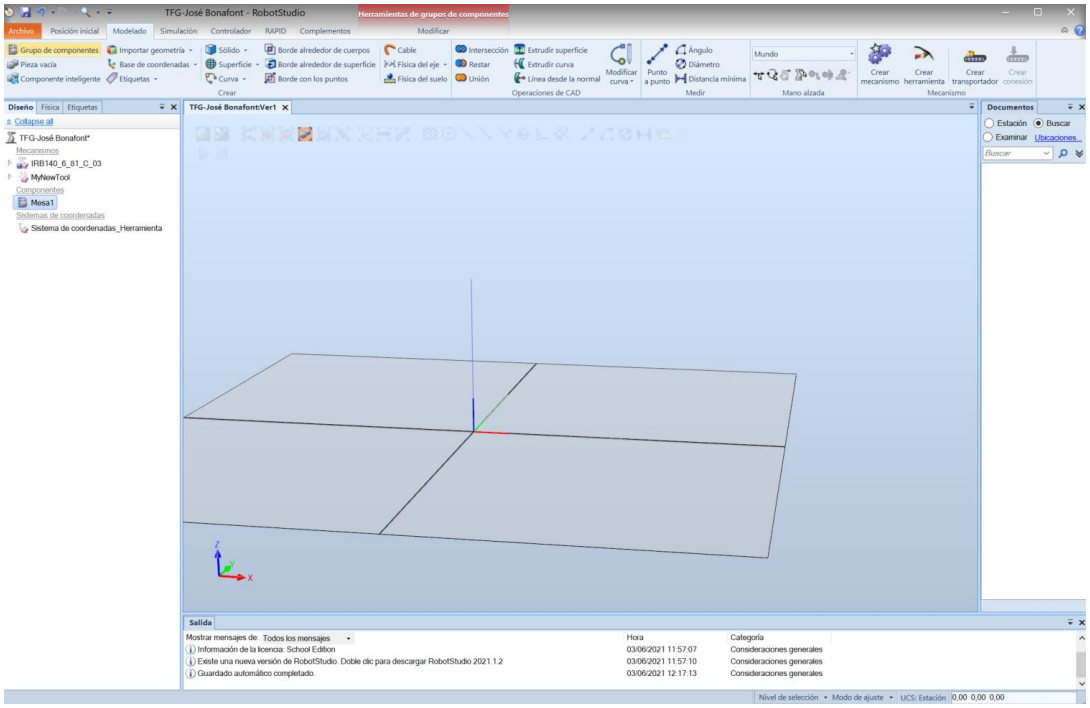


Figura 19 - Crear grupo de componentes

En este grupo de componentes que acabamos de crear añadiremos todos los sólidos que van a componer la mesa principal, dónde se encuentra la cinta transportadora. Es decir, tendremos que crear un tetraedro (tablero de la mesa) y cuatro cilindros (patas de la mesa).

Modelado → Sólido → Tetraedro

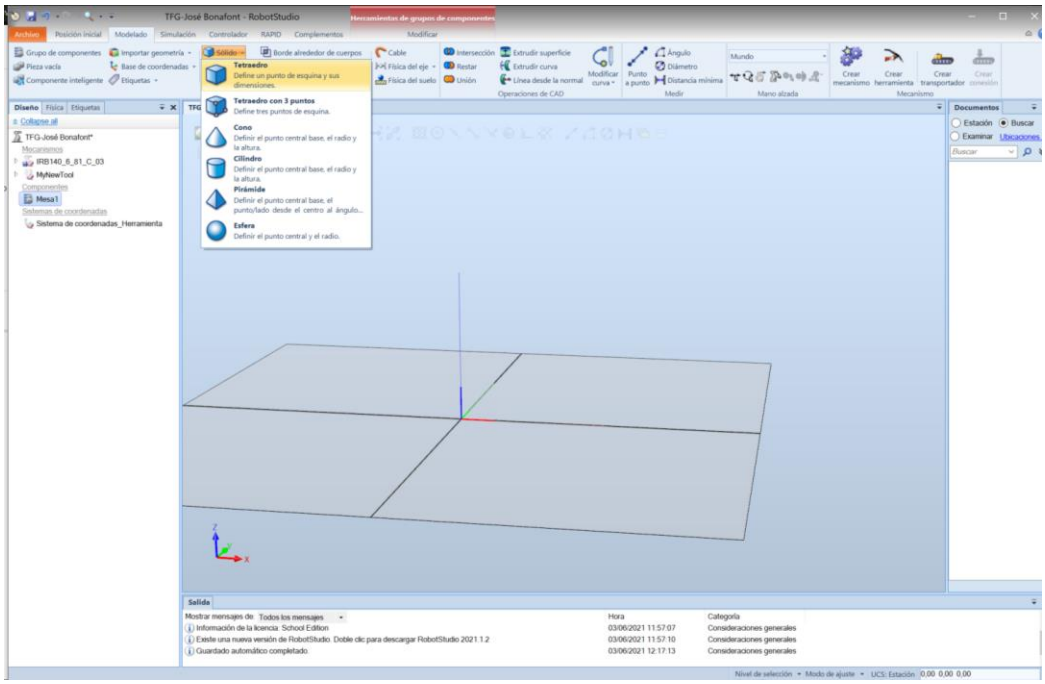


Figura 20 - Crear sólido

Al seguir estos pasos se nos abrirá un cuadro que tendremos que rellenar con las coordenadas dónde se situará la pieza que queremos crear y sus dimensiones. Por comodidad usaremos el sistema de coordenadas predeterminado y una vez modelado todo el entorno lo desplazaremos hasta su posición correcta respecto al robot.

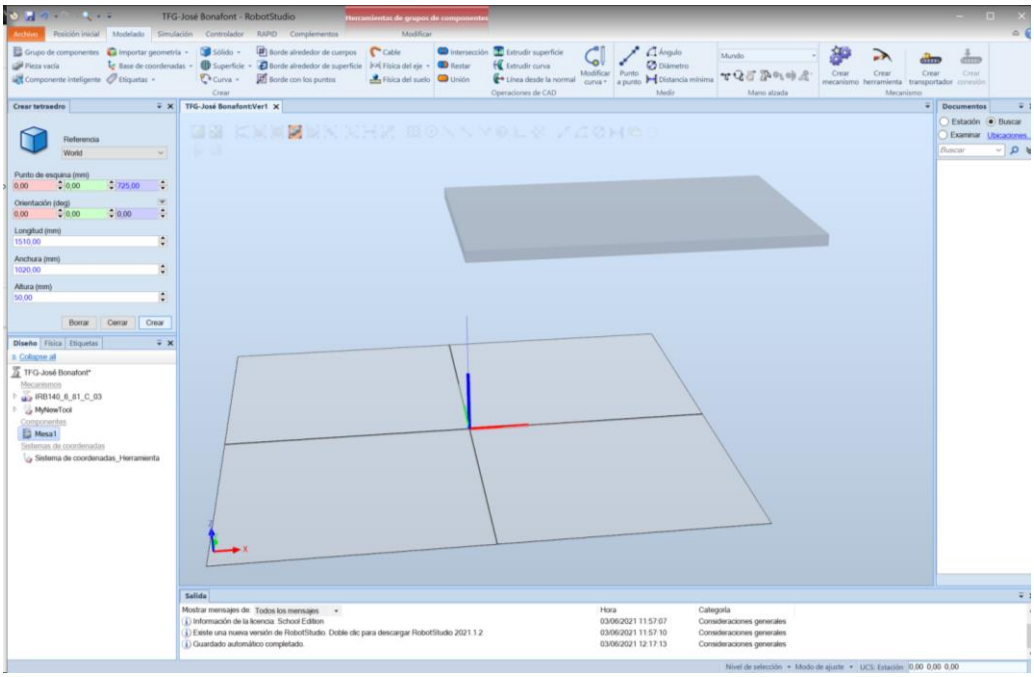


Figura 21 - Crear tetraedro

Repetiremos el proceso, pero esta vez creando cilindros, para modelar las patas de la mesa y completar el modelado de la mesa principal.

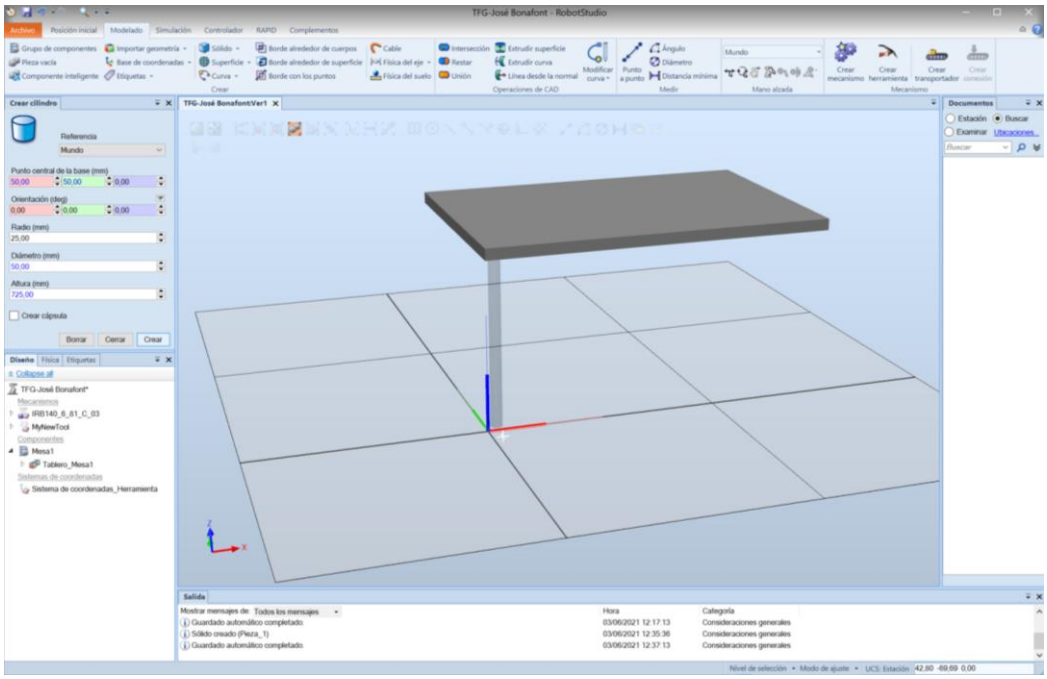


Figura 22- Crear cilindro

Para no repetir el proceso por cada una de las patas de la mesa, copiaremos la pata que ya hemos creado y la pegaremos 3 veces dentro del grupo de componentes donde estamos trabajando.

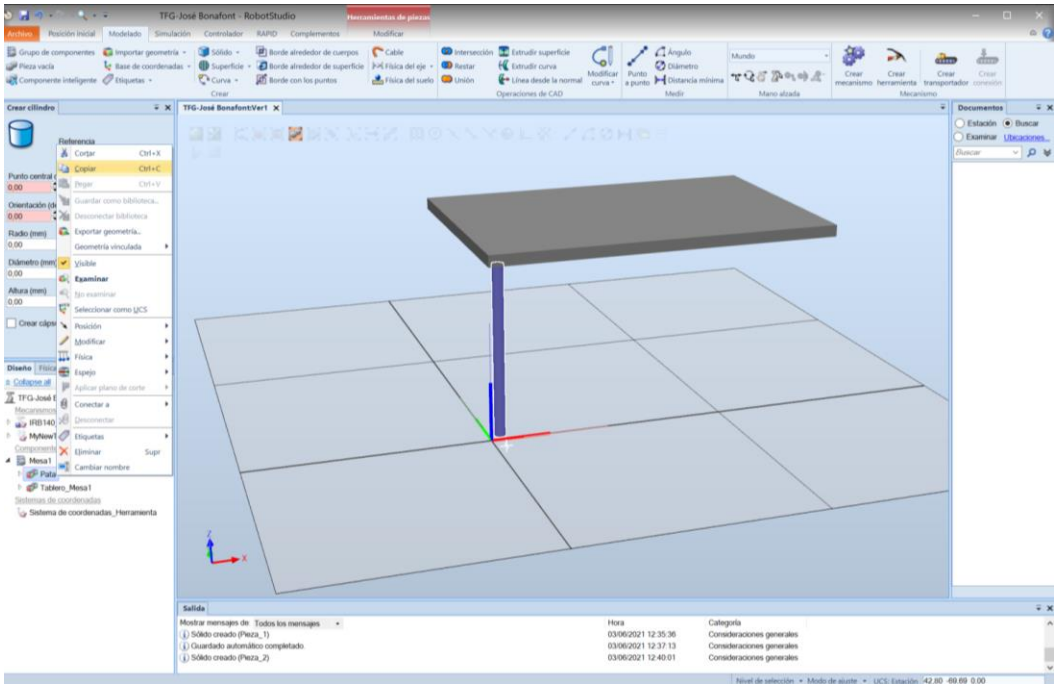


Figura 23 - Duplicar sólidos

Por último, ahora que tenemos las cuatro patas creadas, desplazaremos las patas duplicadas hasta sus esquinas correspondientes de la mesa ayudándonos de la herramienta *Mano alzada* para desplazarlas rápidamente.

Modelado → Mano alzada → Mover

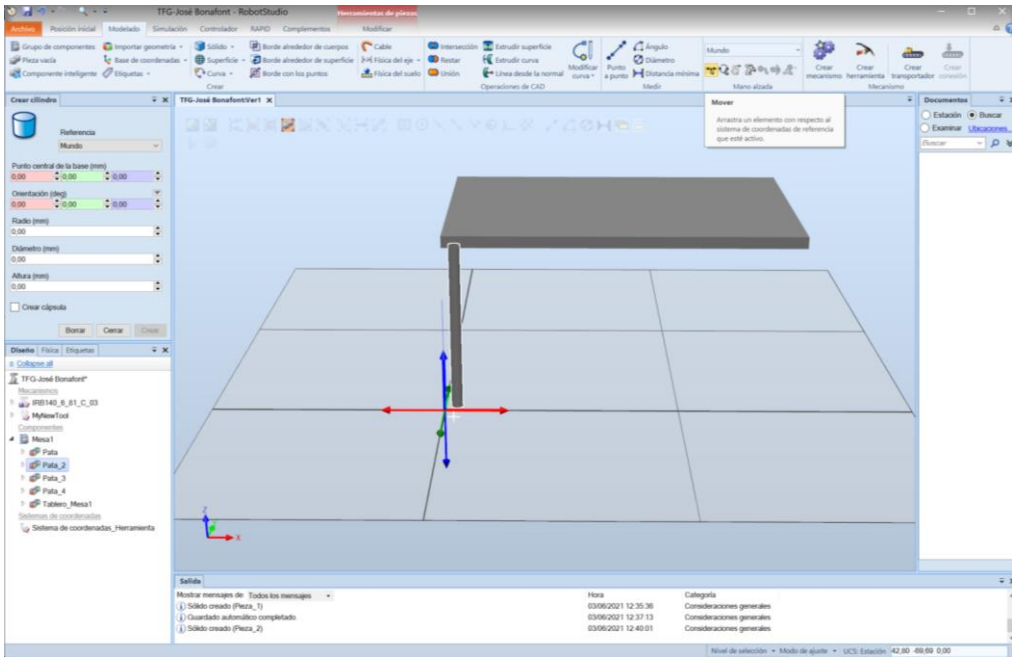


Figura 24 - Mover sólidos

Como podemos observar en la Figura 23, aparecen unas flechas que podremos usar para desplazar el sólido que seleccionemos y situarlo dónde queramos, eligiendo el eje sobre el que se moverá. De esta manera colocaremos las patas de la mesa en sus respectivos sitios.

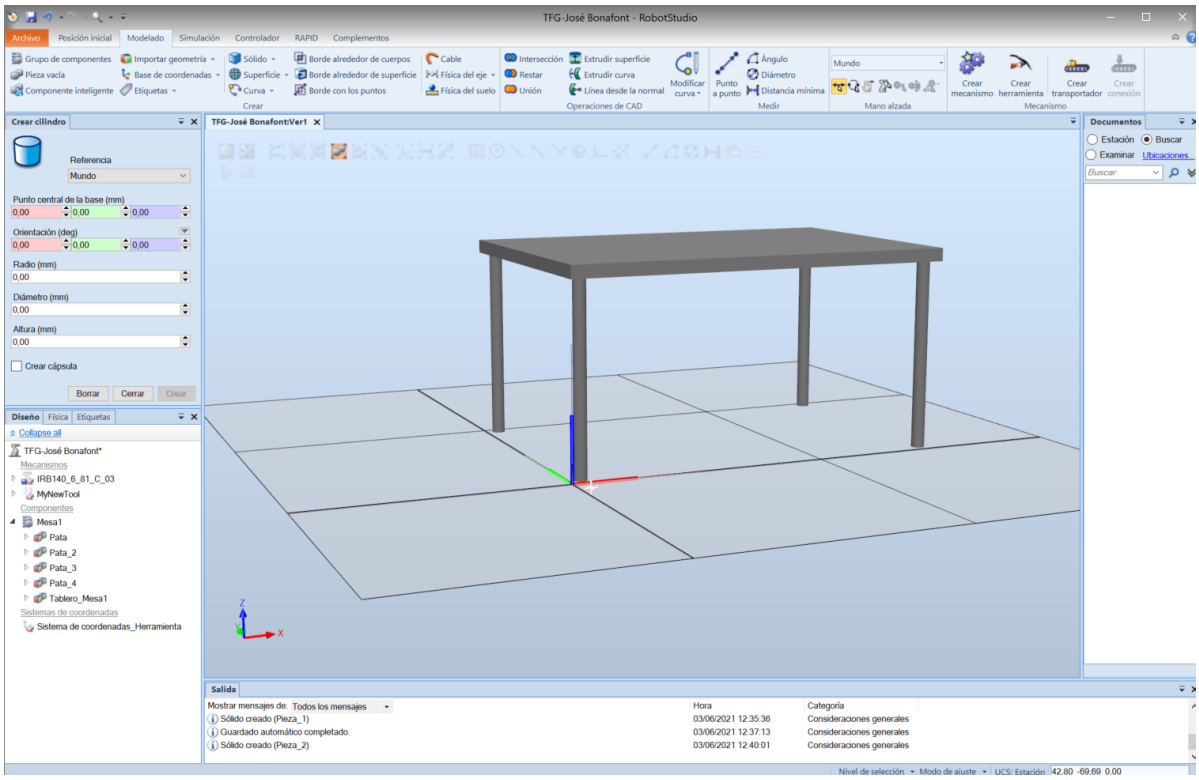


Figura 25 - Mesa principal

4.7 Texturas y colores

Una vez creada la pieza podemos darle un aspecto mas parejo a la realidad añadiéndole texturas y colores a los materiales que la componen. Realmente RobotStudio ofrece una gran cantidad de posibilidades a la hora de personalizar los sólidos que hemos creado, aunque en nuestro caso será una cosa secundaria y no nos complicaremos en exceso al ser simplemente una cuestión estética y que no afecta al proyecto.

Para personalizar las piezas deberemos seguir la siguiente ruta:

Herramientas de piezas → Modificar → Apariencia de gráficos → Editar materiales

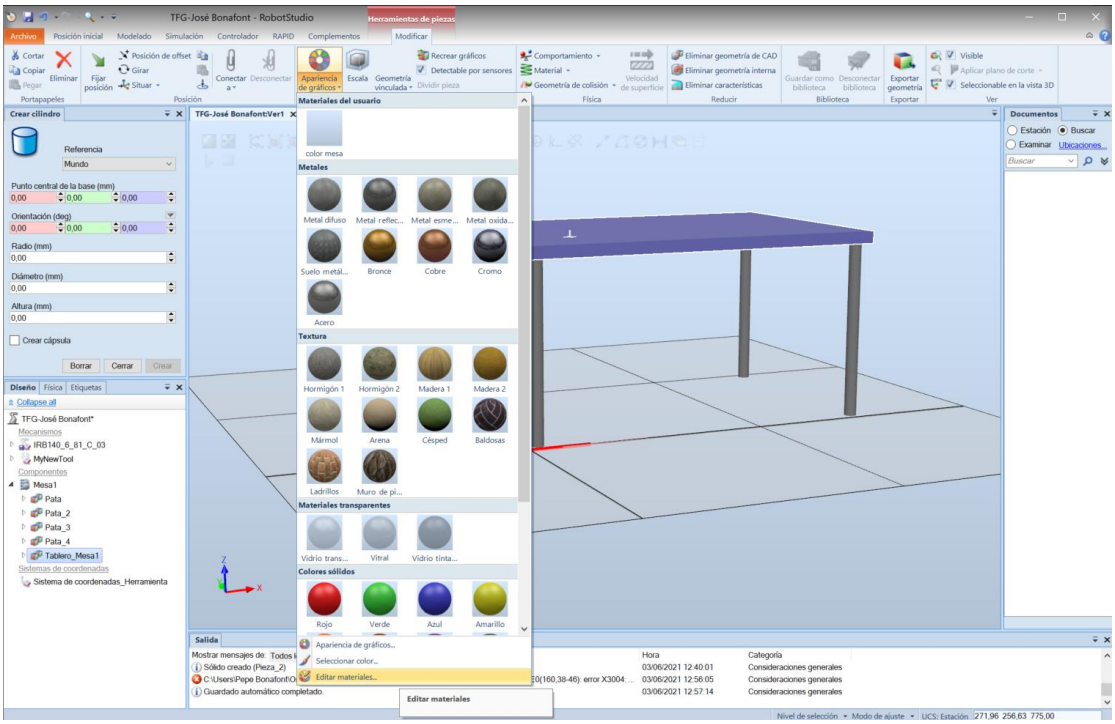


Figura 26 - Editar material

Una vez hecho esto se nos abrirá una ventana (Figura 25) donde nos ofrecerá todas las opciones para editar materiales que presenta RobotStudio, podemos editar cara a cara las texturas y colores del material utilizando las bibliotecas predeterminadas del programa o importando nuestras propias bibliotecas.

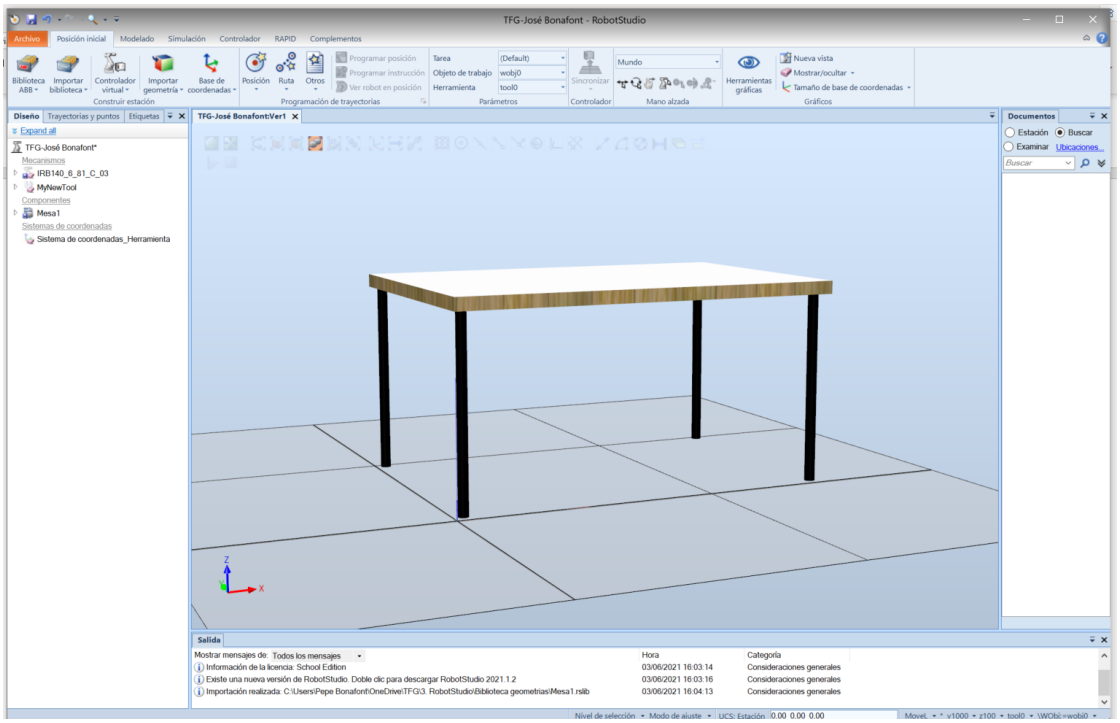


Figura 27 - Mesa principal con texturas

4.8 Guardar componentes como bibliotecas

Una vez tengamos todo el entorno creado, repitiendo el mismo proceso que hemos seguido para crear la mesa principal con todos los componentes que lo componen, obtendríamos el siguiente resultado:

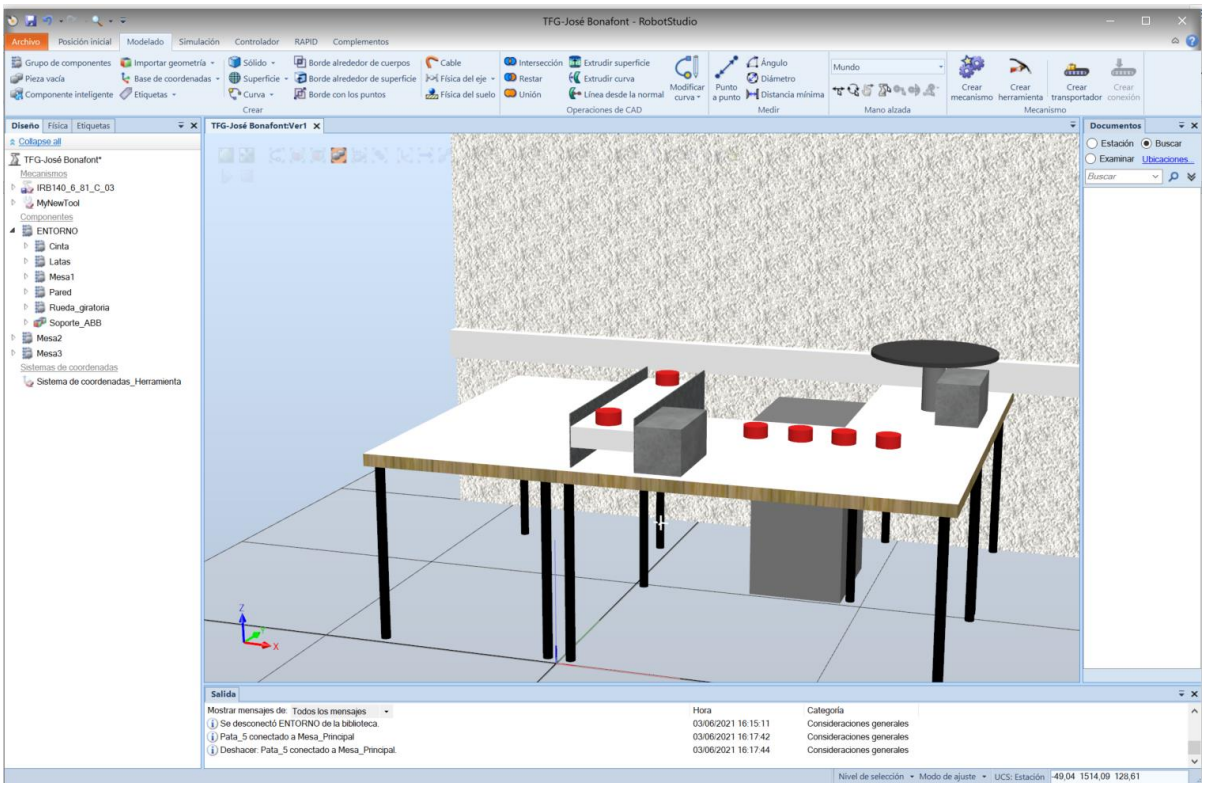


Figura 28 - Entorno del robot

Es interesante conocer que el RobotStudio permite guardar e importar componentes creados en el propio programa, ahorrándonos tener que repetir volver a modelar estas piezas para un futuro nuevo proyecto o una modificación de este.

Simplemente tendremos que hacer clic derecho en el componente o grupo de componentes que queremos conservar y seleccionar *Guardar como biblioteca*.

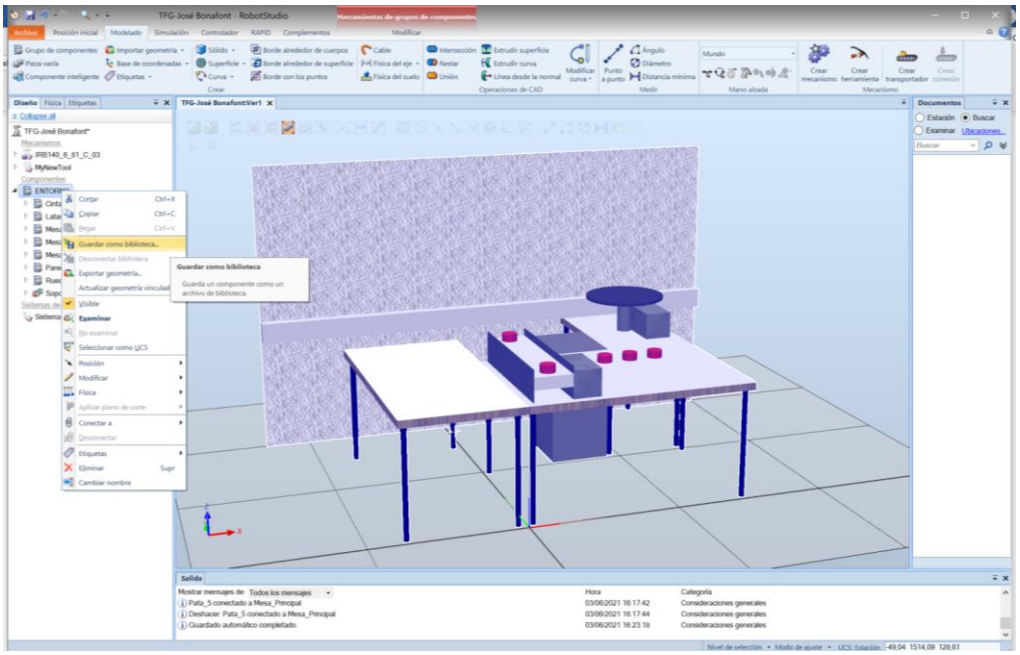


Figura 29 - Guardar como biblioteca

4.9 Implementación del robot en el entorno

Al volver a hacer visible el robot IRB-140, una vez recreado el entorno donde va a ir situado, podemos observar que no se encuentra en la posición dónde le correspondería estar, ya que se encuentra situado sobre el eje de coordenadas predeterminado del programa, igual que el entorno que hemos creado.

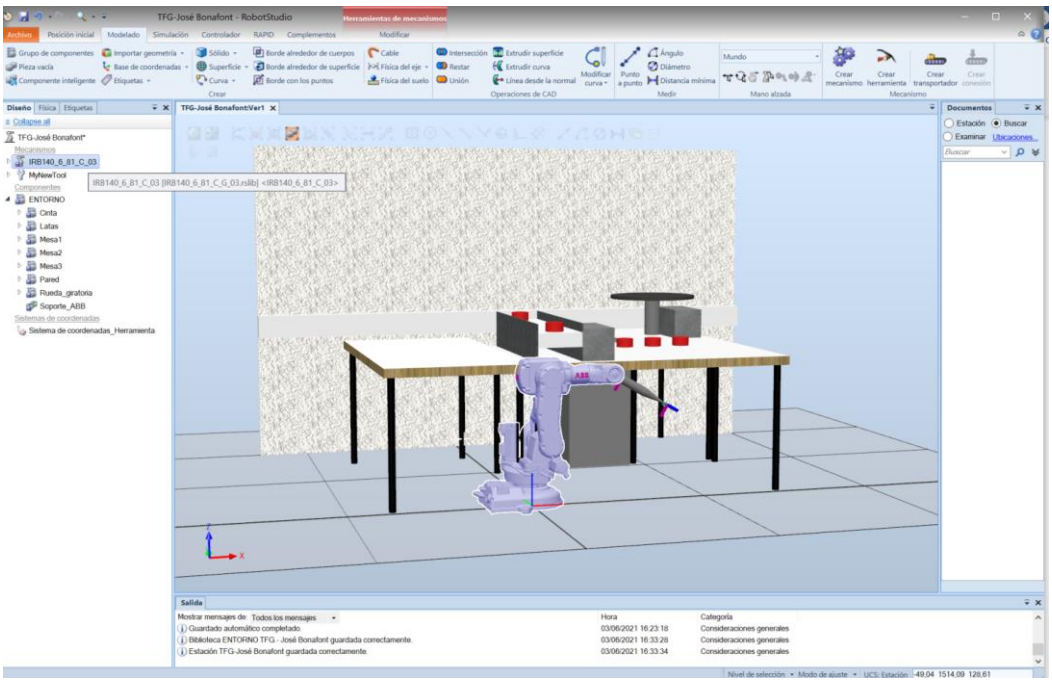


Figura 30 - Robot mal situado en el entorno

Para solucionar este problema tenemos dos opciones:

- Desplazar el Robot hasta la posición que le corresponde
- Desplazar el entorno hasta situarlo correctamente respecto al robot

En nuestro caso vamos a optar por la segunda opción por dos motivos principalmente:

- El robot se mantendrá en el eje de coordenadas, por lo que posteriormente nos será más fácil programar sus movimientos y trayectorias
- El plano de trabajo coincidirá con el plano de la mesa

Para desplazar el entorno a la posición y con la orientación que queremos tendremos que crear un nuevo sistema de coordenadas, de la misma manera que ya lo hemos hecho para crear la herramienta, en la posición que deseamos respecto a las coordenadas predeterminadas del programa.

Modelado → Base de coordenadas → Crear

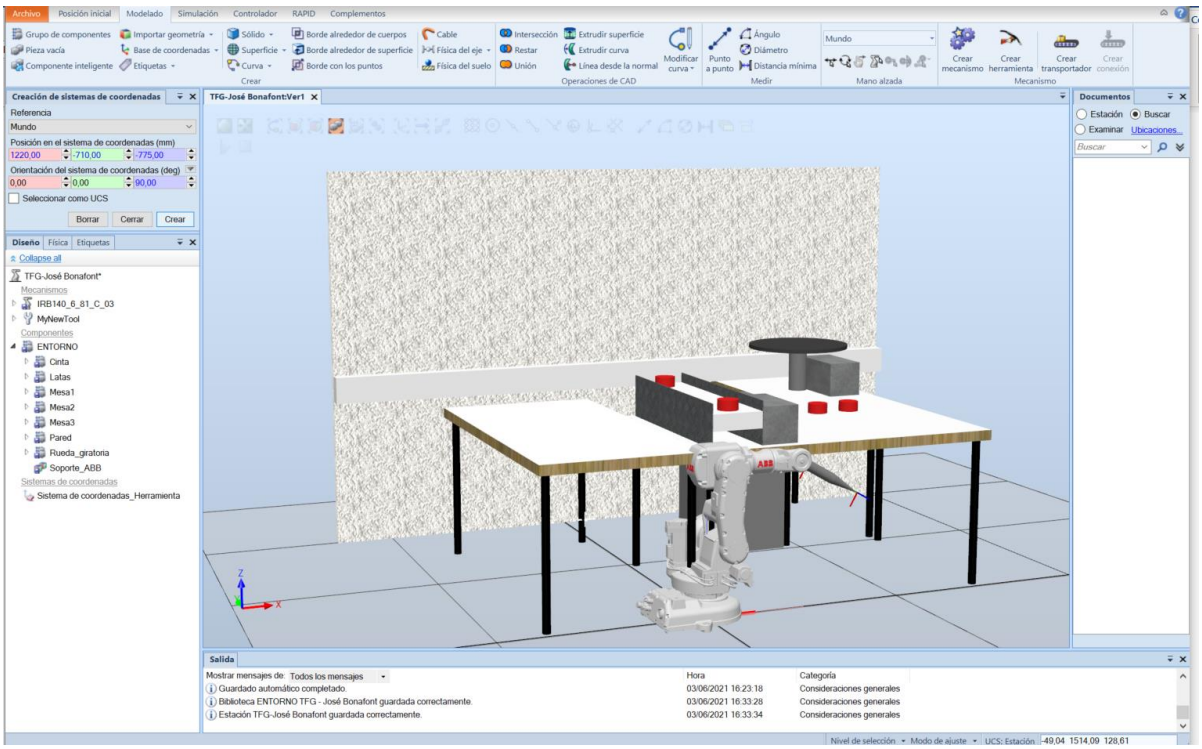


Figura 31 - Crear base de coordenadas del entorno

Por último, una vez creada la nueva base de coordenadas, que hemos nombrado *Sistema de coordenadas_Entorno,* solo tendríamos que mover el entorno hasta su posición.

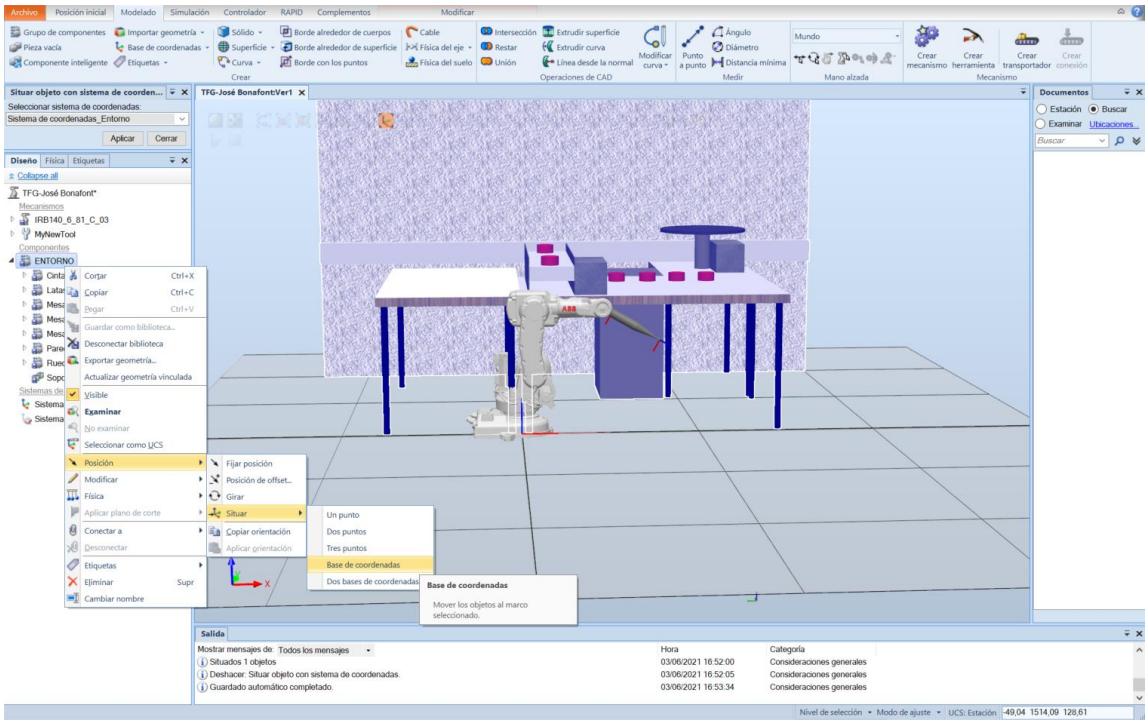


Figura 32 - Situar entorno en la base de coordenadas

Tras realizar todos estos pasos, como podemos observar en la figura 32, el robot ya se encuentra situado correctamente respecto al entorno y tenemos nuestra estación lista para poder iniciar la programación de sus tareas.

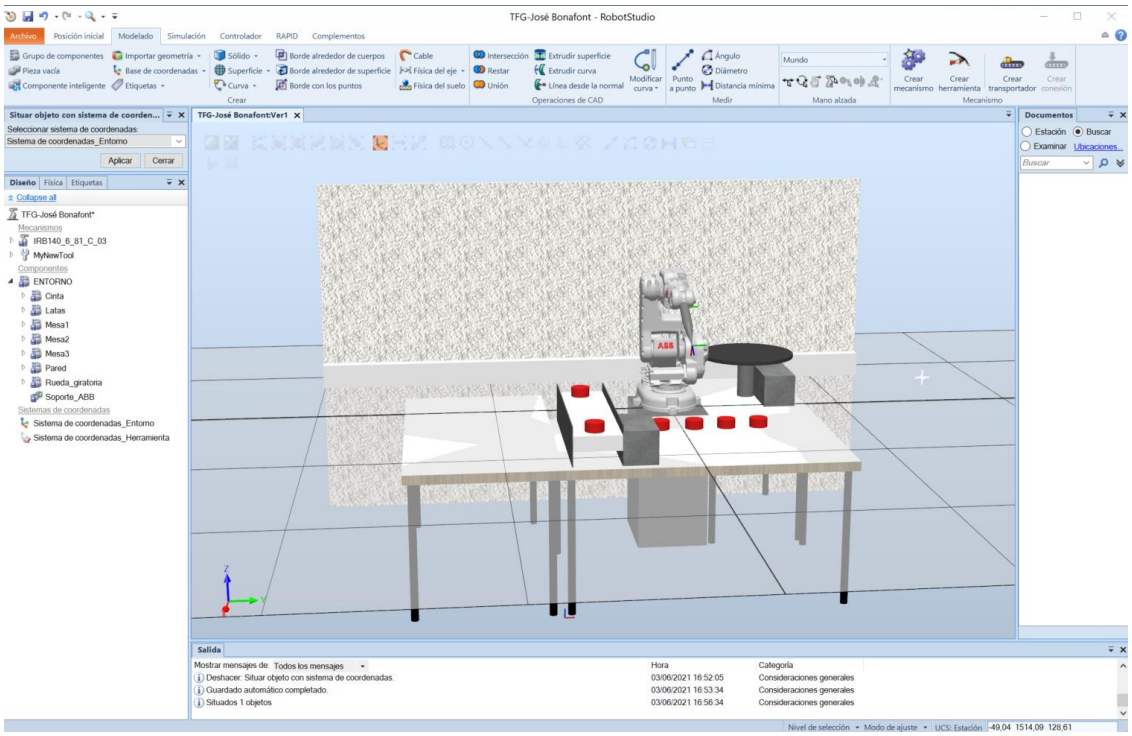


Figura 33 - Robot correctamente situado en su entorno

5. Desarrollo de pruebas de simulación

La siguiente tarea que vamos a hacer en este proyecto es programar el robot para que realice las acciones que deseamos siguiendo las trayectorias que le ordenemos.

En este proyecto, como hemos mencionado antes, realizaremos un Pick & Place: el robot recogerá las piezas que vengan desde la cinta y las situará en 4 puntos diferentes de la mesa.

Pese a ser una tarea muy básica, es algo realmente frecuente en la industria. Por ejemplo, para separar piezas en un proceso de selección, amontonar en palés, manipular piezas muy calientes o para realizar trabajos en entornos con atmósferas peligrosas para los operarios.

Para programar los movimientos del robot tenemos varias maneras de hacerlo, ayudándonos del programa utilizando los llamados "work objects" o escribiendo directamente el código de programación en lenguaje RAPID.

A continuación, haremos una pequeña introducción al lenguaje de programación RAPID y explicaremos como hacerlo de ambas formas para al finalizar poder decantaremos por una de ellas, observando las ventajas e inconvenientes que presentan.

5.1 Introducción al lenguaje de programación RAPID

RAPID es el lenguaje de programación de alto nivel creado por ABB. Esta programación se compone de un programa principal y de módulos secundarios.

En este proyecto, como RAPID se trata de un lenguaje de programación de alto nivel, solo explicaremos los conceptos más básicos de este lenguaje, dando por hecho muchas cosas necesarias para poder comprender este lenguaje de programación.

Para poder realizar la programación de la simulación y el robot, nos hemos ayudado del *"Technical reference manual. RAPID instructions, Functions and Data types"* que ofrece ABB desde su web de manera totalmente gratuita y que adjuntamos en la bibliografía de este proyecto.

5.1.1 Módulos del programa

Como hemos mencionado antes, el lenguaje de programación RAPID consta de un programa principal y de otros módulos secundarios. En nuestro caso, si vamos a la pestaña RAPID y desplegamos nuestro archivo del programa observamos que se compone de un módulo principal llamado *Module1* y de otro secundario llamado *CalibData.*

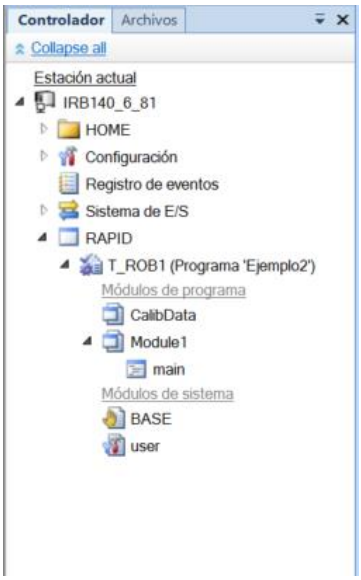


Figura 34 - Módulos del programa

El programa principal, *Module1*, se inicia con el comando "MODULE Module1" y se cierra con "ENDMODULE". Este programa incluirá las posiciones o puntos objetivos del programa y todas las instrucciones pertinentes para que el robot realice la tarea que deseamos.

Para crear los diferentes módulos que compone el programa principal, iniciaremos con el comando "PROC" seguido del nombre del programa y "()". Para cerrarlo utilizaremos el comando "ENDPROC".

El otro programa, *CalibData*, se inicia con el comando "MODULE CalibData" y se cierra con "ENDMODULE". Este programa, en cambio, se encargará de recoger la información y los ajustes de la herramienta que hemos creado anteriormente en este proyecto.

5.1.2 Posiciones objetivo del robot

Para crear posiciones objetivo del robot dentro del *Module1,* deberemos empezar escribiendo "CONST robtarget" seguido del nombre de esta posición, las coordenadas en la estación y la configuración de los ejes para ese punto. Por ejemplo:

```
CONST robtarget
home:=[[775,0,525],[0.5,0,0.87,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

5.1.3 Movimientos del robot

Los módulos que hemos explicado como crear antes pueden contener una serie de instrucciones, entre ellas, los movimientos del robot que harán posible que el robot realice las trayectorias que deseamos.

Para definir los movimientos del robot tendremos que indicar el tipo de movimiento, el destino del movimiento, la velocidad a la que lo realizará, el error que le permitiremos al robot y la herramienta que usará. Por ejemplo:

MoveJ home, v300, z50, MyNewTool;

Por tanto, para poder definir estos movimientos, será necesario conocer los diferentes tipos de movimientos que puede realizar el robot. En RAPID hay tres movimientos principales:

- MoveL: es un movimiento lineal, es decir, el robot se moverá de un punto a otro en línea recta.

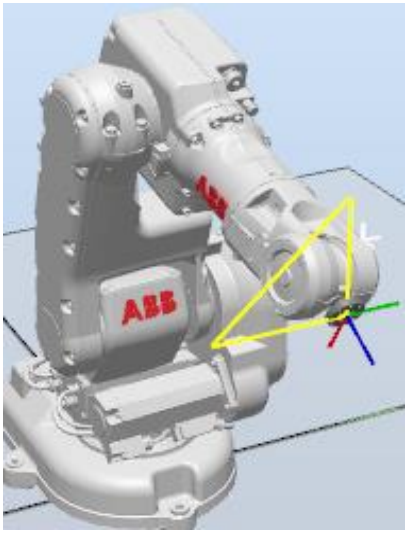


Figura 35 – MoveL

- MoveJ: es un movimiento libre. El robot se desplazará de la manera que el considere óptima de un punto a otro, ya que es probable que para realizar un movimiento totalmente recto deba mover más ejes que con un movimiento libre.

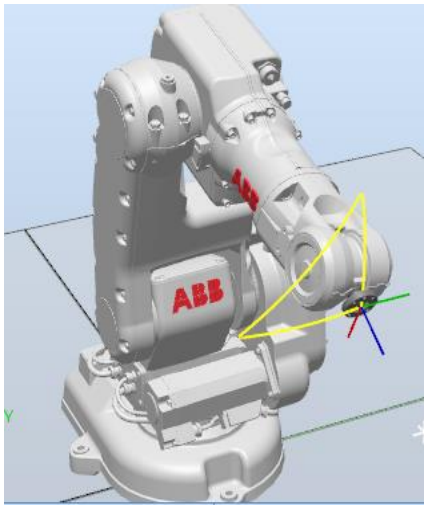


Figura 36 – MoveJ

- MoveC: sirve para crear trayectorias circulares. Para realizar este tipo de movimientos es necesario definirlo con dos puntos. En este caso no utilizaremos este tipo de movimiento.

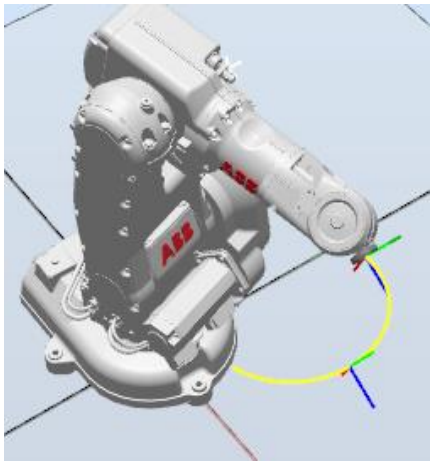


Figura 37 – MoveC

En cuanto a las velocidades pueden programarse de 5 a 7000 mm/s. La velocidad máxima puede variar según el modelo del robot. En nuestro caso no utilizaremos velocidades muy altas por seguridad.

El siguiente dato es el error del robot en el movimiento. Cuanto mayor es el error menor será la precisión tendrá la herramienta y, por tanto, peor será el trabajo. Este error puede ir desde "fine" que sería 0mm de error hasta z200 que son 200mm de error. Si elegimos z0, el error será aproximadamente de 0.3mm.

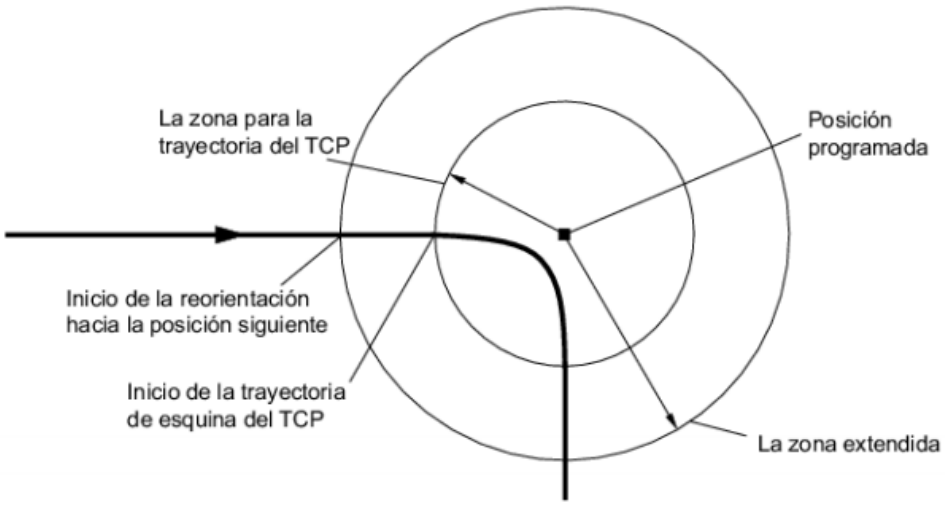


Figura 38 - Error de precisión entre dos puntos

5.2 Creación de trayectorias con objetos de trabajo

Para empezar con la creación de trayectorias primero hay que asegurarse que el controlador del robot se encuentra activo, sin él el programa no nos dejará realizar las tareas necesarias para la programación de este.

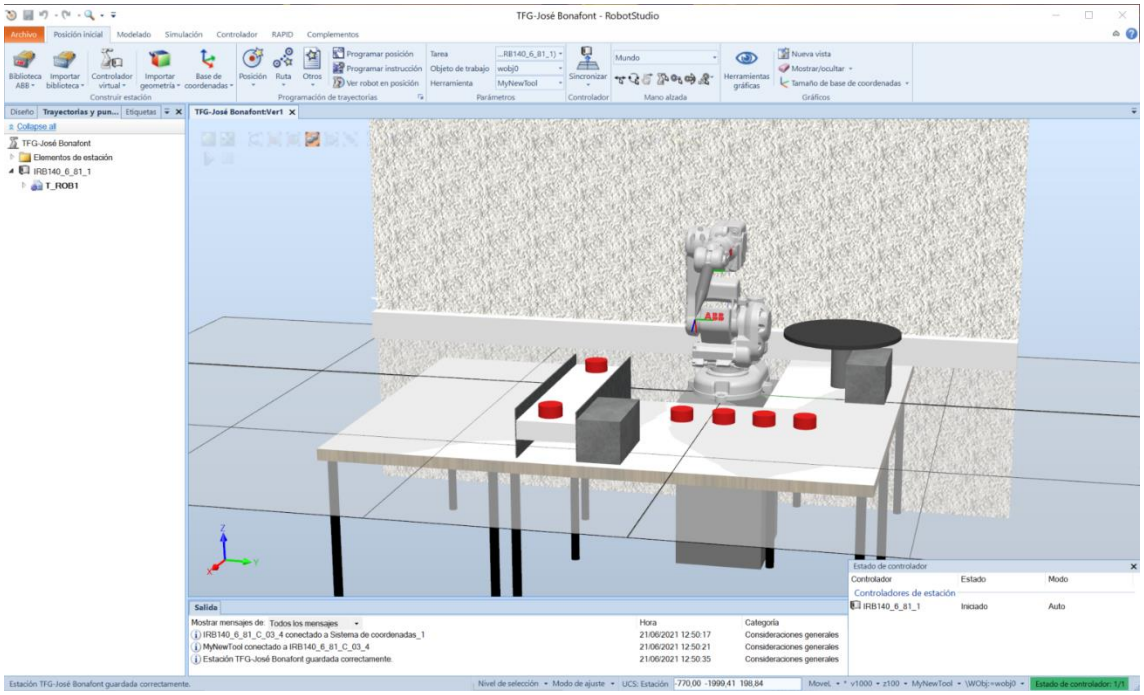


Figura 39 - Estado del controlador

5.2.1 Creación de los puntos de trabajo

El siguiente paso será crear los puntos objetivo, es decir, los puntos dónde queremos que el puntero de la herramienta del robot se sitúe durante el trabajo, definiendo la trayectoria a seguir.

Deberemos ir a la pestaña *Posición inicial*, presionar el desplegable *Posición* y seleccionar *Crear punto.* Se nos abrirá un cuadro para poder crear los puntos objetivo que queremos.

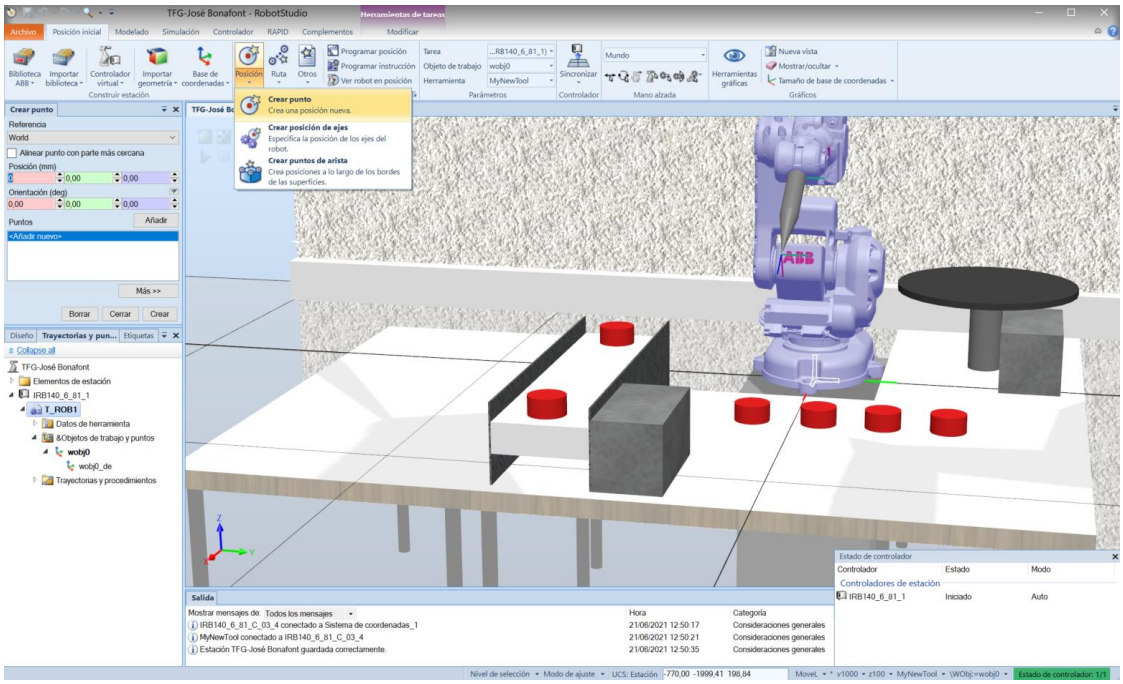


Figura 40 - Crear punto

Como queremos que los puntos objetivo se sitúen en los centros de las caras superiores de los objetos, tendremos que activar las opciones *Selección de superficie* y *Ajustar a centro* que se encuentran en la ventana donde podemos visualizar al robot y su entorno.

Figura 41 - Selección de superficie

Figura 42 - Ajustar a centro

Una vez activadas estas opciones ya podemos crear los puntos objetivo haciendo clic en la cara superior de las diferentes posiciones donde puede encontrarse la pieza.

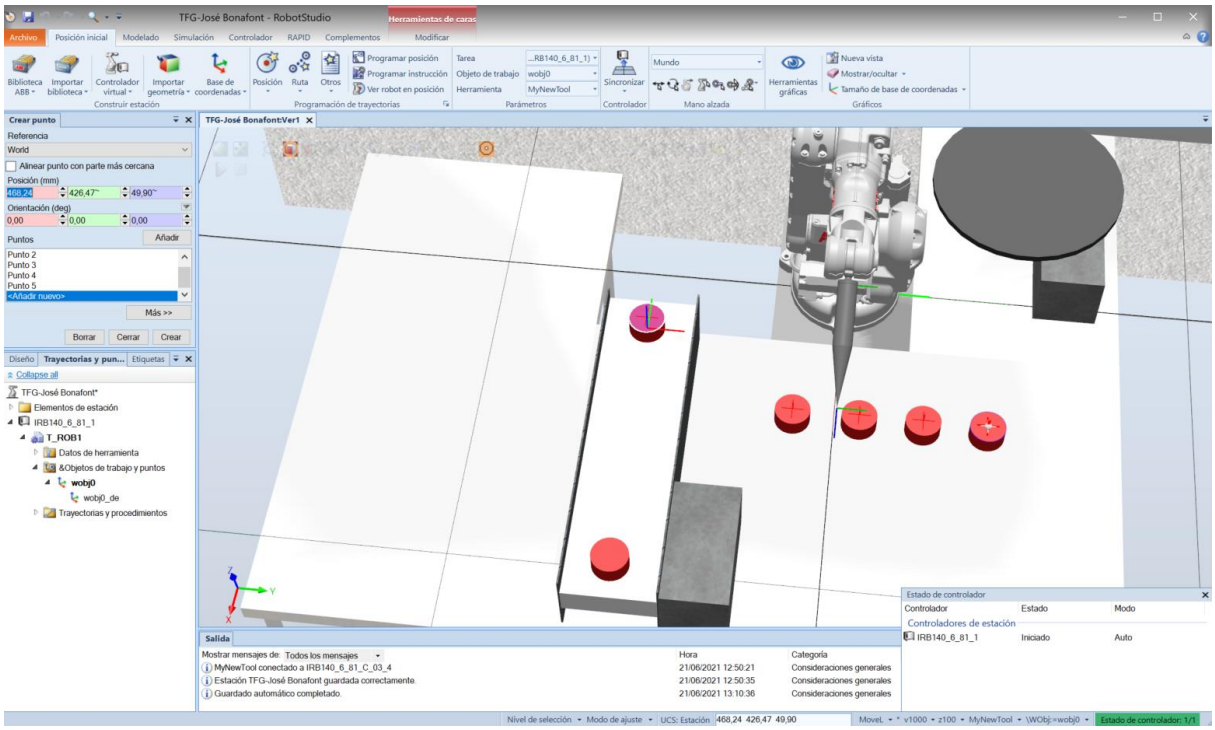


Figura 43 - Selección de puntos

Como podemos observar se han añadido cinco puntos en la pestaña *Crear punto,* cuando le demos al botón Crear, estos cinco puntos se añadirán en la carpeta &*Objetos de trabajo y puntos* dentro de la pestaña *Trayectoria y puntos.* Cambiaremos los nombres de los puntos para que más tarde nos resulte más sencillo identificarlos.

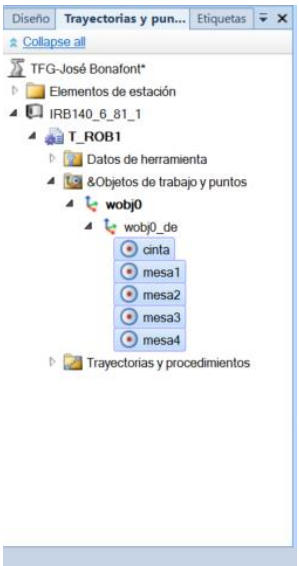


Figura 44 - Puntos de trabajo

Además de estos puntos que hemos definido ya, necesitaremos otros puntos iguales a los que tenemos, pero desplazados en el eje z para poder bajar de incidir de manera totalmente perpendicular sobre las piezas, asegurándonos un perfecto agarre de la ventosa que tiene el robot como herramienta.

Por lo tanto, tendremos que realizar el mismo proceso que antes, pero copiando las piezas y situándolas encima de ellas a la altura que deseamos.

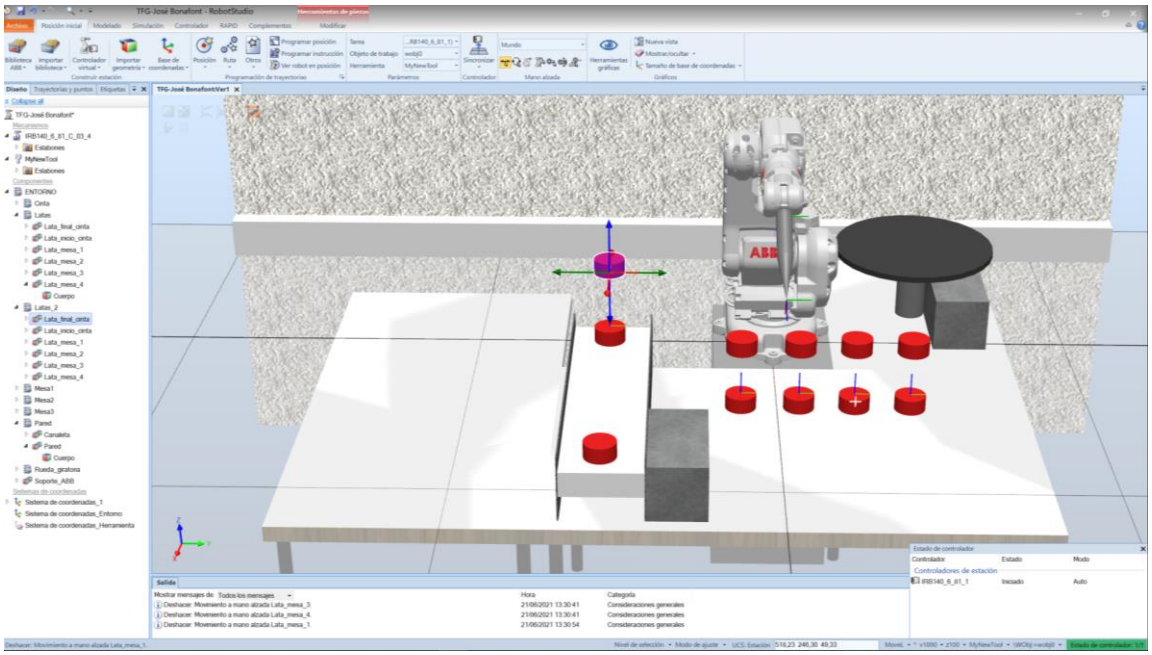


Figura 45 - Piezas desplazadas en el eje Z

Crearemos los puntos de estas añadiéndoles el _up al final del nombre para diferenciarlas de las otras y las haremos invisibles.

Figura 46 - Puntos desplazados en el eje Z

Para acabar con esta parte, tendremos que definir un último punto correspondiente a la posición de reposo del robot o posición *home.* Simplemente tendremos que colocar el robot en la posición que queramos, en nuestro caso lo dejaremos como está, y pulsaremos *Programar posición*, dentro de la pestaña *Posición inicial*.

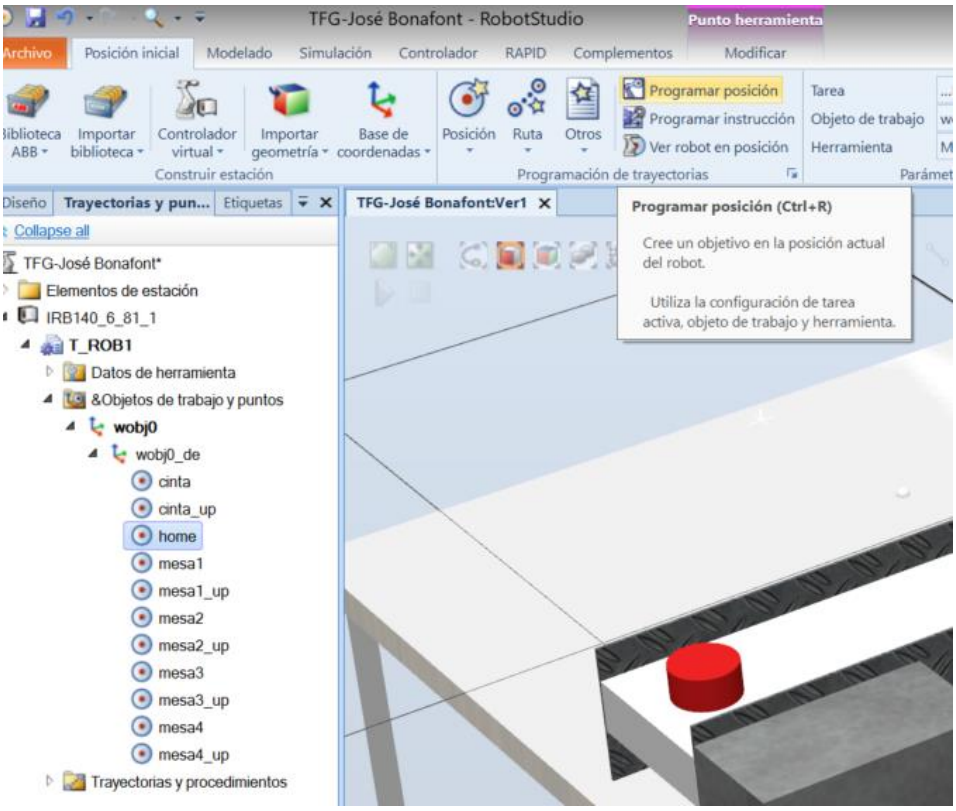


Figura 47 - Creación del punto de reposo

5.2.2 Orientación de la herramienta

Una vez tenemos los puntos creados, y antes de crear las trayectorias, comprobaremos la orientación con la que el RobotStudio ha colocado la herramienta en cada una de las piezas.

Para ver la orientación de la herramienta deberemos hacer clic derecho en el punto, creado anteriormente, seleccionar la opción ver herramienta en posición y escoger la herramienta del robot, en nuestro caso llamada *MyNewTool.*

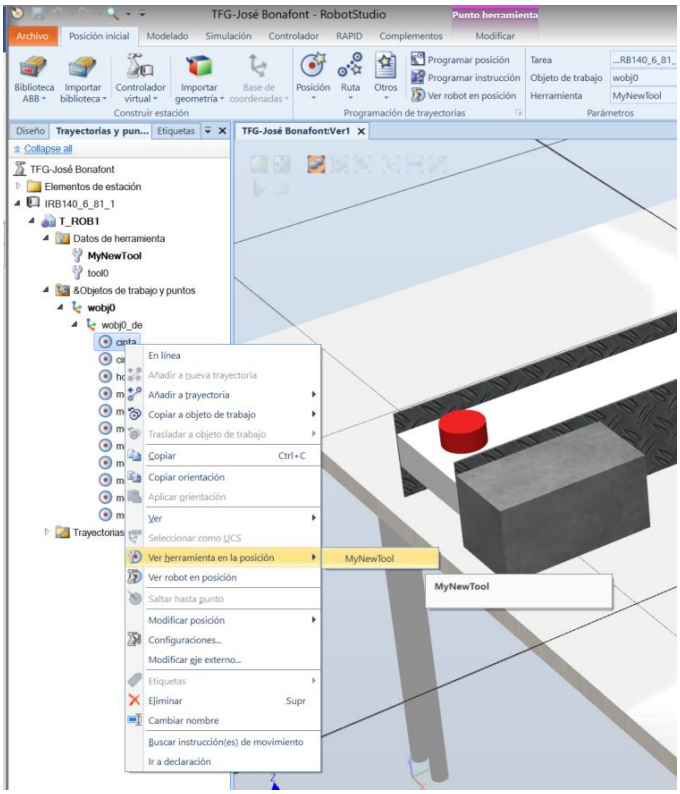


Figura 48 - Ver herramienta en posición

Como podemos observar en la figura 42, la orientación de la herramienta está mal en todos los puntos, incluso se puede observar que la herramienta atraviesa los sólidos creados, como la cinta y la mesa, por lo que seria totalmente imposible para el robot adoptar esa posición.

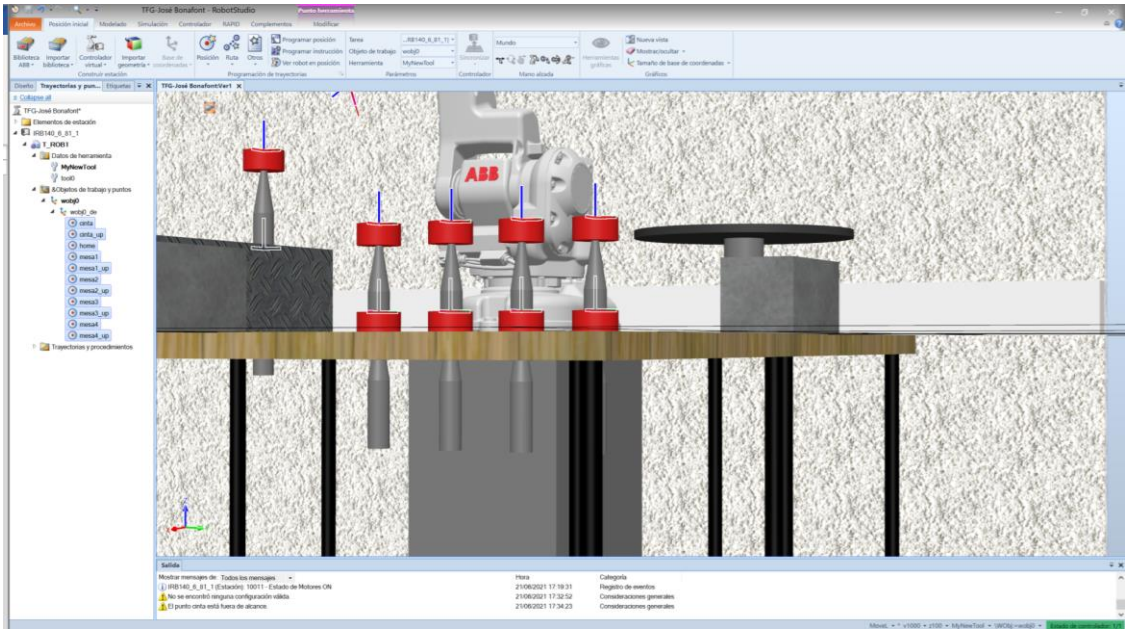


Figura 49 - Herramienta con orientación errónea

Para reorientar la herramienta correctamente deberemos rotar 180 grados en el eje Y. Haremos clic derecho en cualquiera de los puntos donde tenemos que reorientar la herramienta, nos colocaremos en *Modificar posición* y haremos clic en *Girar*.

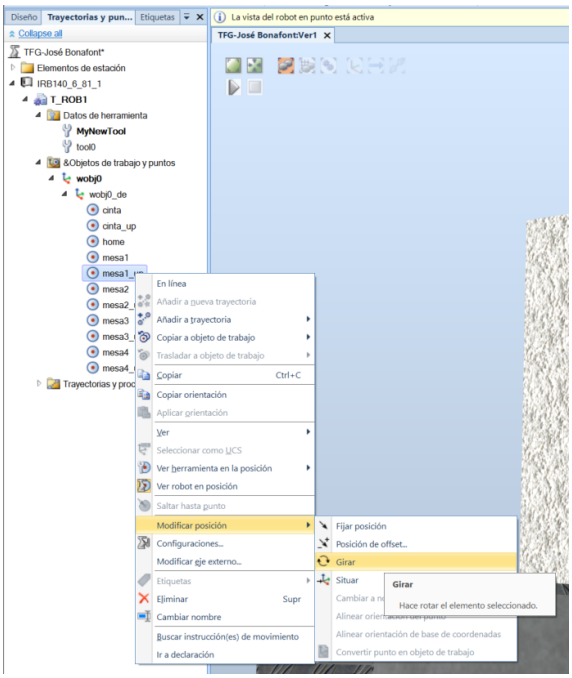


Figura 50 - Girar punto

De esta manera se nos abrirá una ventana donde podremos reorientar dicho punto, como hemos mencionado antes, deberemos girar 180 grados en el eje Y.

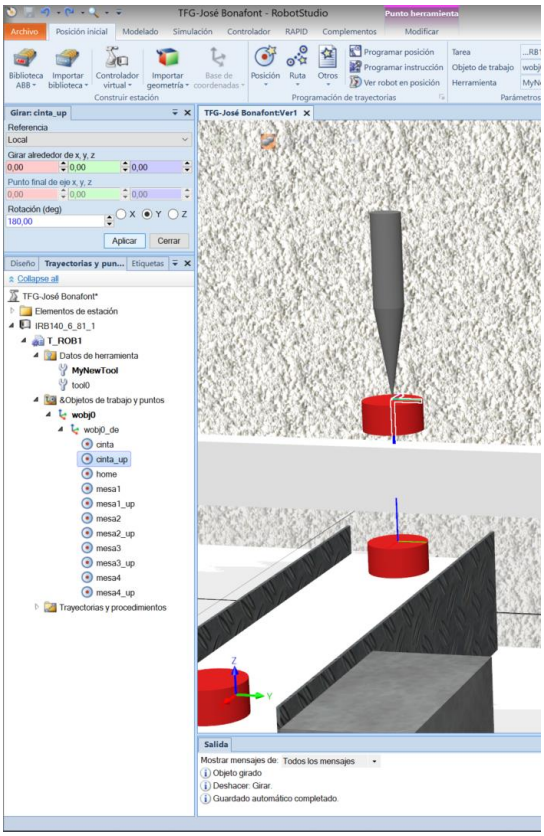


Figura 51 - Giro en el eje Y

Ahora que ya tenemos la orientación correcta en ese punto, como se observa en la figura 44, sólo tendremos que copiar la orientación de la herramienta en este punto y pegarla en los otros puntos. Lo haremos pulsando clic derecho en el punto en el que hemos reorientado la herramienta y seleccionando *Copiar orientación.*

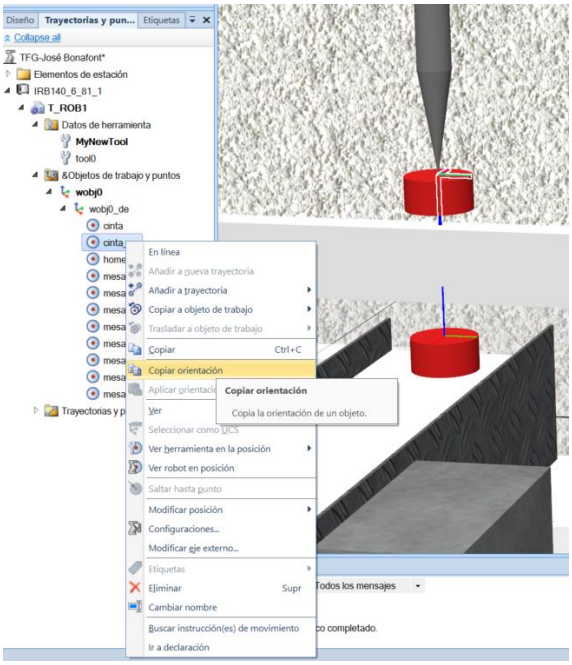


Figura 52 - Copiar orientación

Seguidamente, seleccionaremos todos los puntos que queremos reorientar (todos excepto el punto *home)* y aplicaremos la orientación.

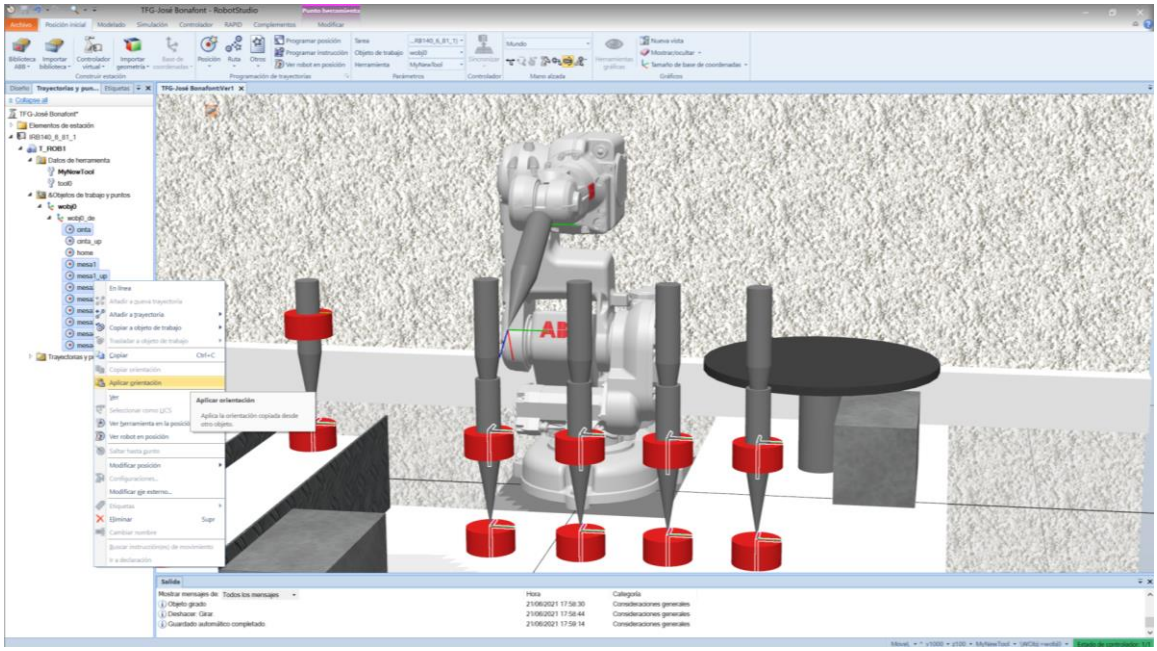


Figura 53 - Aplicar orientación

Como podemos observar en la figura 46 la orientación de la herramienta ya es correcta en todos los puntos.

Por último, y como algo complementario, podemos activar la opción *Ver robot en posición* nos permitirá observar la posición que adoptará el robot cuando se sitúe en cada uno de los puntos y con la orientación de la herramienta que hemos configurado anteriormente. También podemos hacer invisibles las piezas flotantes ya que nos las vamos a volver a utilizar.

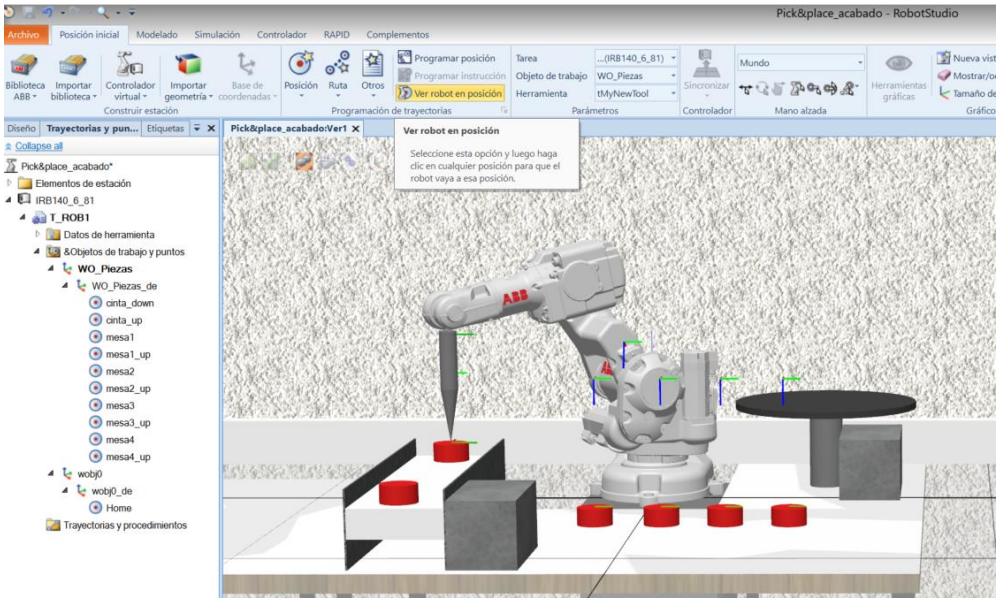


Figura 54 - Ver robot en posición

5.2.3 Creación de las trayectorias

Una vez tenemos los puntos creados con la orientación correcta de la herramienta podemos proceder a la creación de las trayectorias que queremos que realice el robot.

Nuestro proyecto tendrá cuatro trayectorias diferentes, una para cada una de las distintas posiciones en las que se pueden situar las piezas en la mesa.

Para poder crear estas trayectorias trabajaremos desde la ventana *Trayectorias y puntos*, la misma con la que hemos creado los puntos y reorientado la herramienta, haciendo clic derecho en la carpeta *Trayectorias y procedimientos* y seleccionando la opción *Crear trayectoria.*

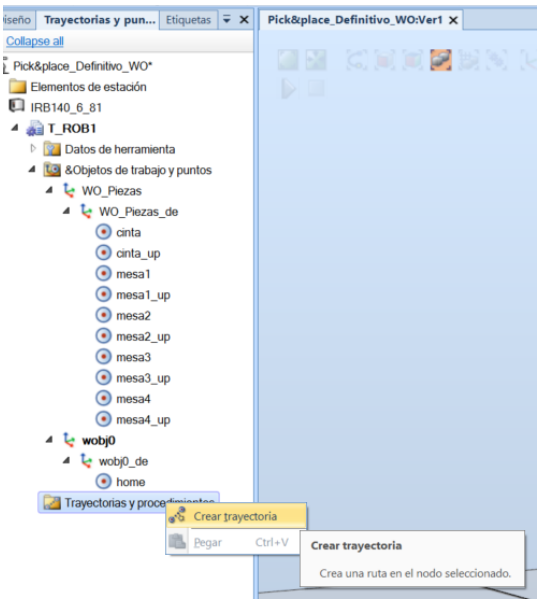


Figura 55 - Crear trayectoria

De esta manera se creará la primera trayectoria (Path_10). Para definir como queremos que sea, deberemos arrastrar los puntos hasta la trayectoria creada con el orden exacto en el que queremos que el robot se mueva. Es decir, para la primera trayectoria arrastraremos:

home → cinta_up → cinta → cinta_up → mesa1_up → mesa1 → mesa1_up → home

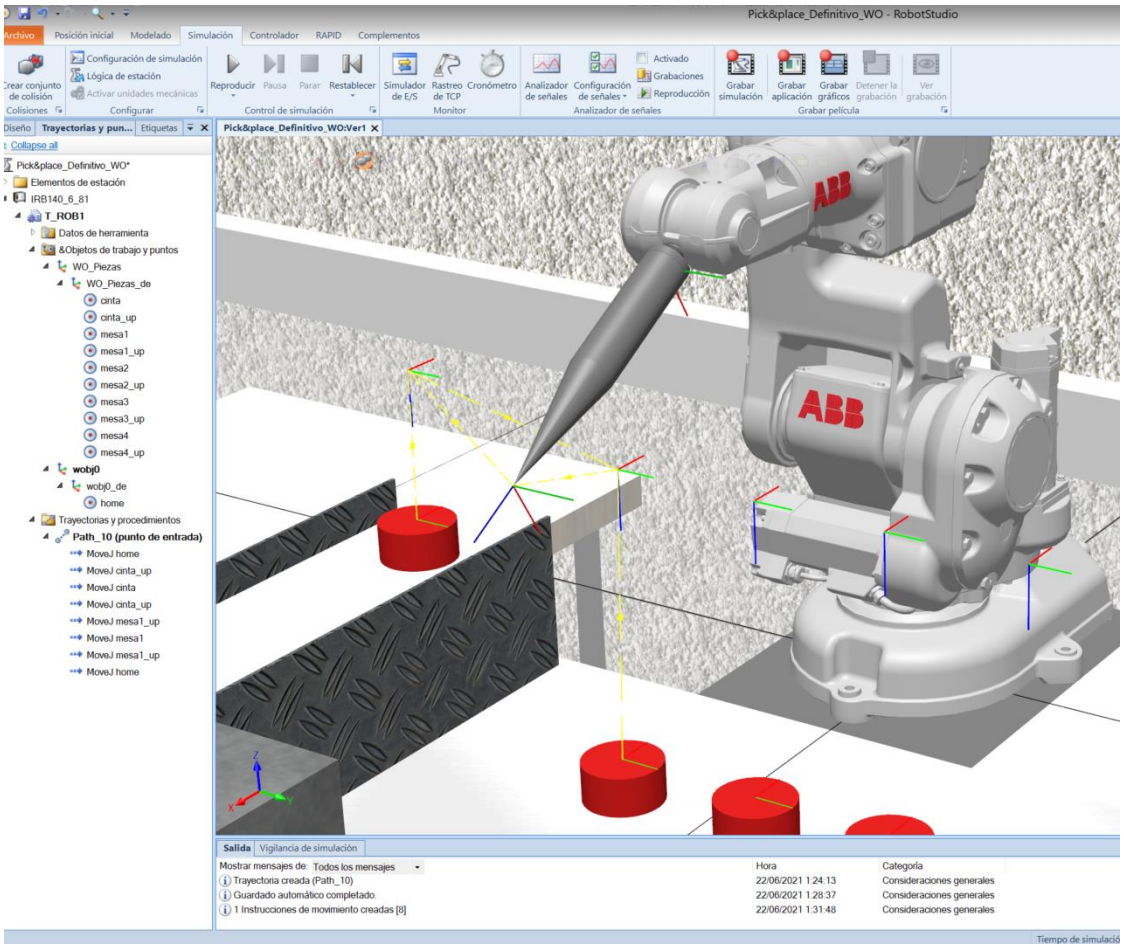


Figura 56 - Primera trayectoria

Como podemos observar en la figura 49, en la ventana de visualización de la estación aparecen unas líneas amarillas que definen la ruta que va a tomar dicho robot en la trayectoria que hemos creado. Además, podemos hacer una previsualización de la ruta para comprobar que no nos hemos equivocado pulsando uno a uno los movimientos que se han creado en la trayectoria que hemos creado.

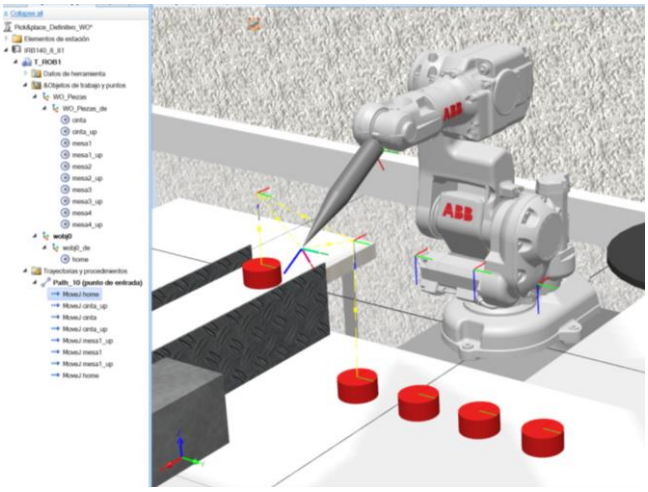


Figura 57 - Movimiento 1

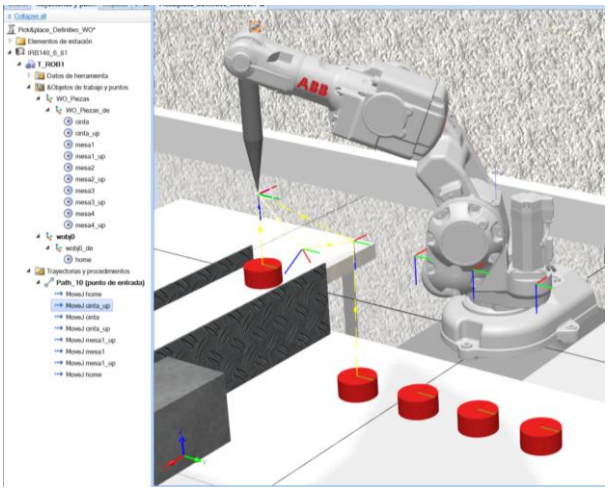


Figura 58 - Movimiento 2

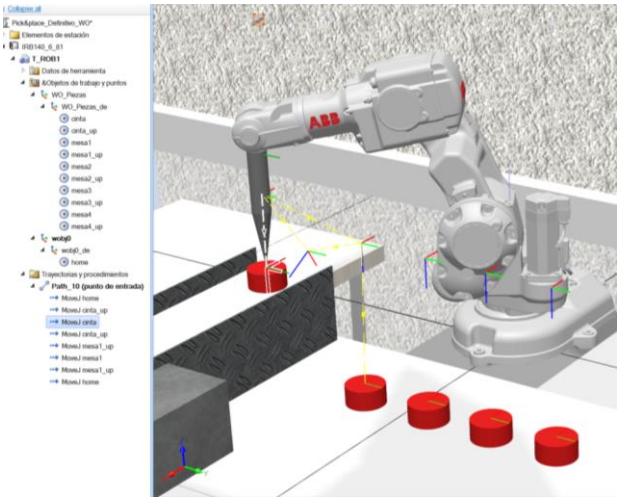


Figura 59 - Movimiento 3

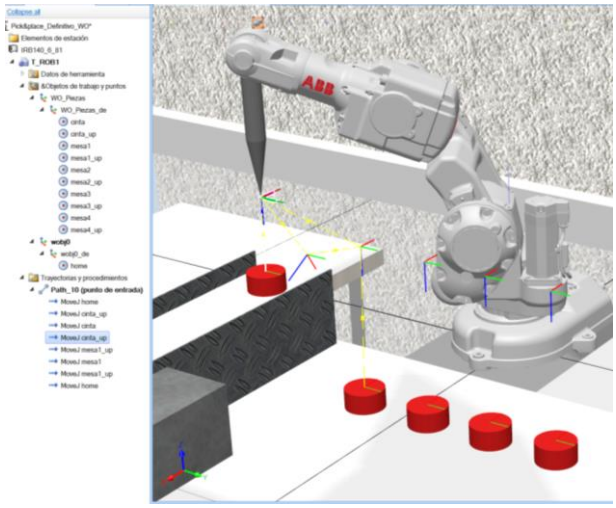


Figura 60 - Movimiento 4

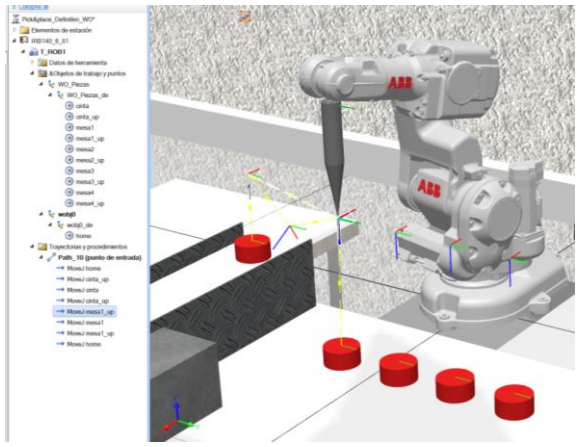


Figura 61 - Movimiento 5

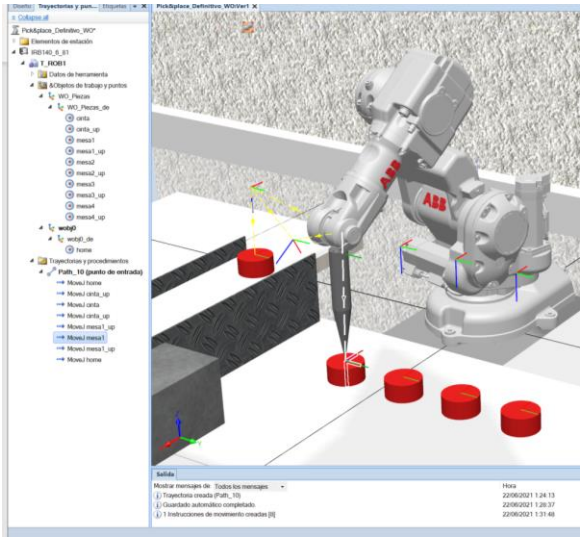


Figura 62 - Movimiento 6

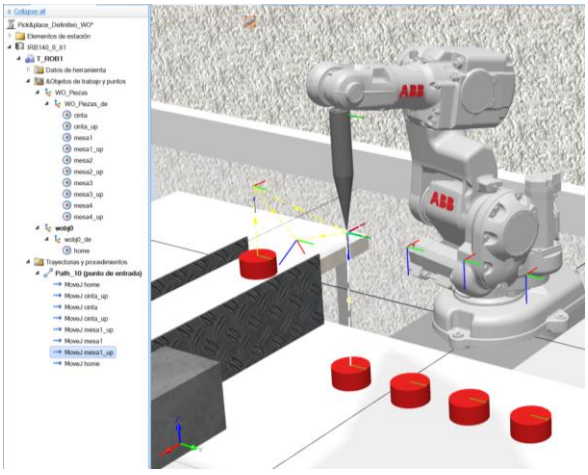


Figura 63 - Movimiento 7

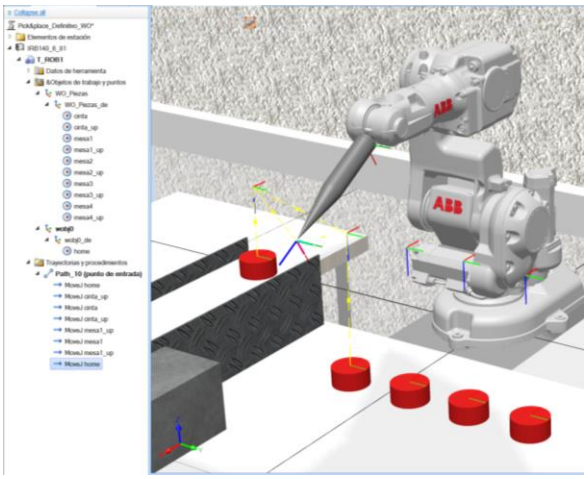


Figura 64 - Movimiento 8

Repitiendo este proceso podríamos crear las otras tres trayectorias que nos faltan, es decir, las trayectorias Path_20, Path_30 y Path_40.

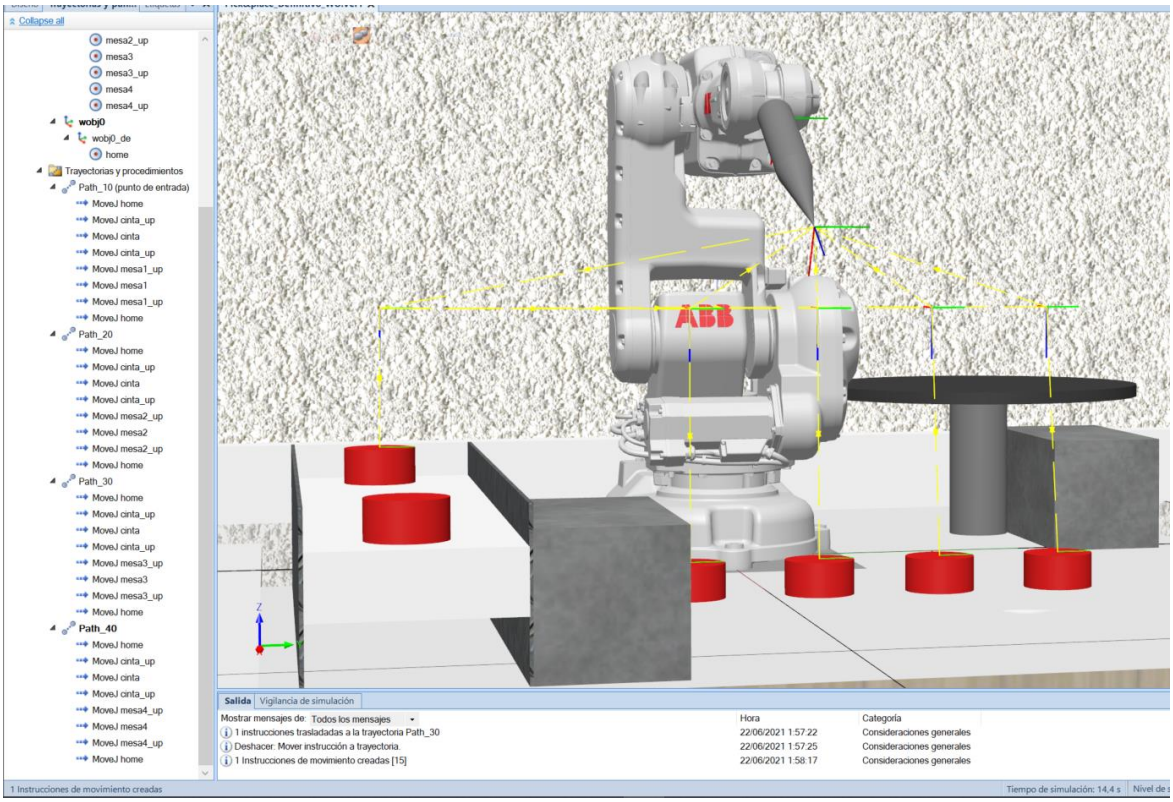


Figura 65 - Trayectorias creadas

5.2.4 Activación del Rastreo de TCP

Llegados a este punto ya podemos simular las diferentes trayectorias que hemos creado. Para poder observar más fácilmente la trayectoria que sigue el puntero de la herramienta durante la simulación deberemos, antes de seguir, activar el *Rastreo de TCP* que se encuentra en la pestaña *Simulación.*

Hecho esto se nos abrirá una ventana donde podremos seleccionar el color de la línea que definirá la trayectoria del puntero de la herramienta cambiando el color primario, además de muchas más opciones que ofrece el programa.

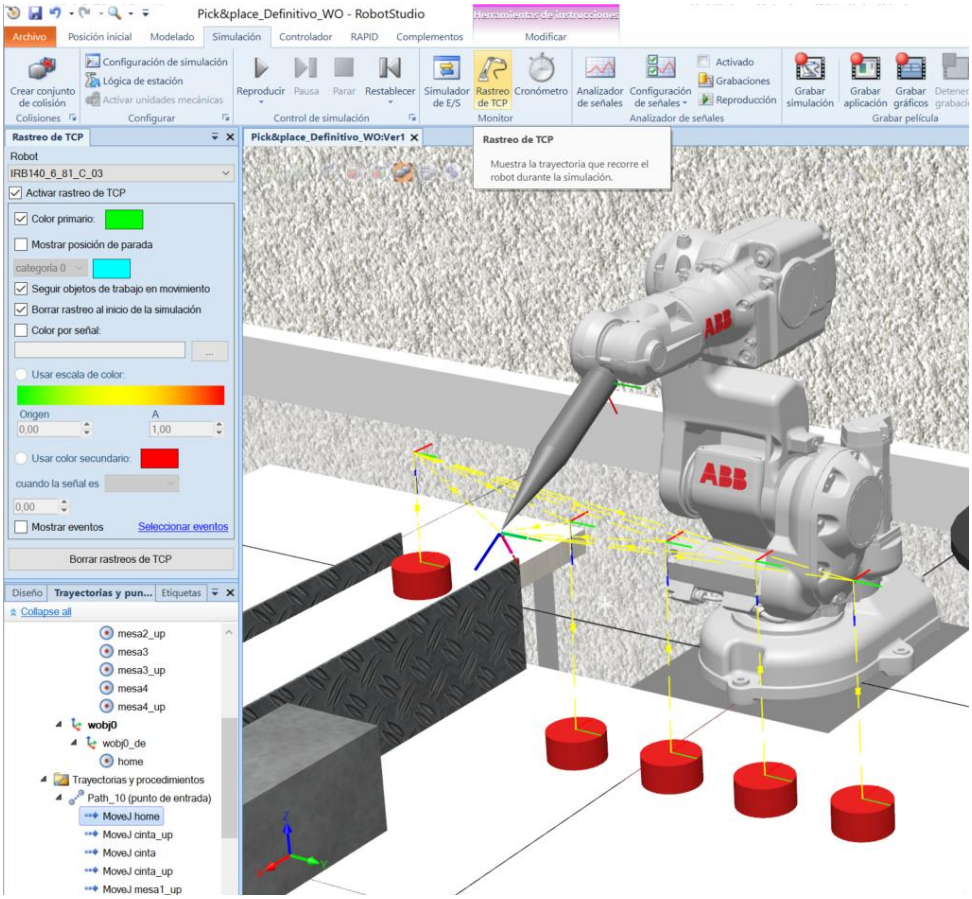


Figura 66 - Activar rastreo TCP

5.2.5 Sincronizar con RAPID

Una vez llegados a este punto ya está todo listo para realizar la simulación de los movimientos que definen las trayectorias que hemos creado.

Para ello, primero tendremos que sincronizar todo lo hecho hasta ahora con el controlador del robot. Esto lo haremos yendo a la pestaña *RAPID* y pulsando el botón *Sincronizar*. Este botón abrirá una ventana en la que dejaremos marcadas todas las casillas para sincronizar todo lo hecho hasta ahora y le daremos al botón *Aceptar.*

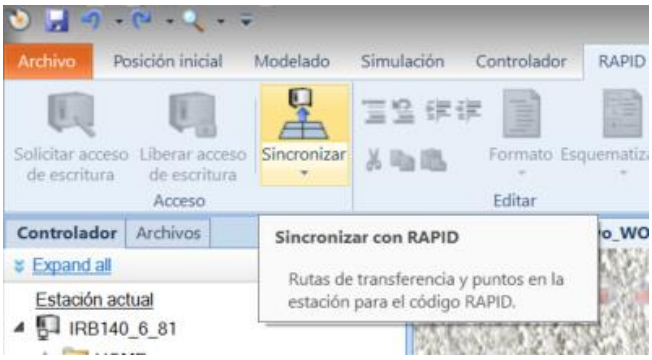


Figura 67 – Sincronizar

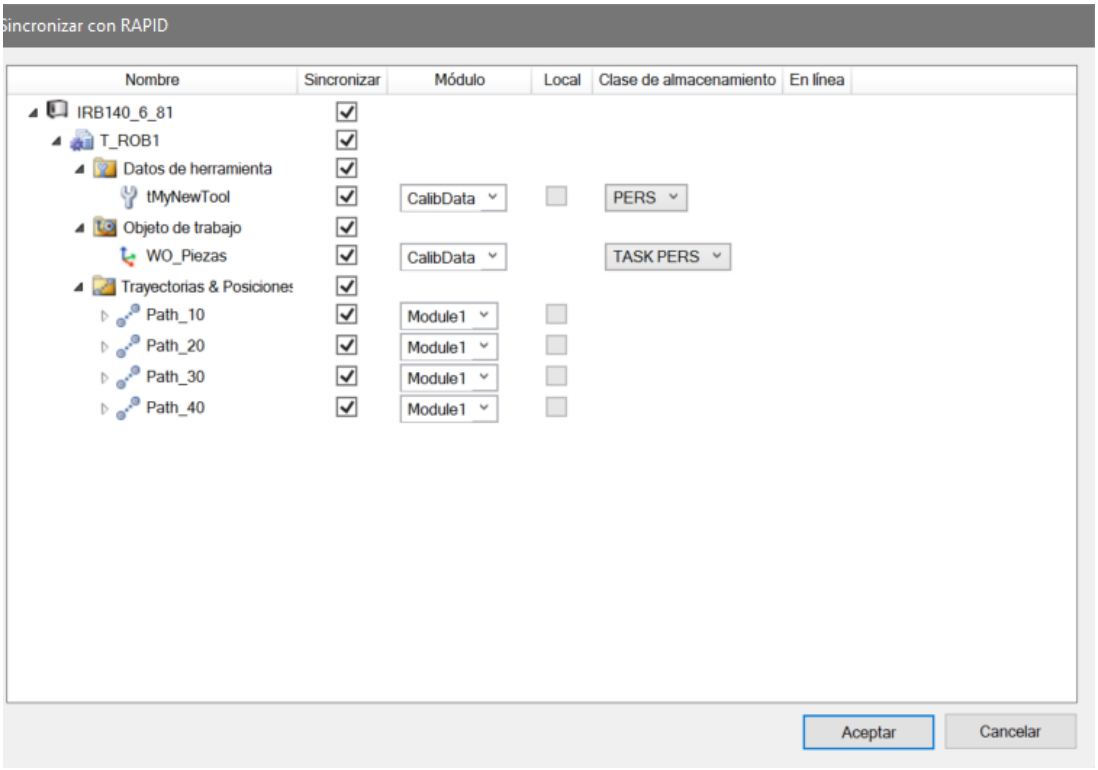


Figura 68 - Sincronizar con RAPID

Al sincronizar con RAPID, el programa generará automáticamente el código RAPID correspondiente a las trayectorias y puntos que hemos programado anteriormente. Este código podremos verlo y modificarlo desde la pestaña RAPID, abriendo el archivo Module1.

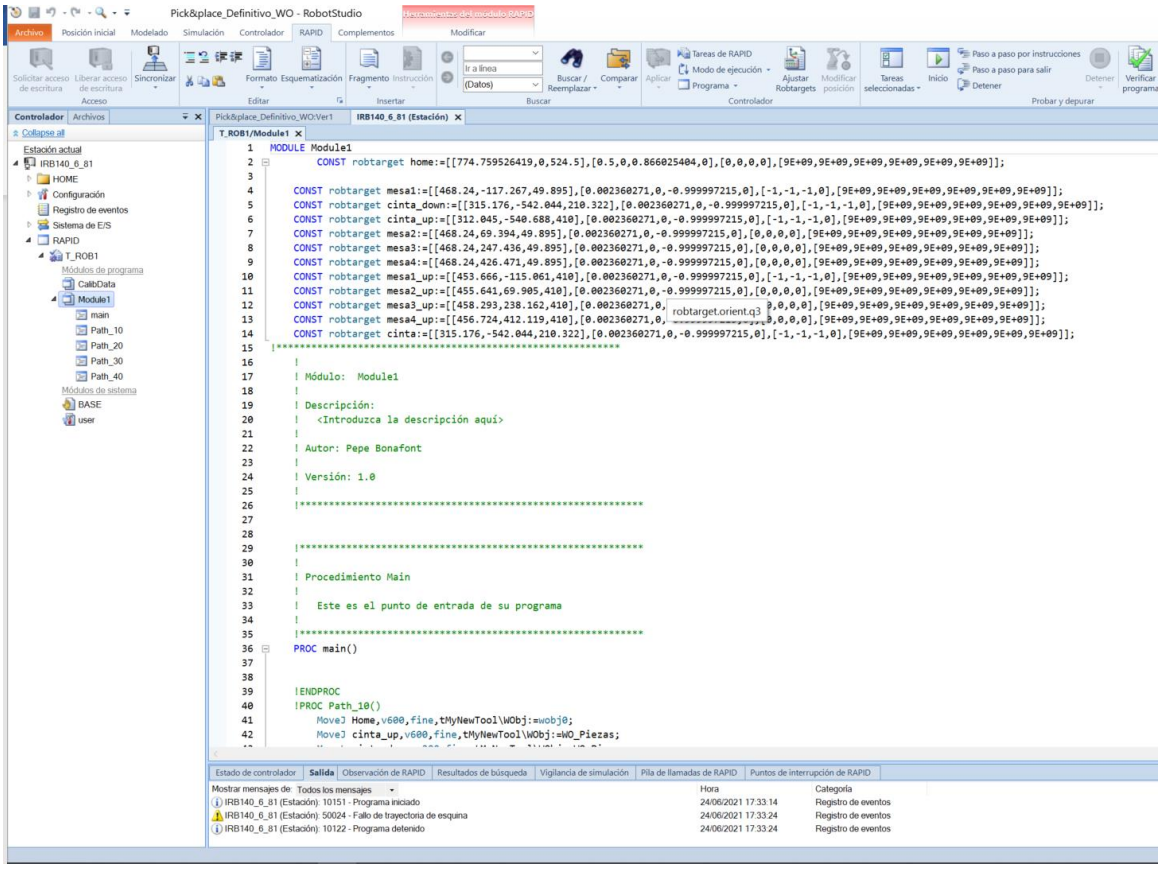


Figura 69 - Ver código RAPID

El código completo que se generaría automáticamente y que observamos en la figura 68 es el siguiente:

```
MODULE Module1
    CONST robtarget
home:=[[774.759526419,0,524.5],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+0
9,9E+09,9E+09,9E+09]];
    CONST robtarget mesa1:=[[468.24,-117.267,49.895],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget cinta_down:=[[315.176,-542.044,210.322],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget cinta_up:=[[312.045,-540.688,410],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa2:=[[468.24,69.394,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa3:=[[468.24,247.436,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa4:=[[468.24,426.471,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa1_up:=[[453.666,-115.061,410],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa2_up:=[[455.641,69.905,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget mesa3_up:=[[458.293,238.162,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa4_up:=[[456.724,412.119,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget cinta:=[[315.176,-542.044,210.322],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !***********************************************************
    !
    ! Módulo:  Module1
    !
    ! Descripción:
    !   <Código generado con Work Objects>
    !
    ! Autor: Pepe Bonafont
    !
    ! Versión: 1.0
    !
    !***********************************************************

    !***********************************************************
    !
    ! Procedimiento Main
    !
    !   Este es el punto de entrada de su programa
    !
    !***********************************************************

  PROC main()
        Path_10;
        Path_20;
        Path_30;
        Path_40;
  ENDPROC

    PROC Path_10()
      MoveJ Home,v600,fine,tMyNewTool\WObj:=wobj0;
      MoveJ cinta_up,v600,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveL cinta_down,v200,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveJ cinta_up,v600,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveJ mesa1_up,v600,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveL mesa1,v200,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveJ Home,v600,fine,tMyNewTool\WObj:=wobj0;
    ENDPROC

    PROC Path_20()
      MoveJ Home,v600,fine,tMyNewTool\WObj:=wobj0;
      MoveJ cinta_up,v600,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveL cinta_down,v200,fine,tMyNewTool\WObj:=WO_Piezas;
      MoveJ cinta_up,v600,fine,tMyNewTool\WObj:=WO_Piezas;
```

```
    MoveJ mesa2_up,v600,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveL mesa2,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ Home,v600,fine,tMyNewTool\WObj:=wobj0;
  ENDPROC

  PROC Path_30()
    MoveJ Home,v800,fine,tMyNewTool\WObj:=wobj0;
    MoveJ cinta_up,v800,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveL cinta_down,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ cinta_up,v800,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa3_up,v800,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveL mesa3,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ Home,v800,fine,tMyNewTool\WObj:=wobj0;
  ENDPROC

  PROC Path_40()
    MoveJ Home,v200,fine,tMyNewTool\WObj:=wobj0;
    MoveJ cinta_up,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ cinta_down,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ cinta_up,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa4_up,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa4,v200,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ Home,v200,fine,tMyNewTool\WObj:=wobj0;
  ENDPROC

 ENDPROC
ENDMODULE
```

5.2.6 Simulación de trayectorias

Ahora que ya está sincronizada la estación con el RAPID dispondremos de dos alternativas para realizar la simulación de las trayectorias creadas anteriormente.

La primera de estas opciones será volver a la pestaña *Simulación*, hacer clic derecho en la trayectoria que queremos simular y seleccionar el botón *Moverse a lo largo de la trayectoria*.

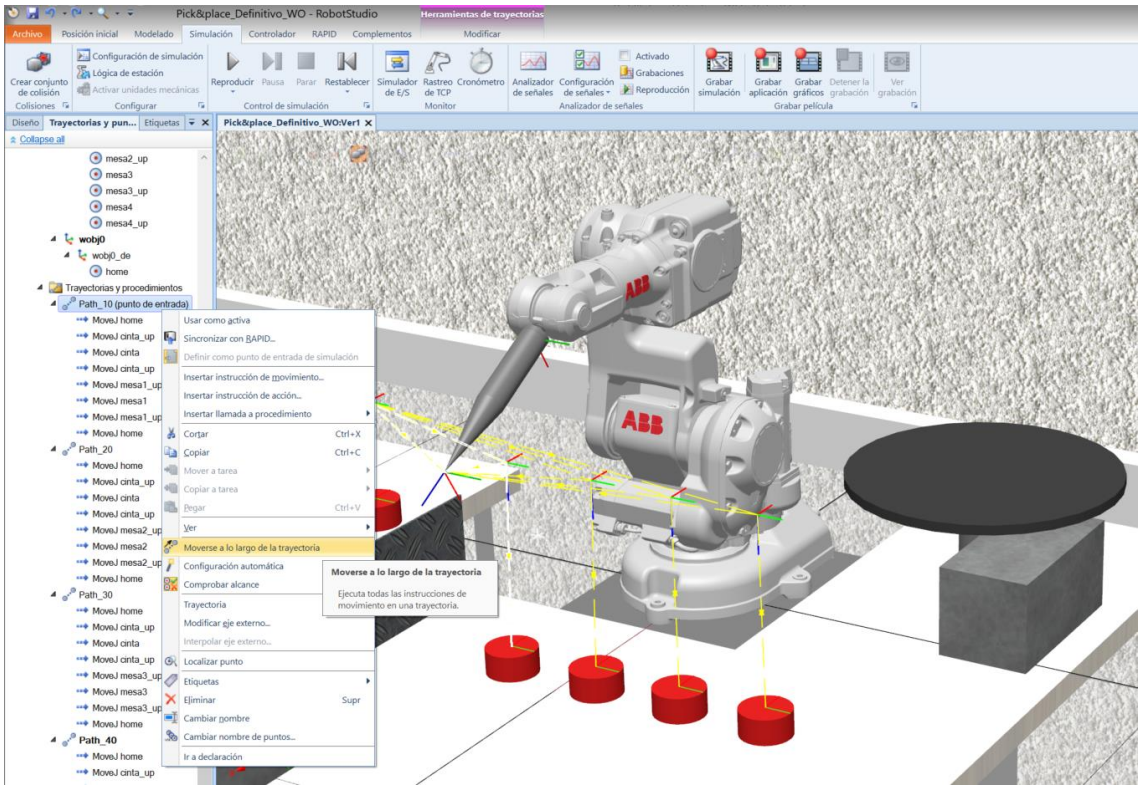


Figura 70 - Moverse a lo largo de la trayectoria

Haciéndolo de esta manera simularemos únicamente la trayectoria que seleccionemos, por ejemplo, simularemos la trayectoria *Path_10*.

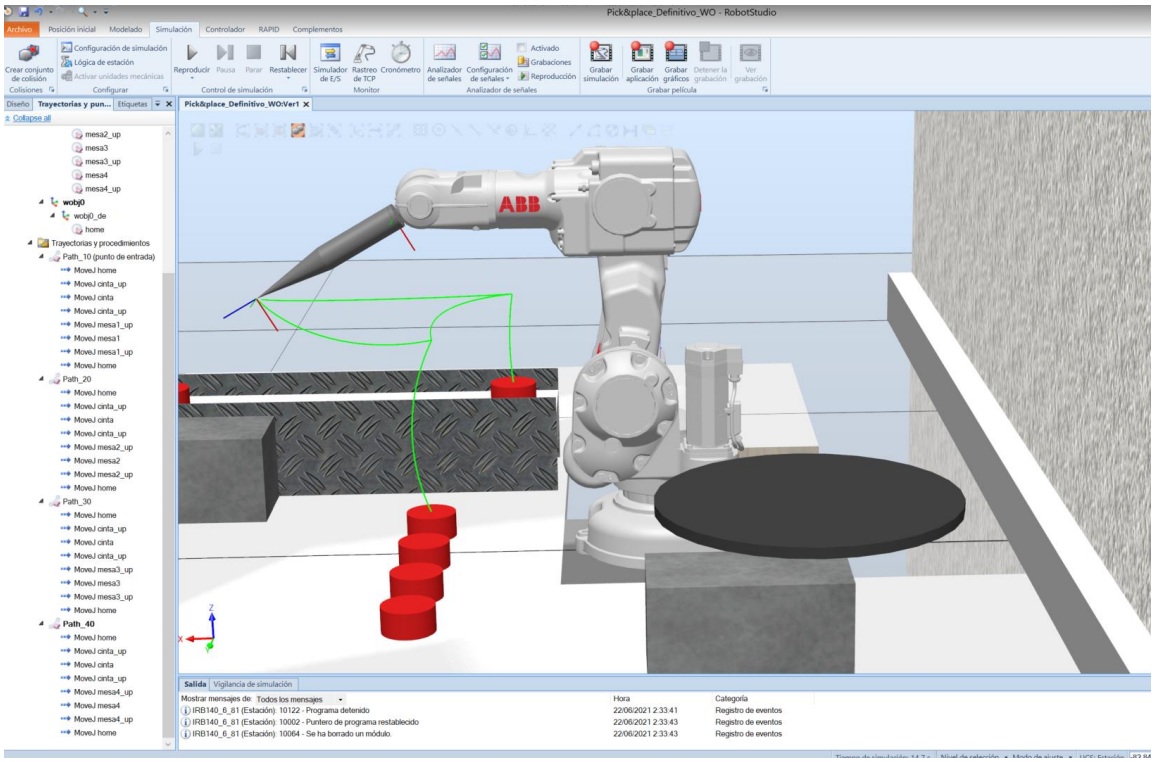


Figura 71 - Simulación de la trayectoria Path_10

La otra opción que ofrece el programa es pulsar el botón *Reproducir* en la pestaña de *Simulación.*

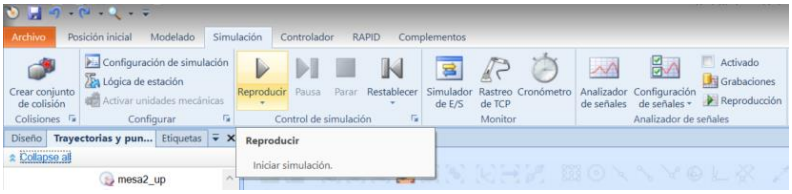


Figura 72 - Reproducir simulación

Si lo hacemos de esta otra manera, el robot simulará todas las trayectorias que hay creadas de manera consecutiva. En la figura 65 podemos observar que, si lo hacemos de esta manera, el rastreo TCP no se colocará correctamente en el puntero de la herramienta, sino en la muñeca del robot. Por lo tanto, de esta manera nos será más difícil identificar si la trayectoria que hemos creado es correcta.

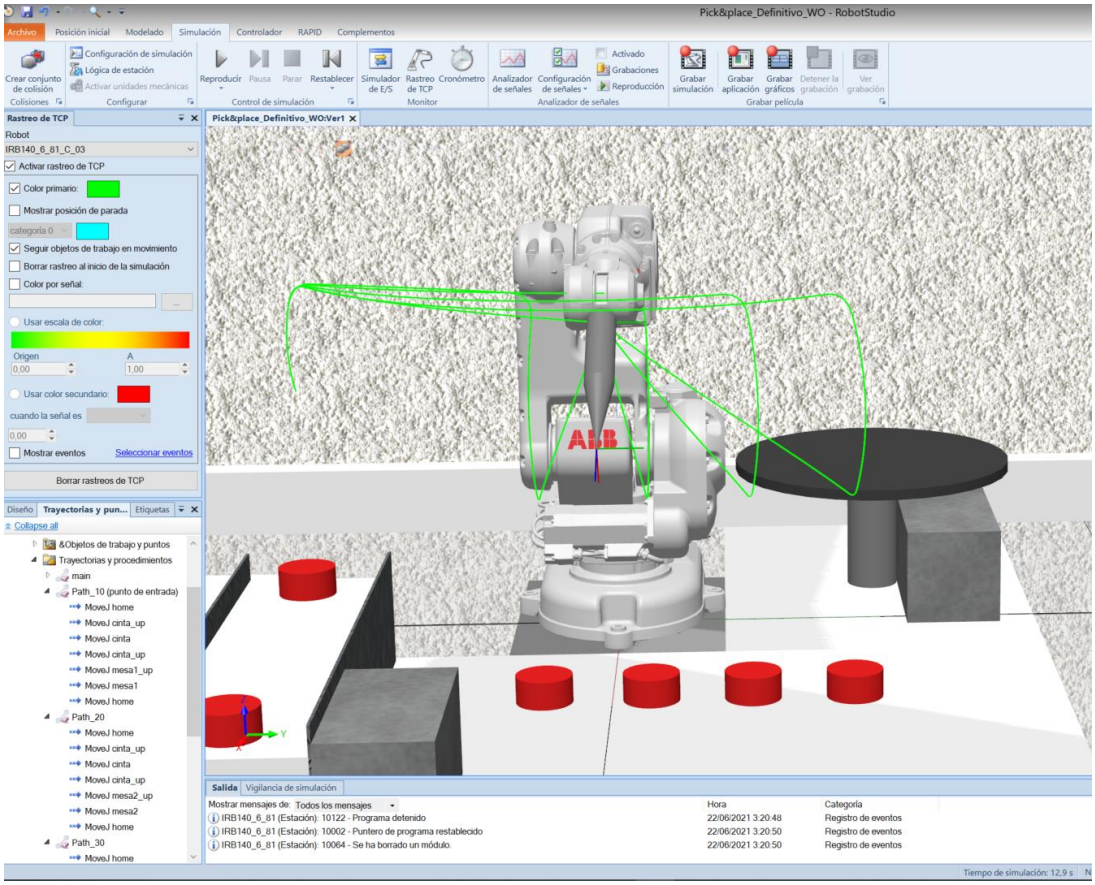


Figura 73 – Rastreo de TCP no válido

5.2.7 Editar instrucciones

Al realizar la simulación de las distintas trayectorias nos hemos dado cuenta de que todos los movimientos que realiza el robot son curvos. Esto es porque el programa, de manera predeterminada, programa los movimientos como *MoveJ* (*Joints)* a una velocidad (*v*) y un error (*z*) determinados también.

En nuestro caso, nos editaremos las instrucciones siguiendo los siguientes criterios:

- Cuando la herramienta realice movimientos lejos de objetos o zonas dónde puede colisionar utilizaremos "MoveJ", porque en este tipo de movimientos no nos importa que el error y la velocidad del robot sean grandes.
- Cuando la herramienta realice movimientos cerca de objetos o zonas con posibilidad de colisión emplearemos "MoveL", estos movimientos serán más lentos y precisos.
- Cuando la herramienta recoja la pieza, hasta que la suelte y se aleje de ella, realizaremos movimientos con velocidades muy reducidas para evitar golpes que puedan dañar el objeto o que se suelte de la ventosa.

Podemos modificar las instrucciones seleccionando en grupo los movimientos que queremos modificar, pulsando clic derecho encima de estos y escogiendo la opción *Editar instrucción.*

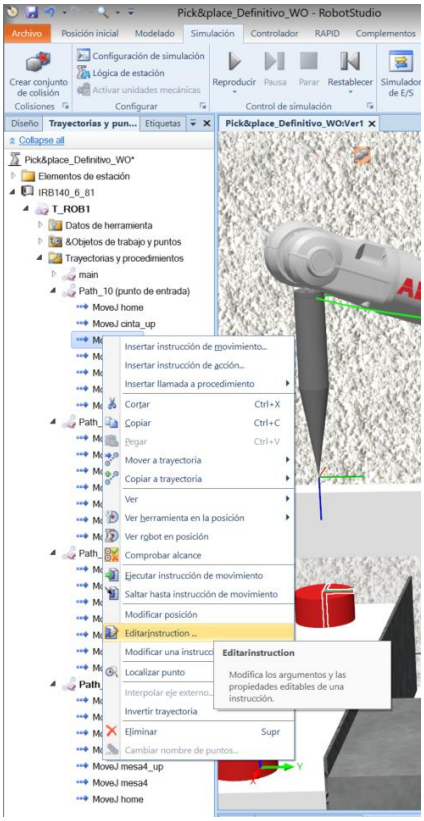


Figura 74 - Editar instrucción

Así pues, se nos abrirá una venta dónde podremos modificar los parámetros de las instrucciones de movimiento.

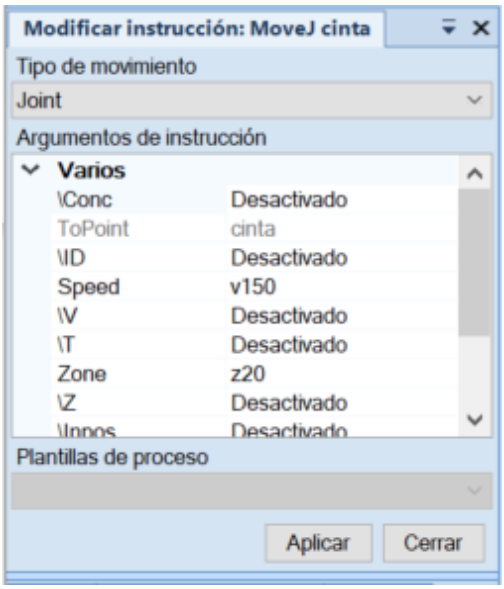


Figura 75 - Modificar instrucción

También podremos modificar las instrucciones directamente sobre el código RAPID. Siguiendo los criterios que hemos dicho anteriormente el código será el siguiente:

```
MODULE Module1
   CONST robtarget
home:=[[774.759526419,0,524.5],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+0
9,9E+09,9E+09,9E+09]];
   CONST robtarget mesa1:=[[468.24,-117.267,49.895],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget cinta_down:=[[315.176,-542.044,210.322],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget cinta_up:=[[312.045,-540.688,410],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa2:=[[468.24,69.394,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa3:=[[468.24,247.436,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa4:=[[468.24,426.471,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa1_up:=[[453.666,-115.061,410],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa2_up:=[[455.641,69.905,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa3_up:=[[458.293,238.162,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST robtarget mesa4_up:=[[456.724,412.119,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
CONST robtarget cinta:=[[315.176,-542.044,210.322],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

!***********************************************************
!
! Módulo:  Module1
!
! Descripción:
!   Instrucciones modificadas
!
! Autor: Pepe Bonafont
!
! Versión: 1.0
!
!***********************************************************




!***********************************************************
!
! Procedimiento Main
!
!   Este es el punto de entrada de su programa
!
!***********************************************************
PROC main()
    Path_10;
    Path_20;
    Path_30;
    Path_40;
ENDPROC

PROC Path_10()
   MoveJ Home,v1000,z5,tMyNewTool\WObj:=wobj0;
   MoveJ cinta_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
   MoveL cinta_down,v150,z0,tMyNewTool\WObj:=WO_Piezas;
   MoveL cinta_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
   MoveJ mesa1_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
   MoveL mesa1,v150,fine,tMyNewTool\WObj:=WO_Piezas;
   MoveJ mesa1_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
   MoveL Home,v1000,z5,tMyNewTool\WObj:=wobj0;
ENDPROC

PROC Path_20()
   MoveJ Home,v1000,z5,tMyNewTool\WObj:=wobj0;
   MoveJ cinta_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
```

```
    MoveL cinta_down,v150,z0,tMyNewTool\WObj:=WO_Piezas;
    MoveL cinta_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa2_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL mesa2,v150,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa2_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL Home,v1000,z5,tMyNewTool\WObj:=wobj0;
  ENDPROC

  PROC Path_30()
    MoveJ Home,v1000,z5,tMyNewTool\WObj:=wobj0;
    MoveJ cinta_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL cinta_down,v150,z0,tMyNewTool\WObj:=WO_Piezas;
    MoveL cinta_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa3_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL mesa3,v150,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa3_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL Home,v1000,z5,tMyNewTool\WObj:=wobj0;
  ENDPROC

  PROC Path_40()
    MoveJ Home,v1000,z5,tMyNewTool\WObj:=wobj0;
    MoveJ cinta_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL cinta_down,v150,z0,tMyNewTool\WObj:=WO_Piezas;
    MoveL cinta_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa3_up,v150,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL mesa3,v150,fine,tMyNewTool\WObj:=WO_Piezas;
    MoveJ mesa3_up,v300,z5,tMyNewTool\WObj:=WO_Piezas;
    MoveL Home,v1000,z5,tMyNewTool\WObj:=wobj0;
  ENDPROC

ENDMODULE
```

5.3 Programación directamente en RAPID

La otra opción, alternativa a usar Work Objects, es programa directamente el código RAPID. Hacerlo de esta manera, abre la posibilidad de simplificar el código y hacerlo más eficiente.

En nuestro caso, al ser muy básico, no podemos modificar mucho el programa. Pero, por ejemplo, podemos crear una función que sirva para recoger la pieza de la cinta, ya que es un movimiento que se repite mucho y nos ahorraríamos escribir bastantes instrucciones:

```
PROC recoger_pieza()

    MoveJ Home,v1000,z5,tMyNewTool;
    MoveJ cinta_up,v300,z5,tMyNewTool;
    MoveL cinta_down,v150,z0,tMyNewTool;
    MoveL cinta_up,v150,z5,tMyNewTool;

ENDPROC
```

Implantando esta función en nuestro programa, el código RAPID quedaría de la siguiente manera:

```
MODULE Module1

    CONST robtarget
home:=[[774.759526419,0,524.5],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+0
9,9E+09,9E+09,9E+09]];
    CONST robtarget mesa1:=[[468.24,-117.267,49.895],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget cinta_down:=[[315.176,-542.044,210.322],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget cinta_up:=[[312.045,-540.688,410],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa2:=[[468.24,69.394,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa3:=[[468.24,247.436,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa4:=[[468.24,426.471,49.895],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa1_up:=[[453.666,-115.061,410],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa2_up:=[[455.641,69.905,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa3_up:=[[458.293,238.162,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget mesa4_up:=[[456.724,412.119,410],[0.002360271,0,-
0.999997215,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget cinta:=[[315.176,-542.044,210.322],[0.002360271,0,-
0.999997215,0],[-1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !***********************************************************
    !
    ! Módulo:  Module1
    !
    ! Descripción:
    !   Implentar función recoger_pieza
    !
    ! Autor: Pepe Bonafont
    !
    ! Versión: 1.0
    !
    !***********************************************************

    !***********************************************************
    !
    ! Procedimiento Main
    !
    !   Este es el punto de entrada de su programa
```

```
!
!************************************************************


PROC recoger_pieza()

    MoveJ Home,v1000,z5,tMyNewTool;
    MoveJ cinta_up,v300,z5,tMyNewTool;
    MoveL cinta_down,v150,z0,tMyNewTool;
    MoveL cinta_up,v150,z5,tMyNewTool;

ENDPROC


PROC main()

  !TRAYECTORIA1
    Recoger_pieza;
    MoveJ mesa1_up,v150,z5,tMyNewTool;
    MoveL mesa1,v150,fine,tMyNewTool;
    MoveJ mesa1_up,v150,z5,tMyNewTool;
    MoveL Home,v1000,z5,tMyNewTool;

  !TRAYECTORIA2
    Recoger_pieza;
    MoveJ mesa2_up,v150,z5,tMyNewTool;
    MoveL mesa2,v150,fine,tMyNewTool;
    MoveJ mesa2_up,v150,z5,tMyNewTool;
    MoveL Home,v1000,z5,tMyNewTool;

  !TRAYECTORIA3
    Recoger_pieza;
    MoveJ mesa3_up,v150,z5,tMyNewTool;
    MoveL mesa3,v150,fine,tMyNewTool;
    MoveJ mesa3_up,v150,z5,tMyNewTool;
    MoveL Home,v1000,z5,tMyNewTool;

 !TRAYECTORIA4
    Recoger_pieza;
    MoveJ mesa4_up,v150,z5,tMyNewTool;
    MoveL mesa4,v150,fine,tMyNewTool;
    MoveJ mesa4_up,v150,z5,tMyNewTool;
    MoveL Home,v1000,z5,tMyNewTool;


  ENDPROC
ENDMODULE
```

6. Desarrollo de pruebas experimentales

Una vez hecho el modelado de la célula robotizada y el desarrollo de las pruebas de simulación, solo nos quedará desarrollar la parte experimental del trabajo de fin de grado.

En esta parte del trabajo comprobaremos que las trayectorias de la simulación son correctas y añadiremos al programa la activación de salidas y entradas necesarias para completar el Pick & Place, como la activación de la cinta, el sensor que detecta las piezas y la herramienta del robot.

6.1 Señales del robot

Al programa que ya teníamos hecho en la simulación, hay que añadir las instrucciones que permiten que las piezas avancen por la cinta, se paren al llegar al sensor y activen/desactiven la herramienta para que recoja o suelte los objetos.

Para poder añadir estas instrucciones en el código RAPID y conseguir que el robot realice la tarea que deseamos, deberemos conocer las señales que hacen posible todo esto.

Las señales que nosotros utilizaremos son tres:

- CONVEYOR_FWD: activa (1) o desactiva (0) la cinta transportadora hacia delante.
- CONVEYOR_OBJ_SEN: sensor de detección de piezas en la cinta transportadora.
- GRIPPER_CLOSE: Abre (1) o cierra (0) la pinza del robot.

6.2 Crear nuevas funciones para el programa

Una vez conocemos las señales que necesarias para este proyecto, podemos crear las funciones que nos permitirán más tarde completar nuestro código RAPID para que el robot realice el Pick & Place que deseamos.

Crearemos tres funciones.

6.2.1 Función avanzar_cinta

Necesitamos una función que haga que la cinta transportadora avance hasta que la pieza llegue al sensor del final de la cinta.

La función será la siguiente:

```
PROC avanzar_cinta()
    SetDO CONVEYOR_FWD,1;  → activa la cinta
    WaitDI CONVEYOR_OBJ_SEN, 1; → espera la señal del sensor
    SetDO CONVEYOR_FWD,0; → detiene la cinta
ENDPROC
```

6.2.2 Función coger_pieza

Esta función se encargará de hacer que el robot se desplace hasta la posición donde se encuentra la pieza una vez se ha detenido la cinta, cogerá la pieza y la levantará hasta que le llegue la instrucción que le indique donde llevarla.

Esta función será:

```
PROC coger_pieza()
    MoveJ Offs(cinta,0,0,225),v300,z5,MyNewTool;
    SetDO GRIPPER_CLOSE,1;
    MoveL cinta,v150,z0,MyNewTool;
    MoveL Offs(cinta,0,0,225),v150,z5,MyNewTool;
ENDPROC
```

6.2.3 Función_dejar pieza

Esta última función hará que el robot, una vez coloque la pieza en su sitio, desactive la herramienta para dejarla y vuelva a la posición de reposo hasta que le llegue una nueva orden.

La escribiremos la función de la siguiente manera:

```
PROC dejar_pieza()
    WaitTime 1; → el programa espera 1s antes de pasar a la siguiente orden
    SetDO GRIPPER_CLOSE,0; → la herramienta se desactiva y suelta la pieza
    WaitTime 0.5; → el programa espera 0,5s antes de pasar a la siguiente orden
ENDPROC
```

6.3 Implementar las funciones en el programa

Para completar el programa que habíamos creado para la simulación deberemos añadir las funciones que acabamos de crear. Con esto, finalizaremos la programación del robot y ya podremos probar experimentalmente el Pick & Place.

Además, para hacer un programa más corto y limpio aún, utilizaremos Offsets para ahorrarnos puntos del programa.

El código RAPID quedaría de la siguiente manera:

```
MODULE Module1

    CONST robtarget
home:=[[774.759526419,0,524.5],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+0
9,9E+09,9E+09,9E+09]];

    CONST robtarget cinta:=[[315,-450,225],[0.002360271,0,-0.999997215,0],[-1,-1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget mesa1:=[[468.24,-117.267,75],[0.002360271,0,-0.999997215,0],[-
1,-1,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !***********************************************************
    !
    ! Módulo:  Module1
    !
    ! Descripción:
    !   <Programa final
    !
    ! Autor: Pepe Bonafont
    !
    ! Versión: 1.0
    !
    !***********************************************************

    !***********************************************************
    !
    ! Procedimiento Main
    !
    !   Este es el punto de entrada de su programa
    !
    !***********************************************************

    PROC main()

      !PROC Path_10()

        avanzar_cinta;
        coger_pieza;

        MoveJ Offs(mesa1,0,0,335),v300,z5,MyNewTool;
        MoveL mesa1,v150,z0,MyNewTool;

        dejar_pieza;

        MoveL Offs(mesa1,0,0,335),v150,z5,MyNewTool;
        MoveJ home,v1000,z5,MyNewTool;
```

```
!PROC Path_20()

    avanzar_cinta;
    coger_pieza;

    MoveJ (mesa1,0,180,335),v300,z5,MyNewTool;
    MoveL (mesa1,0,180,0),v150,z0,MyNewTool;

    dejar_pieza;

    MoveL Offs(mesa1,0,180,335),v150,z5,MyNewTool;
    MoveJ home,v1000,z5,MyNewTool;

  !PROC Path_30()

    avanzar_cinta;
    coger_pieza;

    MoveJ (mesa1,0,360,335),v300,z5,MyNewTool;
    MoveL (mesa1,0,360,0),v150,z0,MyNewTool;

    dejar_pieza;

    MoveL Offs(mesa1,0,360,335),v150,z5,MyNewTool;
    MoveJ home,v1000,z5,MyNewTool;

  !PROC Path_40()

    avanzar_cinta;
    coger_pieza;

    MoveJ (mesa1,0,540,335),v300,z5,MyNewTool;
    MoveL (mesa1,0,540,0),v150,z0,MyNewTool;

    dejar_pieza;

    MoveL Offs(mesa1,0,540,335),v150,z5,MyNewTool;
    MoveJ home,v1000,z5,MyNewTool;

  ENDPROC
```

```
 PROC avanzar_cinta()
      SetDO CONVEYOR_FWD,1;
      WaitDI CONVEYOR_OBJ_SEN, 1;
      SetDO CONVEYOR_FWD,0;
   ENDPROC

   PROC coger_pieza()
      MoveJ Offs(cinta,0,0,225),v300,z5,MyNewTool;
      SetDO GRIPPER_CLOSE,1;
      MoveL cinta,v150,z0,MyNewTool;
      MoveL Offs(cinta,0,0,225),v150,z5,MyNewTool;
   ENDPROC

   PROC dejar_pieza()
      WaitTime 1;
      SetDO GRIPPER_CLOSE,0;
      WaitTime 0.5;
   ENDPROC

ENDMODULE
```

6.4 Probar programa experimentalmente

Para finalizar con este proyecto, explicaremos todos los pasos que tendremos que seguir para poder probar nuestro programa de manera experimental en el laboratorio.

6.4.3 Secuencia de encendido

Para empezar, deberemos encenderlo, colocando el interruptor que se encuentra en la caja del controlador en ON.

Figura 76 - Interruptor ON

A continuación, introduciremos nuestro pendrive (donde debemos tener guardado el programa que hemos guardado y queremos probar) en la ranura USB que se encuentra en la caja del controlador.

Figura 77 - Ranura USB

Más tarde, comprobaremos que la llave del tubo que aspira aire en la ventosa se encuentra abierta y, por tanto, la herramienta podrá funcionar.

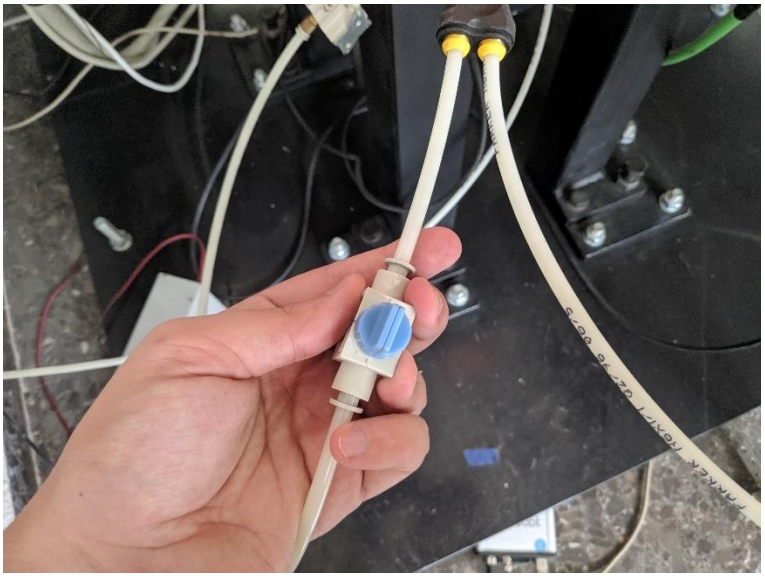


Figura 78 - Llave abierta

Por último, nos aseguraremos de que magnetotérmico de la cinta transportadora se encuentra subido.

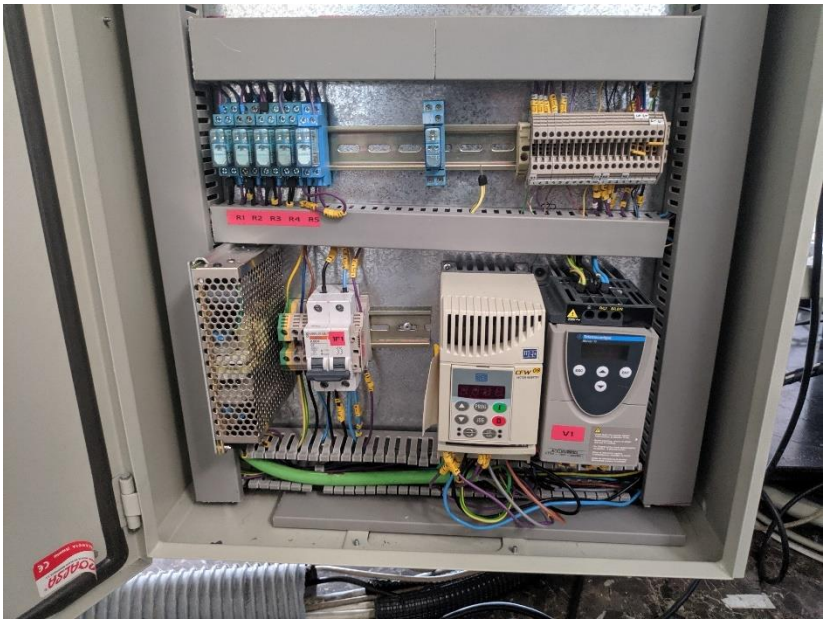


Figura 79 - Magnetotérmico

6.4.4 Secuencia de calibrado

Una vez realizada la secuencia de encendido, manejaremos todo desde el FlexPendant.

Figura 80 – FlexPendant

Figura 81 - Botón de emergencia y hombre muerto

Como podemos ver en la figura 80 y en la figura 85, el FlexPendant se compone de las siguientes partes:

- Pantalla táctil
- Botonera
- Joystick (movimiento vertical, horizontal y de rotación)
- Botón de paro de emergencia
- Pulsador de hombre muerto

El FlexPendant se encenderá solo una vez pongamos el interruptor que se encuentra en la caja del controlador en ON.

La secuencia que tendremos que realizar para calibrar el robot será la siguiente:

1) Pulsar el botón “ABB”

Figura 82 - Botón ABB

2) Entrar en el menú “Calibración”

Figura 83 - Menú calibración

3) Pulsar “Cuentarrevoluciones no actualizados”

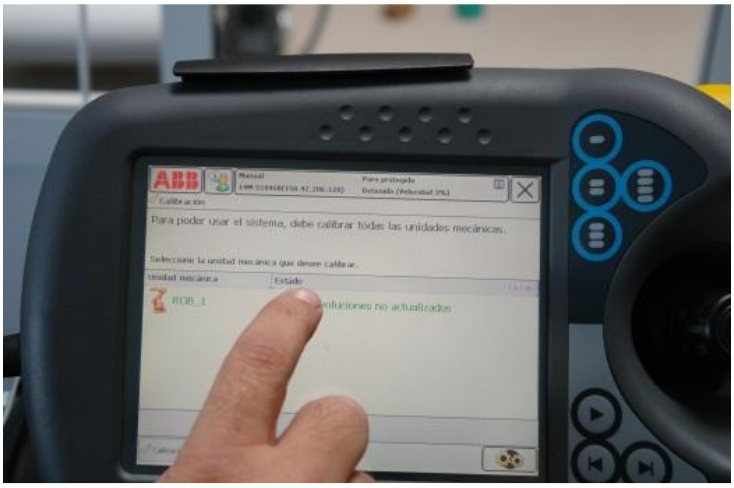


Figura 84 - Cuentarrevoluciones no actualizados

4) Pulsar “Actualizar cuentarrevoluciones”

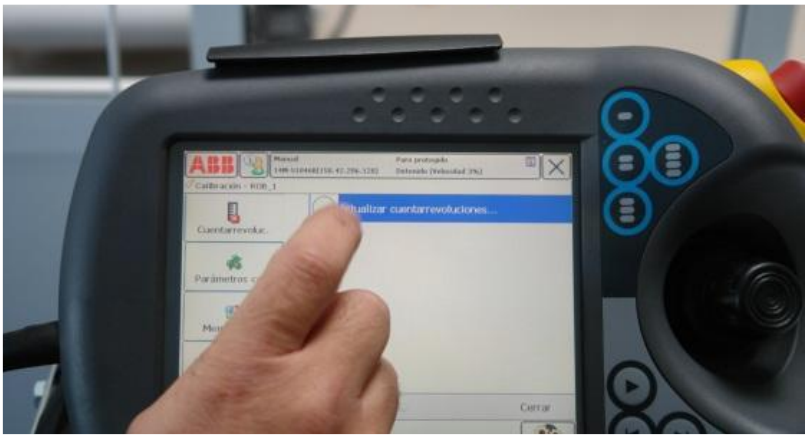


Figura 85 - Actualizar cuentarrevoluciones

5) Confirmar

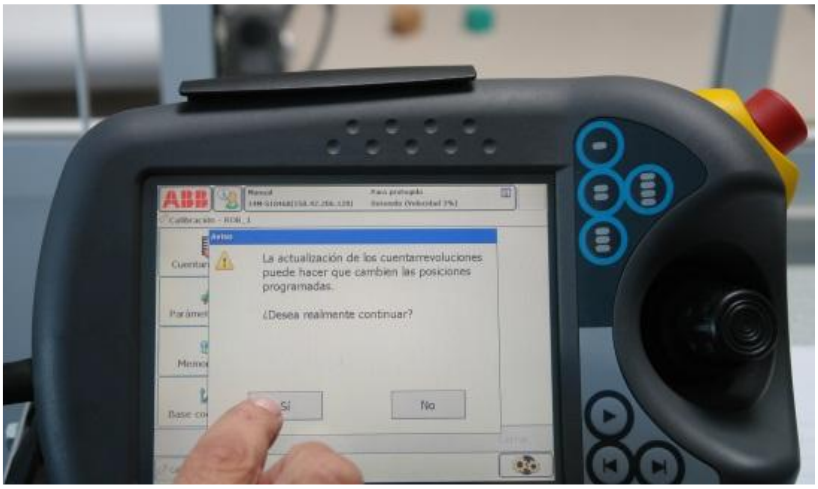


Figura 86 - Confirmar

6) Actualizar

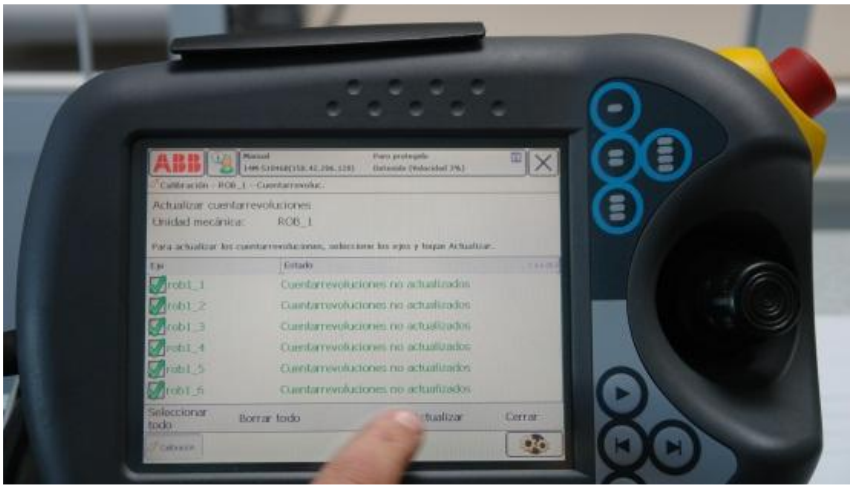


Figura 87 - Actualizar

7) Confirmar actualización

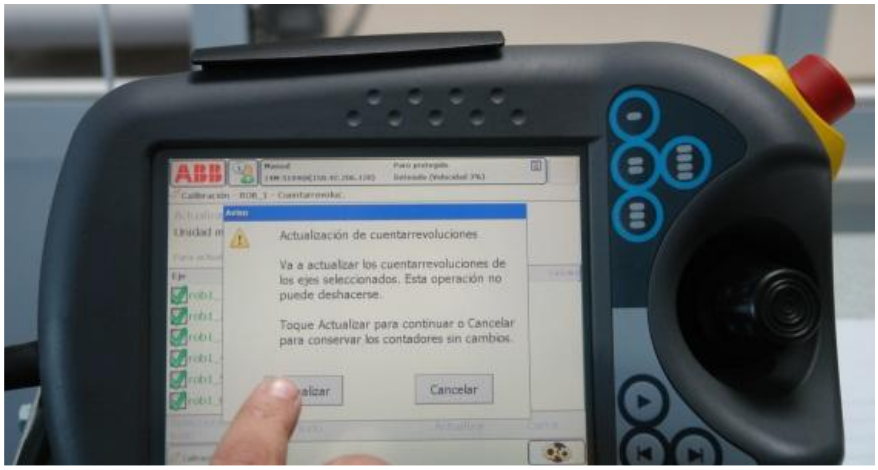


Figura 88 - Confirmar actualización

6.4.5 Cargar programa

Una vez calibrado el robot, ya podremos cargar nuestro programa para probarlo. Para ello, seguiremos los siguientes pasos:

1) Pulsamos el botón “ABB” para acceder al menú

Figura 89 - Menú ABB

2) Entramos en “Ventana de producción”

Figura 90 - Ventana de producción

3) Seleccionamos la memoria extraíble

Figura 91 - Unidad extraíble

4) Buscamos la carpeta donde tenemos el archivo del programa

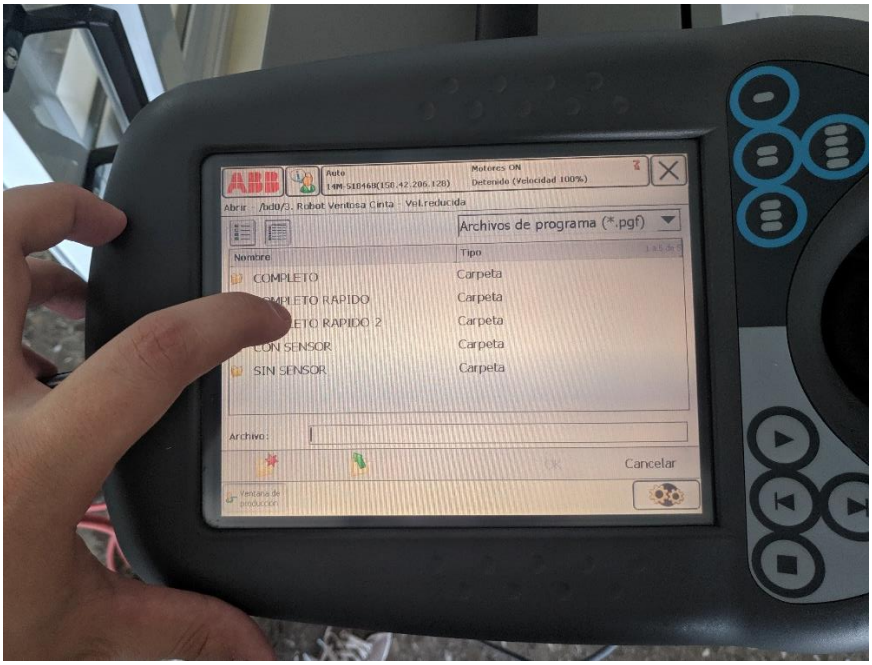


Figura 92 - Abrir carpeta

5) Seleccionamos nuestro programa

Figura 93 - Seleccionar programa

6) Esperamos a que el programa cargue

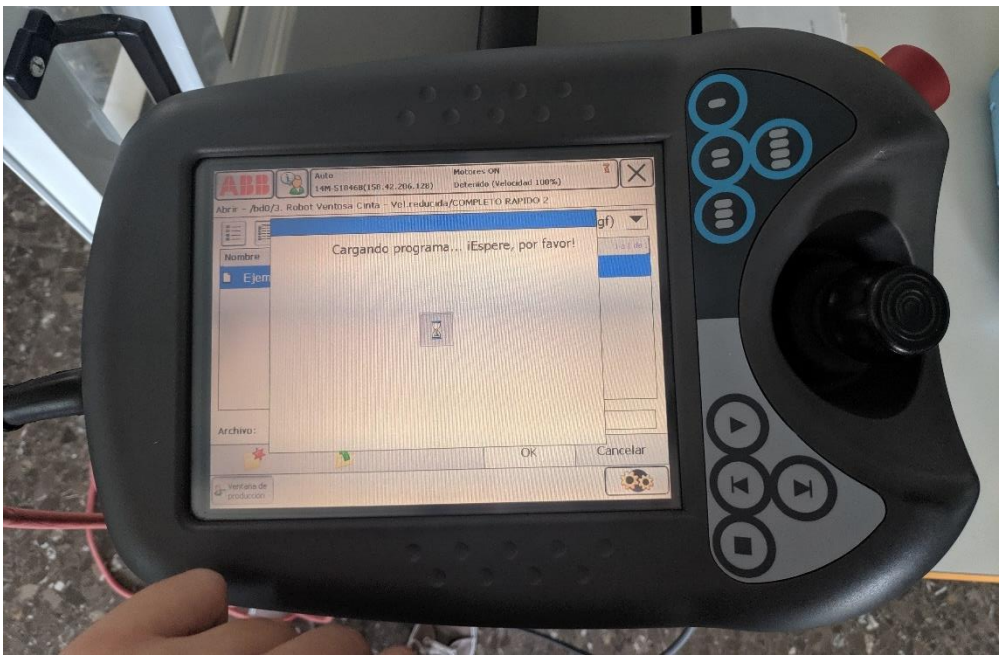


Figura 94 - Esperar carga del programa

Después de seguir estos 7 sencillos pasos, se nos abrirá una ventana con el código de programación en RAPID y ya podremos ejecutar el programa, aunque deberemos tener en cuenta el modo de actuación en el que está puesto el controlador y el estado de los motores.

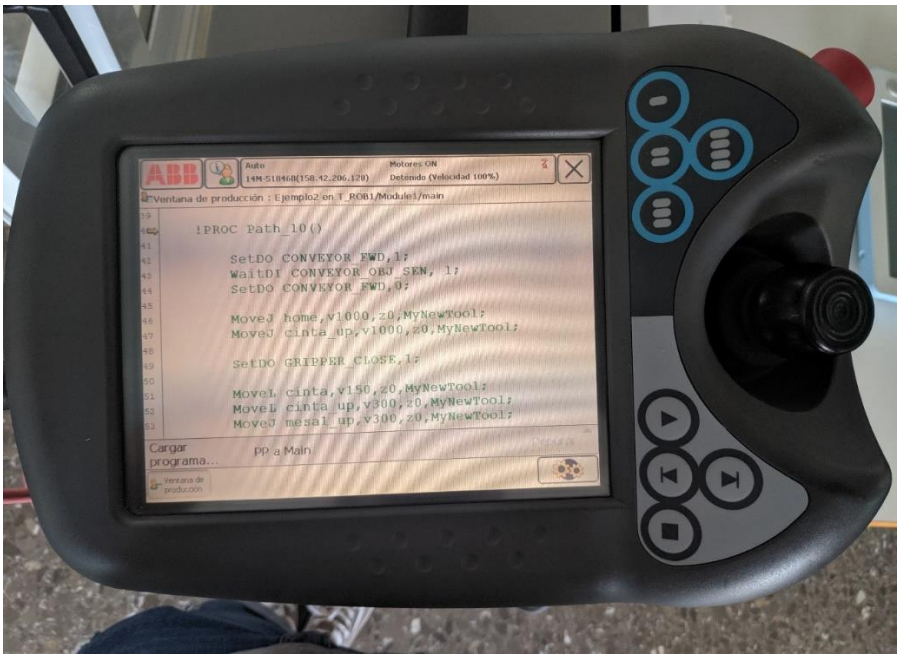


Figura 95 - Programa abierto y listo

6.4.6 Modos del controlador

El control tiene tres modos de funcionamiento que diferenciaremos a continuación. Se puede cambiar de un modo a otro girando la llave, como vemos en la figura 96.

Figura 96 - Selección de modo de actuación

- **Modo automático:** Corresponde a la posición de la izquierda. En este modo, el operario no tendrá que mantener el interruptor de hombre muerto pulsado. Una vez pulse el botón “play”, el programa se ejecutará automáticamente y sólo se detendrá si pulsamos el botón “stop” o el botón de emergencia.

- **Modo manual a velocidad reducida:** Corresponde a la posición central. En este modo, el operario tendrá que mantener el interruptor de hombre muerto pulsado en todo momento o el programa se parará. En este modo de funcionamiento, el operario podrá pasar de una en una las órdenes del programa o darle al botón “play” para que se ejecuten todas, una detrás de otra.

Este modo es perfecto para probar por primera vez un programa, ya que los movimientos se ejecutan a una velocidad reducida y, por tanto, es mucho más seguro.

- **Modo manual 100%:** corresponde a la posición de la derecha. Es exactamente igual que el modo anterior, pero funciona a la velocidad programada.

6.4.7 Cargar motores

Por seguridad, el controlador cada vez que se apaga o cambia de modo de actuación descarga los motores para que no se pueda activar el programa accidentalmente.

Para cargar los motores, deberemos pulsar el botón que se encuentra justo encima de la llave donde seleccionamos el modo de actuación del robot.

Si los motores están cargados, la luz del botón permanecerá encendida.

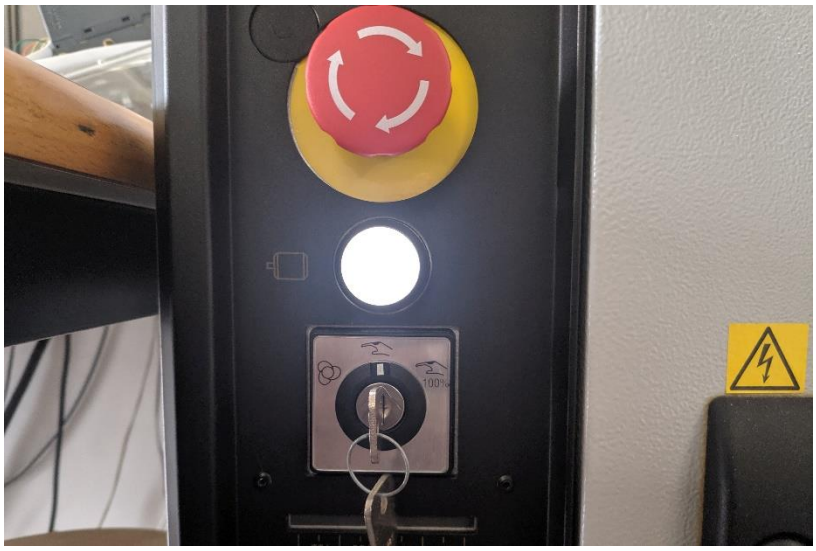


Figura 97 - Cargar motores

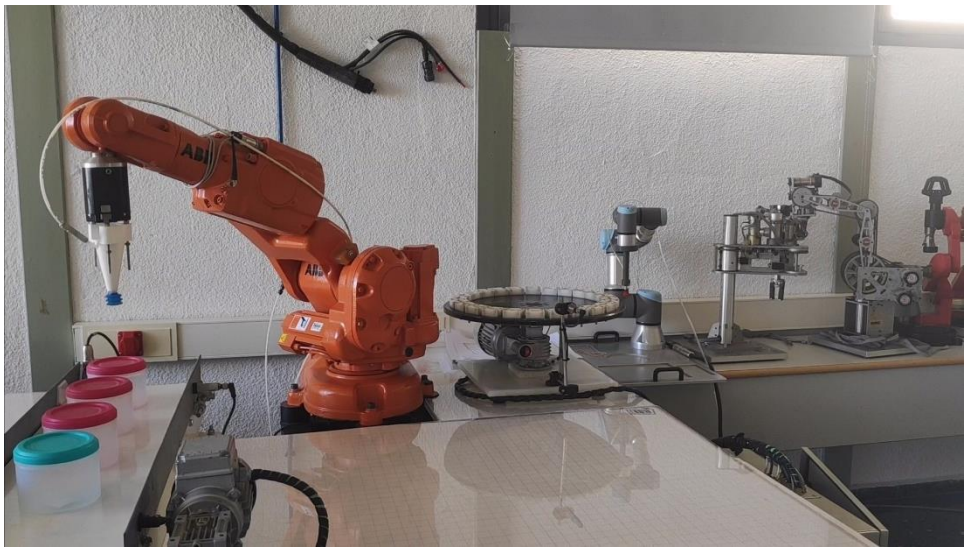
Con esto, ya sabríamos todo lo necesario para experimentar con el robot y comprobar que el programa que hemos creado funciona.

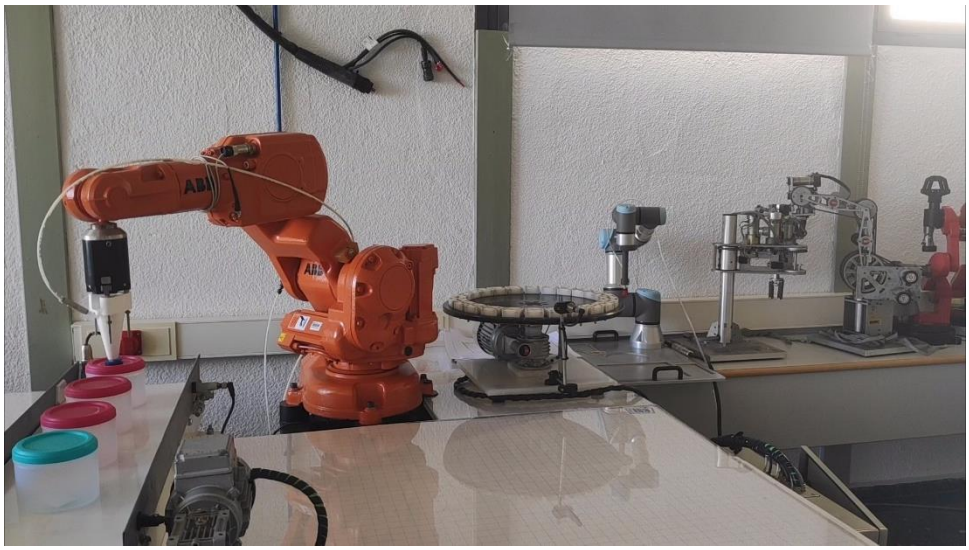
6.4.8 Simulación paso a paso

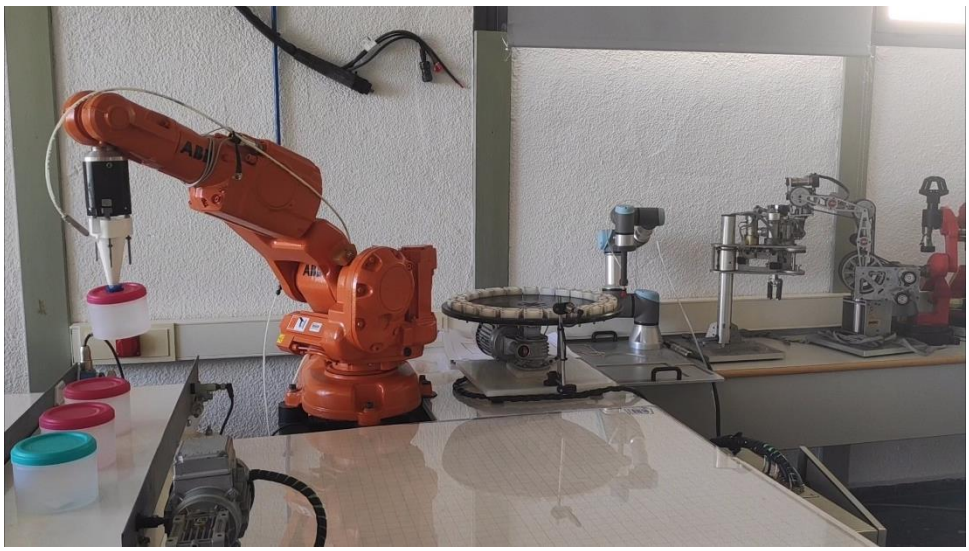
En este apartado, mostraremos la simulación con una secuencia de imágenes utilizando los frames más importantes de un vídeo que grabamos donde se muestran los movimientos del robot para la ejecución del pick and place.

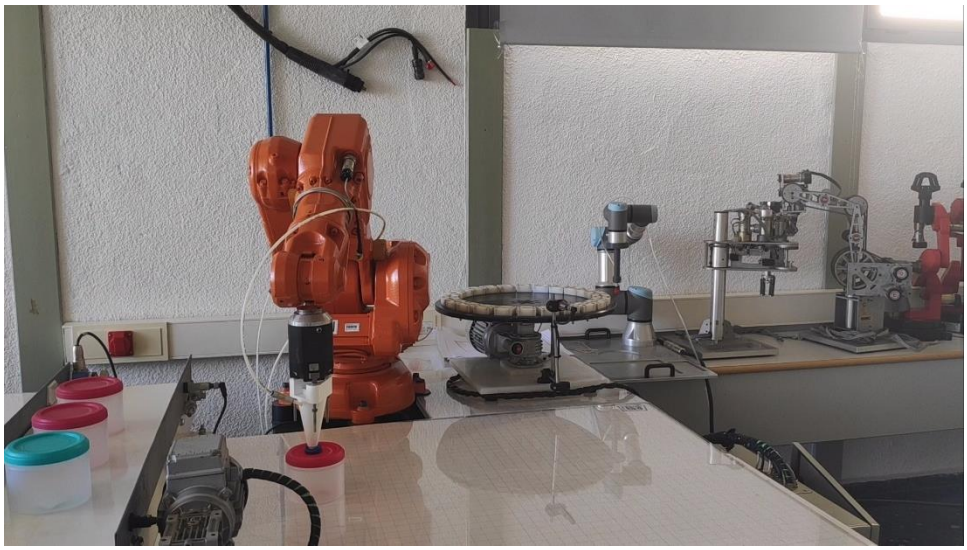
El vídeo completo se puede ver en el siguiente enlace: https://youtu.be/BpXFS7lImgE

Primera trayectoria:

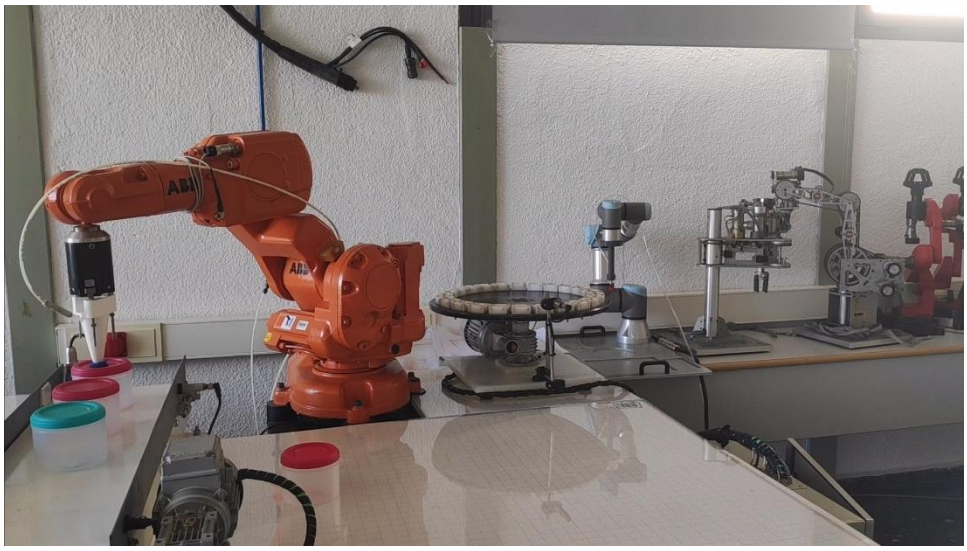


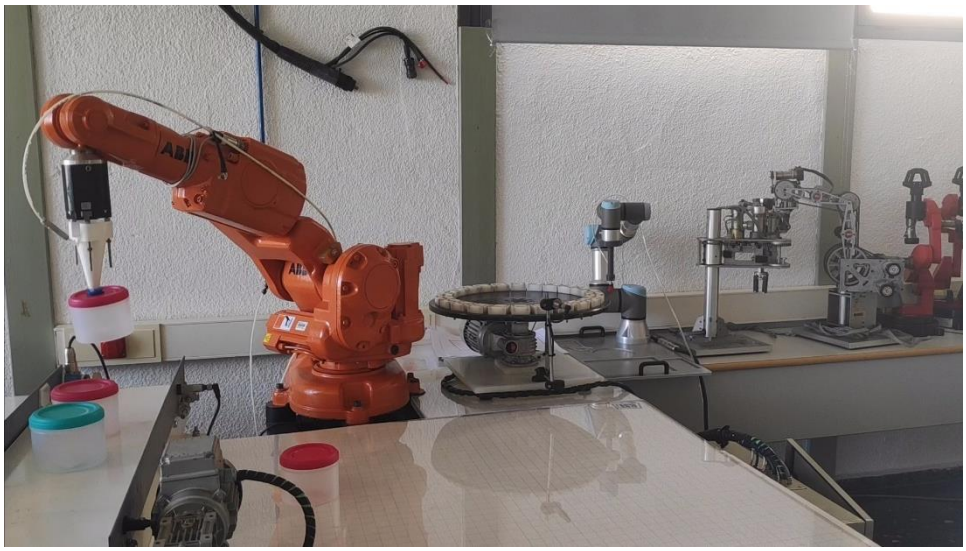






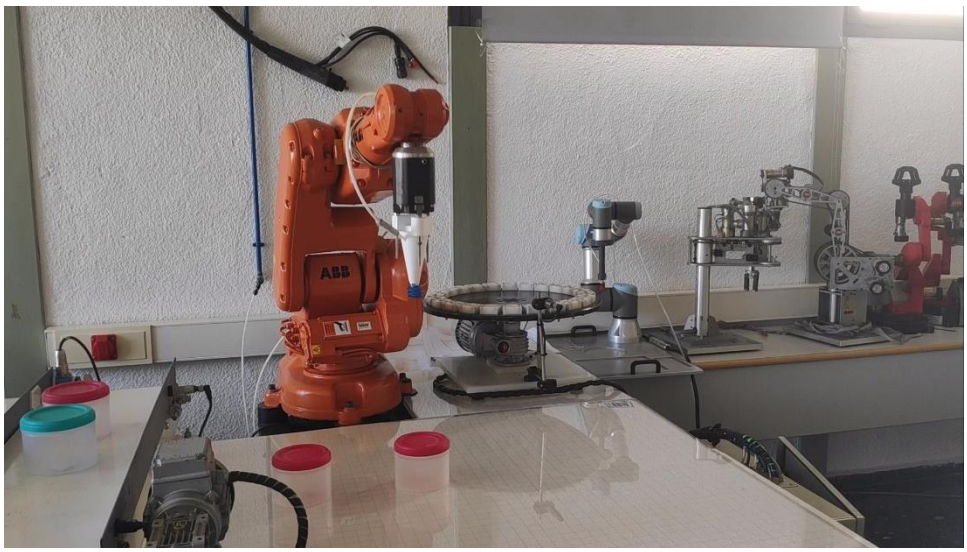
Segunda trayectoria:



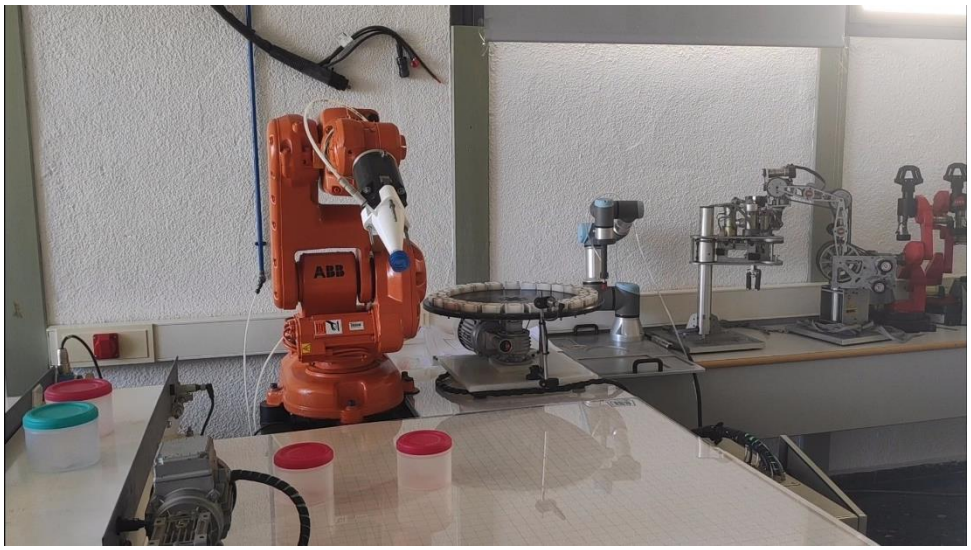



ABB

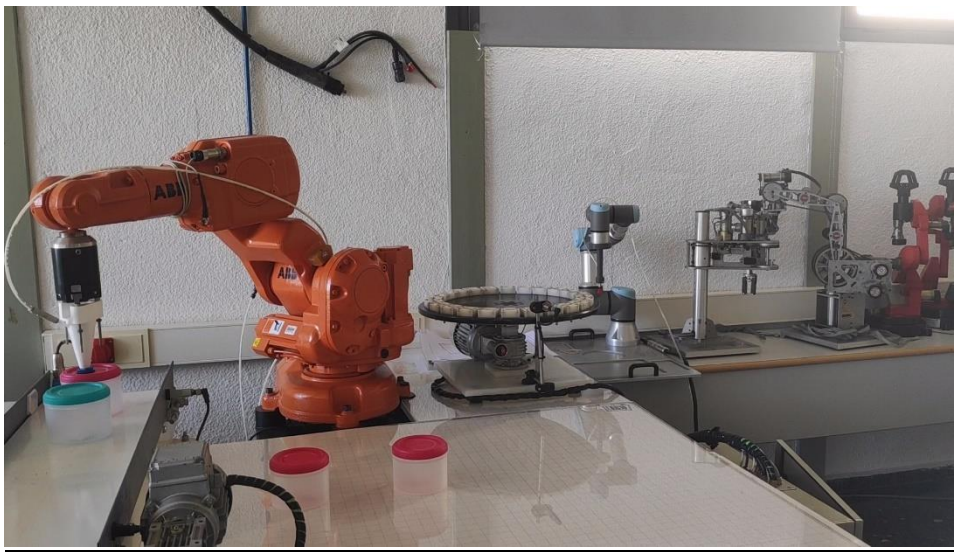

ABB

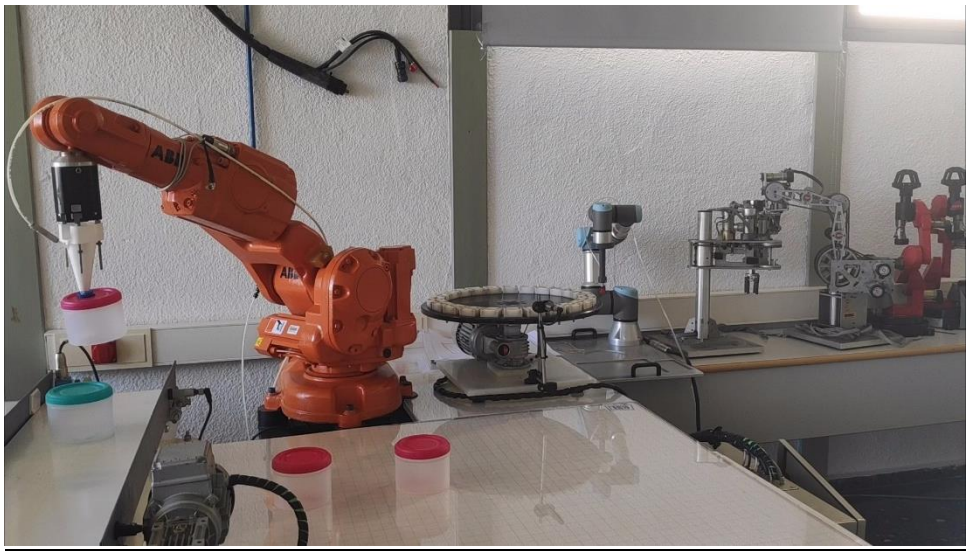


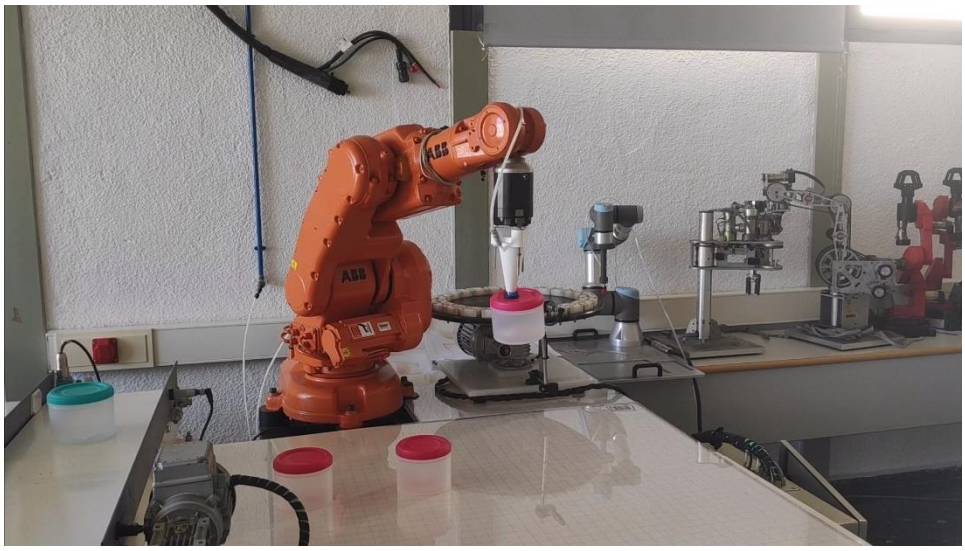
ABB

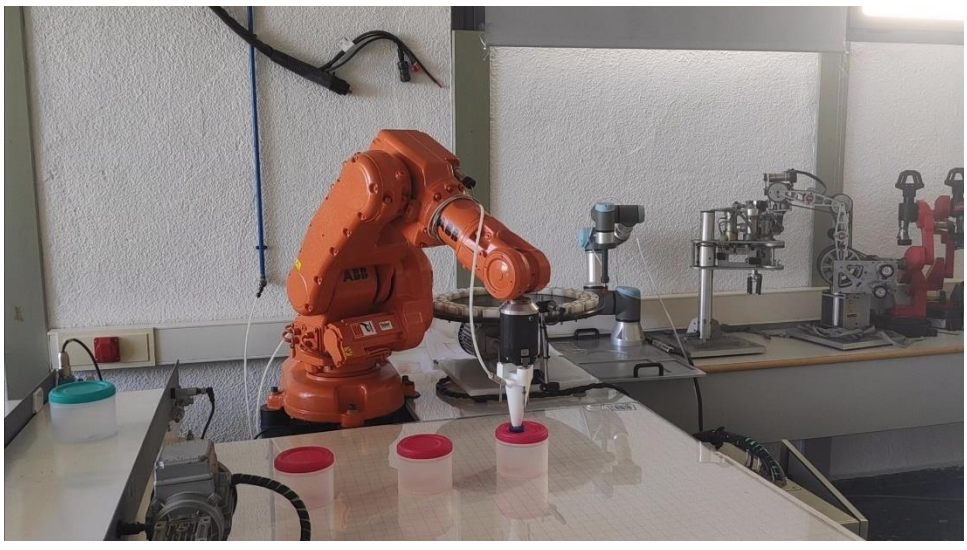


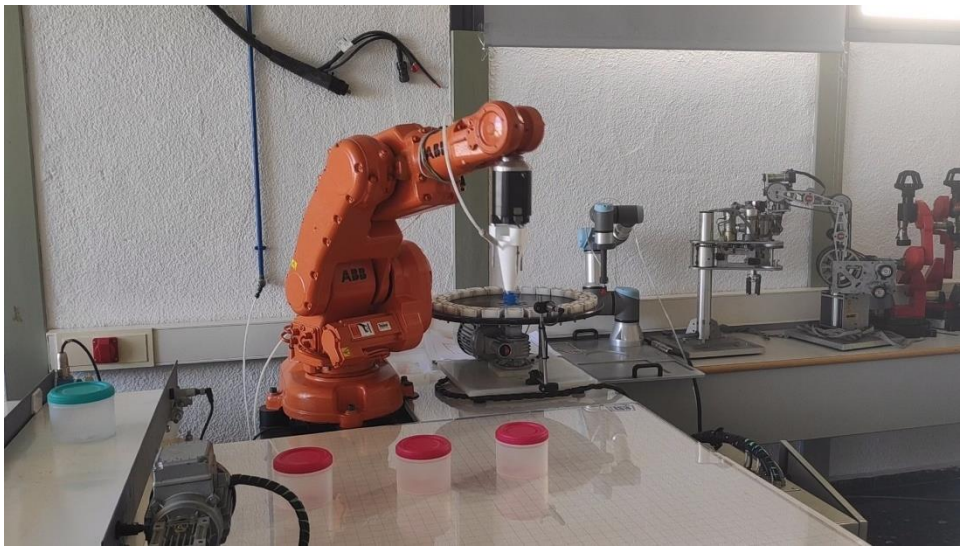
Tercera trayectoria:

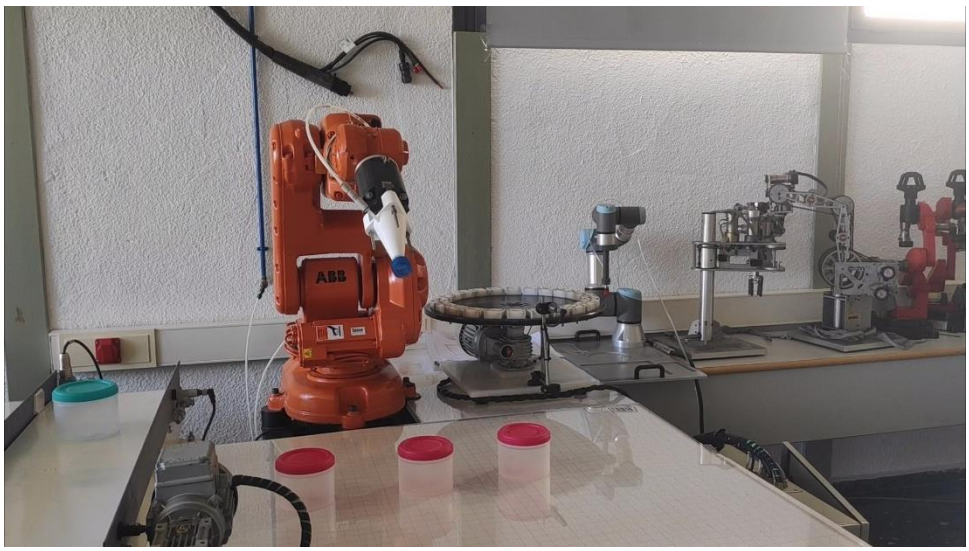




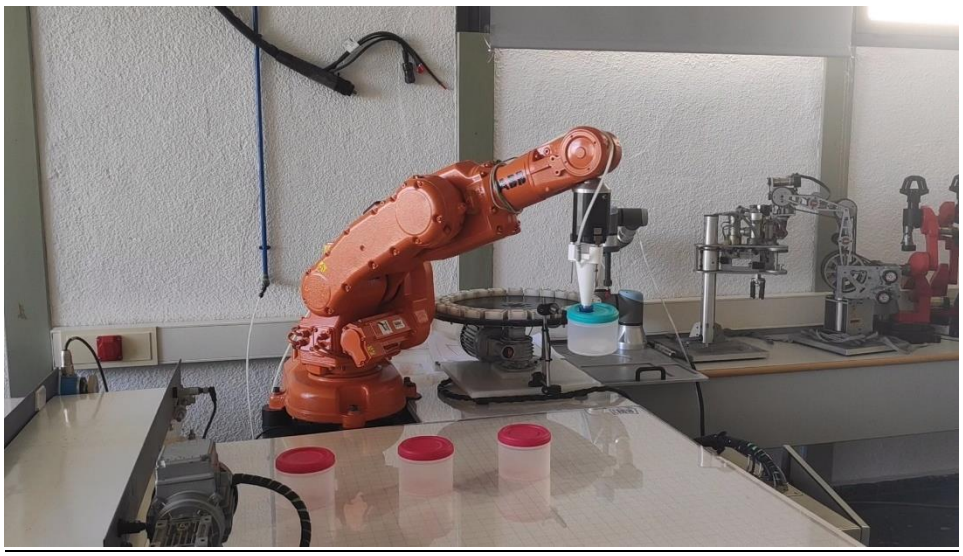


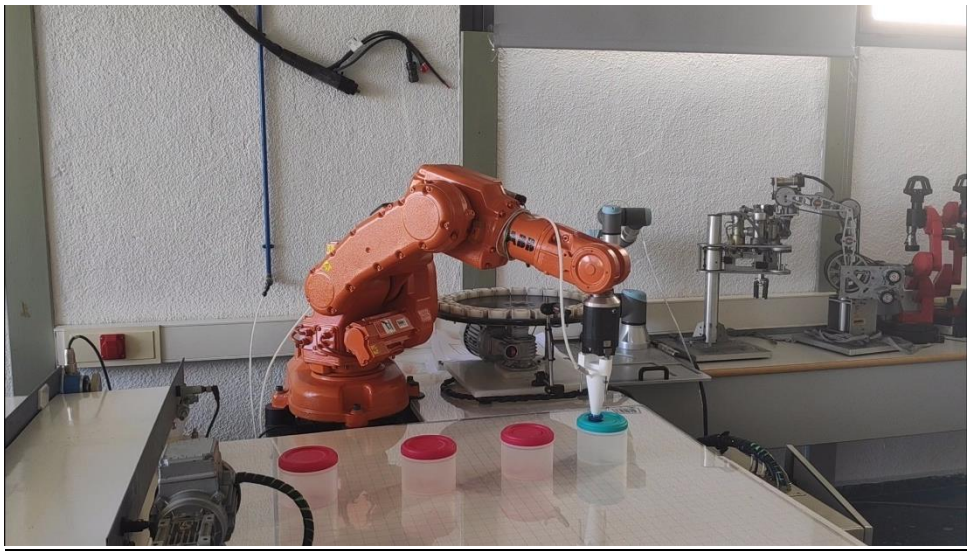


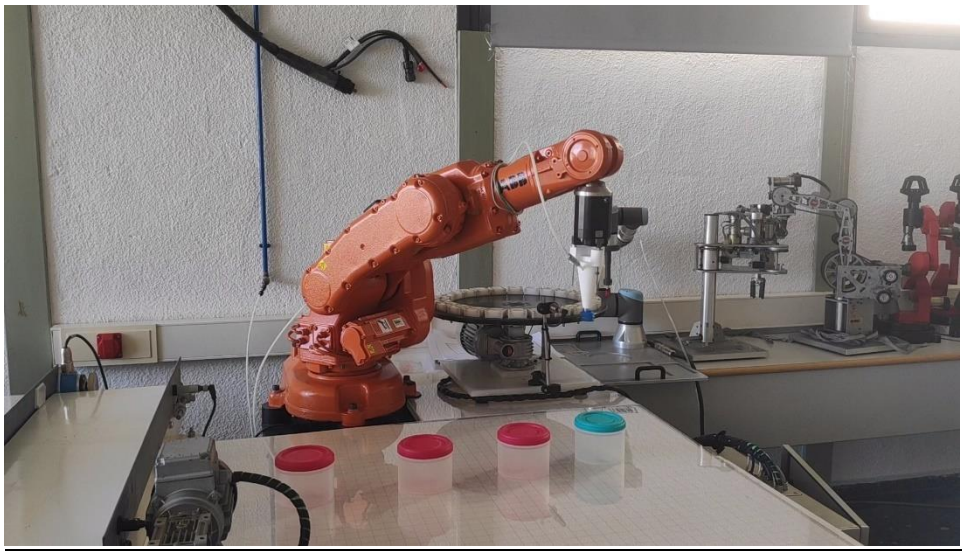




Cuarta trayectoria:







7. Conclusiones

Como conclusiones después de realizar el proyecto que nos incumbe, podemos decir que la elaboración de una célula robotizada y el desarrollo de pruebas de simulación previas a la realización de pruebas de manera experimental presenta números ventajas, pero también presenta unos inconvenientes para tener en cuenta, ya que dependiendo del tipo de trabajo que se quiera realizar puede llegar a ser contraproducente.

La ventaja más significativa de crear el modelado de una célula robotizada y desarrollar pruebas de simulación en RobotStudio que hemos encontrado después de realizar el proyecto, como era de esperar, es que, al probar el programa en el robot de manera experimental, las trayectorias funcionaban como queríamos y en ningún momento hemos tenido riesgo de colisión con ningún objeto de la estación.

Otra ventaja es que al realizar la simulación nos aseguramos de que el programa que teníamos en mente y hemos diseñado es correcto y se encuentra dentro del espacio de trabajo del robot. De esto nos dimos cuenta porque en las primeras simulaciones que hicimos había puntos del programa donde el robot no llegaba y, por tanto, nos saltaba un error en la simulación avisándonos, pudiendo rectificar el programa antes de probarlo en el laboratorio de robótica.

Sin embargo, la principal desventaja que presenta el crear el modelado de una célula robotizada y el desarrollo de pruebas de simulación previas a la realización de pruebas experimentales es que hay que emplear una gran cantidad de tiempo, y más si no somos expertos en el programa como era nuestro caso.

La conclusión que obtengo de este trabajo de fin de grado es que la elaboración de una célula robotizada y el desarrollo de pruebas de simulación previas a la realización de pruebas experimentales tiene muchísimas ventajas y vuelven el proceso mucho más seguro, pero no compensa, por la carga de tiempo que supone, hacerlo para programas y trabajos tan sencillos como el de nuestro proyecto, sino más bien, para grandes proyectos que requieran una programación compleja.

8. Bibliografía

- **Manuales:**

  - RAPID manual del operador. ABB.

https://library.e.abb.com/public/5240940e78c27430c1257b67004c6759/3HAC032104-es.pdf

  - Manual técnico RAPID. ABB.

https://library.e.abb.com/public/b227fcd260204c4dbeb8a58f8002fe64/Rapid_instructions.pdf

- **Fichas técnicas:**

  - Información del robot IRB 140. ABB.

https://new.abb.com/products/robotics/es/robots-industriales/irb-140/datos

- **Videotutoriales:**

  - Juan Carlos Martín Castillo. YouTube.

https://www.youtube.com/channel/UCsRQGtuqCm1-d-whRYOSV2Q

  - Arizaa. Youtube.

https://www.youtube.com/channel/UC5eFeNylNpHDAaIgr0F7dBw

- **Información recopilada en clases:**

  Información y documentación recopilada de la asignatura de Sistemas Robotizados impartida en el Grado de Ingeniería Eléctrica de la Universidad Politécnica de Valencia.

- **Normativa:**

  - Real Decreto 488/1997, de 14 de abril, sobre disposiciones mínimas de seguridad y salud relativas al trabajo con equipos que incluyen pantallas de visualización.

https://www.boe.es/buscar/act.php?id=BOE-A-1997-8671

DOCUMENTO II

PLIEGO DE CONDICIONES

1. Objeto

El objetivo del pliego de condiciones es establecer las obligaciones a cumplir por las partes contratantes durante la realización de este Trabajo de Fin de Grado: Modelado de una célula robotizada y desarrollo de pruebas de simulación y experimentales.

2. Condiciones de los materiales

Los materiales necesarios para la realización de este trabajo han sido un ordenador de sobremesa, dos pantallas, teclado, ratón y accesorios ofimáticos como auriculares, y unidades de memoria portátiles. Además de un asiento y mesa.

Todos estos materiales tienen que estar regidos por el **REAL DECRETO 488/1997**, de prevención de riesgos laborales, que determinan las garantías y responsabilidades precisas para establecer un adecuado nivel de protección a los trabajadores frente a riesgos derivados de las condiciones de trabajo. En este proyecto tiene que ver sobre distancias mínimas de seguridad y salud frente a ordenadores.

3. Descripción de los equipos

3.1 Hardware

- Ordenador de sobremesa
  - Procesador: Intel Core i5-8400
  - Placa base: B360M-DS3H
  - Memoria RAM: 16GB
  - Memoria: 1TB de SSD

- Pantallas
  - Ozone 27” IPS 144HZ 1ms
  - NOC 22” 1080p 5ms

- Teclado
  - KROM Kernel TKL

- Ratón
  - Logitech G305

3.2 Software

- Windows 10
- RobotStudio 2020
- Microsoft Office 2019
- Google Chrome
- Microsoft OneDrive 365

4. Normativa

Normativa regida por el **REAL DECRETO 488/1997.**

1. **Pantalla**

- Los caracteres deberán estar bien definidos y configurados de forma clara, y tener una dimensión suficiente, disponiendo de un espacio adecuado entre los caracteres y los reglones.
- La imagen de la pantalla deberá ser estable, sin fenómenos de destello, centelleos u otras formas de inestabilidad
- El usuario de terminales con pantalla deberá poder ajustar fácilmente la luminosidad y el contraste entre los caracteres y el fondo de la pantalla, y adaptarlos fácilmente a las condiciones de entorno.
- La pantalla deberá ser orientable e inclinable a voluntad, con facilidad para adaptarse a las necesidades del usurario.
- Podrá utilizarse un pedestal independiente o una mesa regulable para la pantalla.
- La pantalla no deberá tener reflejos ni reverberaciones que puedan molestar al usuario.

2. **Teclado**

- El teclado deberá ser inclinable e independiente de la pantalla para permitir que el trabajador adopte una postura cómoda que no provoque cansancio en los brazos o las manos
- Tendrá que haber espacio suficiente delante del teclado para que el usuario pueda apoyar los brazos y las manos.
- La superficie del teclado deberá ser mate para evitar reflejos.
- La disposición del teclado y las características de las teclas deberán tender a facilitar su utilización
- Los símbolos de las teclas deberán resaltar suficientemente y ser legibles desde la posición normal del trabajo.

3. Mesa o superficie de trabajo

- La mesa o superficie de trabajo deberán ser poco reflectantes, tener dimensiones suficientes y permitir una colocación flexible de la pantalla, del teclado, de los documentos y del material accesorio
- El soporte de los documentos deberá ser estable y regulable y estará colocado de tal modo que se reduzcan al mínimo los movimientos incómodos de la cabeza y los ojos
- El espacio deberá ser suficiente para permitir a los trabajadores una posición cómoda.

4. Asiento de trabajo

- El asiento de trabajo deberá ser estable, proporcionando al usuario libertad de movimientos y procurándole una postura confortable.
- La altura de este deberá ser regulable.
- El respaldo deberá ser reclinable y su altura ajustable
- Se pondrá un reposapiés a disposición de quienes lo deseen.

DOCUMENTO III

PRESUPUESTO

En este documento analizaremos los constes de realización del proyecto. Los desglosaremos de la siguiente manera:

- Costes de amortización del material empleado
- Costes de mano de obra
- Coste total sin IVA y con IVA

1. Coste de amortización del material empleado

Este coste es la relación del tiempo empleado en un dispositivo o programa durante el proyecto con el su tiempo de vida total.

$$Coef.Amoritización = \frac{Tiempo\ del\ proyecto}{Tiempo\ de\ amortización}$$

El tiempo de realización de este proyecto han sido aproximadamente 3 meses y se ha considerado el tiempo de amortización de los dispositivos empleados de 6 años. Por otra parte, la licencia del RobotStudio (para estudiantes) es gratuita.

CONCEPTO	DESCRIPCIÓN	FABRICANTE	PRECIO (€)	COEFICIENTE AMORTIZACIÓN	TOTAL(€)
Hardware	Robot IRB 140	ABB	30.000	0,042	1260
Hardware	Ordenador sobremesa	ASUS	800	0,042	33,6
Software	Microsoft Windows 10	Microsoft	50	0,042	2,1
Software	Office Profesional 2010	Microsoft	90	0,042	3,78
Software	RobotStudio	ABB	-----	-----	-----
COSTE TOTAL DE AMORTIZACIÓN DEL MATERIAL EMPLEADO					1299,48

2. Coste de la mano de obra

En este punto se contabilizarán todos los costes de organización, investigación, diseño y desarrollo del software del proyecto.

ORGANIZACIÓN DEL PROYECTO			
TAREA	HORAS	PRECIO (h)	TOTAL (€)
Planificación	16	25	400
Investigación y documentación	50	25	1250
DISEÑO			
Modelado de la célula robotizada	30	25	750
Desarrollo del Software en RobotStudio	30	25	750
Programación RAPID	10	25	250
Simulación en el laboratorio	24	25	600
COSTE TOTAL DE LA MANO DE OBRA			**4000**

3. Coste total

Los precios de los apartados anteriores han sido sin tener en cuenta el IVA. En este apartado haremos un resumen del coste total con IVA y sin IVA.

CONCEPTO	COSTE (€)
Coste de amortización del material empleado	1299,48
Coste de mano de obra	4000
Total sin IVA	**5299,48**
IVA (21%)	1112,9
Coste total del proyecto	**6412,38**