UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL




# "Programación de robots ABB para la realización de tareas de producción automatizada."

TRABAJO FIN DE GRADO

Septiembre - 2021

AUTOR: Francisco José Martínez Peral

TUTOR/ES: Carlos Pérez Vidal

Jorge Borrell Mendez

## AGRADECIMIENTOS

En primer lugar, agradecer a mis padres, por todo el apoyo incondicional que dan día a día, y a mi familia y amigos, por estar siempre a mi lado.

Por supuesto, agradecerle a Carlos Pérez Vidal la oportunidad de poder realizar este proyecto y ser mi tutor, por toda la ayuda e implicación que ha dedicado en el proyecto. De igual manera, agradecer a Jorge Borrell Mendez por ser co-tutor en el proyecto y por toda la ayuda y conocimientos compartidos.

Agradecer Universidad Miguel Hernández por su colaboración y poner a mi disposición los programas necesarios para realizar este proyecto, así como la licencia y un servidor para la utilización de la licencia del software de RobotStudio.

## RESUMEN

El presente trabajo se basa principalmente en dos partes. La primera, pretende diseñar y programar una estación robótica la cual realiza un proceso de Pick&Place de plantillas de zapatos y su posterior clasificación. La segunda parte , se basa en la programación de un programa usando Python para la obtención de datos a partir de archivos de AutoCAD y su envío directamente al robot con lo que conseguiremos una mayor automatización del proceso.

Primero, realizaremos un estudio del estado del arte actual en el uso de robots paralelos, los cuales han sido los elegidos para este proyecto. El siguiente paso será realizar un breve tutorial del funcionamiento de RAPID y RobotStudio. Adquiridos estos conocimientos, se diseñará la estación virtual mediante RobotStudio. Dicha estación realizará el proceso de corte de planchas de un material mediante una máquina CNC y a continuación, gracias a una cinta transportadora, estas plantillas serán transportadas hasta la posición de recogida, donde los robots, suspendidos en un pórtico sobre la cinta, serán los encargados de realizar la tarea de Pick&Place y su posterior clasificación y colocación de las plantillas. Finalmente, nos centraremos en la simulación de la estación, con el análisis y evaluación del resultado obtenido para comprobar su correcto funcionamiento.

En la segunda parte del proyecto, pasaremos a diseñar un programa en Python para leer los archivos obtenidos de una maquina CNC con el diseño del corte realizado, proporcionados por la empresa Comelz y con los que se obtendrán los datos de posición, rotación, y el nombre de cada uno de los modelos de plantillas presentes en el fichero. Estos datos serán enviados mediante un socket al robot en RobotStudio. Con estos datos, el robot será capaz de realizar la tarea de Pick&Place de una forma más automatizada y su posterior clasificación.



**Palabras clave:** ABB, controlador, estación robotizada, proceso industrial, robot paralelo, programación RAPID, RobotStudio, simulación, CNC, plantillas de zapatos, AutoCAD, Python, socket.

## ABSTRACT

This work is mainly based on two parts. The first part aims to design and program a robotic station which performs a Pick&Place process of shoe insoles and their subsequent classification. The second part is based on the programming of a program using Python to obtain data from AutoCAD files and send them directly to the robot, thus achieving greater automation of the process.

First, we will carry out a study of the current state of the art in the use of parallel robots, which have been chosen for this project. The next step will be to carry out a brief tutorial on the operation of RAPID and RobotStudio. Once this knowledge has been acquired, the virtual station will be designed using RobotStudio. This station will carry out the process of cutting sheets of a material using a CNC machine and then, thanks to a conveyor belt, these templates will be transported to the pick-up position, where the robots, suspended on a gantry above the belt, will be in charge of carrying out the Pick&Place task and their subsequent classification and placement of the templates. Finally, we will focus on the simulation of the station, with the analysis and evaluation of the result obtained to check its correct operation.

In the second part of the project, we will design a Python program to read the files obtained from a CNC machine with the design of the cut made, provided by the company Comelz and with which we will obtain the position data, rotation, and the name of each of the template models present in the file. This data will be sent via a socket to the robot in RobotStudio. With this data, the robot will be able to perform the Pick&Place task in a more automated way and its subsequent classification.



**Keywords:** ABB, controller, robotic station, industrial process, parallel robot, RAPID programming, RobotStudio, simulation, CNC, shoe inserts, AutoCAD, Python, socket.

# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1. INTRODUCCIÓN

## 1.1. Motivación

Hoy en día, la mayoría de los procesos industriales están en vía de ser automatizados. Con la automatización de los procesos industriales, las empresas consiguen una mejora en la optimización de los tiempos de producción y un mayor rendimiento en economía de costes.

Los procesos que involucran grandes producciones tanto diarias como anuales, al mismo tiempo que una gran exigencia de precisión y calidad son candidatos para su automatización mediante la robótica.

Este proyecto se ha basado en la automatización de uno de esos procesos, el corte de un material mediante una máquina de control numérico computarizado o CNC es muy común en grandes establecimientos industriales, pero también en talleres de pequeña y mediana envergadura.

Con esta automatización de este proceso se quiere conseguir que principalmente mejorar la productividad y la reducción de costes, pero también se persigue el liberar a operarios de realizar tareas repetitivas y peligrosas y con las que con robots serían más rápidas y precisas.

## 1.2. Objetivos

El presente trabajo tiene como objetivo principal la automatización del proceso de recogida y clasificación de piezas que han sido cortadas previamente por una máquina de control numérico computarizado o CNC.

Para ello, desarrollaremos un prototipo funcional inteligente con el que podremos pronosticar el resultado del proceso, el cual consiste en una estación basada en robótica de una línea de corte de plantillas para zapatos mediante una maquina CNC.

Con el proceso se quiere conseguir que los robots sean capaces de realizar una tarea de Pick&Place de las plantillas, al mismo modo que estas son clasificadas por su tamaño.

Al mismo tiempo, se irán adquiriendo conocimientos y habilidades necesarios para poder diseñar y programar una estación de Pick&Place mediante robots con el software de simulación de RobotStudio.

Una vez que se haya podido programar la estación consiguiendo los objetivos planteados, pasaremos a realizar un trabajo de investigación con el que se conseguirá obtener desde un archivo de AutoCAD, los datos de las posiciones y rotación de las plantillas, el cual es proporcionado por la máquina CNC. Mediante y programa de Python haremos que estos datos obtenidos sean enviados al robot mediante un socket entre la estación y el código de Python.

Con este proceso, conseguimos que el proceso se automatice aún más, ya que de esta forma las posiciones son enviadas desde el archivo obtenido de la CNC y se le comunique al robot.

### 1.3. Resumen del trabajo

La realización de este proyecto se podría decir que se ha ido realizando en diferentes fases, desde un planteamiento inicial hasta la simulación de la estación y posterior programación en Python.

En primer lugar, se pensó en el diseño de la estación, donde se acordó la colocación de una maquina CNC, la cual fue aportada por Carlos Pérez. Una cinta transportador que moviera las piezas hasta el lugar donde se encuentran los robots para realizar la tera de Pick&Place.

En cuento a los robots, era imprescindibles que se trataran de unos robots paralelos, los cuales tienen las características idóneas para realizar este tipo de tarea. Dichos robots debían de estar situados encima de la cinta transportadora, por lo que se diseñó un pórtico donde fueran colocados.

Las plantillas serán colocadas por los robots en una mesas situadas a los lateras de la cinta transportadora, dentro del espacio de trabajo de los robots.

Una vez terminado el diseño de la estación, el siguiente paso era comenzar con la programación de los movientes de los robots. Se utilizarán diferentes sensores, mediante los cuales junto con las señales de entrada/salida se consigue hacer que los robots empiecen a moverse en un momento determinado.
Se definirán la posición y rotación de cada una de las plantillas cuando la plancha donde se encuentran se encuentre en el lugar de recogida de los robots. De esta forma, las posiciones y rotaciones son definidas a mano, con el programa que realizaremos con Python, estos datos serán obtenidos del archivo creado en la maquina CNC.

Finalmente, una vez diseñada la estación y programado en RAPID los movimientos de los robots, pasaremos a realizar la simulación de la estación. Una vez simulada, realizaremos un estudio del resultado obtenido para comprobar que el funcionamiento es el correcto.

La segunda parte del proyecto se trata de realizar un programa en Python con el que se puede extraer datos de un archivo de AutoCAD, y esos archivos sean enviado al robot directamente en RobotStudio.

En esta fase del proyecto, se realiza un estudio del código del programa de Python, con una explicación de cada una de las funciones usadas y que resultado se obtiene de ellas.

Primero, intentaremos crear un script en AutoCAD con el que podamos extraer datos de una archivo .dxf donde se encuentra la forma en que ha sido cortada la plantilla. Desde Python seremos capaces de iniciar AutoCAD, abrir el archivo y ejecutar el script que guardará los datos en un archivo de texto. A partir del archivo de texto, deberemos de almacenar los datos que debamos de mandarle al robot en otro archivo de texto que crearemos en la carpeta HOME, dentro del controlador del robot.

Desde RAPID podremos acceder a dichos archivos que se habrán creado tras usar un socket con el que comunicamos Python con RAPID y con el que le decimos al robot que puede obtener los datos porque ya se ha creado el archivo.

Por último, veremos el resultado que se obtiene al simular el proceso completo.
Los programa utilizados durante el proyecto son los siguientes:

- ABB – RobotStudio 2020, versión de RobotWare 6.11.01.
- Autodesk AutoCAD 2020 versión para estudiantes.
- Visual Studio Code para la programación en Python.

### 1.4. Estructura de la Memoria del TFG

La memoria del presente Trabajo Fin de Grado está distribuida en ocho capítulos, en los que se explica en detalle su ejecución, y de los que vamos a dar una breve explicación en las siguientes líneas:

- **Capítulo 1. Introducción:** Se expone la motivación y el contexto del presente trabajo, los objetivos que sigue, un resumen y la estructura de la memoria para una mejor visión general de sus contenidos.

- **Capítulo 2. Estado del arte:** Se analiza el estado del arte, con una revisión de las herramientas y métodos aplicados en la industria, y los diferentes modelos de robots paralelos en el mercado.

- **Capítulo 3. Metodología:** Se realiza una pequeña introducción al programa de software RobotStudio ABB versión 6.11.01, con el aprendizaje y creación de una estación en un entorno de simulación. Donde se incorpora un robot y su controlador, definiendo un sólido, la herramienta de soldadura, objetos de trabajo, objetivos y trayectorias. Además, se realiza la sincronización de los datos del programa, donde aprendemos el lenguaje de programación textual RAPID, la creación de componentes inteligentes (SC), señales de E/S y la programación de la lógica de la estación. Finalmente, se muestra cómo simular la estación y su grabación para la reproducción posterior.

- **Capítulo 4. Implementación de un proceso de recogida de plantillas de zapatos en una línea automatizada con maquina CNC:** En la estación vamos a implementar el proceso de Pick&Place y clasificación de las plantillas, definiendo cada una de las posiciones y movimientos de los robots.

- **Capítulo 5. Desarrollo de la comunicación AutoCAD-Python-RobotStudio:** En este apartado veremos el programa realizado en Python con el que se obtendrán los datos de las plantillas desde un archivo .DXF de AutoCAD, su envío a RobotStudio para que sean leídos, cree los movimientos del robot y realice la tarea de Pick&Place .

- **Capítulo 6. Resultados:** Al finalizar la programación de la estación, se ha realizado una de simulación de esta, donde se puede observar por pantalla el correcto funcionamiento del sistema automatizado creado y se ha grabado para poder reproducirlo en cualquier momento.

- **Capítulo 7. Conclusiones y Líneas futuras de trabajos.** Se exponen las conclusiones del proyecto, comparándolas con los objetivos marcados. Además, se proponen algunas mejoras y nuevas líneas de investigación para fututos trabajos.

- **Capítulo 8. Bibliografía:** En este capítulo se indican todas las referencias bibliográficas utilizadas en este trabajo, ordenadas con código para su cita en la memoria, así como las páginas web y videotutoriales usados durante el proyecto.

- **Capítulo 9. Anexos:** En este último capítulo se añade información complementaria, como lo son los planos de los objetos diseñados en AutoCAD, las señales de entrada/salida utilizadas en las estaciones, los códigos d ellos programas creados y las hojas de características de los robots utilizados en las estaciones.

# 2. ESTADO DEL ARTE

## 2.1. Introducción

El actual entorno industrial es muy cambiante por lo que es necesario un cambio constante en él para conseguir un producción más rápida, económica y versátil. La robótica es una de las principales soluciones utilizadas por las empresas para alcanzar estos objetivos.

## 2.2. Historia de la robótica

La primera vez que la palabra robot fue usada fue en el año 1921, cuando el escritor checo Karel Capek estrenó en el teatro nacional de Praga su obra "Rossum's Universal Robot". Su origen proviene de la palabra eslava "*robota",* que se refiere al trabajo realizado de una manera forzada.

Posiblemente el término hubiera desaparecido si no hubiese sido por los escritores de ciencia ficción. Fue el escritor americano, de origen ruso, Isaac Asimov (1920-1992) el máximo impulsor de la palabra robot.

En octubre de 1945 publicó en la revista "*Galaxy Science Fiction"* una historia en la que por primera vez enunció sus tres "*Leyes de la robótica*":

1. Un robot no puede perjudicar a un ser humano, ni con su inacción permitir que un ser humano sufra daño.
2. Un robot ha de obedecer las órdenes recibidas de un ser humano, excepto si tales órdenes entran en conflicto con la primera ley.
3. Un robot debe proteger su propia existencia mientras tal protección no entre en conflicto con la primera o segunda ley.

Se le atribuye a Asimov la creación del término "robotics" (robótica), ya que desde su obra literaria, ha contribuido a la divulgación y difusión de la robótica.

Figura 2.1: Isaac Asimov

En 1956, George C. Devol y Joseph F. Engelberger comienzan a trabajar en la utilización industrial de las máquinas, fundando la empresa Consolidated Controls Corporation, que más tarde se convirtió en Unimation ( Universal Automation).

Hacia el 1960 instalan su primera máquina Unimate en la fábrica de General Motors de Trenton, Nueva Jersey, en una aplicación de fundición por inyección [5].

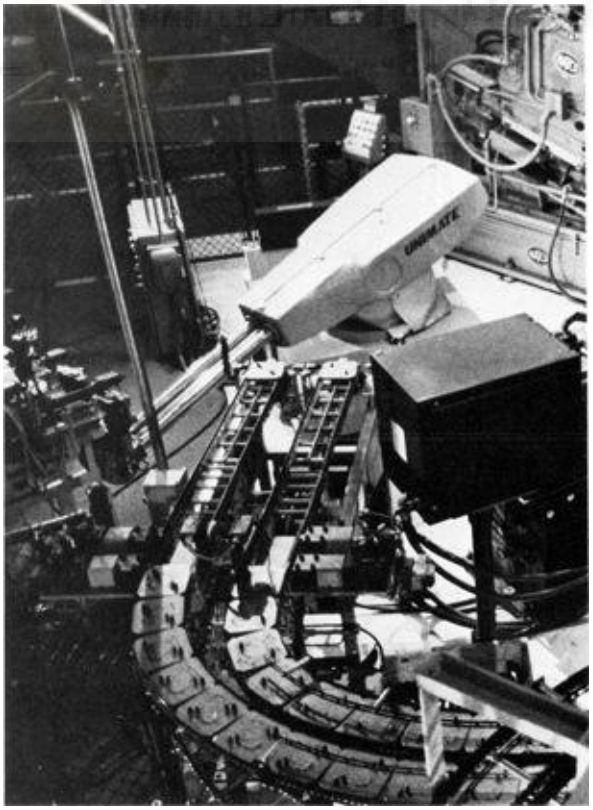


Figura 2.2: Robot Unimate en la planta de General Motors. (1961)

A partir de los años 80, se produjo una explosión en el desarrollo de la robótica, considerando esa década como el inicio de la Era Robótica, donde su fabricación y venta aumentaron en un 80%.

A partir de ella, surgió la que hoy conocemos como robótica inteligente, en la que con el uso de la tecnología adecuada y la inteligencia artificial, los robots han ido ganando independencia en sus tareas y cooperan con el ser humano, tomando decisiones en tiempo real.

La robótica inteligente ha supuesto una evolución de la robótica industrial gracias al uso de nuevas tecnológicas como lo son el Big Data, el IoT y los sistemas de visión e inteligencia artificial, como se redacta en [12].

### 2.3. Tipos de robots industriales

Tal y como se desarrolla en [13], tomamos como referencia a uno de los organismos internacionales más reconocido como lo es la Federación Internacional de Robótica (IFR, International Federation of Robotics), el cual define las normas ISO.

Según la norma ISO 8373, nos referimos a un robot industrial como un manipulador multifuncional, controlado automáticamente y que pueden cumplir diferentes tareas, convirtiéndolos en una herramienta indispensable a la hora de optimizar los procesos y ahorrar en gastos. También, se definen como reprogramables, ya que pueden ser calibrados constantemente para la tarea de que tengan asignada.

Lo que caracteriza a un robot industrial son:

- Los grados de libertad, es decir, el número de articulaciones que tiene.
- La zona de trabajo disponible para operar el robot.
- La carga que deben de sostener durante la tarea.
- La velocidad a la que deben de realizar las acciones.
- El nivel de programabilidad.

A continuación, se va a distinguir entre los distintos tipos de robots industriales que hay. En la figura 2.3 se puede ver una representación de los distintos tipos de robots que hay en el mercado:


Figura 2.3: Tipos de robots industriales

- **Robot cartesiano:** se trata de un robot de coordenadas cartesianas, los cuales funcionan en tres ejes ( x,y,z ) moviéndose en línea recta y formando ángulos también rectos. Ofrecen una gran precisión, rentabilidad y facilidad en su programación y uso, por lo que son idóneos para movimientos lineales de gran precisión.

- **Robot SCARA:** la palabra SCARA proviene de las siglas Selective Compliant Assembly Robot Arm. Son conocidos por sus rápidas tiempos de trabajo, su elevada repetitividad, gran capacidad de carga y su amplio campo de aplicación.

- **Robot angular o antropomórfico:** este robot tiene 3 articulaciones de posicionamiento y simula los movimientos de un brazo humano. Podemos encontrar robots de 5, 6 o 7 ejes.

- **Robot Colaborativo:** se trata de un robot antropomórfico, pero de dimensiones y peso menor al descrito anteriormente. Son robots más manejables, de fácil programación y no necesitan medidas de portación específicas.

- **AGV:** (Automatic Guided Vehicle) son robots que se identifican como vehículos autónomos cuya aplicabilidad va muy ligada a la logística en la empresa.

## 2.4. Robots Delta

Un robot delta está formado por un efector final con n grados de libertad, y por una base fija, que están unidos entre sí por al menos dos cadenas cinemáticas independientes. La actuación es producida por n actuadores simples, y cuando estos están bloqueados, el manipulador permanece en su posicion. La idea básica de robot delta es el uso de paralelogramos, que permite que un eslabón de salida permanezca con una orientación fija con respecto a un eslabón de entrada, tal y como se desarrolla en [8], donde se puede observar diferentes análisis basándose en robots delta.

Los robots delta, o también llamados robots Spider, permiten realizar movimientos delicados y precisos. Son robots que trabajan a una velocidad elevada y tiene un bajo consumo [12]. Los robots paralelos presentan un gran potencial en operaciones de Pick&Place de alta velocidad en muchos sectores como el alimentario, el electrónico y muchas otras industrias ligeras [6].

Tal y como se desarrolla en [7], la operación de recogida y colocación es una tarea frecuente en la mayoría de las líneas de producción. En muchas aplicaciones, la operación Pick&Place se diseña considerando el hecho de que la ubicación inicial y final del objeto manipulado se encuentra en una región delimitada y el trazado del efector final se encuentra en un determinado subespacio. Uno de los principales retos es aumentar la capacidad de recogida y colocación en un tiempo especificado.

Según [14], algunos de los robots tipo Delta de las principales marcas en el mercado son las siguientes:

- **ABB** [15]

  El robot Delta de la serie IRB 360 FlexPicker de ABB está considerado uno de los robots más rápido del sector. Entre sus características cabe destacar la flexibilidad a altas velocidades, la capacidad de carga, que puede llegar a ser hasta de 8kg y el software de visión integrado.


Figura 2.4: Robot Delta IRB 360 FlexPicker de ABB

- **Fanuc** [16]

  Los robots delta de Fanuc La serie M-3iA ofrece fiabilidad, rapidez y versatilidad. Pueden llegar hasta los 12kg de carga en su modelo M-3iA/6ª con un alcance de 1350mm.

Figura 2.5: Robot delta M-3iA de Fanuc

- **Omron** [17]

  Los robots de la serie X-Delta de Omron son una de las aplicaciones más fiables del mercado y disponen de un controlador Sysmac NJ que permite controlar hasta 8 robots a la vez.

Figura 2.6: Robot delta de la marca Omron

# 3. METODOLOGÍA

## 3.1. Introducción al Programa RobotStudio

Tal y como se desarrolla en [1], el programa RobotStudio es un software que permite crear, programar y simular estaciones de robots industriales, el cual utiliza el lenguaje RAPID. Diseñado y patentado por la empresa ABB, líder a nivel mundial en el sector de la robótica.

El programa RobotStudio está basado en un VirtualController de ABB, una copia exacta del software real en el que se ejecutan sus robots en producción. Lo que permite programar los robots offline (fuera de línea) en un ordenador, sin necesidad de para la producción y realizar simulaciones muy realistas, donde se utilizan programas de robots reales y archivos de configuración idénticos a los utilizados en una instalación.

El programa proporciona herramientas que ayudan a incrementar la rentabilidad de un sistema robotizado mediante tareas de como formación, programación y optimización, sin afectar a la producción, lo que proporciona numerosas ventajas como son un arranque más rápido, incremento en la productividad, reducción de riesgos y un arranque más rápido.

En la actualidad se encuentra disponible la versión 2021.1.1 de RobotStudio, pero para la realización de este Trabajo de Fin de Grado se ha empleado RobotStudio 2020.5 y la versión 6.11.01 de RobotWare.
ABB ofrece una versión de prueba de 30 días que se debe activa para trabajar con el programa. En el presente proyecto, la Universidad Miguel Hernández de Elche ha facilitado una licencia de red, conectada mediante un servidor de licencias de red.

### *3.1.1. Creación de una estación*

Una vez iniciado el programa e instalada la licencia, el primer paso es crear una nueva estación en la pestaña “Archivo”, apartado “Nuevo” y en “Estaciones”, donde tenemos varias opciones a la hora de crear una nueva estación:

- **Solución con Estación vacía:** crea una estructura de archivos de solución con una estación vacía.
- **Solución con Estación y Controlador de robot:** crea una solución que contiene una estación y un controlador del robot, que tiene una lista con los robots ABB que podemos utilizar.
- **Estación vacía:** crea una estación vacía.


Figura 3.1: Creación de una nueva estación

Una vez creada la estación, el programa dispone de 7 pestañas en la parte superior:

- **Archivo:** Donde tenemos diferentes opciones como son guardado, abrir, nuevo, compartir, imprimir y opciones del programa.
- **Posición Inicial:** Es la pestaña principal del programa, donde se puede añadir robot, controlador, bibliotecas, coordenadas e importar geometrías. Además de realizar la programación de trayectorias, seleccionar la herramienta y el espacio de trabajo, sincronizar, movimientos del robot y herramientas gráficas.
- **Modelado:** En esta pestaña podemos crear grupos de componentes, componentes inteligentes, importar geometrías, crear sólidos, superficies y curvas, también podemos realizar operaciones de edición de CAD y crear mecanismos y herramientas.

- **Simulación:** Se pueden crear conjuntos de colisión, editar la lógica de la estación, configuración de simulación, control de la simulación, monitorizar, analizar las señales y realizar la grabación de la simulación de la estación.
- **Controlador:** En esta pestaña podemos añadir un controlador a la estación, reiniciarlo, acceder a la configuración del controlador y abrir la paleta FlexPendant.
- **RAPID:** Se utiliza para la programación del robot en leguaje RAPID.
- **Complementos:** Desde esta pestaña se pueden descargar algunos complementos e instalar paquetes como Painting, Machinig, Cutting, Palletizing.


Figura 3.2: Pestañas del programa RobotStudio

Para la realización de esta introducción vamos a cargar el robot ABB IRB 120. Para ello, desde la pestaña “Posición inicial”, pinchamos en “Biblioteca ABB” y seleccionamos el robot que queremos, como podemos ver en la figura 3.3.

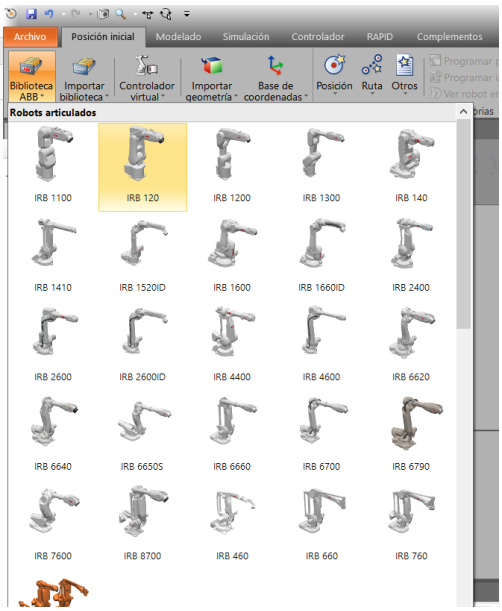



Figura 3.3: Robots de la Biblioteca ABB

Y cargamos en robot en la estación, como en la figura 3.4.

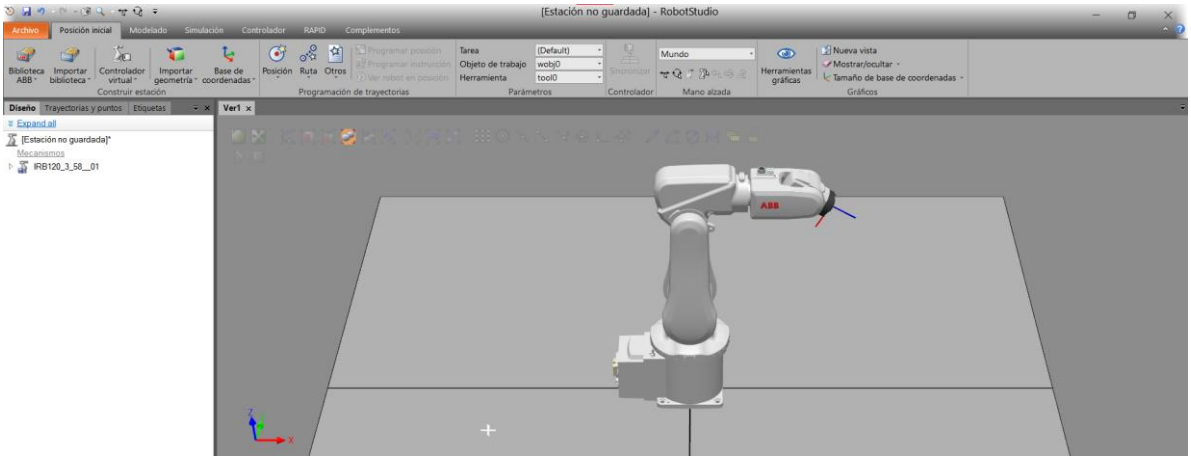



Figura 3.4: Robot ABB IRB 120.

### *3.1.2. Creación de la herramienta de un robot*

Todos los robots llevan como extremo efector una herramienta para poder realizar las determinadas tareas como pintura, soldadura, pulido o Pick&Place. Es necesario que el robot tenga una herramienta ya que con ella se programa las operaciones y se crean los objetivos, esta herramienta se encuentra situada en la muñeca del robot.

En este tutorial, vamos a crear una herramienta a través de la pestaña “Modelado”, donde podemos crear un tetraedro, cilindro, pirámide, esfera o cono, el cual usaremos. Pincharemos en “Sólido” y escogeremos “Cono”, como se muestra en la figura 3.5.

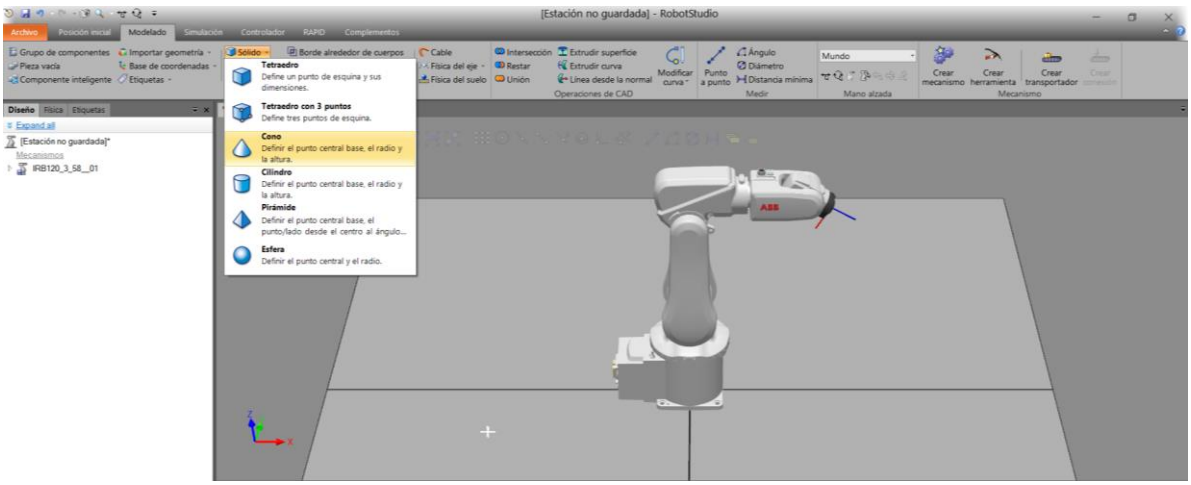



Figura 3.5: Creación de sólido (Cono)

A continuación, configuraremos los parámetros del cono con un radio de 25 mm, diámetro de 50 mm y una altura de 100 mm, y le daremos a crear. Se creará un sólido en forma de cono en la posición (0,0,0), ya que no se ha especificado una posición determinada.

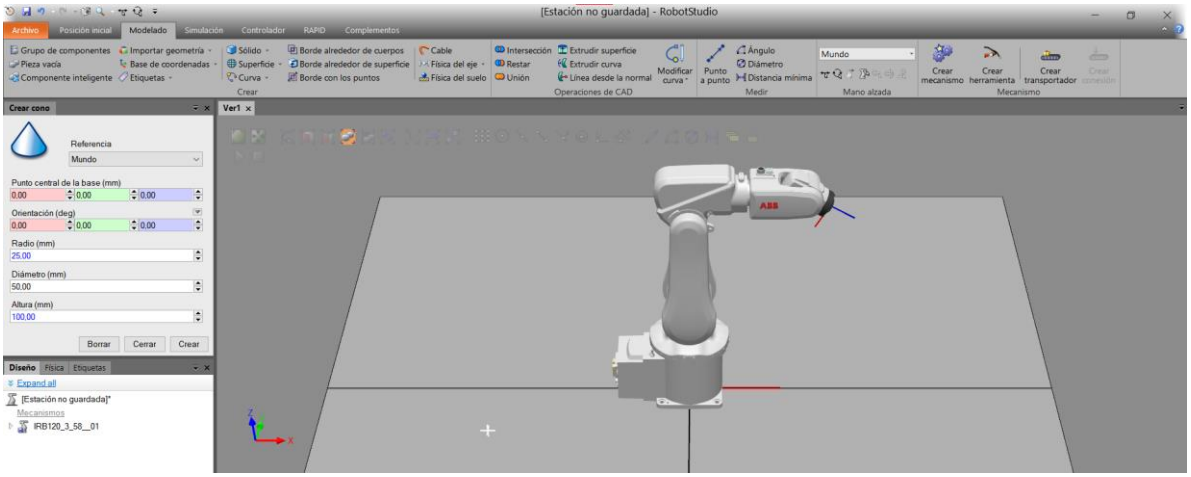



Figura 3.6: Definición de los parámetros del cono

En el siguiente paso, consiste en transformar la geometría en forma de cono creada en una herramienta, este paso es necesario para que el programa interprete la pieza como una herramienta del robot. Para hacer esto, en la pestaña “Modelado”, seleccionamos la opción “Crear herramienta” de la figura 3.7.

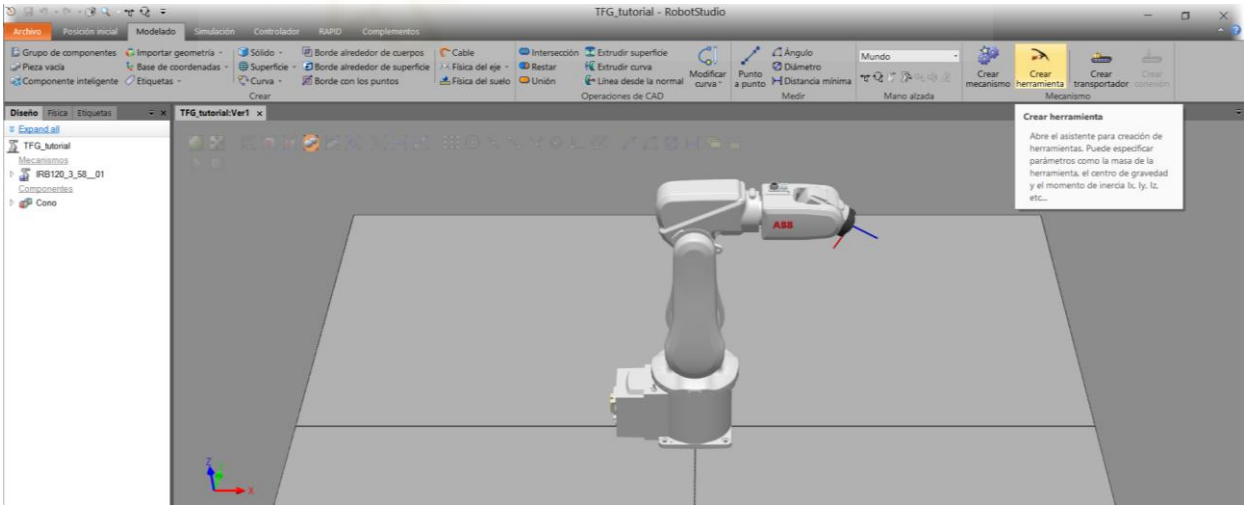



Figura 3.7: Creación de la herramienta

Lo primero que debemos de hacer es definir un nombre a la herramienta, en este caso la hemos llamado “Cono_tool”, seleccionamos que es una pieza existente “Cono”, a continuación indicamos una masa de la herramienta y si sabemos el centro de gravedad y los momentos de inercia se pueden añadir a la información de la herramienta.

El siguiente paso podemos añadir la posición y la orientación relativa al extremo de la herramienta, para que las posiciones del robot que se creen posteriormente se realicen respecto al extremo de la herramienta. Esta posición se denomina TCP (Tool Central Point), una vez definida su posición sobre la pieza, seleccionamos el boto de la flecha "→" y seleccionamos "Terminado", como podemos ver en la figura 3.8.

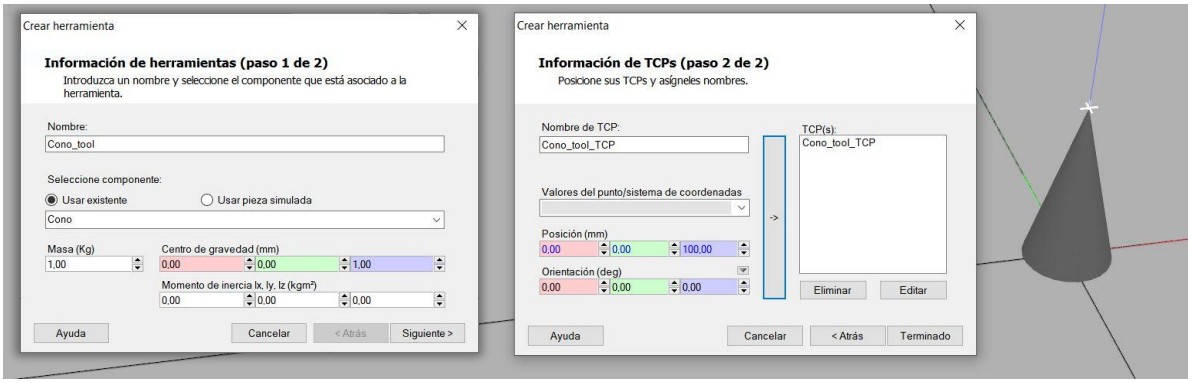



Figura 3.8: Configuración de la herramienta

Para finalizar, lo que debemos de hacer es conectarla al extremo del robot, para realizar esto podemos seleccionar la herramienta "Cono_tool" y arrastrarla al robot IRB120, a continuación, seleccionamos "Si" cuando pregunta si queremos actualizar la posición de la herramienta y automáticamente, la herramienta se encuentra conectada al robot, como podemos ver en la figura 3.9.

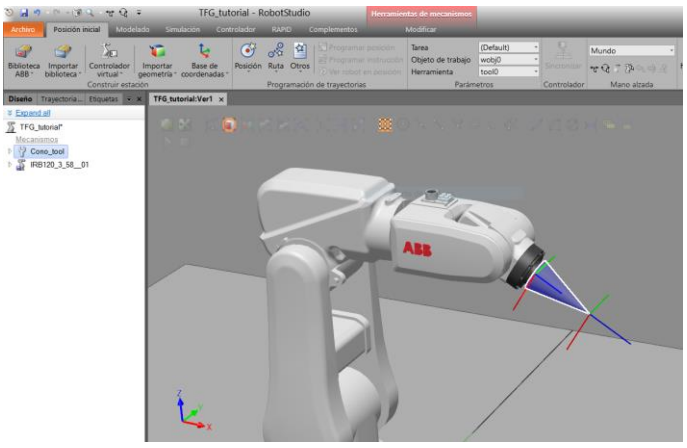



Figura 3.9: Robot IRB120 con herramienta conectada

También, tenemos la opción de "Importar geometría" en la pestaña de "Posición inicial", con la que podemos añadir una herramienta creada en otro programa, como en la figura 3.10.

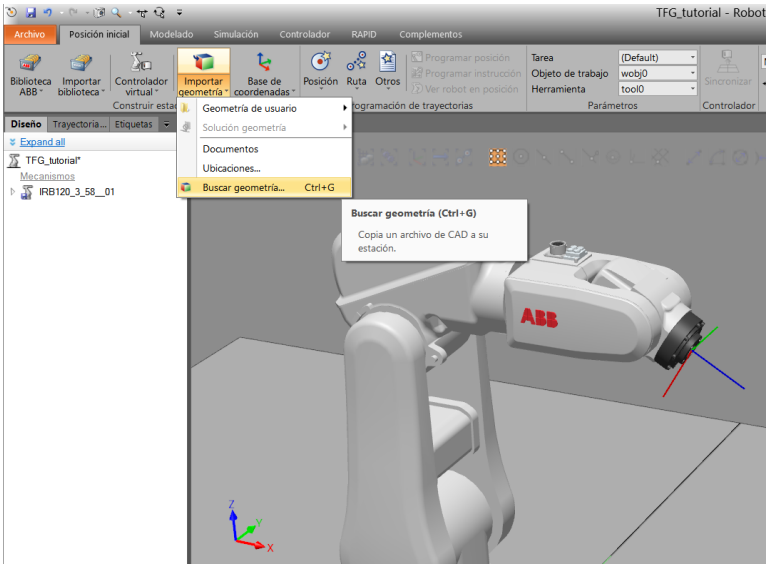



Figura 3.10: Importar geometría

Otra opción es añadir una herramienta de la biblioteca del programa, donde podemos encontrar algunas herramientas. En la pestaña de “Posición inicial”, haciendo clic en “Importar Biblioteca” y luego abriendo el desplegable de “Equipamiento”, vemos que hay variedad de herramientas y objetos que podemos añadir a nuestra estación. Podemos ver en la figura 3.11 la biblioteca de RobotStudio.

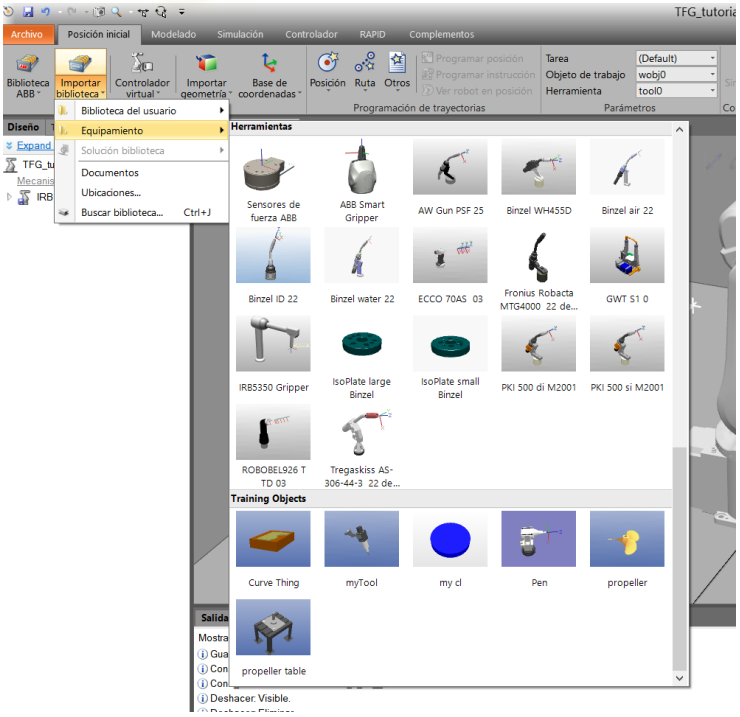



Figura 3.11: Biblioteca RobotStudio (Equipamiento)

Para este tutorial, hemos añadido la herramienta de soldadura de la biblioteca del programa llamada “RA_MTG4000_22_2” y la hemos conectado al robot como hemos realizado en el paso anterior, arrastrando la herramienta al robot. En la figura 3.12 podemos ver el robot IRB120 con la herramienta conectada en su extremo.

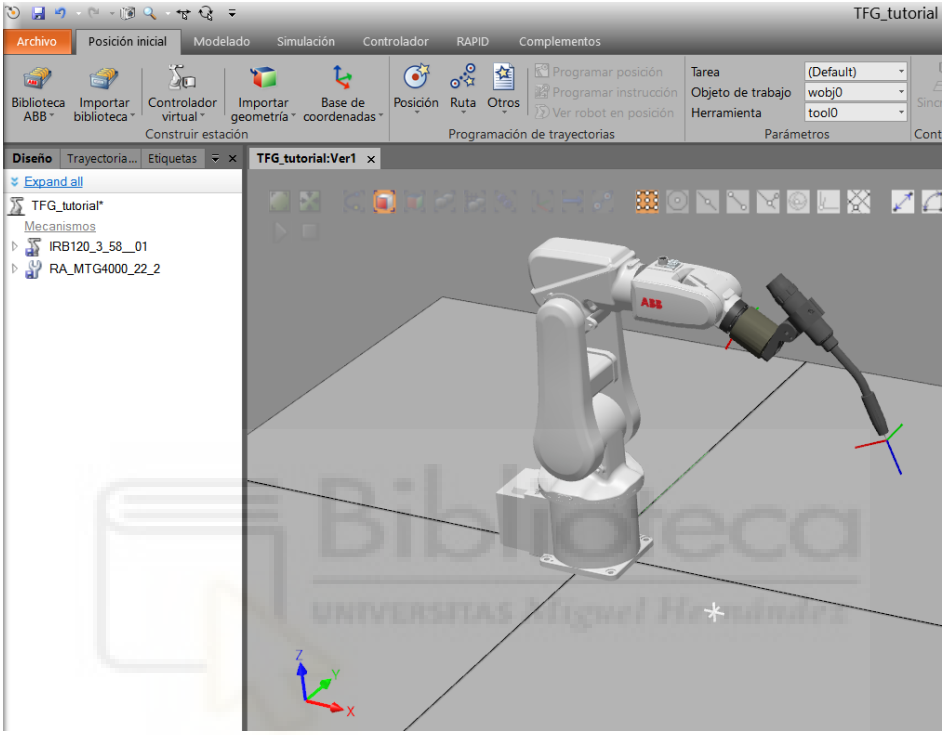



Figura 3.12: Robot IRB120 con herramienta de soldadura

### *3.1.3. Creación del controlador de un robot*

De momento, tenemos la herramienta conectada al robot, pero aún no podemos ni programar trayectorias ni realizar movimientos con el robot. Para realizar esto, es necesario crear un controlador asociado al robot y dotarlo de inteligencia.

Para añadir el controlador, en la pestaña “Posición Inicial”, en “Controlador Virtual” seleccionamos “Desde diseño”, como se puede ver en la figura 3.13.

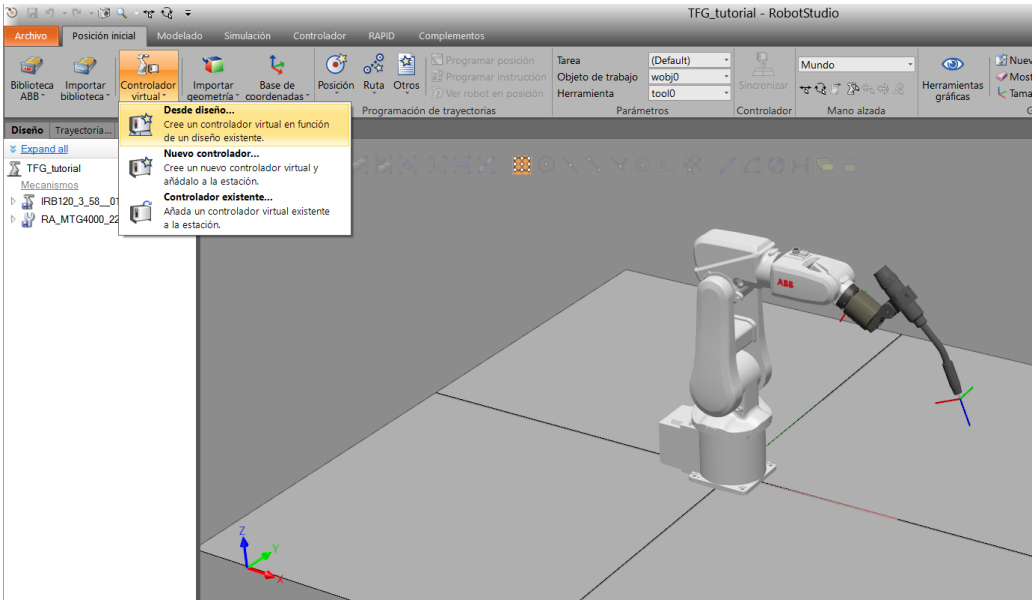



Figura 3.13: Creación de un controlador desde diseño

A continuación, le damos un nombre al controlador y pulsamos en “Siguiente”, luego seleccionamos el mecanismo al que estamos creando el controlador, en este caso “IRB120_3_58__01” y pulsamos de nuevo en “Siguiente”, podemos ver este proceso en la figura 3.14.

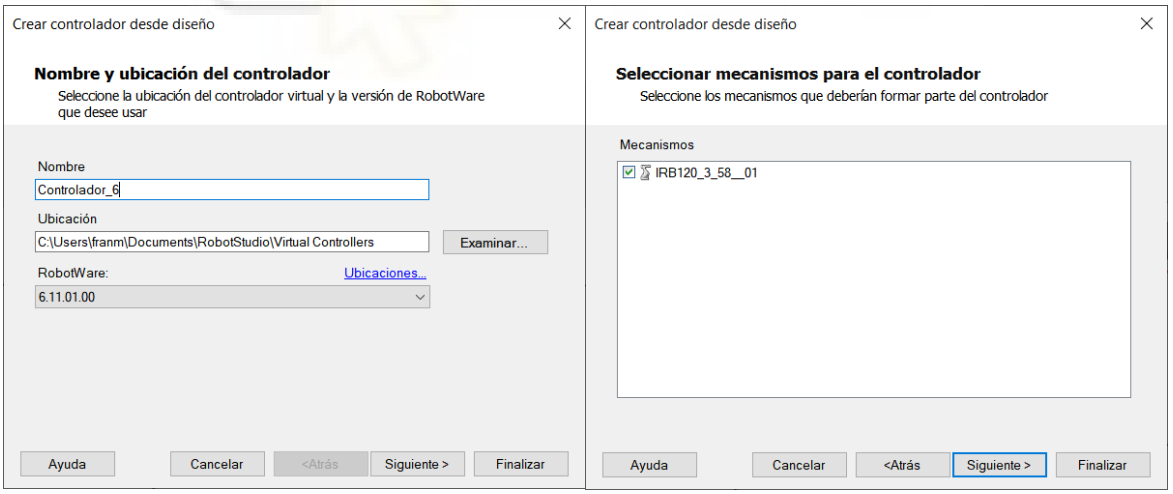



Figura 3.14: Creación del controlador

Para acabar de configurar el controlador podemos cambiar las opciones que viene por defecto en “Opciones…”, como por ejemplo podemos cambiar el idioma en el apartado “Default Language” a Español, luego le damos a “Aceptar” y por último a “Finalizar”, como podemos ver en la figura 3.15.

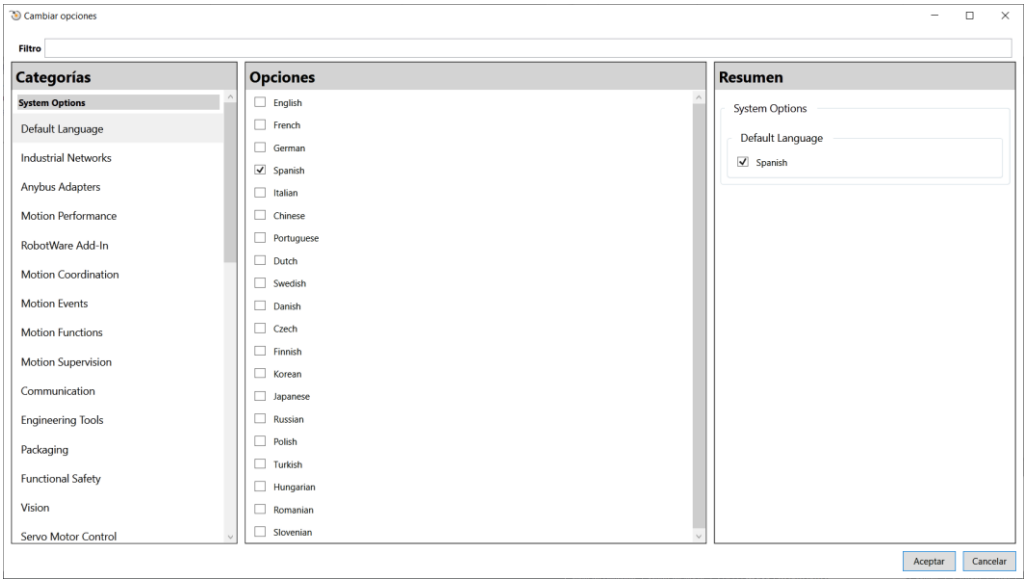



Figura 3.15: Cambiar opciones del Controlador

Una vez realizado los pasos previos, ya tenemos creado el controlador del robot.

### *3.1.4. Creación de planos de trabajo y posiciones*

Antes de crear el plano de trabajo y los objetivos, necesitamos un objeto donde se realizará la trayectoria. La pieza que vamos a crear va a ser un sólido, en concreto se tratará de un cilindro con radio de 100mm y una altura de 200mm. Lo colocaremos en una posición que se encuentra a 550mm en el eje X respecto de la base del robot (0,0,0), donde la herramienta del robot podrá alcanzar sin problemas al cilindro. Le damos a “Crear” y le cambiamos el nombre a “Cilindro”.

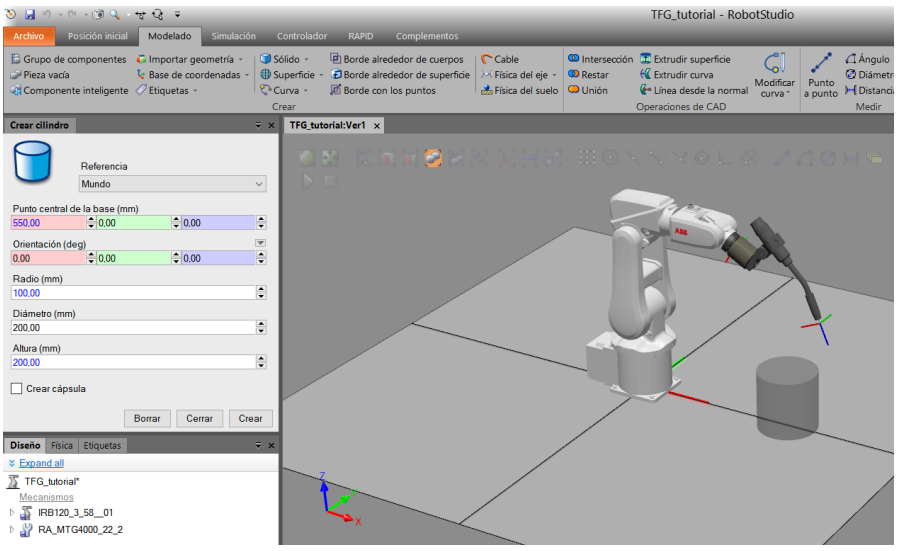



Figura 3.16: Creación de un Cilindro.

Ahora debemos de crear un objeto de trabajo (Work Object), para ello, en la pestaña “Posición inicial”, seleccionamos “Otros” en el apartado “Programación de trayectorias” y después, en “Crear objeto de trabajo”, como se puede ver en la figura 3.17.

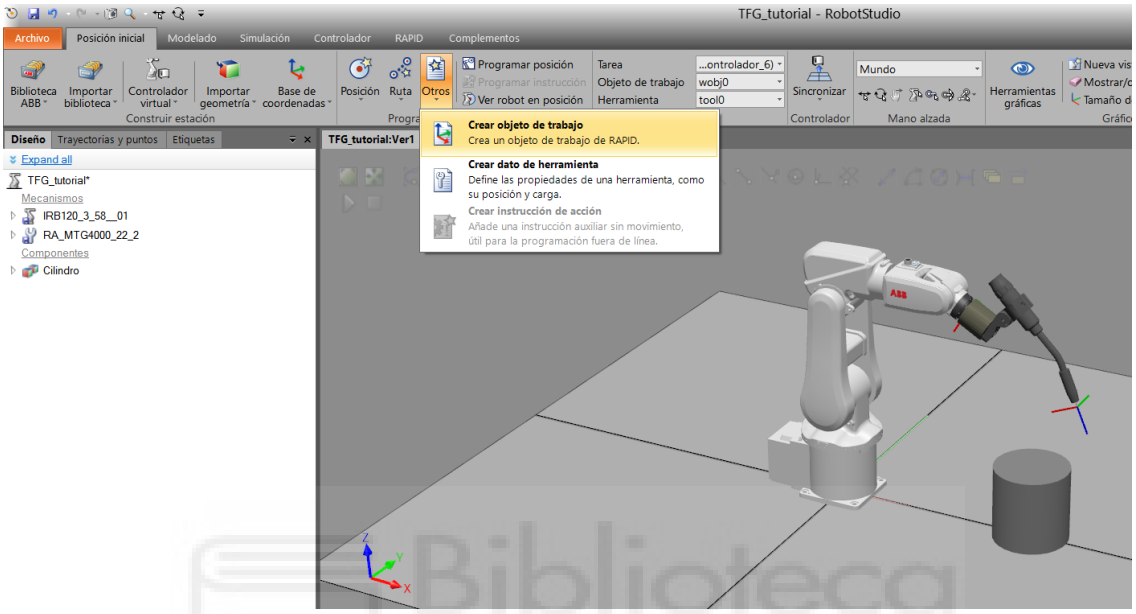



Figura 3.17: Creación de un Objeto de Trabajo

El nombre del objeto de trabajo lo dejamos por defecto “Workobject_1” (se puede cambiar de nombre), en el apartado Sistema de coordenadas usuario, seleccionamos Posición X,Y,Z y podemos cambiar las coordenadas por defecto, para ello pinchamos en la casilla roja (coordenada X) y seleccionamos el punto que donde queremos que se encuentre el punto de trabajo.

En este caso, vamos a seleccionar el centro de la cara superior del cilindro (anteriormente debemos seleccionar en los iconos tanto de Selección de pieza como de Ajustar a centro), como podemos ver en la figura 3.18. Finalmente, pinchamos en “Accept” y después en “Crear”, y ya tendremos el objeto de trabajo creado.

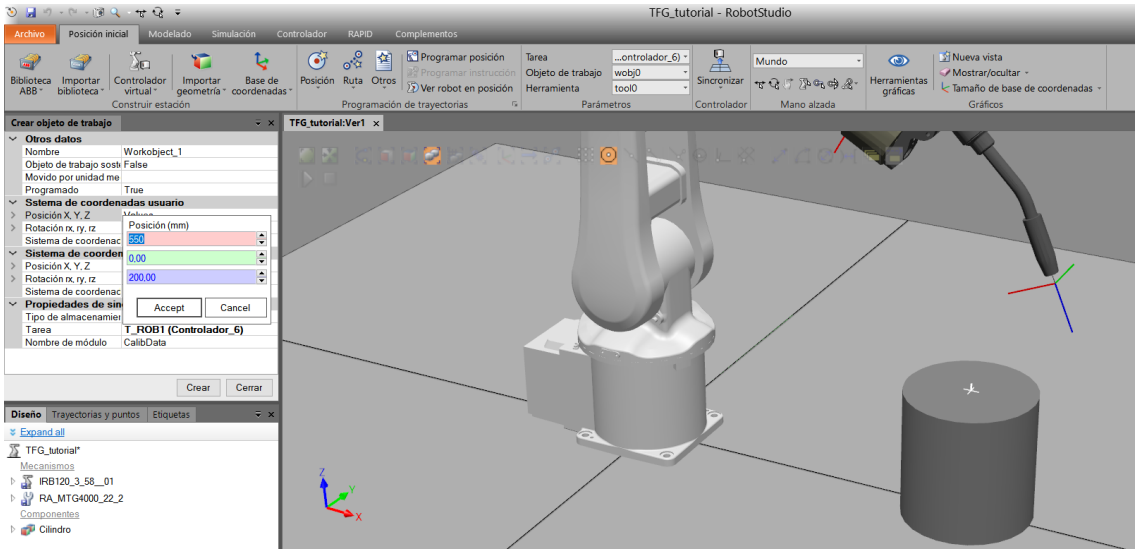



Figura 3.18: Colocación del objeto de trabajo

El siguiente paso es el de crear los objetivos, pero antes de ello debemos de cambiar el Objeto de trabajo y la herramienta. Para hacer esto, debemos de ir a la pestaña “Posición inicial”, y en “Parámetros” seleccionarlos, así, los objetivos que creemos pertenecerán a dicho objeto de trabajo y se utilizará la herramienta seleccionada.

Una vez que los hemos seleccionado, procedemos a crear las posiciones, en la pestaña “Posición inicial”, “Posición” y seleccionamos “Crear objetivo”, como vemos en la figura 3.19.

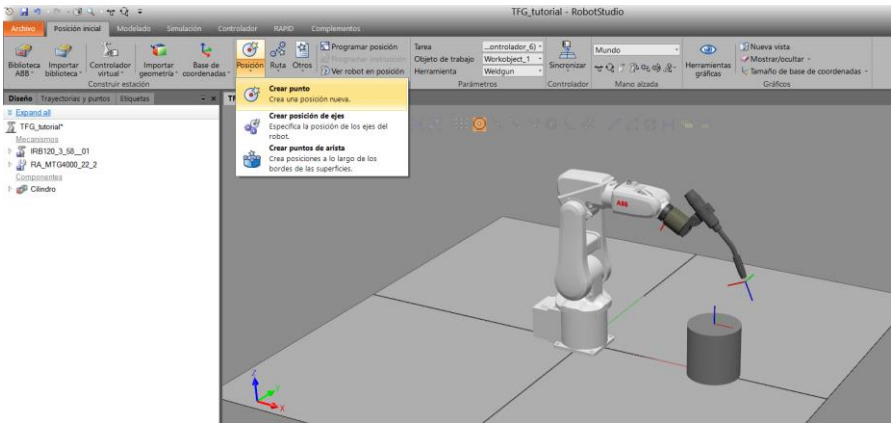



Figura 3.19: Crear punto

A continuación, en la figura 3.20, creamos cuatro puntos en dos cuadrantes de la cara superior del cilindro, es necesario seleccionar la casilla roja de Posición (coordenada X) para poder coger la posición de los puntos. Para finalizar, pinchamos en crear para crear los puntos.

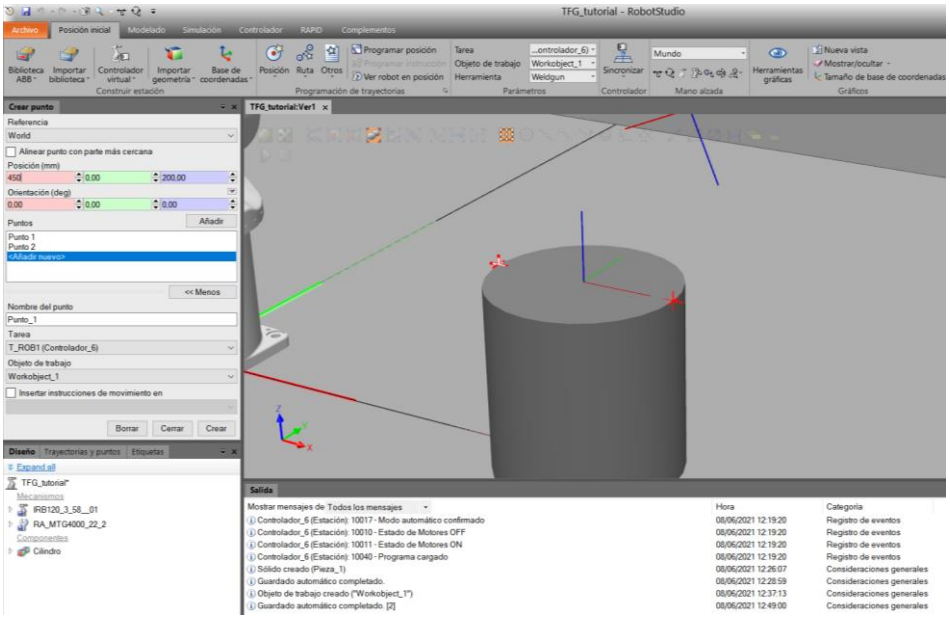



Figura 3.20: Creamos dos puntos como objetivos

También, crearemos una posición inicial, para así situar la herramienta en una posición fuera de los puntos objetivo. Esta posición inicial la crearemos en el objeto de trabajo “wobj0”.

### *3.1.5. Creación de trayectorias*

Una vez creado el conjunto de puntos, vamos a visualizar la posición de la herramienta “RA_MTG4000_22_2” en los puntos creados. La orientación que viene por defecto puede que no sea la idónea o que no pueda ser alcanzada por el robot. Escogemos el primer punto creado como “Target_10”, hacemos clic con el botón derecho del ratón y seleccionamos “Ver herramienta en la posición” y pinchamos en “RA_MTG4000_22_2”, ahora la herramienta se visualiza en “Target_10” que hemos creado anteriormente.

Como se puede ver en la figura 3.21, este punto con la orientación actual de la herramienta no puede ser alcanzado por el robot, porque si hacemos clic en “Ver robot en posición”, aparece un mensaje de que el punto Target_10 se encuentra fuera de alcance.

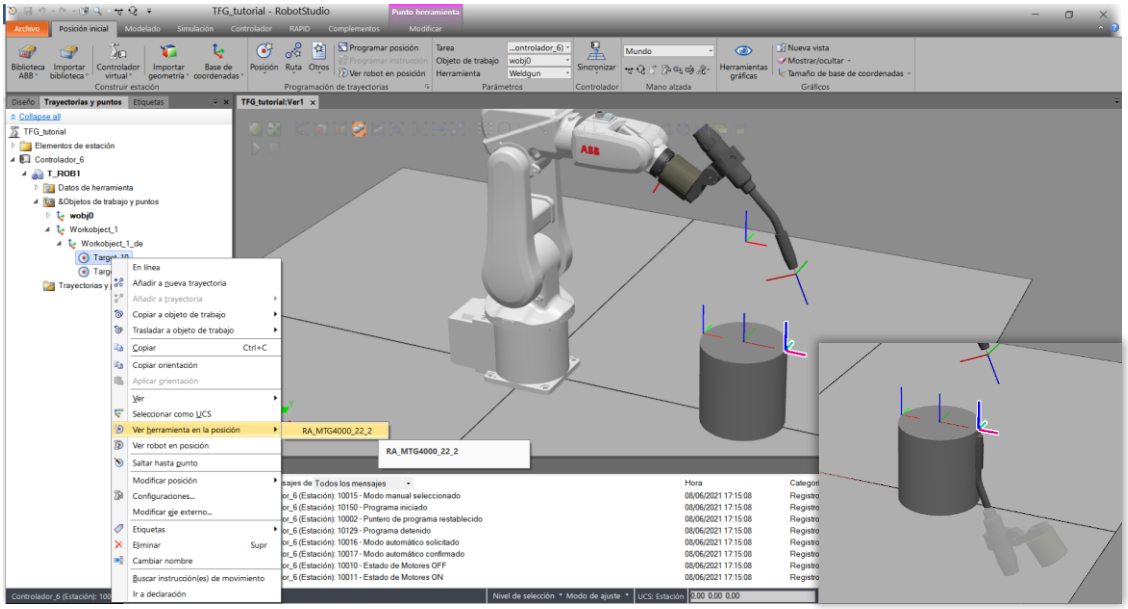


Figura 3.21: Ver herramienta en posición

Para colocar la herramienta con la orientación con la que pueda ser alcanzada por el robot, será necesario realizar un giro al sistema de eje de coordenadas. Hacemos clic derecho con el ratón sobre “Target_10” otra vez, seleccionamos “Modificar posición” y pinchamos en la opción girar como aparece en la figura 3.22.

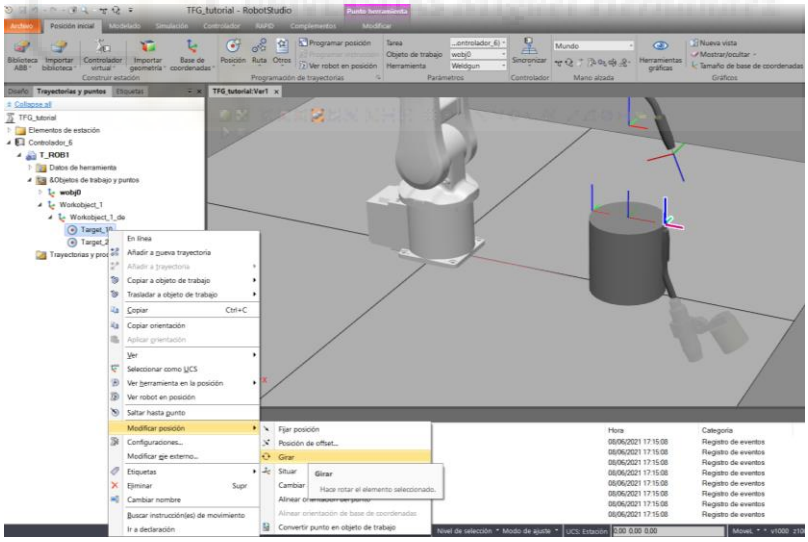


Figura 3.22: Girar posición de la herramienta

Debemos de realizar diversos giros hasta llegar a una posición donde el robot alcance el objetivo. Estos giros pueden ser en los ejes de coordenadas X, Y, Z, en nuestro caso realizaremos dos giros, un primer giro 180 grados respecto al eje X, y un segundo giro también a 180 grados pero ahora respecto al eje Z. Con la opción “Ver robot en posición” podemos ver que el robot alcanza el objetivo, como se puede ver en la figura 3.23.

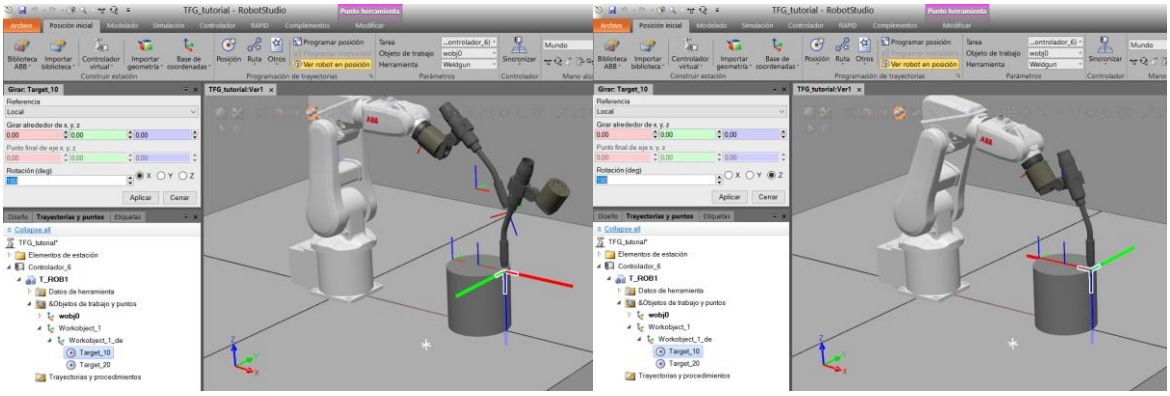


Figura 3.23: Giros de 180º en el eje X y en el eje Z

Ahora, para realizar la trayectoria tenemos dos opciones, una es crear una trayectoria vacía, en la que debemos de añadir los objetivos a la trayectoria y la otra es crear una trayectoria automática en la que necesitaremos una geometría o curva. En nuestro caso vamos a realizar una “Trayectoria automática”, seleccionándolo en la pestaña “Posición Inicial” y en la opción “Ruta”.

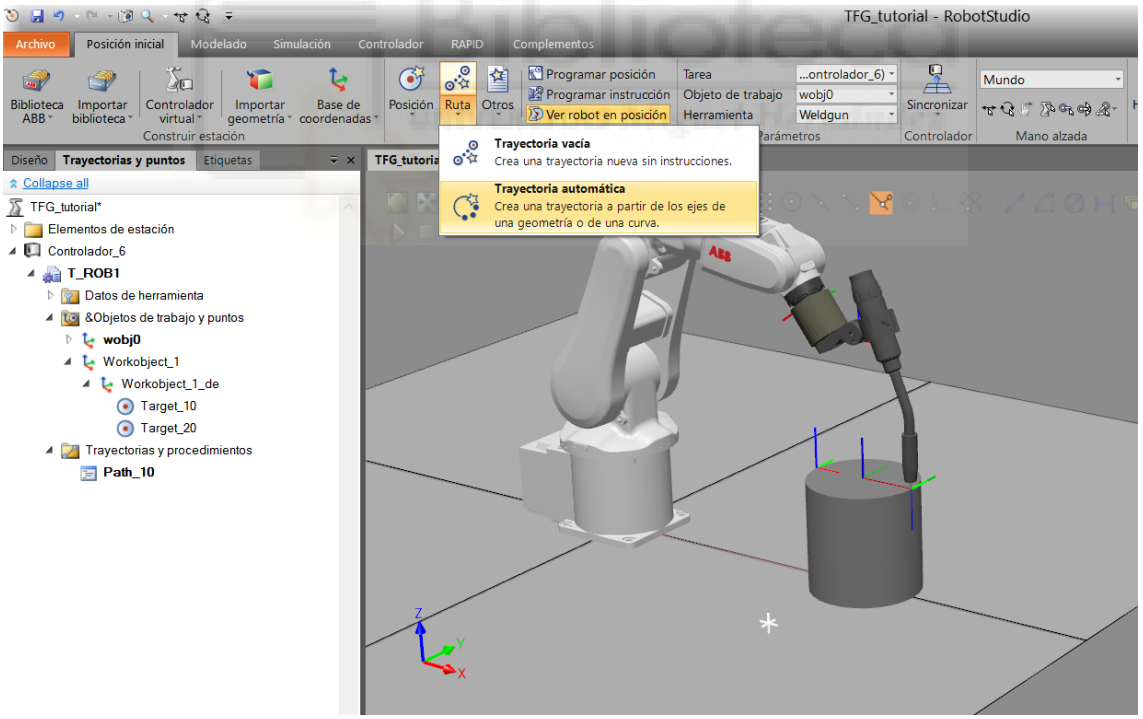



Figura 3.24: Creación de una trayectoria automática

En este tutorial vamos a realizar una trayectoria circular a lo largo de la arista superior del cilindro. Para la trayectoria automática necesitaremos seleccionar de que arista del cilindro queremos crear la trayectoria, y luego de que tipo, en nuestro caso seleccionamos “Circular”, como podemos ver en la figura 3.25.

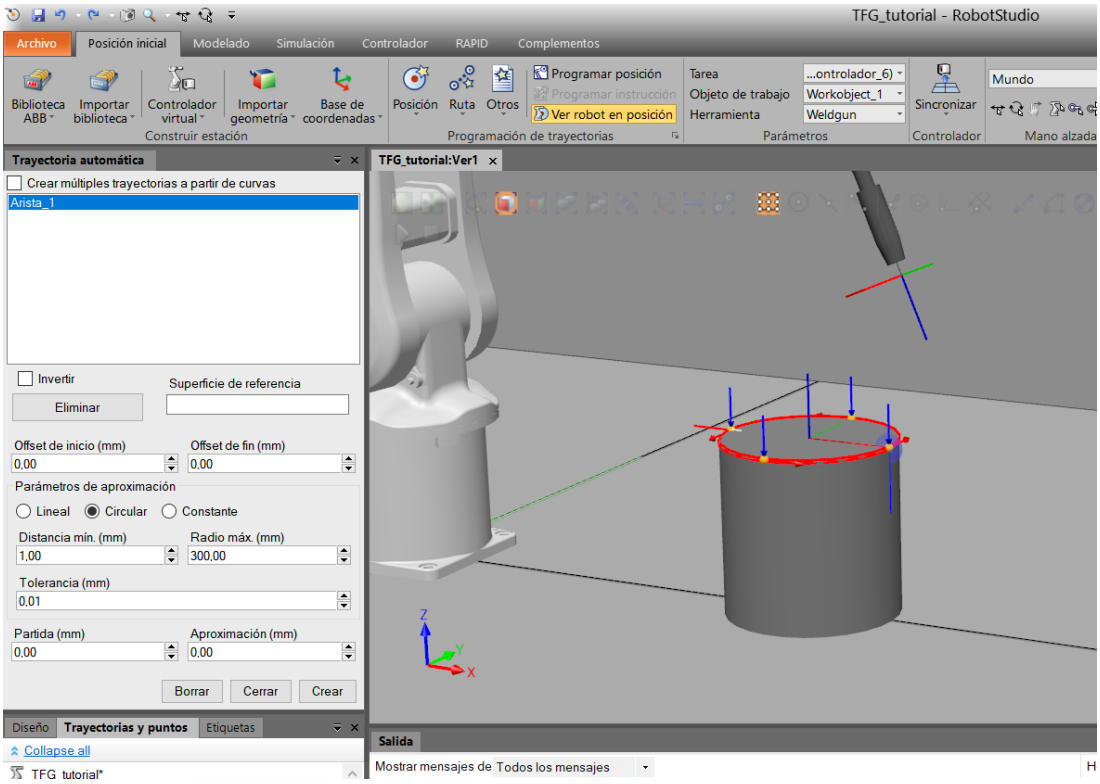



Figura 3.25: Parámetros trayectoria automática circular

Le damos a “Crear”, y ahora podemos ver que ha creado una serie de puntos con los que realiza la trayectoria circular a lo largo de la arista superior en el Path_10. Debemos de indicarle la orientación con lo que hacemos el mismo proceso que anterior mente para copiar y aplicar orientación. También, incluiremos al principio y al final del Path_10 un movimiento al punto de “Inicio”, como podemos ver en la figura 3.26.

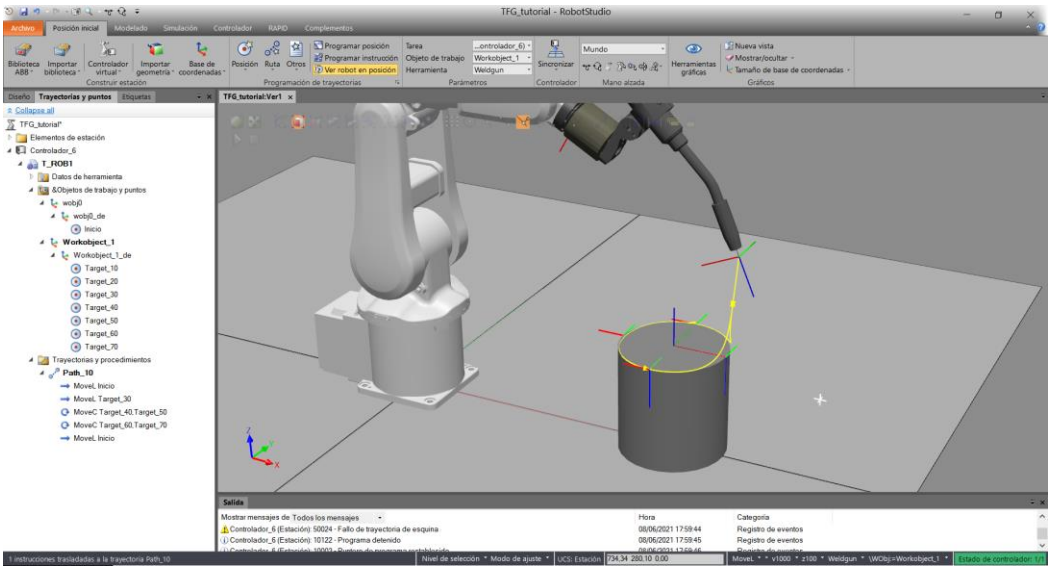



Figura 3.26: Trayectoria Path_10

Una vez tenemos la orientación correcta para que el robot alcance el objetivo , podemos copiar y pegar la orientación al resto de objetivos. Esto lo podemos hacer desde “Taget_10”, botón derecho y “Copiar orientación” y en el resto de los objetivos, clic derecho y “Aplicar orientación”, como podemos ver en la figura 3.27. De esta forma el robot alcanzará todos los puntos con la misma orientación.

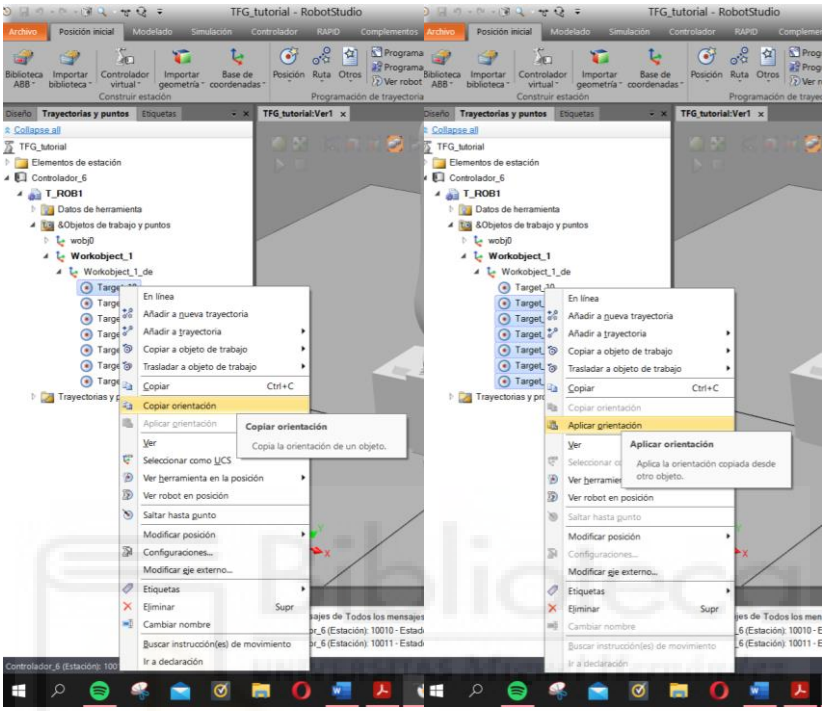



Figura 3.27: Copiar y Aplicar orientación

Una vez modificada todas las orientaciones de los objetivos, podemos realizar la configuración de los ejes del robot. Para realizar la configuración, hacemos clic con el botón derecho sobre Path_10, “Configuraciones”, seleccionamos “Configuración automática” y elegiremos “Todas las instrucciones de movimiento”, como se muestra en la figura 3.28.

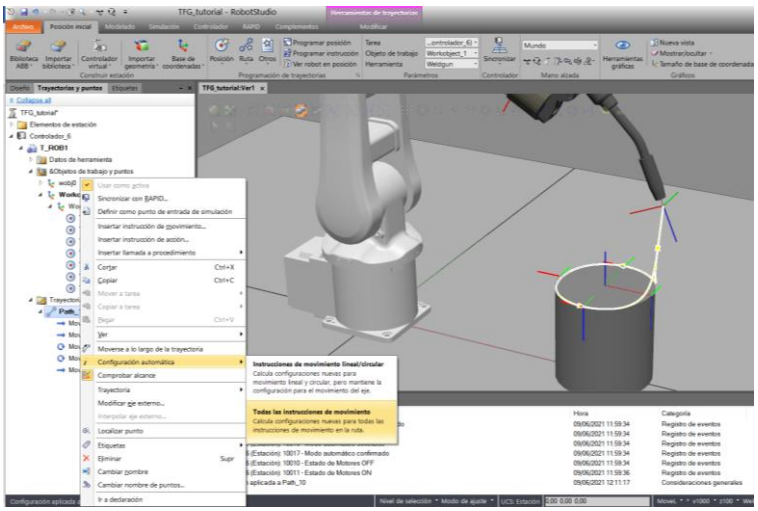



Figura 3.28: Configuración automática de Path_10

En las trayectorias tenemos varias opciones que podemos cambiar, como son el tipo de movimiento, velocidad, precisión, herramienta y objeto de trabajo. Para cambiar algún dato, seleccionamos un punto o varios de la trayectoria y en la parte de abajo a la derecha de la ventana del programa tenemos las siguientes opciones:

- **Plantilla de Instrucciones:** donde podemos cambiar el tipo de movimiento a MoveL, MoveJ o MoveAbsJ, en el apartado 3.1.6 veremos que significa cada tipo de movimiento.
- **Speed:** en este apartado podemos cambiar la velocidad con la que se mueve el robot, por defecto aparecerá v1000, siendo la velocidad en mm/s.
- **Zone:** esta opción se refiere a la precisión del movimiento, los valores referencian al error en distancia (en mm) que hay desde el TCP de la herramienta del robot al punto objetivo en el momento en el que se ejecuta la instrucción y el robot está pasando por dicha posición. Donde fine es el más preciso y z200 el menos preciso, por defecto aparecerá z100.
- **Herramienta:** se podrá escoger entre las herramientas que tengamos asociadas como herramientas al robot.
- **Objeto de trabajo:** el último apartado podremos cambiar el objeto de trabajo.

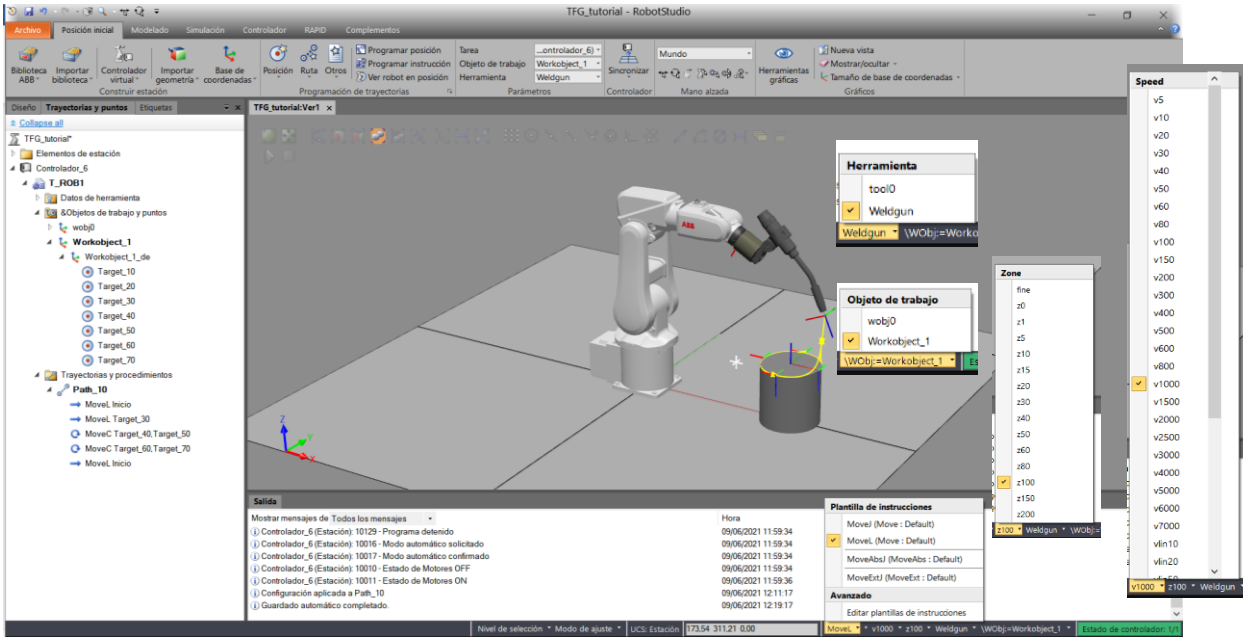



Figura 3.29: Cambio de parámetros en las trayectorias

Otra forma de editar las instrucciones es haciendo clic en el botón derecho en cada instrucción de movimiento de la trayectoria y seleccionando “Editar instrucción”, tendremos las mismas opciones que hemos comentado antes. Como vemos en la figura 3.30.

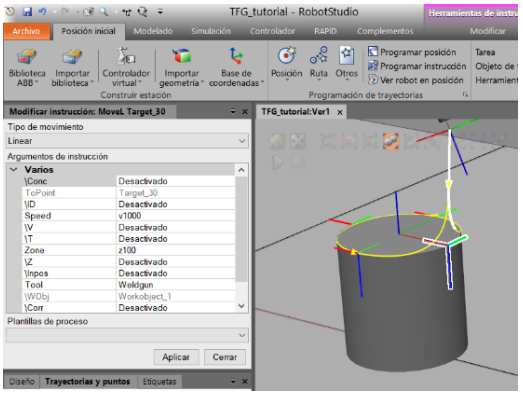



Figura 3.30: Modificar una instrucción

### *3.1.6. Introducción a RAPID*

Tal y como se desarrolla en [4], RAPID es un lenguaje de programación textual de alto nivel desarrollado por la empresa ABB. Una aplicación RAPID consta de un programa y una serie de módulos del sistema. El programa es una secuencia de instrucciones que controlan el robot, en general consta de tres partes:

- **Rutina principal (main):** es la rutina principal donde se inicia la ejecución.
- **Conjunto de subrutinas:** estas rutinas sirven para dividir el programa en partes más pequeñas con el fin de obtener un programa dividido en módulos
- **Datos del programa:** en esta parte se definen posiciones, valores numéricos, sistemas de coordenadas, variables, datos de herramientas, etc.

Para realizar la programación en RAPID, en primer lugar hay que sincronizar la estación con el controlador virtual. Para realizar esto debemos de ir a la pestaña “Posición Inicial”, seleccionamos “Sincronizar” y pinchamos en “Sincronizar con Rapid”, por último, daremos “Aceptar” como podemos ver en la figura 3.31.

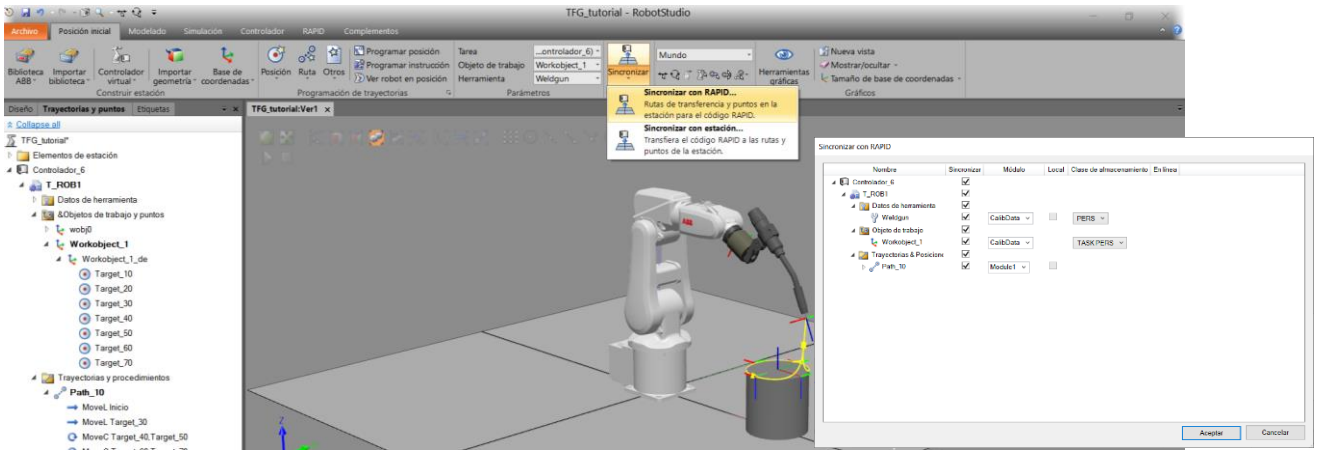



Figura 3.31: Sincronización con RAPID

Al sincronizar conseguimos que los datos de la herramienta, objeto de trabajo con todos los objetivos y las trayectorias creadas sean volcados al programa, apareciendo los módulos de programa “CalibData” y “Module1” necesarios para la programación del robot en Rapid.

Ahora pasamos a programar, para ello debemos ir a la pestaña de “RAPID” y dentro del controlador, abrimos el desplegable llamado “Rapid” y luego “T_ROB1”, haciendo doble clic derecho “Module1” podemos ver, gracias a la sincronización que hemos realizado previamente, definidos los puntos como constantes robtarget y también las trayectorias con sus movimientos.

El programa esta englobado en el módulo del programa llamado “Module1”. Para definir el módulo, tenemos una instrucción inicial “MODULE Module1” y una instrucción final con “ENDMODULE”.

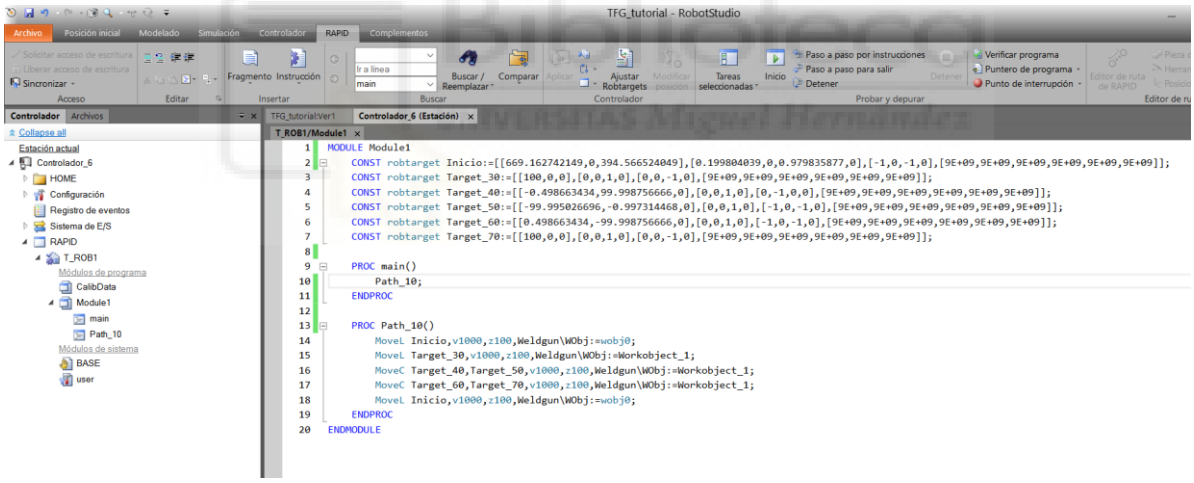



Figura 3.32: Programa RAPID (Module1)

A continuación de definir el módulo, debemos de realizar la declaración de datos mediante el comando “CONST”, seguido de la palabra “robtarget”, el nombre del objetivo, sus coordenadas relativas al objeto de trabajo y la configuración de ejes del robot en dicho punto, en la siguiente figura podemos ver un ejemplo de definición de un punto.

```
CONST robtarget Inicio:=[[669.162742149,0,394.566524049],[0.199804039,0,0.979835877,0],[-1,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09];
```

Figura 3.33: Definición de un punto en RAPID

Lo siguiente es la declaración de un programa, para ello se debe comenzar con “PROC nombre_porgrama ()” y se finaliza con “ENDPORC”. Dentro del programa podemos incluir las instrucciones de movimiento, esperas a señales, etc. que queremos.

En cuanto a las instrucciones de movimiento “Move” que hemos utilizado antes, existen 3 principales tipos de movimiento:

- **MoveL:** se trata de un desplazamiento del extremo del robot hasta el punto indicado en línea recta.
- **MoveC:** se trata de un desplazamiento del extremo del robot hasta el punto indicado siguiendo un círculo, se necesitan dos posiciones, la primera de ellas define un punto intermedio del círculo que debe realizar la trayectoria y la otra, define el punto final del círculo y de la trayectoria.
- **MoveJ:** se trata de un desplazamiento del extremo del robot hasta el punto indicado de forma rápida, sin garantizar cuál es la trayectoria seguida (no existe una coordinación de velocidad entre los distintos ejes del robot).

En la imagen 3.34 encontramos un ejemplo de instrucción de movimiento, donde en primer lugar se define el tipo de movimiento “MoveL”, seguido del punto de destino “Inicio”, la velocidad del movimiento “v1000”, la precisión “z100” y la herramienta que está utilizando el robot y el objeto de trabajo “Weldgun\WObj:=wobj0”.

```
MoveL Inicio,v1000,z100,Weldgun\WObj:=wobj0;
```

Figura 3.34: Instrucción de movimiento en RAPID

- **Instrucciones para la utilización de entradas y salidas**

Dos tipos de operaciones importante que vamos a necesitar son comprobar el valor de una entrada y fijar el valor de una salida.

Para fijar el valor de las salidas utilizamos las siguientes instrucciones:

- **Set:** fija el valor de una salida digital a 1.
- **Reset:** fija el valor de una salida digital a 0.
- **SetDO:** fija una salida digital a un valor simbólico (activado o desactivado).
- **SetAO:** fija el valor de una salida analógica.

La instrucción principal para comprobar el valor de una entrada es la instrucción **WaitDI** (por ejemplo, WaitDI Pieza,1;), esta instrucción hace que el robot espere hasta que la señal “Pieza” tenga el valor indicado. También comprobamos el valor de una señal de entrada con instrucciones de comparación, como lo son las instrucciones IF señal = 1 THEN…, IF señal < 5 THEN…

- **Instrucciones de control de flujo de ejecución**

Son instrucciones que también son utilizadas en otros lenguajes de programación:

- **IF ___ THEN:** ejecuta una serie de instrucciones si se cumple una determinada condición indicado detrás del IF.
- **FOR:** se trata de un bucle que repite una sección del programa un determinado número de veces.
- **WHILE:** se trata de un bucle que repite una sección del programa mientras se cumpla una condición dada detrás del WHILE.
- **TEST/CASE:** ejecuta diferentes instrucciones en función del valor de un dato (similar al switch-case del lenguaje C).
- **GOTO:** salto incondicional a un punto del programa, debemos de definir el punto al que debe de saltar anteriormente con “nombre_punto_salto:”

- **Variables y expresiones**

El lenguaje RAPID permite definir variables u otros tipos de datos. Definimos las variables para crear expresiones aritméticas o lógicas mediante una serie de operadores comunes (suma, producto, comparación…).

Los principales tipos de datos en RAPID son ConfData, JointTarget, Load Data, MotSetData, Num, Orient, Pos, Robjoint, Robtarget, StopPointData, ToolData, ZoneData.

Continuando con el tutorial, debemos de cambiar la precisión “Zone” de las trayectorias para que tenga mayor precisión, para ello hemos cambiado el valor “z100” de todas las instrucciones a “fine”, de este modo puede llegar hasta todos los puntos de forma que no tengamos ningún error y la trayectoria pase por la arista del cilindro.

En la figura 3.35, podemos ver todos los cambios realizados en las instrucciones de movimiento.

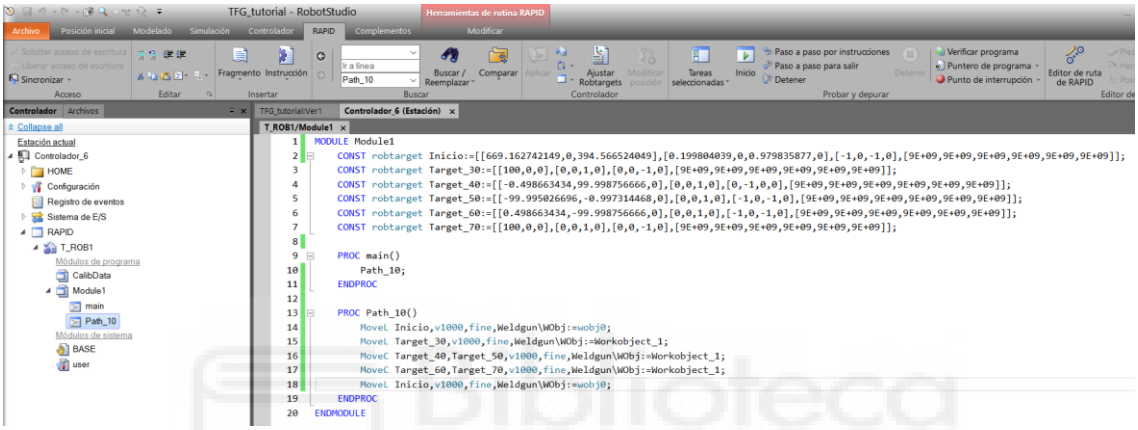



Figura 3.35: Cambios realizado en RAPID

Ahora, vamos a sincronizar con la estación desde RAPID para que se actualicen todos los cambios realizados. Para ello, en el apartado “Acceso”, clicamos en “Sincronizar” y seleccionamos “Sincronizar con estación”. Y luego le damos a “Aceptar”, como podemos ver en la figura 3.36.

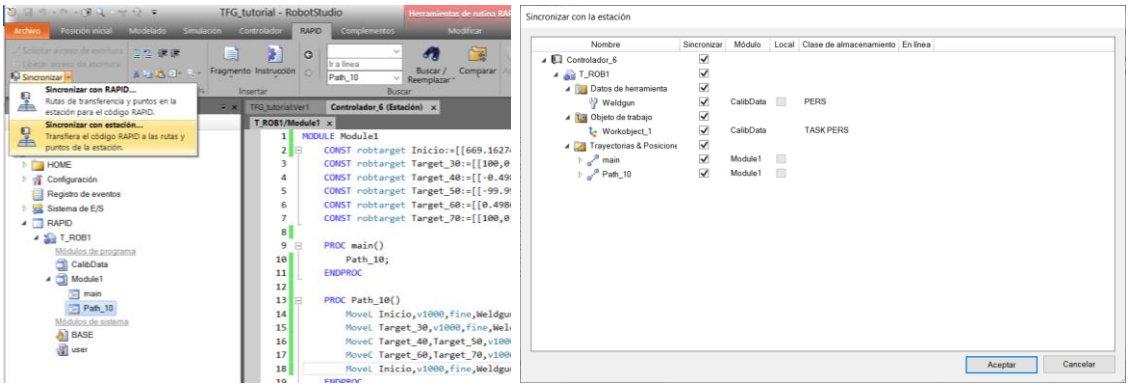



Figura 3.36: Sincronización con la estación desde RAPID

Si vamos a la pestaña de “posición inicial”, podemos ver que todos los cambios se han realizado y que ahora el robot alcanza toda la arista del cilindro, como podemos ver en la figura 3.37.

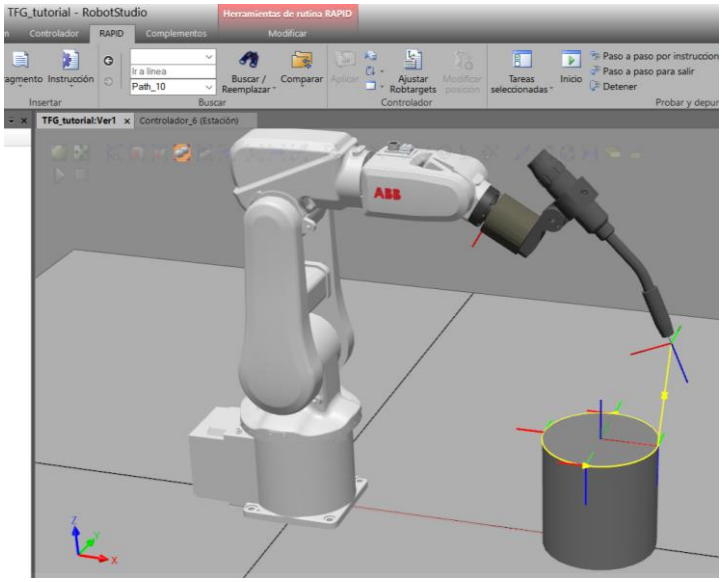



Figura 3.37: Mejora de la precisión

## 3.2. Diseño y programación de una Estación

### *3.2.1. Diseño en AutoCAD de la geometría de la estación*

En primer lugar, para el diseño de la estación se han de crear los sólidos que la componen, pero el programa RobotStudio no es suficiente ni a la hora de crear un sólido de diseño ni con los objetos que ya vienen incluidos en el programa, como son las cintas transportadoras, mesas, vallas…

En este trabajo se ha utilizado el software de diseño AutoCAD 2020, para realizar el diseño de las piezas necesarias. El conjunto de planos de las piezas que se han creado con AutoCAD y que posteriormente se han importado a RobotStudio se encuentran en el Anexo 9.1 de este trabajo.

El proceso para adjuntar la pieza en el programa de RobotStudio es el siguiente, lo primero que debemos de hacer es exportar desde el programa AutoCAD la pieza que hemos creado, en el icono de AutoCAD a la izquierda, vamos a “Exportar” y seleccionamos como “Otros formatos”.

Aparecerá una venta emergente donde debemos de seleccionar el tipo de archivo (en nuestro caso seleccionamos .sat, que es el tipo de archivo que admite RobotStudio) y el lugar donde debemos de guardarlo, por último, le damos a Guardar y seleccionamos el sólido que forma la pieza, se puede ver el proceso en la siguiente figura 3.38.

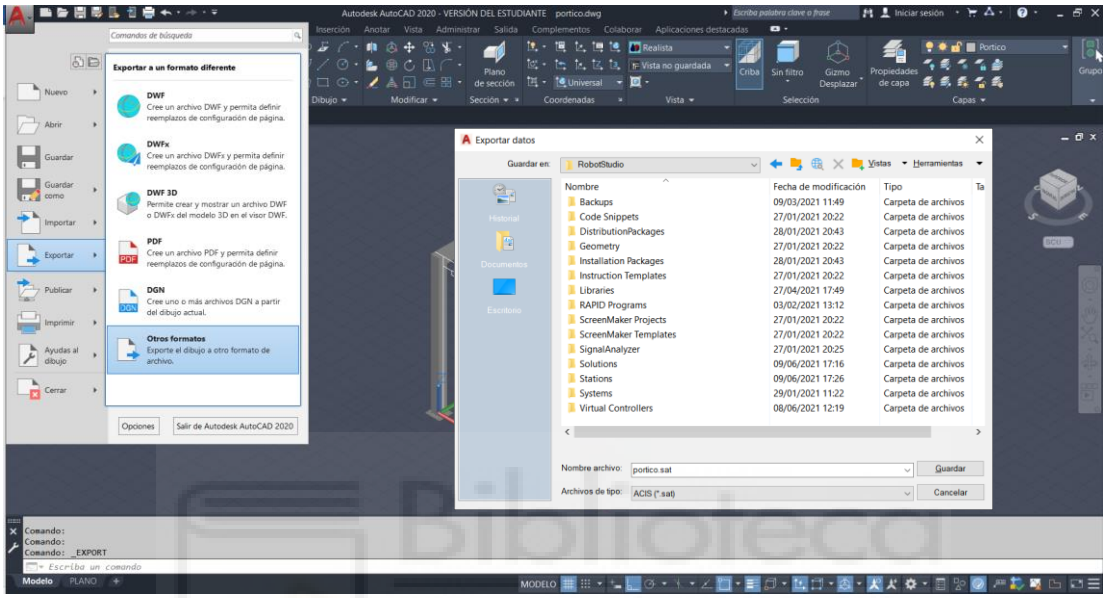



Figura 3.38: Exportar sólido desde AutoCAD

Una vez creado, vamos a RobotStudio y para añadirlo debemos de ir a la pestaña “Posición inicial”, seleccionamos “Importar Geometría” y hacemos clic en “Buscar Geometría”, vamos donde hayamos guardado el archivo .sat que queremos importar y lo abrimos. RobotStudio solamente permite exportar sólido y superficies.

### *3.2.2. Creación y diseño de los Smart Components*

Los Smart Components (SC) o componentes inteligentes, son elementos asociados a los sólidos, piezas o robots que tenemos en nuestra estación. Los cuales tiene un comportamiento controlado por señales y propiedades del sistema.

En la figura 3.39, podemos ver como se crear un Smart Component. Vamos a la pestaña de “Modelado” y hacemos clic en “Componente Inteligente”.

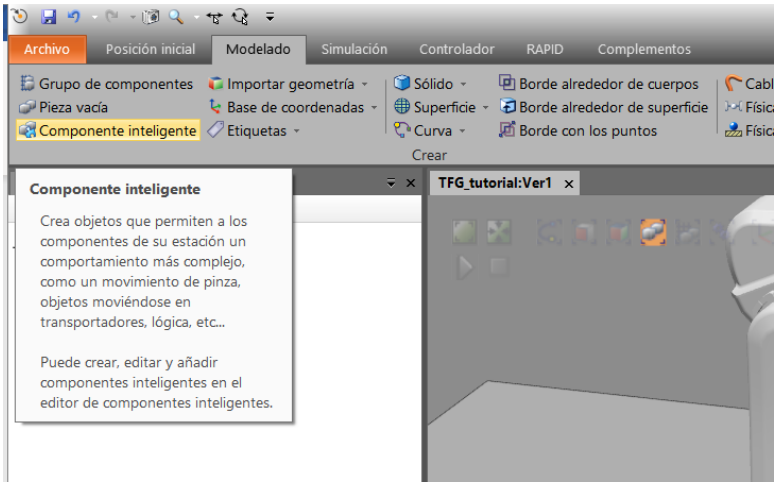



Figura 3.39: Creación de un Componente Inteligente (SC)

Los Componente inteligente los podemos agrupar en 6 categorías distintas: Señales y propiedades, Primitivos paramétricos, Sensores, Acciones, Manipuladores, Controlador, Física, PLC y Otros.

- **Señales y propiedades:** es la primera categoría que encontramos, en ella podemos encontrar puertas lógicas, multiplexores, contadores, temporizadores, expresiones matemáticas, convertidores y otros elementos que permiten modificar una señal o las propiedades del sistema a programa. Lo vemos en la figura 3.40.

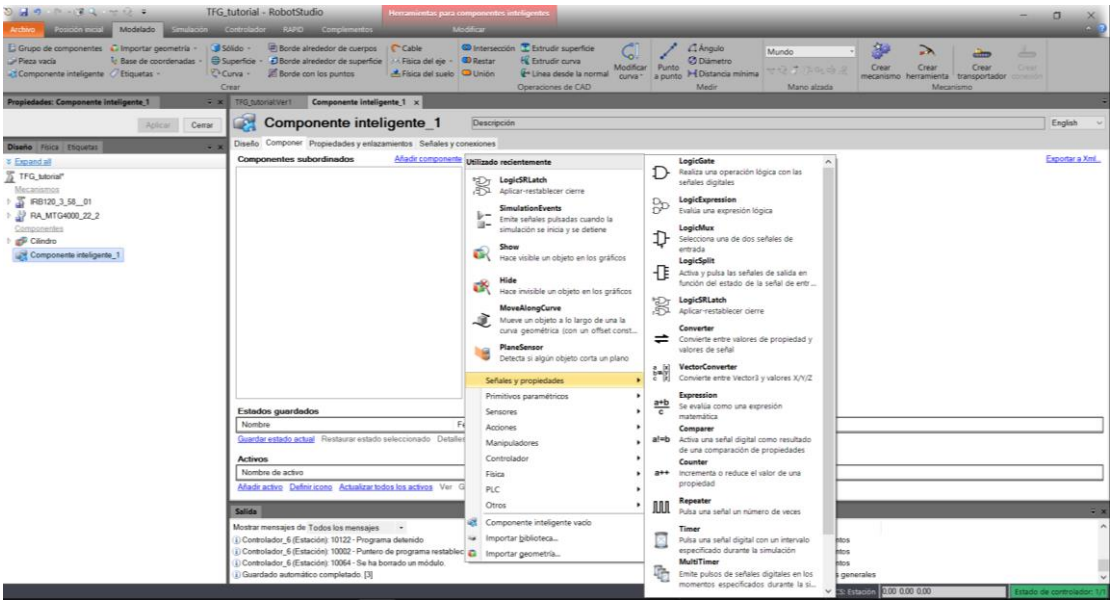



Figura 3.40: Categoría de Señales y propiedades en SC

- **Primitivos paramétricos:** En esta categoría se encuentran los componentes necesarios para crear de forma automática sólidos y líneas, así como generar copias de piezas ya existentes.

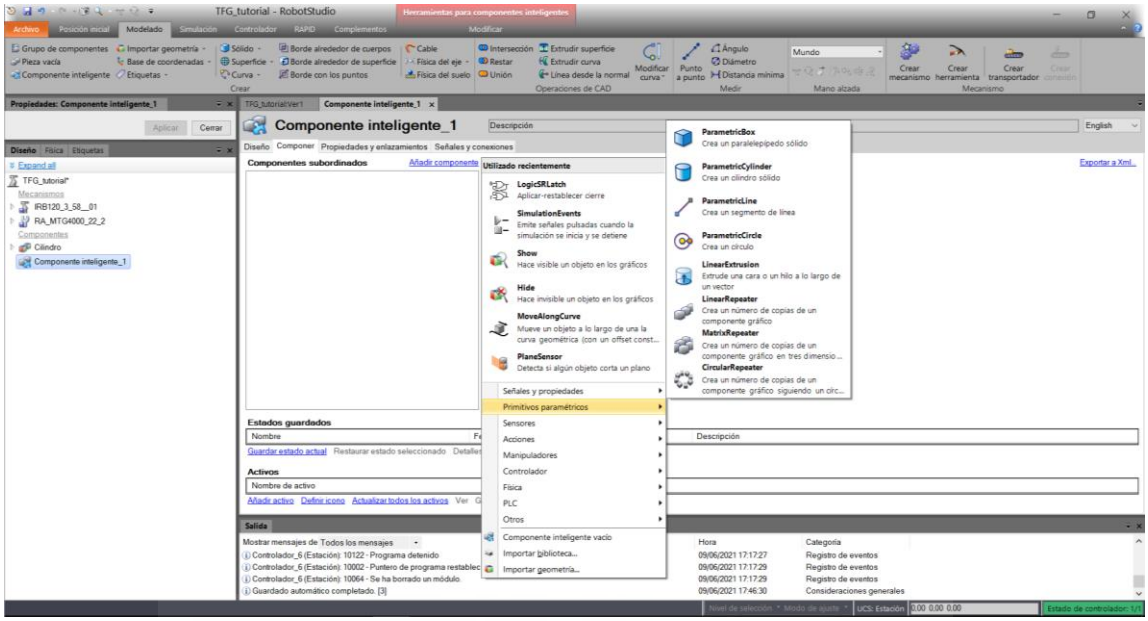



Figura 3.41: Categoría de Primitivos paramétricos en SC

- **Sensores:** Esta categoría reúne el conjunto de sensores a utilizar en los procesos de producción automático de este trabajo. Incluye sensores de colisión, de línea, de superficie, volumétricos, etc.

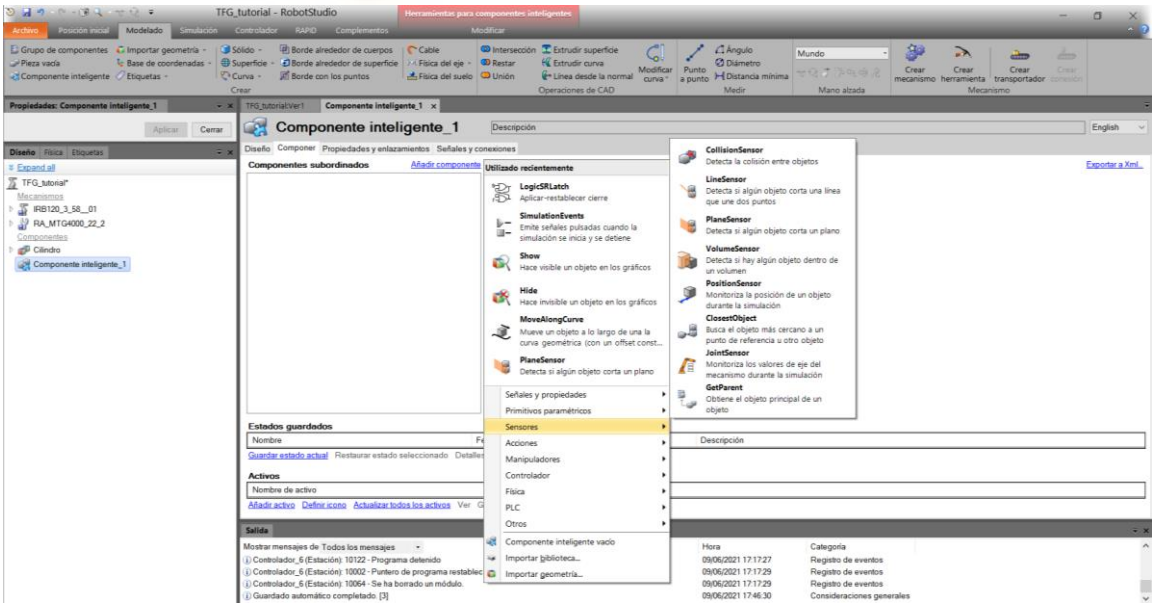



Figura 3.42: Categoría de Sensores en SC

- **Acciones:** Hace referencia a componentes inteligentes como conectar o desconectar dos objetos entre sí, eliminar un objeto o simplemente hacerlo visible/invisible.

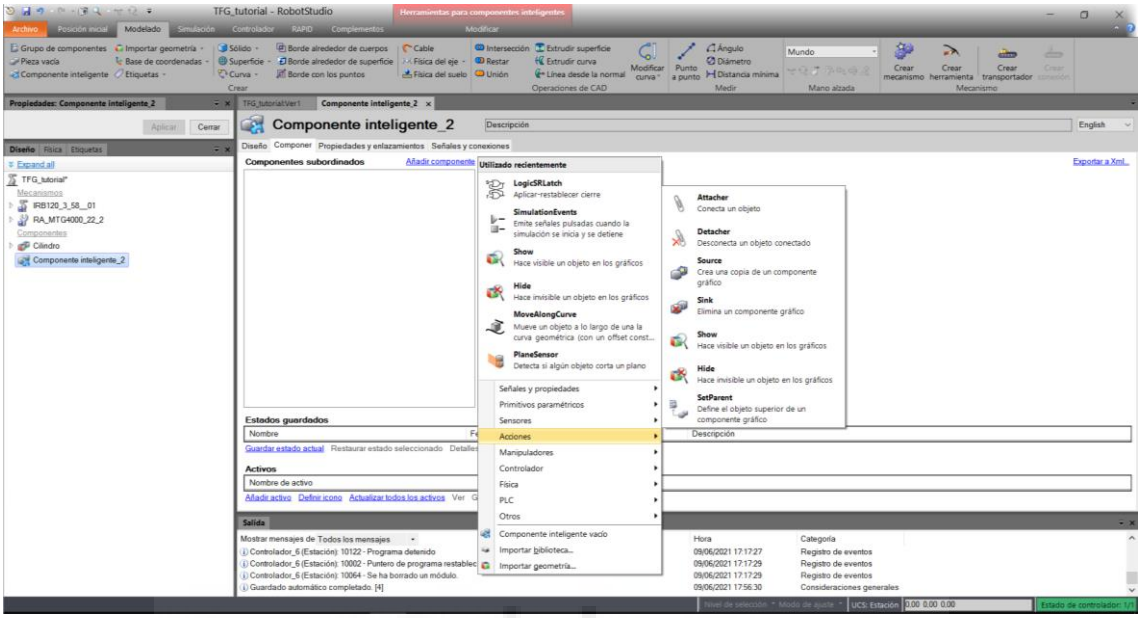



Figura 3.43: Categoría de Acciones en SC

- **Manipuladores:** en este apartado reúne los componentes necesarios para mover un objeto de forma lineal, hacerlo rotar un determinado ángulo , moverse a lo largo de una curva o spline, o posicionarlo en un lugar específico. También permite mover los ejes del mecanismo de un brazo robótico a la posición elegida.

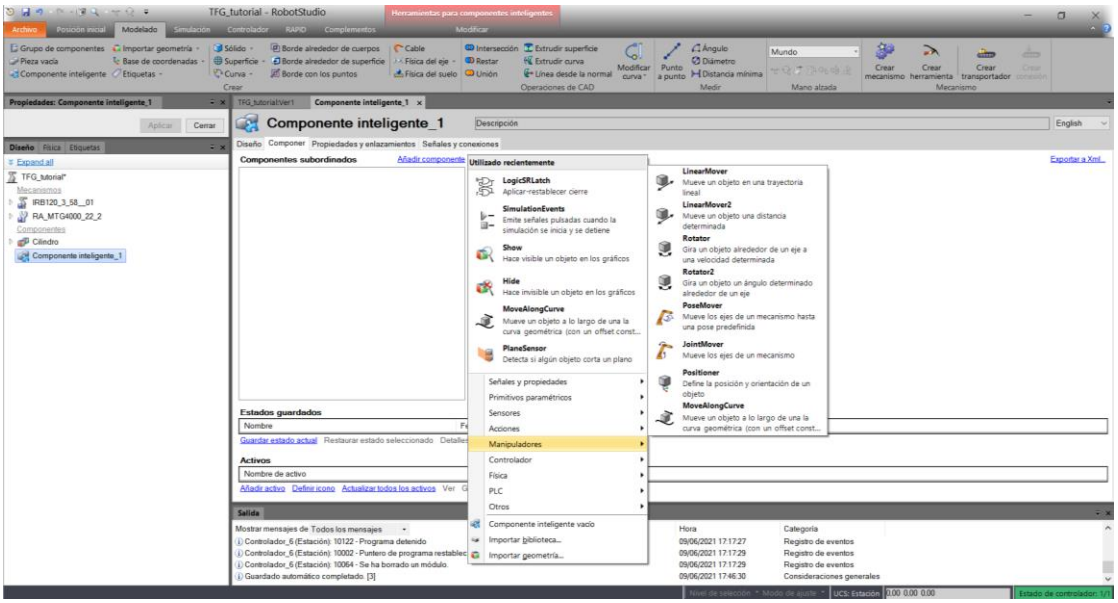



Figura 3.44: Categoría de Manipuladores en SC

- **Controlador:** en este apartado encontramos elementos para establecer u obtener el valor de una variable en RAPID.

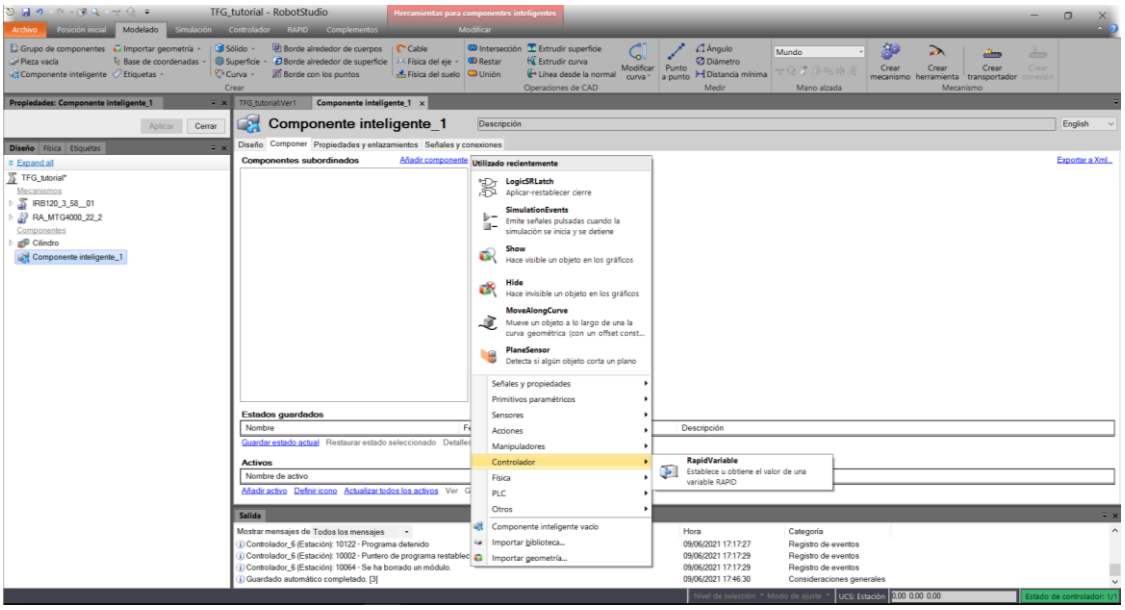



Figura 3.45: Categoría de Controladores en SC

- **Física:** en esta categoría encontramos elementos que permiten controlar la física de un objeto o que los ejes de un robot.

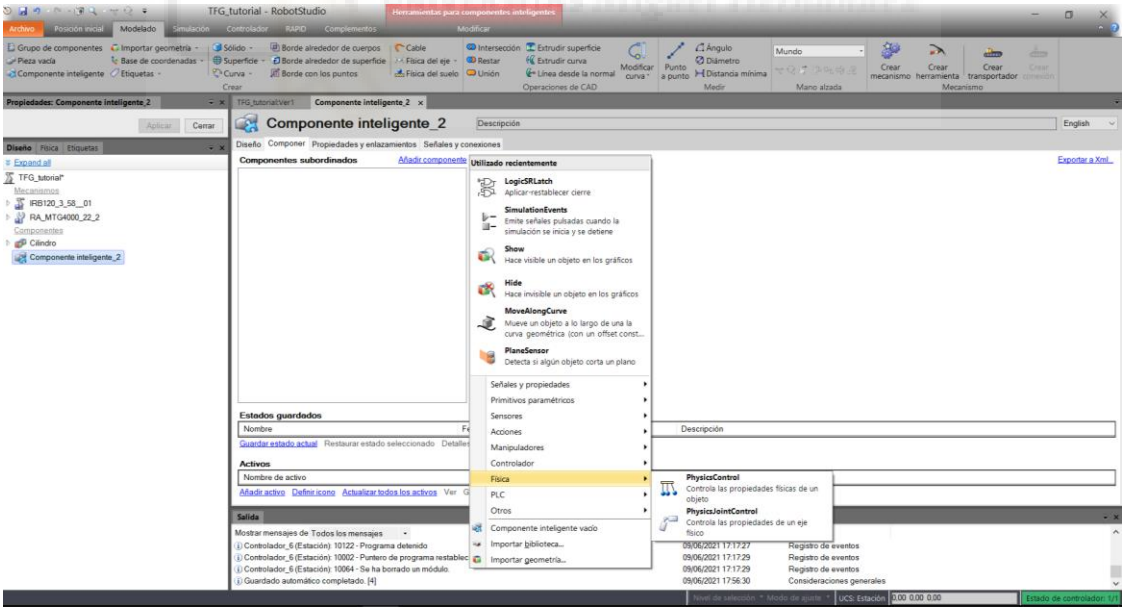



Figura 3.46: Categoría de Física en SC

- **PLC:** en este apartado encontramos un elemento que permite realizar la conexión con Siemens SIMIT.

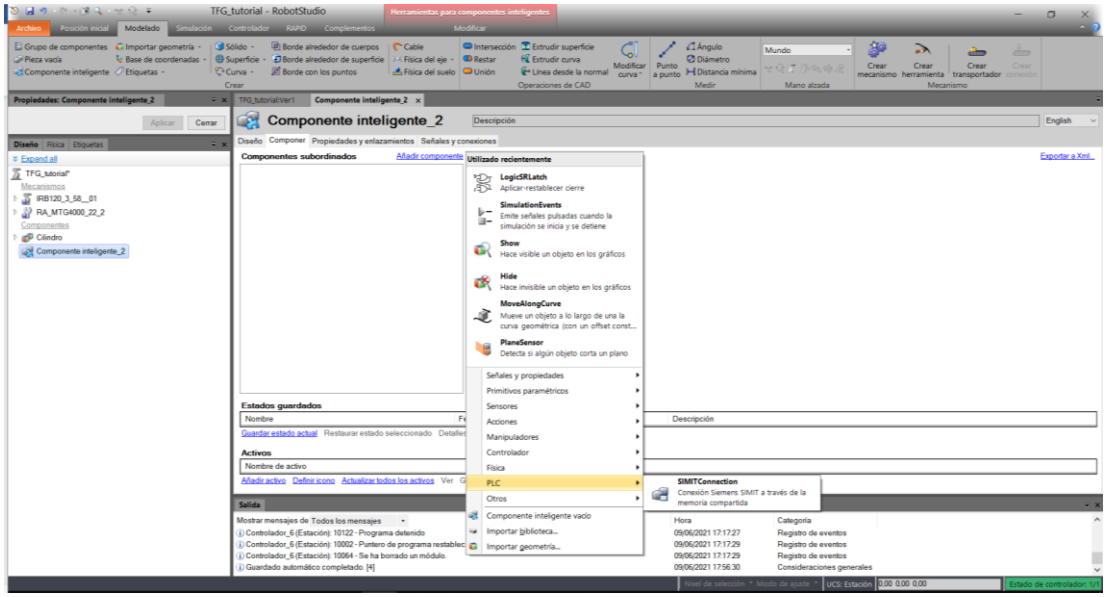



Figura 3.47: Categoría de PLC en SC

- **Otros:** En la última categoría se encuentran un conjunto de SC de distinta variedad. Representación de una cola de objetos, generación de un número aleatorio, detención de la simulación, reproducción de un sonido, etc.

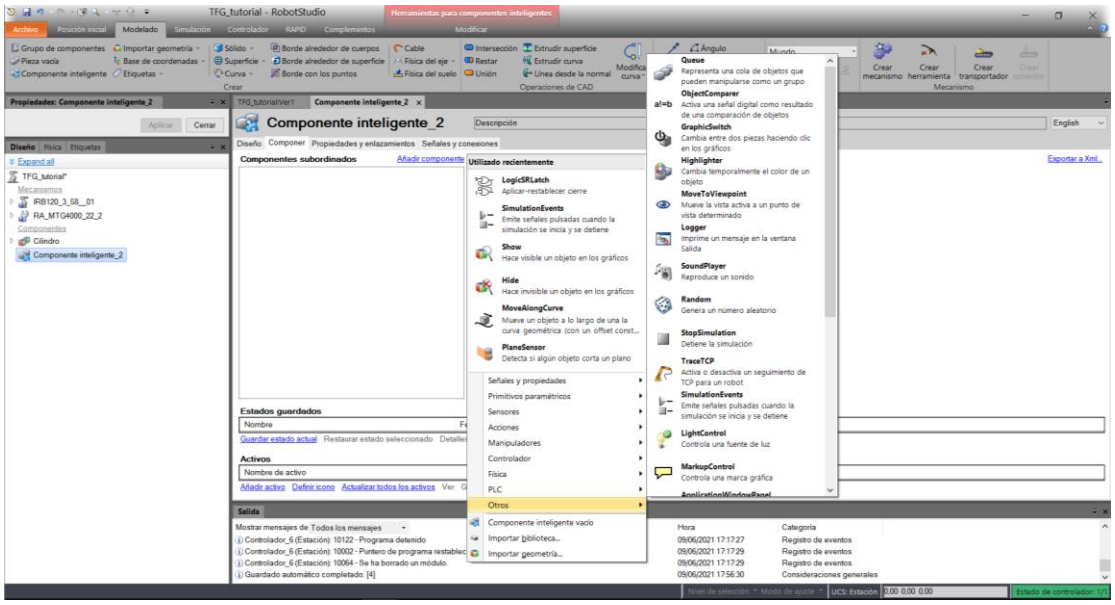



Figura 3.48: Categoría de Otros en SC

Dentro del Smart Componente, tenemos una pestaña de Señales y conexiones, donde podemos añadir señales de entrada o salida correspondientes a este componente, además también podemos realizar las conexiones entre los diferentes objetos. Lo podemos ver en la figura 3.49.

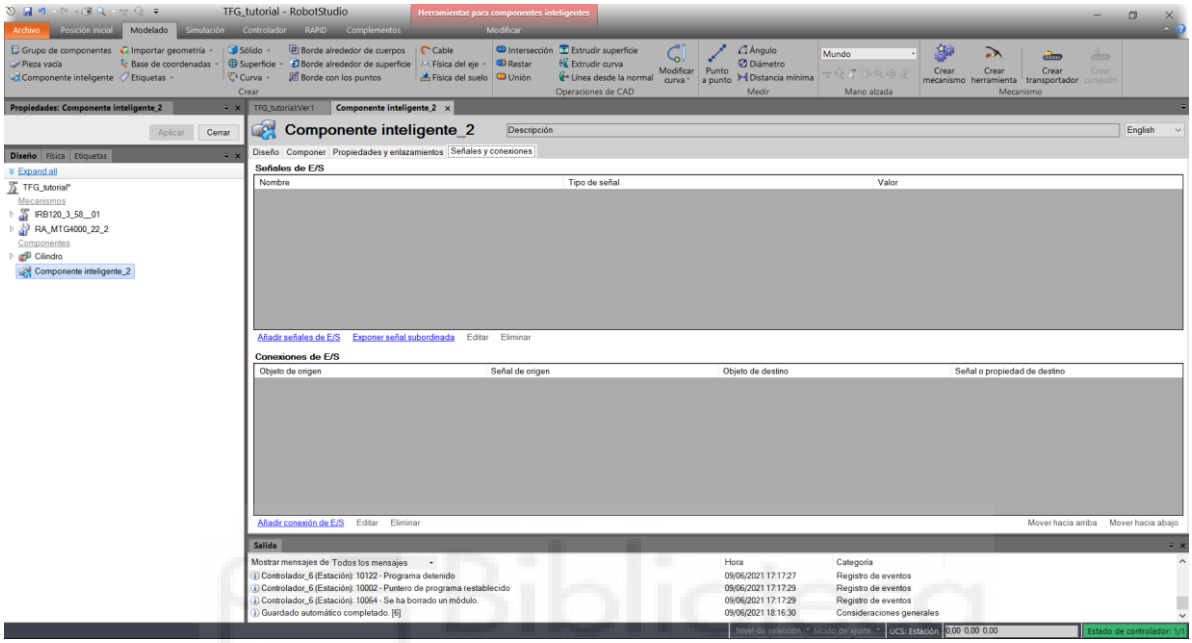



Figura 3.49: Señales y Conexiones en SC

Luego encontramos una pestaña de Diseño, donde podemos ordenar los objetos que hayamos incluido en el Componente Inteligente, cambiar su configuración, añadir entradas/salidas o conectar mediante flechas los distintos objetos.

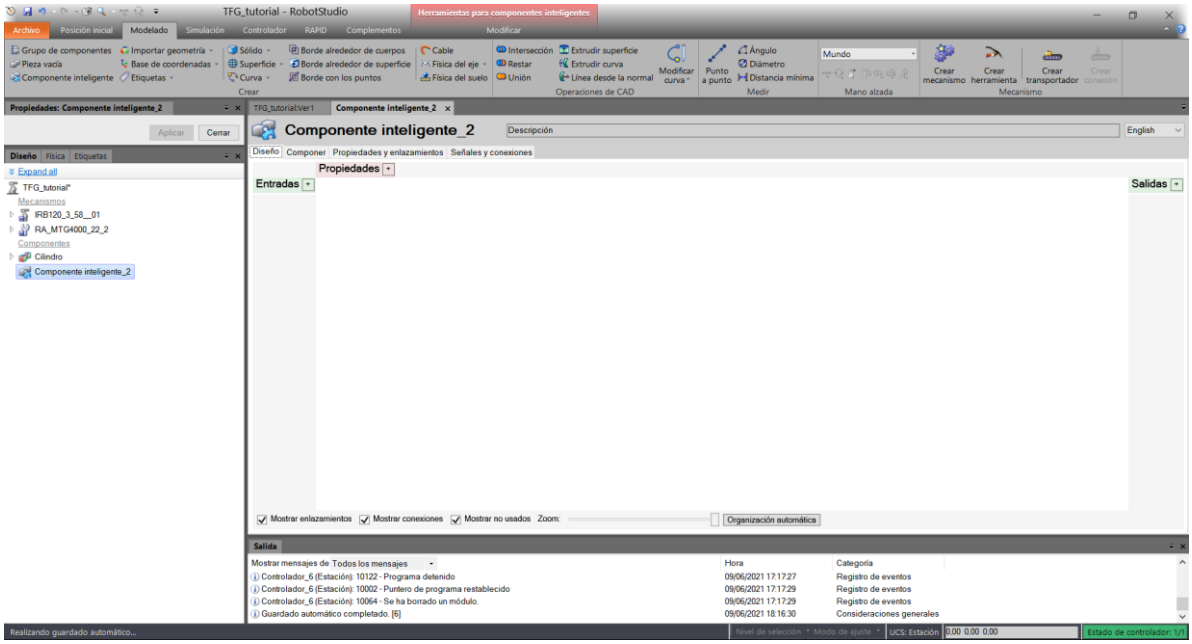



Figura 3.50: Diseño en SC

### *3.2.3. Creación de señales E/S*

Una vez creada la geometría de la estación, el siguiente paso es “Atribuirle inteligencia” al robot que tenemos en la estación. El sistema de E/S maneja las señales de entrada y salida intercambiadas con el controlador.

Para añadir una señal al controlador, desde la pestaña “Controlador”, seleccionamos “Configuración” y hacemos clic en “Añadir señales…”, como se puede ver en la figura 3.51.

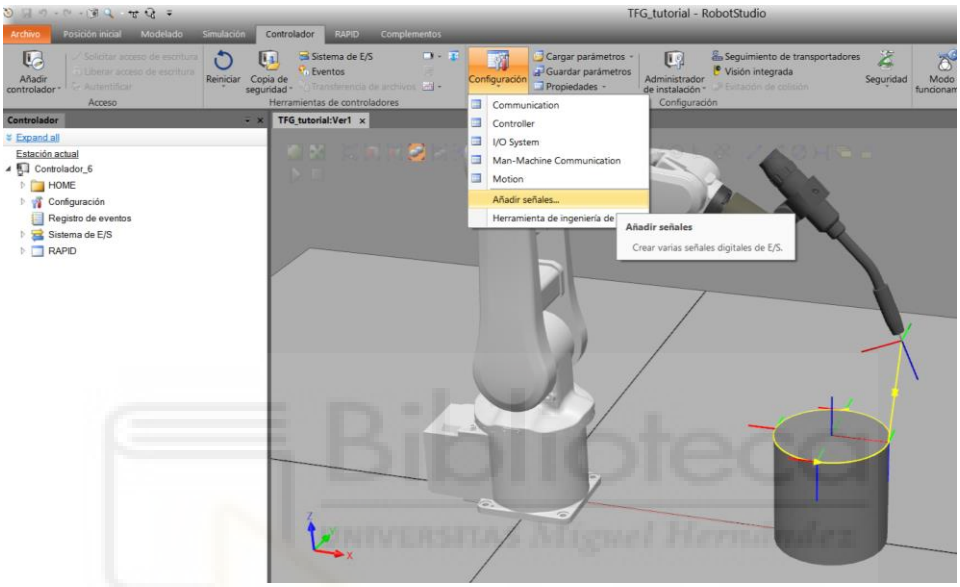



Figura 3.51: Añadir señales al controlador

Para añadir una señal, primero debemos de elegir el tipo de señal queremos añadir, las más comunes suelen ser las señales de Entrada Digital o Salida Digital, en este caso vamos a añadir un Entrada Digital.

A continuación, le damos un nombre a la señal (“Activar_robot”) y en asignación a dispositivo seleccionamos “ninguno” y para finalizar pulsamos Aceptar, como en la figura 3.52.

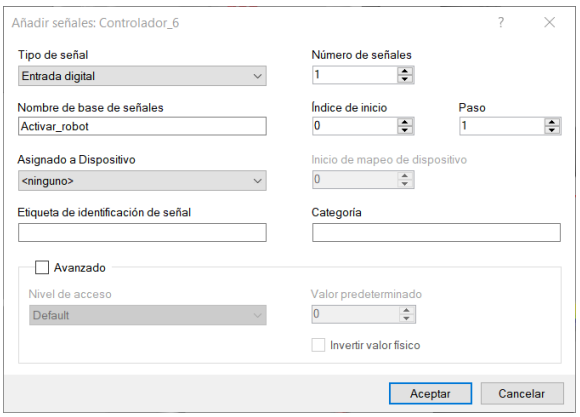



Figura 3.52: Crear Entrada Digital en controlador

Lo más importante a la hora de crear señales es que demos de Reiniciar el controlador cada vez que realicemos algún cambio o añadamos una señal nueva, para que se guarden y sincronicen correctamente los cambios realizados. Como vemos en la figura 3.53, vamos a la pestaña “Controlador” y hacemos clic en “Reiniciar” para reiniciar el controlador.

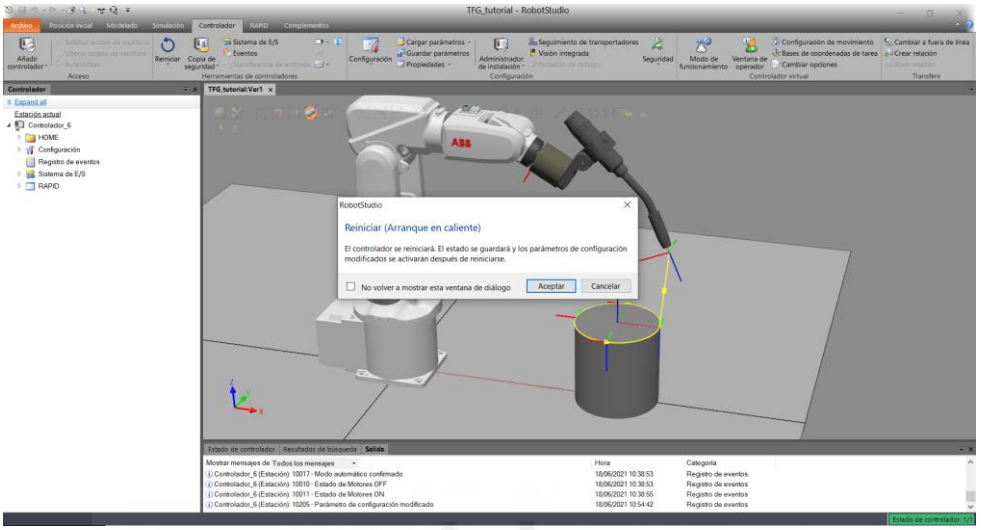



Figura 3.53: Reinicio del controlador

Otra forma de añadir señales es, en la pestaña del “Controlador”, vamos a la ventana de la izquierda, dentro de “Controaldor_6”, en el desplegable “Configuración”, hacemos clic en “I/O System”.

Para añadir una nueva señal, haciendo clic en el botón derecho en “Signal”, pinchamos en “Nuevo Signal”. También, es posible editar, copiar o eliminar señales creadas anteriormente.

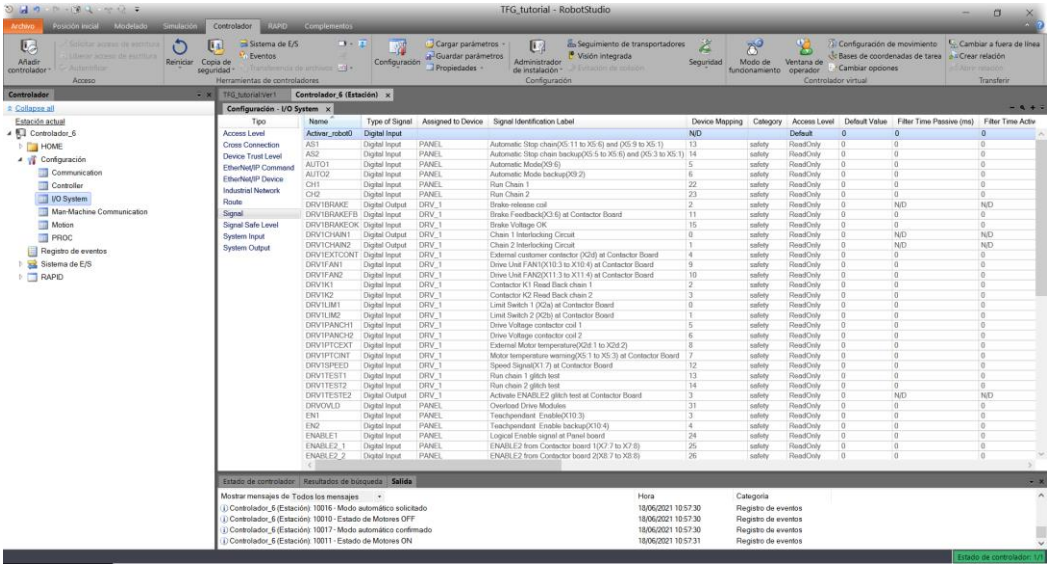



Figura 3.54: Configuración de señales en controlador

Una vez añadidas todas las entradas y salidas necesarias, se crearán un conjunto de señales del controlador que corresponderán a las señales de entrada y salida de los Componentes Inteligentes que sean necesarias para el control de los robots.

En el Anexo 9.2 se encuentra el listado de entradas y salidas de este trabajo.

### *3.2.4. Lógica de la estación*

Tras haber creado el controlador del robot y los SC, es necesario conectarlos entre sí, paso importante para que el sistema coordinado de los robots y de los sólidos inteligentes funcionen correctamente durante la simulación.

Para ello, accedemos a la pestaña “Simulación” y dentro de ésta, hacemos clic en “Lógica de estación”. La lógica de la estación estará formada por un conjunto de bloques correspondientes a los SC que hayamos creado y el controlador del robot. Para poder observar dicho esquema de bloques una vez abierta la lógica de estación, al igual que con los SC, hay que pulsar sobre la pestaña “Diseño”, como vemos en la figura 3.55.

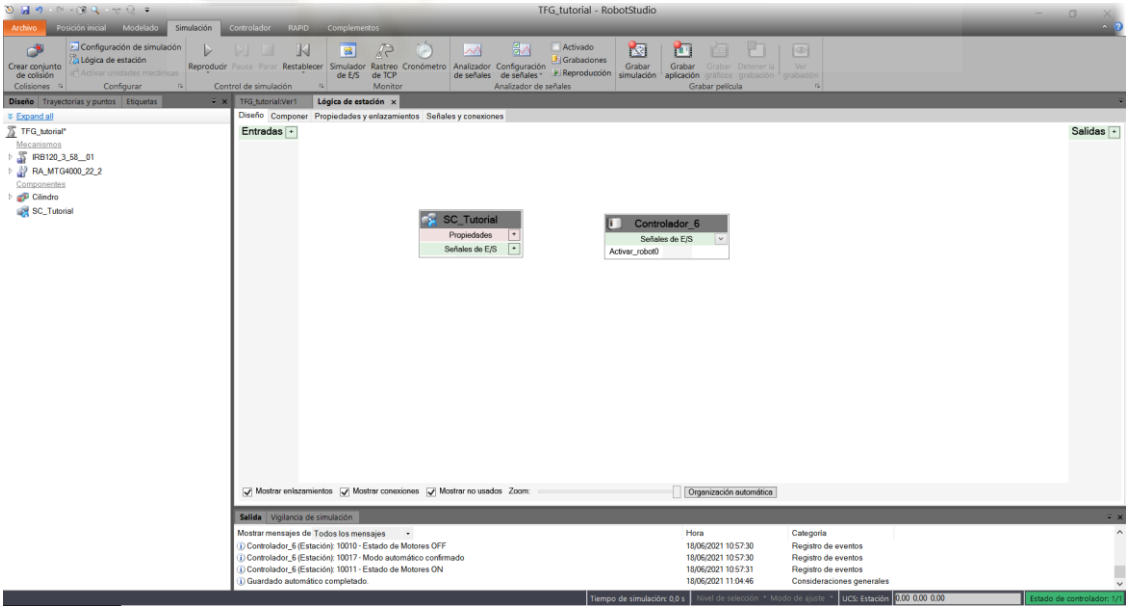



Figura 3.55: Lógica de la estación

## 3.3. Simulación de una estación

Una vez finalizada la programación de la estación, solo queda realizar la simulación, y así observar el funcionamiento del sistema automatizado que hemos creado. Para observar paso a paso la simulación en el programa RAPID, desde la pestaña RAPID, pulsamos en “Verificar programa”, seleccionamos “Aplicar” para que se guarden los cambios y finalmente hacemos clic en “Inicio” para comenzar la simulación.

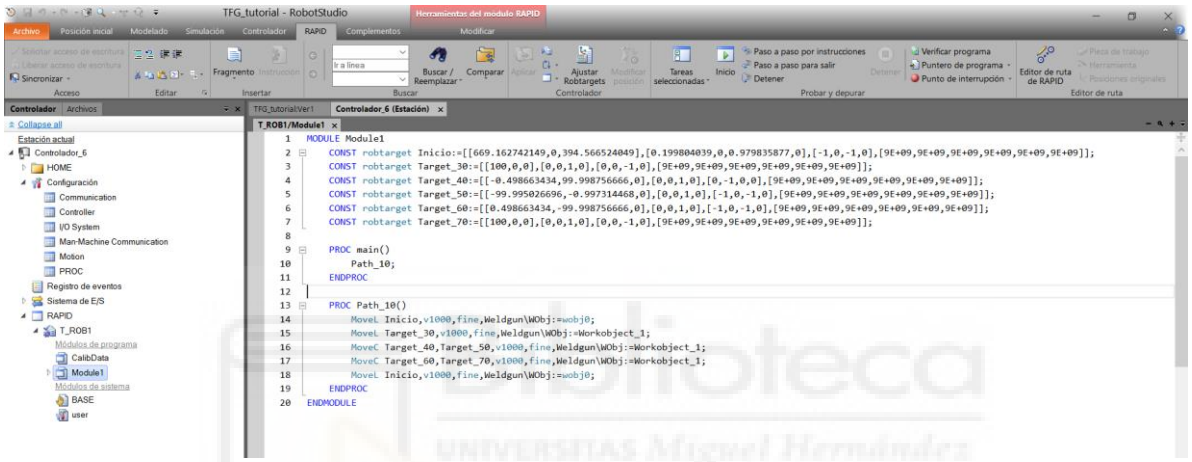


Figura 3.56: Compilación de un programa en RAPID e inicio de la simulación

Para realizar la simulación, podemos configurarla de varios modos, en la pestaña de “Simulación”, haciendo clic en “Configuración de simulación”, aparece un panel con las configuraciones como la que vemos en la figura 3.57.

Seleccionando “Controaldor_6”, aparece unos ajustes donde podemos seleccionar si queremos que se inicie automáticamente cuando comience la simulación. También, podemos seleccionar el modo de ejecución, podemos seleccionar que la tarea se realice una vez, seleccionando “Un solo ciclo” o escogemos el modo “Continuo”, donde la simulación no finaliza y lo realiza de forma continua.

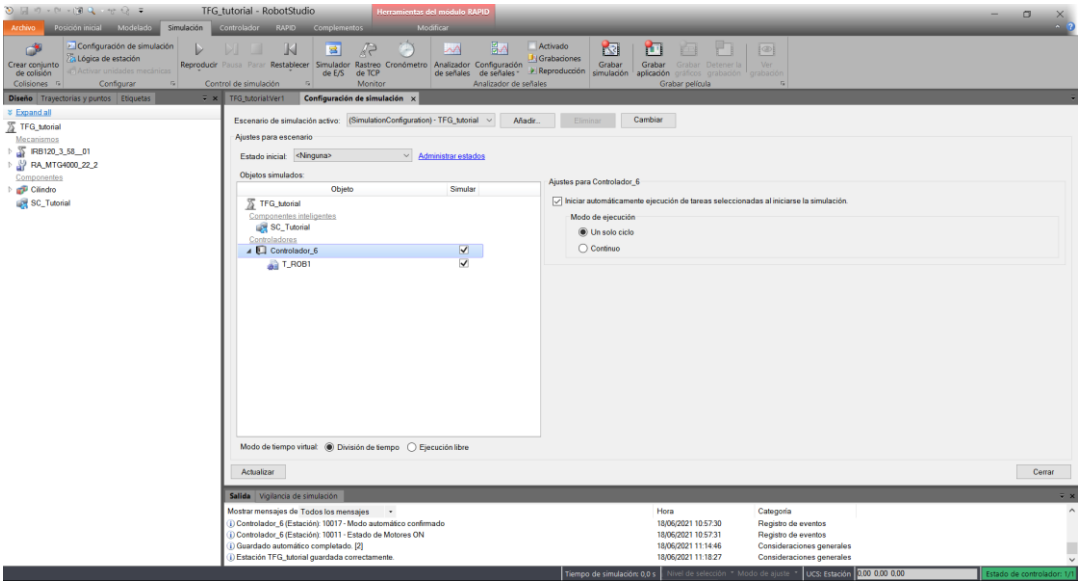



Figura 3.57: Configuración de la simulación, ajuste de Controlador_6

Otra opción, podemos cambiar de la simulación con los ajustes de T_ROB1 e indicamos que solamente haga la trayectoria seleccionada “Path_10”, o que realice el programa principal indicando “main”, como vemos en la figura 3.58.

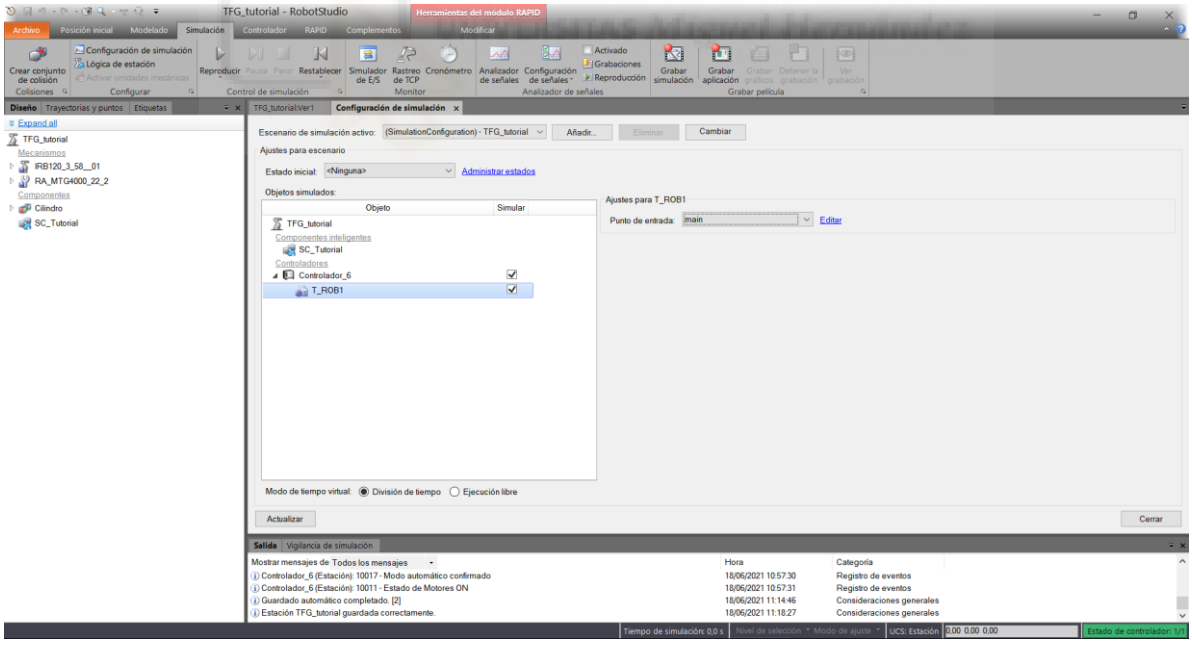



Figura 3.58: Configuración de la simulación, ajuste de T_ROB1

Además, una opción es la de cambiar la configuración de grabación de la simulación en “Archivo”, “Opciones” y “Grabadora de pantalla”.

En la figura 3.59, podemos reproducir la simulación en la pestaña “Simulación”, debemos de darle a “Reproducir”. También, podemos pausar la simulación y si vamos clicando varias veces en “Pausar”, se irá moviendo lentamente.

Otra opción es la de “Parar” la simulación y de “Restablecer”, de esta última forma, si se ha movido los objetos o el robot vuelvan al estado inicial de la simulación.

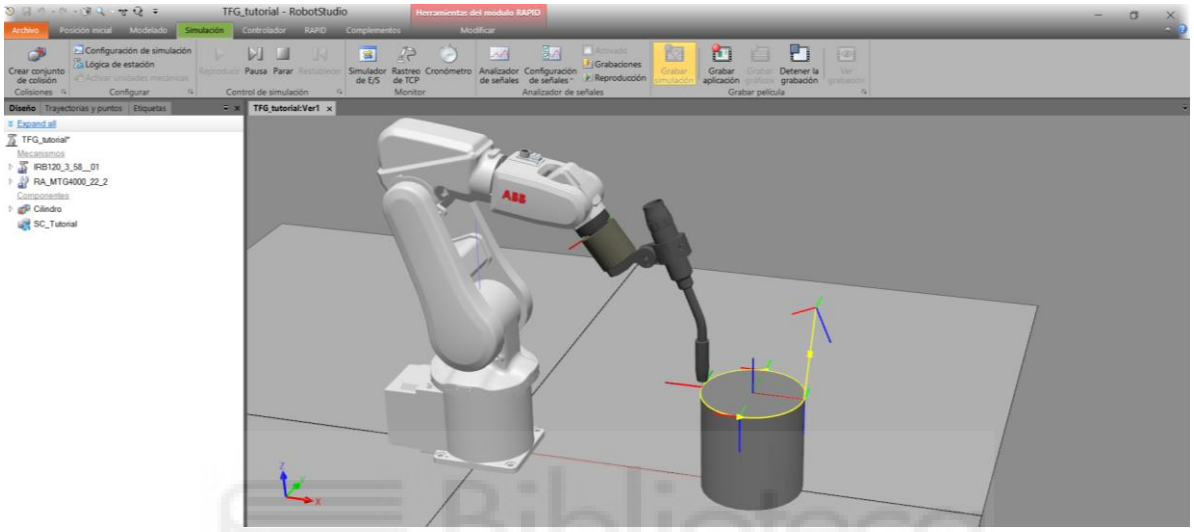



Figura 3.59: Simulación y grabación

# 4. IMPLEMETACIÓN DE UN PROCESO DE RECOGIDA DE PLANTILLAS DE ZAPATOS EN UNA LÍNEA AUTOMATIZADA CON MAQUINA CNC

## 4.1. Elección de los Robots

En la estación de recogida de las plantillas se va a instalar un pórtico fijo que se encontrara encima de la cinta transportadora que desplaza las planchas con las plantillas cortadas. En el pórtico, se situarán dos robots que sean capaces de realizar la tarea definida.

Los robots necesarios deben de cumplir una serie de especificaciones entre las que se destacan ser rápidos, muy precisos, gran capacidad de repetitividad, el suficiente alcance para abarcar la mitad de la cinta transportador y mesas laterales, y uno mínimo tiempo de ciclo durante la tarea de Pick&Place.

Basándonos en las características que deben de tener los robots, hemos escogido el robot ABB IRB360, ya que cumple todas las especificaciones y se trata de un robot muy utilizado en las tareas de Pick&Place, siendo uno de los más rápidos del mercado.

Una vez escogido el robot, en la tabla 4.1 podemos observar que existen diferentes modelos, con diversas versiones de tamaño y carga.

| Versiones IRB360 | Capacidad de carga (Kg) | Área de trabajo. Diámetro (mm) |
|---|---|---|
| IRB 360-1/1130 | 1 | 1130 |
| IRB 360-3/1130 | 3 | 1130 |
| IRB 360-1/1600 | 1 | 1600 |
| IRB 360-6/1600 | 6 | 1600 |
| IRB 360-8/1130 | 8 | 1130 |

Tabla 4.1: Versiones del robot ABB IRB 360

Entre todas las versiones del robot de la anterior tabla, se ha escogido el modelo de robot IRB 360-6 / 1600, mostrado en la figura 4.1. Se ha escogido este modelo ya que el área de trabajo que ofrece es la adecuada para la estación, al mismo tiempo que puede soportar grandes cargas en el caso que se utilice un material pesado para el diseño de plantillas.

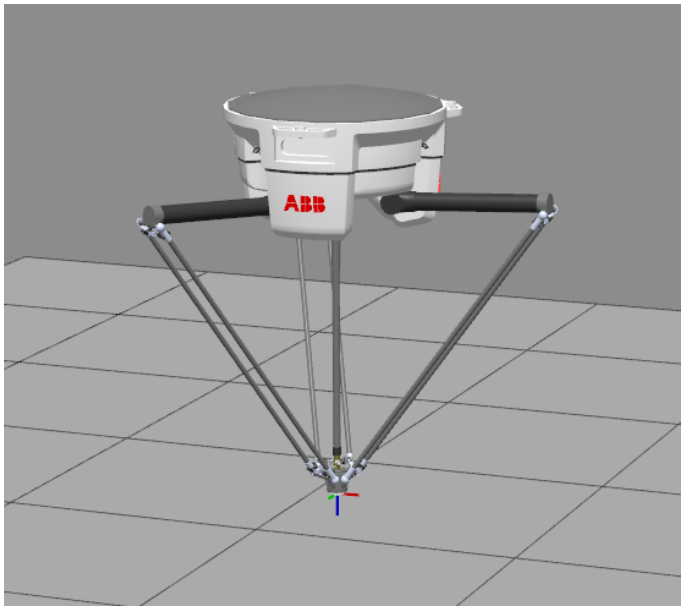



Figura 4.1: Robot ABB IRB 360 - 6 / 1600

## 4.2. Elección de la Herramienta

La herramienta elegida para el proyecto ha sido una herramienta una en forma de Ventosa, la cual permite coger las piezas creando el vacío entre la ventosa y la pieza, y soltarla una vez que el robot se encuentre en la posición deseada eliminando el vacío creado.

En los dos robots se han conectado la misma herramienta ya que los dos realizan la misma tarea de Pick&Place.

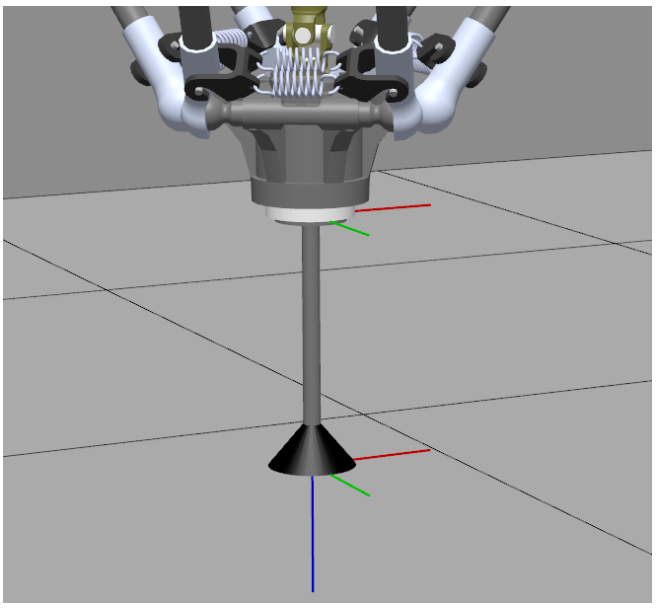


Figura 4.2: Herramienta de ventosa

## 4.3. Diseño de los sólidos de la estación

Los sólidos creados para la estación se han diseñado utilizando el software de CAD Autodesk AutoCAD 2020, los planos correspondiente a todos los sólidos se encuentran en el anexo 9.1.

En primer lugar, como se ha comentado en el apartado 4.2, la herramienta utilizada en la estación ha sido diseñada en AutoCAD, como podemos ver en la figura 4.3.

La herramienta ha sido creada de forma que la ventosa tenga la suficiente superficie como para poder crear el vacío con las plantillas que debe de coger. Además, se ha creada con una longitud determinada que la aleja del extremo efector del robot, lo que permite que no choque ninguna pieza con el robot.

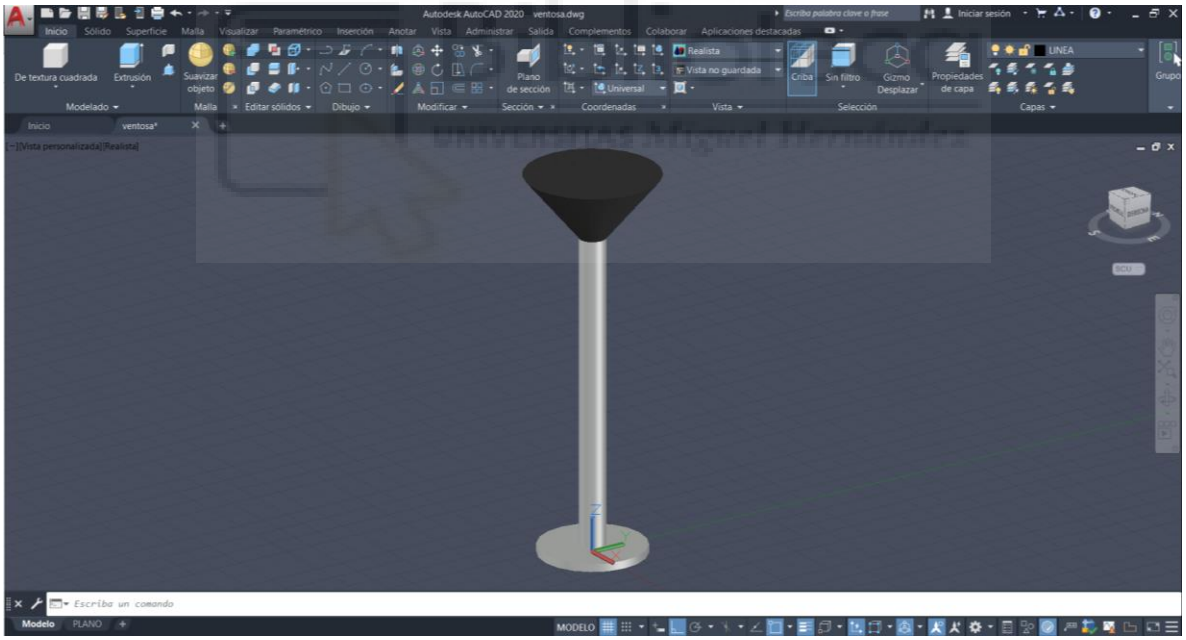



Figura 4.3: Herramienta Ventosa

El siguiente sólido diseñado ha sido el pórtico, mostrado en la figura 4.4. El pórtico es el soporte de los dos robots, que se encuentran situados sobre la cinta transportadora que mueve las planchas con las plantillas.

El pórtico ha sido diseñado con la altura necesaria para que los robots alcancen a todos los puntos durante el proceso de Pick&Place, y también con la anchura correcta para que quepa la cinta transportadora.
Se trata de un pórtico fijo el cual se puede anclar al suelo con las chapas que se encuentran en los extremos de las patas.

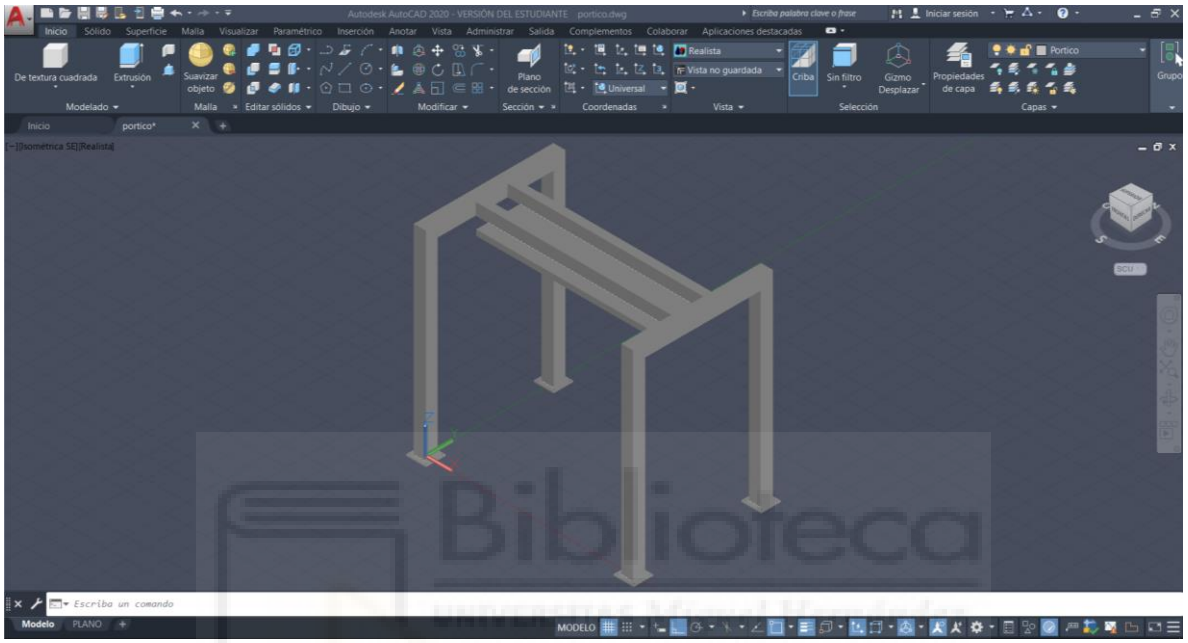


Figura 4.4: Pórtico fijo

Otro sólido diseñado en CAD fueron las mesas situadas a los lados de la cinta transportador y el pórtico. Estas mesas fueran diseñadas para que los robots depositarán en ellas las plantillas una vez clasificadas.

La mesa ha sido diseñada de forma que tiene una bancada grande en la parte superior y dos cajones para almacenaje. En la estación se han situado una cada lado, cada una de ellas para el robot de dicho lado. Podemos ver una imagen del diseño de la mesa en la figura 4.5 a continuación.

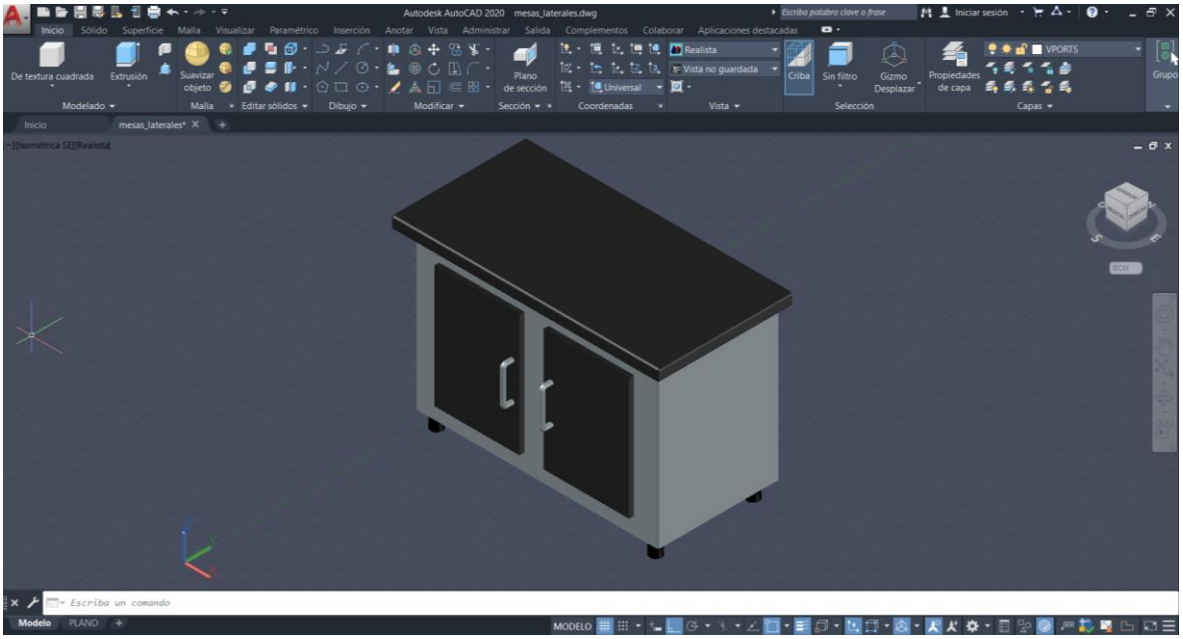


Figura 4.5: Mesa lateral

El ultimo sólido creado ha sido las plancha de plantillas usadas en el proyecto.

A partir de las medidas de la cinta transportadora, se han creado dos modelos de plantillas, una donde solo ha plantillas de pie derecho y otra con plantillas de pie izquierda. Además, en la plancha han sido diseñadas tres tamaños distinto de plantillas para que los robot pueden clasificarlas por tamaño.

En la figura 4.6 podemos ver ambos diseños de las plantillas.

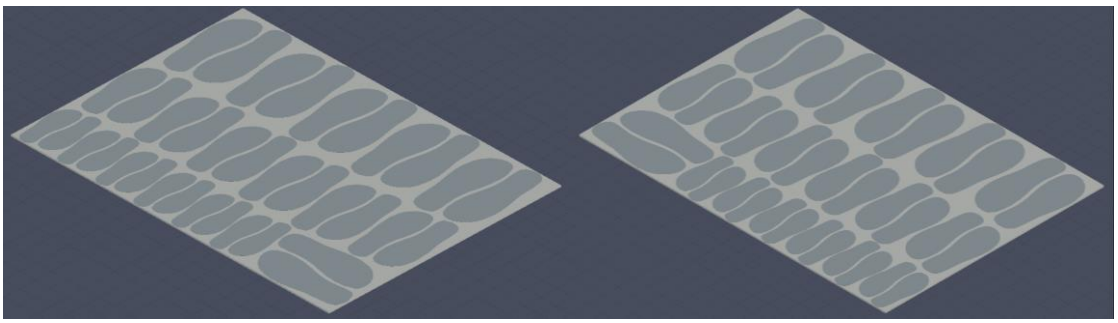


Figura 4.6: Plantillas (pie izquierdo y pie derecho)

### *4.3.1. Otros solidos importados*

La máquina CNC ha sido importada y añadida a la estación, gracias a la aportación de Carlos Pérez Vidal de una estación de ejemplo la cual contenía la máquina. Podemos verla en la figura 4.7.

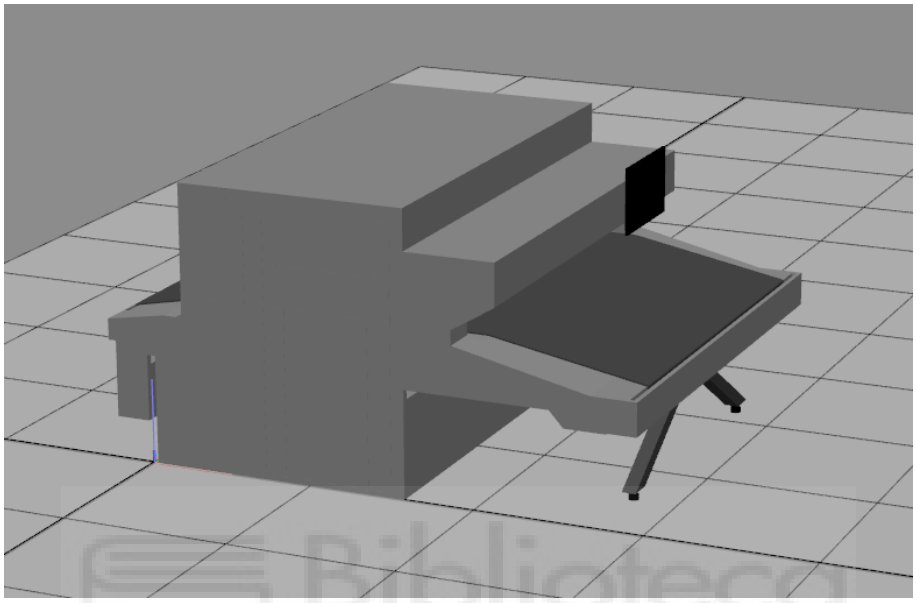


Figura 4.7: Máquina CNC

La máquina está formada por un tapiz rodante sobre el que se desplazan las planchas que van a ser cortadas, dentro se encuentra una herramienta de corte que corta las planchas para que salgan por la cinta con el diseño que se la haya indicado.

### *4.3.2. Otros solidos añadidos desde la biblioteca de RobotStudio*

El resto de los sólidos utilizados en la estación has sido importado desde la biblioteca de RobotStudio.

Al final de la cinta se han colocado dos pallets, uno para cada lado de la cinta ya que vendrán dos planchas al mismo tiempo. Lo ponemos ver en la figura 4.8.

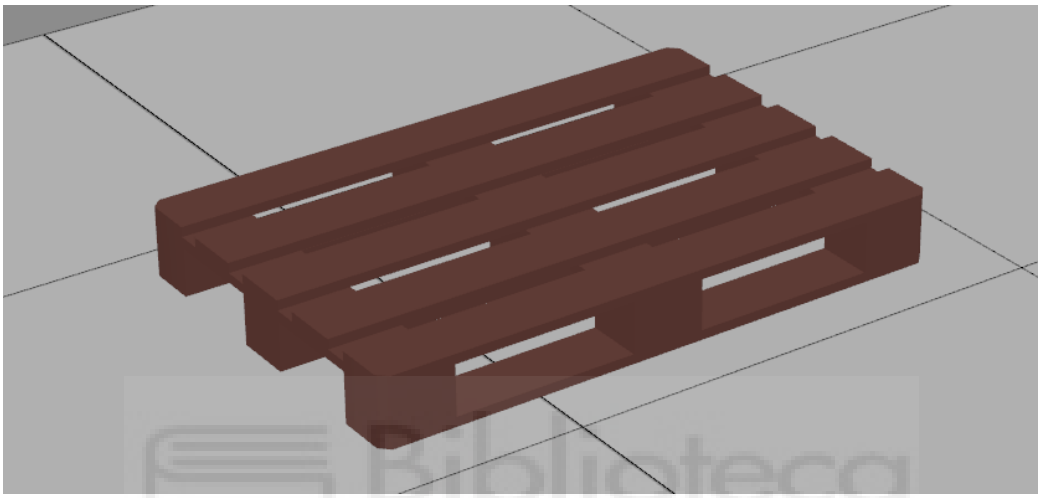


Figura 4.8: Pallet

En la figura 4.9 podemos ver la caja, que se colocaran encima de los pallets. Creada mediante solidos en la pestaña de “Modelado”.

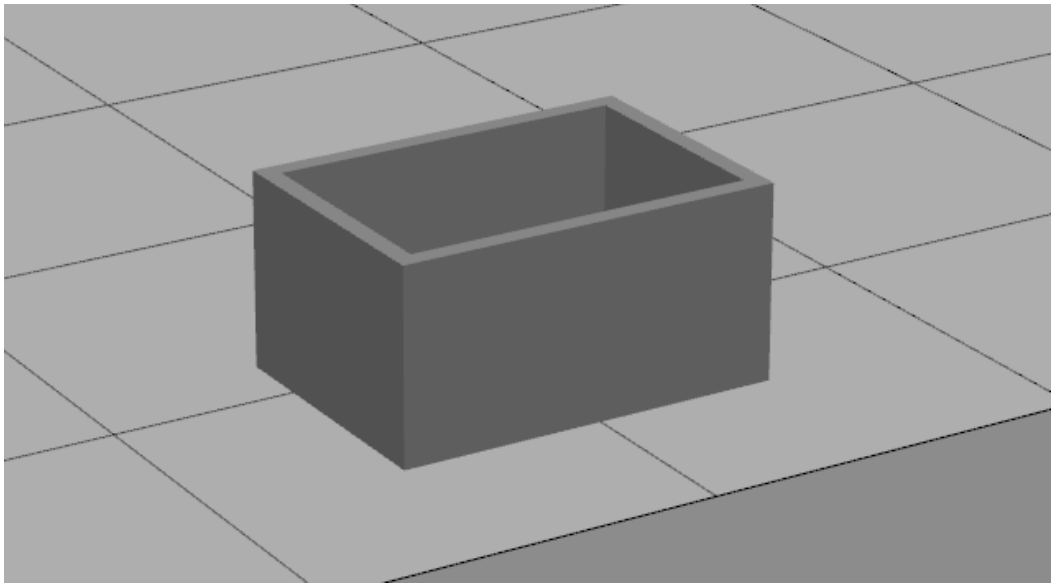


Figura 4.9: Caja

Las vallas, mostradas en la figura 4.10, delimitan la zona a la que no se puede pasar si los robots están en funcionamiento.

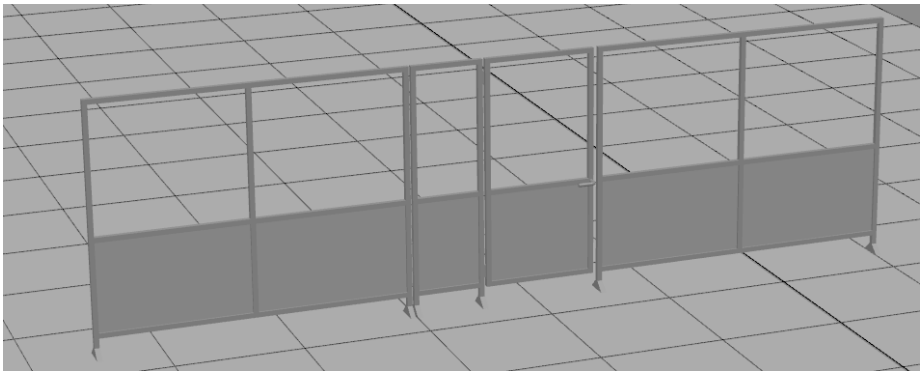


Figura 4.10: Vallas

La estación tiene un controlador para ambos robots como el de la figura 4.11.

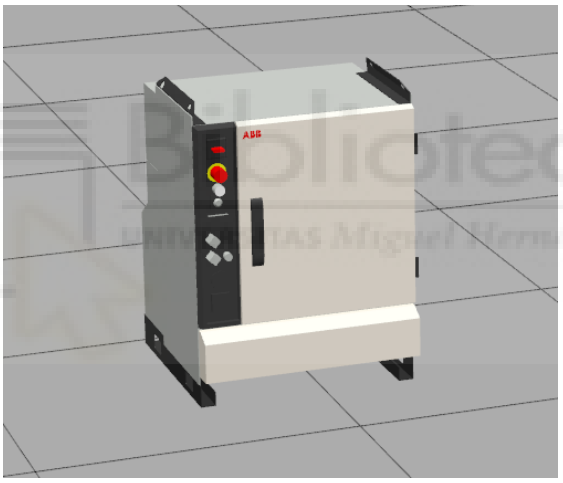


Figura 4.11: Controlador

El FlexPendant para el control del robot, se puede ver en la figura 4.12.

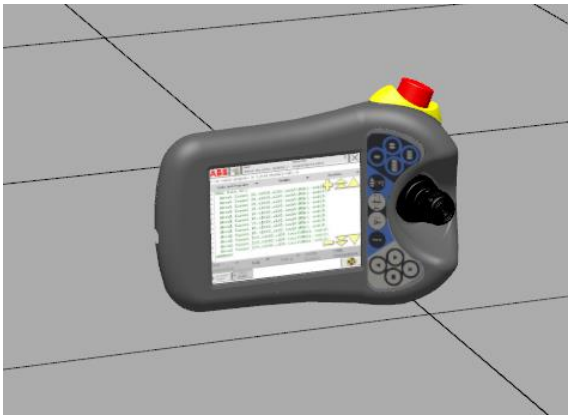


Figura 4.12: FlexPendant

Se muestra también un operario en la estación en la figura 4.13. Será el encargado de controlar el correcto funcionamiento de los robots, así como de retirar las plantillas una vez hayan sido recogidas y clasificadas.

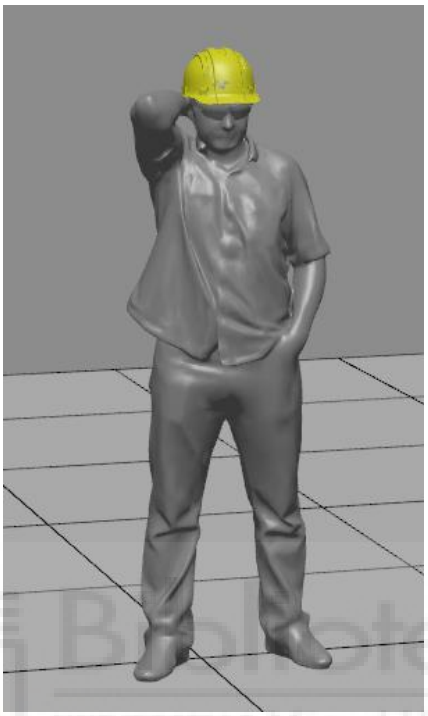


Figura 4.13: Operario

Por último, se ha colocado una carretilla elevadora con la que el operario podrá recoger los pallets con las cajas que contendrán la parte que se desecha.

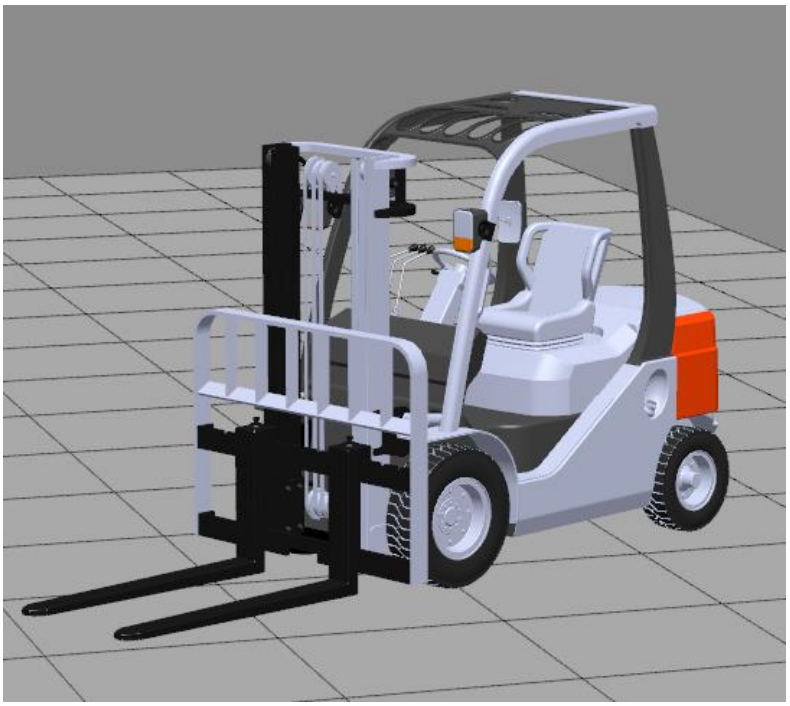


Figura 4.14: Carretilla elevadora

## 4.4. Creación de la estación en RobotStudio

### *4.4.1. SC_Cintas*

Uno de los Smart Components creados es el de las cintas, que se muestra en la figura 4.15. y está formado por los siguientes componentes:

- MoveAlongCurve: Mueve un objecto ( Object ) a lo largo de una curva geométrica (usando un constante Offset), se le pude indicar una velocidad determinada ( Speed (mm/s) )y en el apartado ( WirePart ) se le indica el cable que se ha creado para recrear el movimiento de las planchas sobre las cinta, este cable se trata de una spline creada a lo largo de la trayectoria que debe de seguir las planchas. La casilla de ( KeepOrientation ) se ha dejado sin seleccionar ya que así conseguimos que la plancha cambie de orientación a lo largo del movimiento de las cintas dependiendo de las pendientes de estas. El movimiento se inicia cuando la señal MARCHA de entrada al Smart Component es activada.

  Encontramos cuatro componentes de MoveAlongCurve diferentes. Los números 1 y 3 se corresponden con las bases de las planchas antes de ser cortadas y su movimiento es cancelado cuando el Sensor_CNC es activado. Los números 2 y 4 son para las planchas ya cortadas, las cuales son canceladas cuando el Sensor_cinta se activa.
- Hide: hace un objeto ( Object ) invisible gráficamente. Encontramos dos componentes en el SC, utilizados para las bases de las planchas que no han sido cortadas, estas planchas son visibles inicialmente y una vez que el Sensor_CNC es activado las planchas se hace visibles.
- Show: hace un objeto ( Object ) visible gráficamente. También encontramos dos componentes, pero ahora están asociados a las planchas que ya están cortadas y en las que se pueden ver las plantillas, las cuales inicialmente no son visible y una vez que el Sensor_CNC se activa se hace visibles.

Gracias a la utilización de los componentes Hide y Show se consigue representar el proceso de corte en la maquina CNC, teniendo al inicio del tapiz rodante la planchas sin contar y una vez que se encuentran en el interior de la maquina cambia a las planchas que están cortadas.

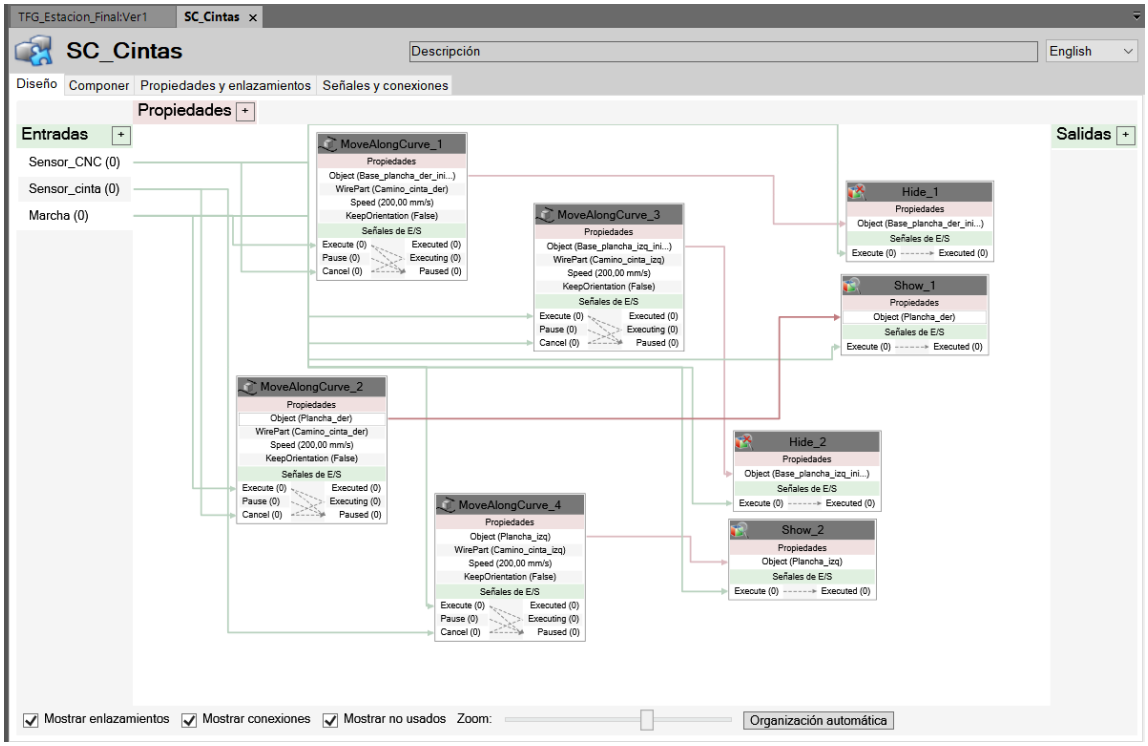



Figura 4.15: SC_Cintas

Se añaden tres entradas al Smart Component de las cintas, la entrada “Sensor_CNC” que es la señal utilizada para activar los componentes de Hide y Show, así como el de cancelar el movimiento de alguna de las planchas cuando esté a “1”. La señal de “Sensor_cinta” se utiliza cuando se encuentre a “1” para cancelar el movimiento del resto de las planchas cuando se encuentren en la posición de recogida. La última señal, la de “Marcha”, es utilizada para iniciar el movimiento de todas las planchas usando los componente de MoveAlongCurve cuando esté a “1”.

### *4.4.2. SC_ventosa_tool_izq*

Este componente se encarga de realizar la acción de crear el vacío y coger una pieza con la ventosa del robot situado a la izquierda en la cinta, y luego soltar dicha pieza dependiendo del valor de las entradas.

La entrada que se ha añadido al Smart Component es la de “vacío”, con ella indicamos que se debe de cativa un “LineSensor”, este sensor se encuentra en el extremo de la ventosa y con el que se detecta la pieza a coger.

El sensor se ha creado indicando una longitud de 59 mm, en los apartados de ( Start ) y ( End ), además, se le ha indicado un radio ( Radius ) de 2 mm. La propiedad de ( SensedPart ) la cual indica que solido ha sido detectado ha sido conectada al componente “Attacher” en la propiedad ( Child ), así como la salida que ofrece el LineSensor de ( SensorOut ) que se activa si ha detectado algún objeto y que activara el “Attacher”.

El componente “Attacher” es el encargado de crear el vacío y coger la pieza dejándola pegada al objeto indicado en la propiedad ( Parent ), que en este caso es la ventosa del robot izquierdo “Mi_Ventosa_izq”.
El componente “Detacher” es el encargado de soltar la pieza y se activa cuando la señal de entrada se encuentra a “0”, tras pasar por un puerta NOT, donde se convierte a “1”, de esta forma cuando no se deba de coger pieza siempre esta activado. La propiedad ( Child ) está conectada al Attacher para así cuando se deba de soltar el objeto sea el que esté cogido por la ventosa.

Se utiliza un “SimulationEvents”, para hacer el ( reset ) de un “LogisSRLatch”, el cual da el valor a la salida del SC “Vacuostato”. Al reset también está conectado la salida del Detacher y al ( Set ) la salida del Attacher.
En la figura 4.16 podemos ver la distribución del SC_ventosa_tool_izq.

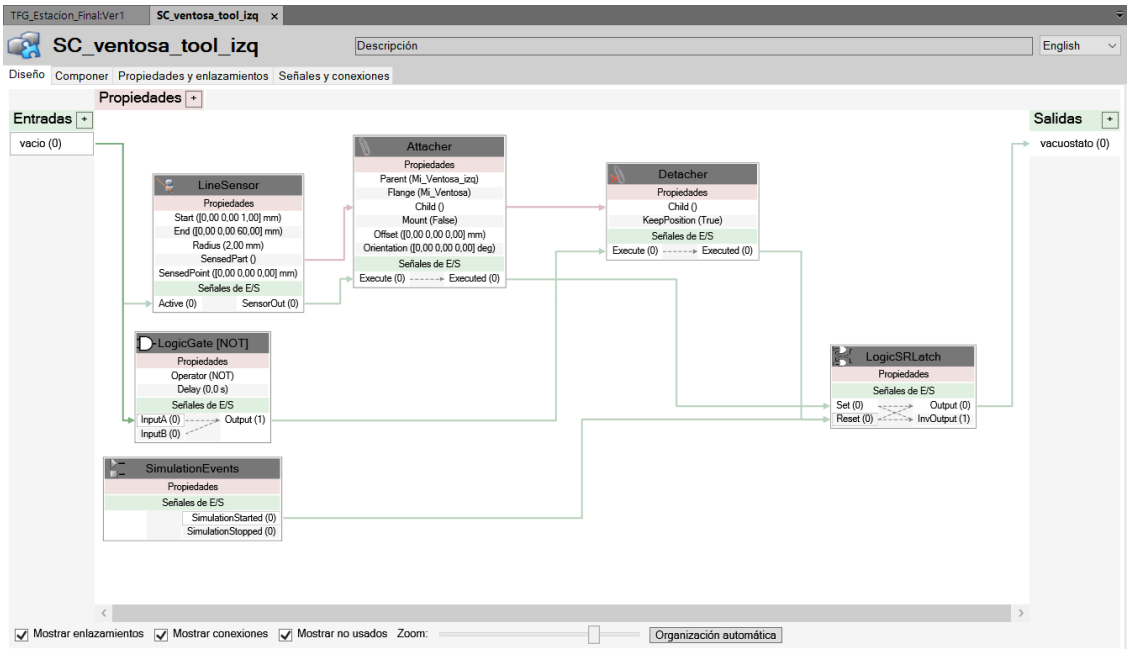



Figura 4.16: SC_ventosa_tool_izq

### *4.4.3. SC_ventosa_tool_der*

Este componente se encarga de realizar la acción de crear el vacío y coger una pieza con la ventos del robot situado a la derecha en la cinta, y luego soltar dicha pieza dependiendo del valor de las entradas.

Los componentes que contiene este Smart Component son los miso que contiene el SC_ventosa_tool_izq, por lo que las conexiones y entradas y salidas son idénticas.

En la figura 4.17, se puede ver la distribución del Smart Component.

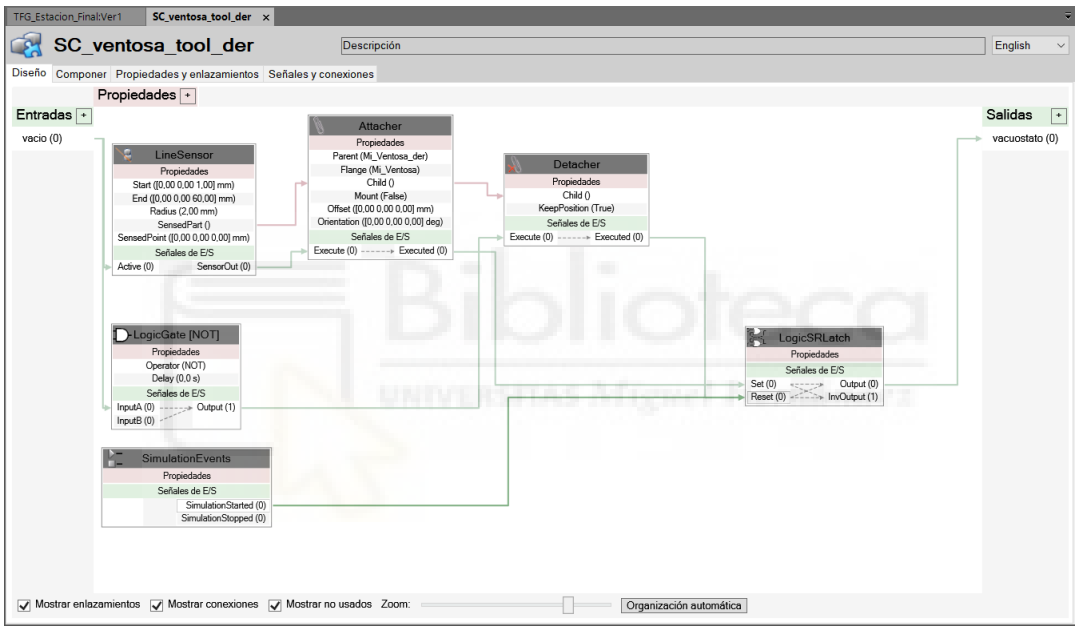



Figura 4.17: SC_ventosa_tool_der

### *4.4.4. Controlador y Lógica de la Estación*

El controlador de los dos robots IRB360_6_1600_der y IRB360_6_1600_izq tendrá 4 señales, 2 de entrada y otras 2 de salida. Las señales de entrada al controlador serán las de “Sensor_CNC” y “Sensor_cinta” conectadas a dos los dos sensores llamados con esos nombres. El Sensor_CNC estará colocad en el interior de la maquina CNC y con el que se indica que la plancha se encuentra en proceso de corte.

El Sensor_cinta se encuentra en la posiciones de recogida de las plantillas justo debajo de los dos robots y con el que se indica que las planchas cortadas están en posición para realizar la tarea de Pick&Place y clasificación por parte de los robots.

Las señales de salida del controlador son “Coger_ven_izq” y “Coger_ven_der” . Como su nombre indica cada una de ellas está destinada a su respectivo robot, la señal Coger_ven_izq se activará a 1 cuando el robot situado a la izquierda en la cinta se encuentre en el lugar para coger una plantilla y se pondrá a 0 cuando esté en el sitio donde debe de soltar la plantilla.

Esta señal está conectada a la entrada del SC_ventosa_tool_izq “vacío”. La otra señal de salida del controlador, llamada Coger_ven_der realiza la misma función que la otra pero ahora asociada al robot situado a la derecha en la cinta, esta señal esta conectad a la entrada del SC_ventosa_tool_der “vacío”.

También, encontramos en la lógica de la estación un componente llamado SimulationEvents el cual al iniciar la simulación se activa su salida “SimulationStarted” y está conectada a la señal de entrada al SC_Cintas “Marcha.” En la figura 4.18 se puede ver la lógica de la estación con sus respectivas conexiones.

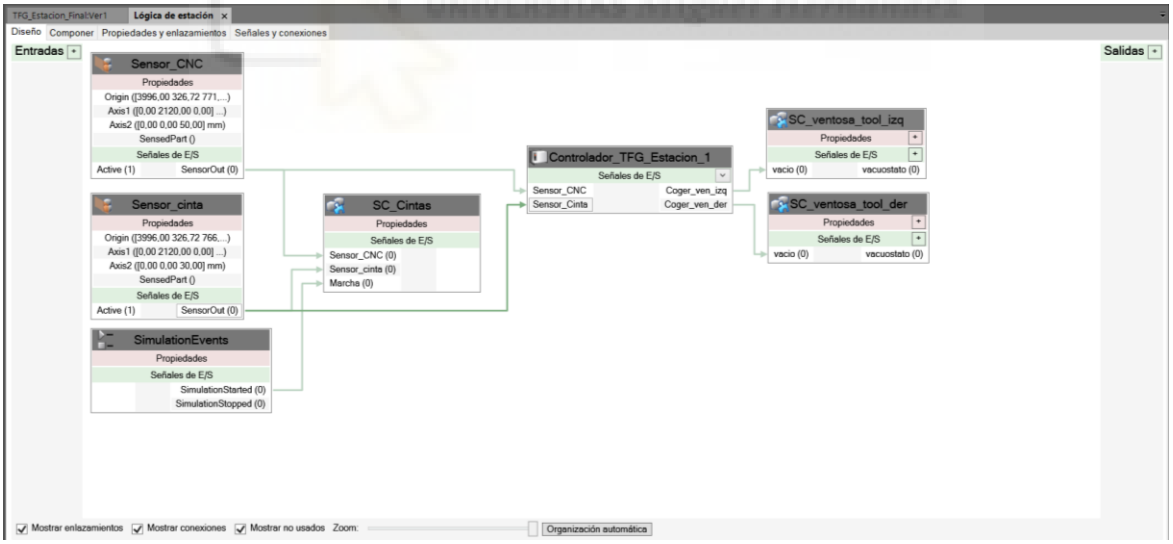



Figura 4.18: Lógica de la estación

### *4.4.5. Programación en RAPID*

La última parte que quedaría para el diseño completo de la estación sería la programación de los robots en PARID dando valores a las distintas entradas y salidas del controlador.

En el controlador encontramos dos programas distintos, uno para el robot IRB360_6_1600_iz colocado a la izquierda de la cinta transportadora, llamado T_ROB1 y otro programa para el robot IRB360_6_1600_der, colocado a la derecha, que se llama T_ROB2.

Ambos programas se han creado mediante diferentes subprogramas PROC. Se han creado un subprograma para la recogida de cada una de las 36 plantillas que hay cortadas en la plancha, también tenemos un subprograma para la colocación de cada uno de los tres tamaños distintos que hay cortados de plantillas, y por último, el subprograma principal llamado “main”, en el que se engloba todas las llamadas a los subprocesos.

Cada uno de los subprogramas creado para las plantillas se llaman desde Planta1, Planta2, … hasta Planta36. En cada uno de ellos, se programa el movimiento del robot hasta el centro de cada una de las plantillas, se activa la salida del controlador “Coger_ven_izq”, la cual activa el Smart Component para coger dicha plantilla, y por último se realiza un movimiento MoveL Offs, para que una vez que una plantilla ha sido cogida, el robot realice un movimiento vertical desde la posición que se encuentra para evitar el choque entre la plantilla y la plancha.

Los tres subprogramas creados para la colocación de las plantillas dependiendo de su tamaño son llamados : “Dejar_plantilla_grande”, “Dejar_plantilla_mediana” y “Dejar_plantilla_pequeña”.
En cada uno de ellos lo que se hace primero es sumar +1 una variable donde se contabiliza el número de plantillas de ese tamaño que han sido recogidas, y se mueve el robot a la posición llamada “Inicio”.

A continuación, comienza el proceso de dejar la plantilla, se empieza con mover el robot a una posición de aproximación al lugar donde se debe de depositar la plantilla, ahora, el robot se desplaza verticalmente para depositar la plantilla encima de la que se ha depositado anteriormente.

Para hacer esto, se ha programado el movimiento mediante un MoveL Offs, creando una variación en a altura ala que se debe de dejar la plantilla multiplicando el alto de cada una de las plantillas ( 5,1mm ) por el número de plantillas que se han recogido de ese tamaño anteriormente, consiguiendo así apilarlas todas en una misma columna.
Una vez que se debe de dejar la plantilla, se pone la salida del controlador “Coger_ven_izq” a 0, de esta forma el robot solará la plantilla. Finalmente, el robot se retira a la posición de aproximación y por último, a la de inicio.

Por último, el subprograma principal “main” es donde se llama a los distintos subprogramas con un orden de plantilla diseñado para que los dos robot no se choquen en ningún momento. Lo primero que se hace en el main es poner los tres contadores de plantillas a 0 para que las primeras plantillas sean depositadas sobre la mesa, luego, se pone la salida del controlador “Coger_ven_izq” a 0 y se mueve al robot a la posición de inicio. Uno de los puntos importante viene ahora, donde mediante una instrucción de “WaitDI Señor_cinta,1” espera hasta que el Sensor_cinta detecte que la plancha ha llegado al lugar de recogida de las plantillas. El resto del subprograma se basa en llamar a la plantilla mediante l instrucción “PlantaX;” donde X es el número de plantilla y después llama al subprograma de colocación dependiendo del tamaño.

El programa del robot situado a la derecha está programado con la misma estructura, donde tenemos un subprograma principal “main”, tres subprogramas para la colocación de las plantilla dependiendo de su tamaño y uno para cada una de las plantillas. Las instrucciones durante el programas son similares, ahora en el main el orden de recogida de las plantillas en distinto para conseguir que no se choquen. La señal usada para darle un valor a la hora de coger y solar con la ventosa las plantillas es la salida del controlador “Coger_ven_der”.

En el anexo 9.3 podemos encontrar ambos códigos RAPID de la estación automatizado con los dos robots paralelos.

A continuación, tenemos un diagrama de flujo que explica el programa diseñado para la estación, siendo el mismo para ambos robots.

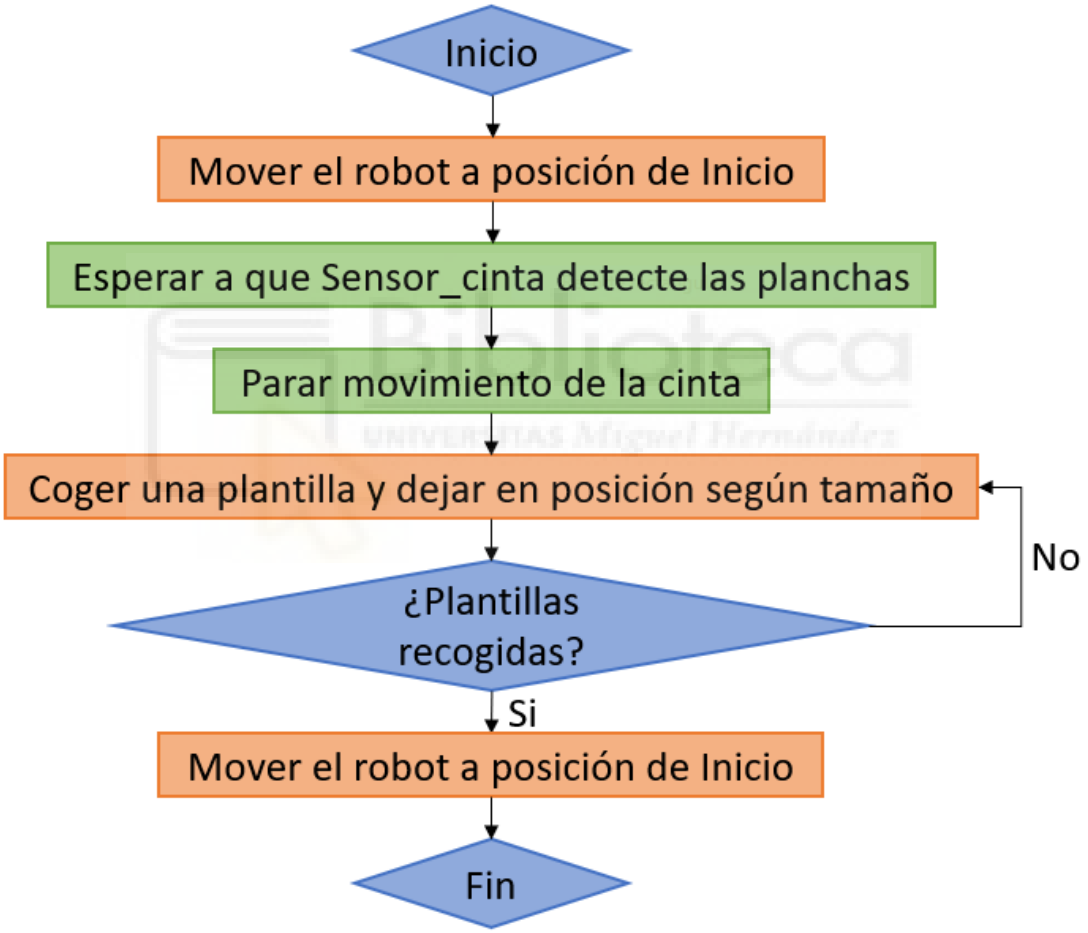



Figura 4.19: Diagrama de flujo del programa

# 5. DESARROLLO DE LA COMUNICACIÓN AUTOCAD-PYTHON-ROBOTSTUDIO.

En esta segunda parte del proyecto, el objetivo era conseguir una mayor automatización del proceso a la hora de obtener datos de las plantillas que los robots deben de coger y clasificar. Anteriormente, las posiciones de las plantillas eran definidas manualmente en RobotStudio una a una y eran creados los movimientos que debía de realizar el robot para alcanzar dichas posiciones.

Ahora, utilizamos el archivo que se obtiene directamente de la maquina CNC, con el diseño de las plantillas que ha sido cortado sobre una plancha. Dicho archivo se trata de un archivo con la extensión .DXF, es decir, se puede abrir con AutoCAD y a partir de él obtener los datos. Los archivos con los distintos modelos de plantillas han sido aportados por una empresa de Italia llamada Comelz CAD CAM España S.L.U. Primeramente, la empresa nos aportó unos diseños donde los cortes eran solamente plantillas de zapato ( fichero N04801.dxf ), pero posteriormente y con el objetivo de realizar la programación del proceso aun avanzada y que no fuera solo para este tipo de corte, le pedimos que nos compartiera otros modelos de corte en los que se cortan más partes de las que componen el zapato (fichero N02027.dxf).

La forma de conseguir obtener los datos de AutoCAD ha sido mediante la programación de un script en el cual se llama a diversos comandos con los que se consigue obtener los datos necesarios, este proceso será explicado más adelante en el apartado 5.1 de la memoria.

Lo más sencillo para poder acceder a AutoCAD y ejecutar todos los pasos era utilizando el lenguaje de programación de Python. Este lenguaje contiene diversas librerías con las que se pueden realizar infinidad de acciones con AutoCAD.

La última parte es la de comunicar RobotStudio y Python, para realizar esto se utilizó un Socket, con el que se envía un mensaje entre ambos programas, siendo Python quien avisa a RobotStudio que los datos ya han sido obtenidos desde AutoCAD. A continuación, vamos a desarrollar cada uno de los programas que se han programado.

## 5.1. Obtener datos de AutoCAD

Como se ha comentado anteriormente, desde AutoCAD se deben de obtener los datos de las plantillas según el modelo que se haya obtenido de la maquina CNC.

Para obtener los datos, la forma más sencilla es utilizando el comando EXTRACDAT de AutoCAD. Con él, se obtiene un archivo de texto con los datos que queremos de las plantillas, ya que el comando permite elegir los datos a extraer, lo que hace más sencillo acceder a ellos desde Python.

Los datos necesarios que se deben de extraer son. las coordenadas X e Y, y la rotación de cada una de las plantillas, así como el nombre del modelo de cada plantillas, con el cual se podrá clasificar en plantillas entre pie derecho o pie izquierdo, en el caso del modelo N04801, y ser separados por modelos, en el caso del archivo N02027. A continuación, se va a explicar los pasos a seguir para obtener los datos de las plantillas, los cuales serán exportados a un archivo de texto.

- **Comando EXTRACDAT, crear una plantilla de datos:**

Al usar el comando EXTRACDAT de AutoCAD para extraer los datos es necesario haber creado una plantilla de extracción de datos anteriormente, y con los parámetros necesarios para que el archivo de texto que se creará nos muestre los datos necesarios.

Lo primero que se debe de hacer es abrir el archivo del que se quiere obtener los datos y ejecutar el comando EXTRACDAT. Se trata de un comando con 8 pasos, los cuales nos permiten editar la plantilla para que se extraigan los datos necesarios de ciertos elementos que hay en el archivo de AutoCAD.

En la figura 5.1 podemos ver el primer paso, en el que se debe seleccionar la opción de crear un nueva plantilla. A continuación, le damos un nombre y daremos aceptar.

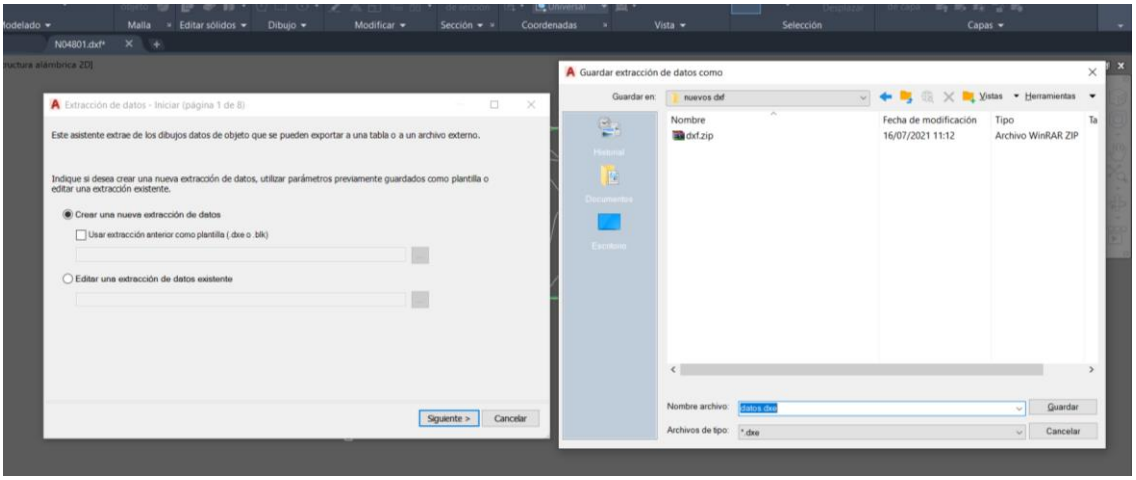



Figura 5.1: Paso 1, creación de nueva plantilla

En el paso dos, se debe de seleccionar los dibujos de los cuales se pueden extraer datos, para ello, seleccionamos la casilla “Incluir dibujo actual” y damos a siguiente, esta parte debe ser realiza en todo los archivos que nos han sido aportados por la empresa, para que no haya problema a la hora de extraer los datos, como se puede observar en la figura 5.2, donde ya se han añadido todos los archivos compartidos.

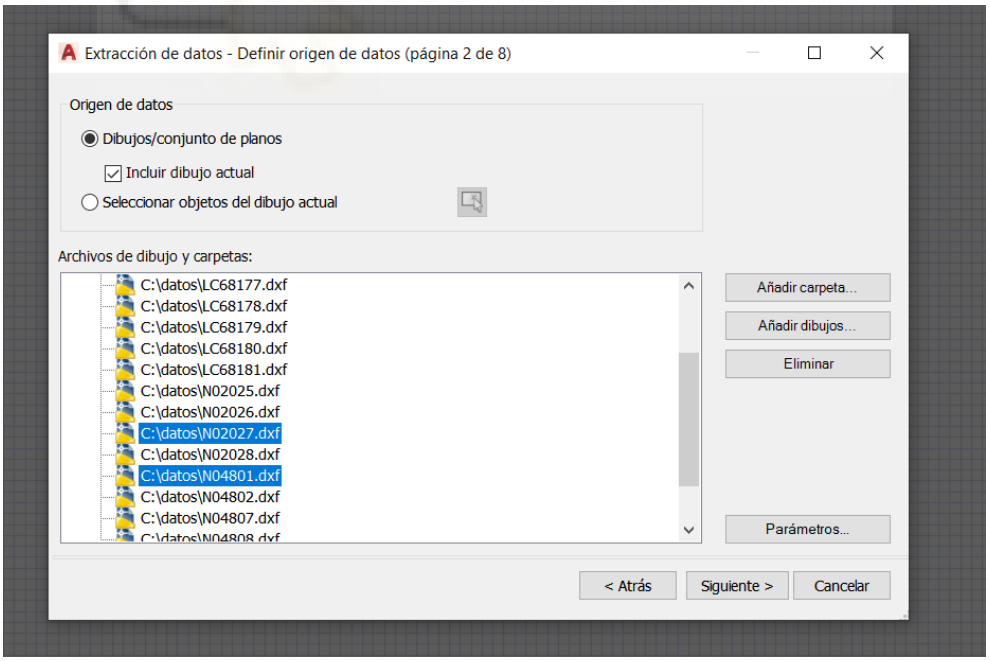



Figura 5.2: Paso 2, selección de los dibujos

En el paso 3 del comando, se debe de seleccionar que objetos del dibujo son de los que se quiere obtener los datos, para poder añadir todos los posibles modelos de los que son necesarios los datos se deben de tener todos los archivos abiertos en AutoCAD y a continuación se debe de seleccionar la casilla de “Mostrar solo bloques” y no haber seleccionado la de “Mostrar sólo objetos actualmente en uso”, de esta forma añadiremos todos los bloques, como vemos en la figura 5.3.

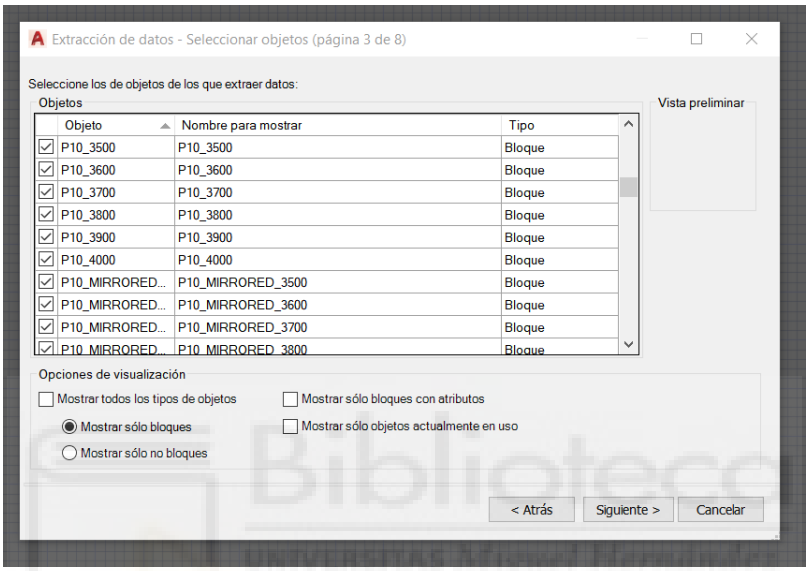



Figura 5.3: Paso 3, selección de objetos

El paso 4 es el más importante, en este paso se elige los datos que queremos que se extraigan al archivo de texto. Primero, en la parte derecha seleccionamos los filtros de “Geometría” y “Varios”, y luego, se seleccionan solo las casillas necesarias, en esta caso son las de “Posición X”, “Posición Y” y “Rotación”, como vemos en la figura 5.4.

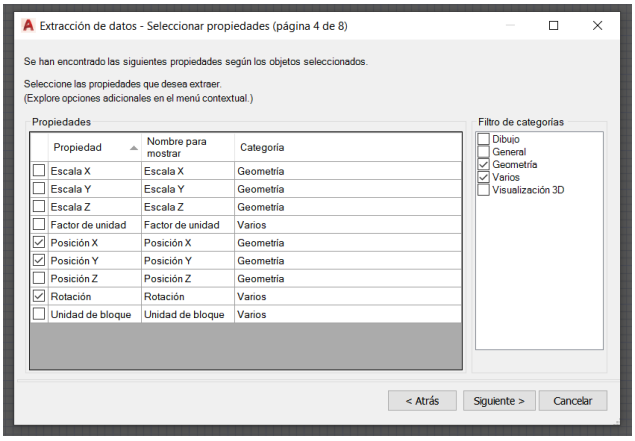



Figura 5.4: Paso 4, selección de los datos a extraer

En el paso 5 ( figura 5.5 ), se puede ver una vista previa de los datos que serán obtenidos, en este caso son los datos del fichero N04801.

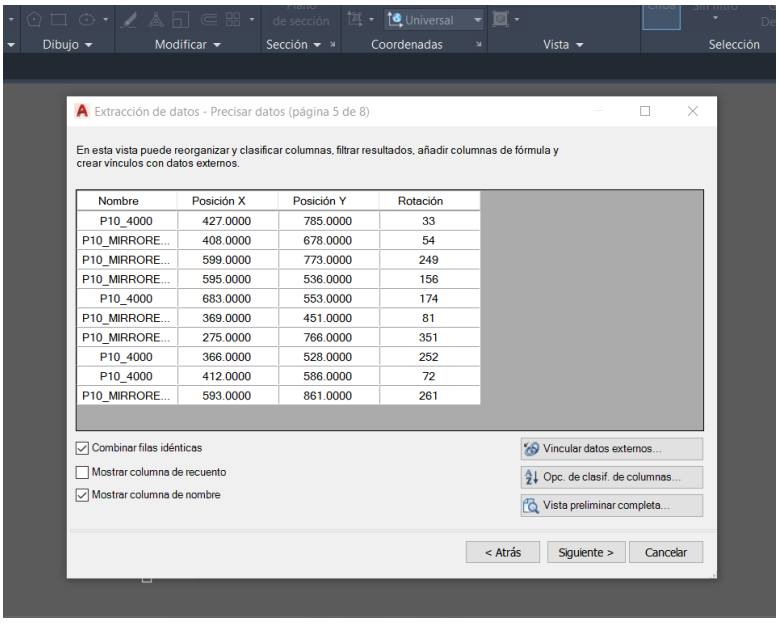



Figura 5.5: Paso 5, vista previa de datos

En el siguiente paso, el 6, es cuando se selecciona que queremos que lo extraiga como un archivo de texto. Para ello, se elige la casilla “Salida de datos a un archivo externo” y se abre el desplegable, donde podemos guardarlo con el nombre que se decida.

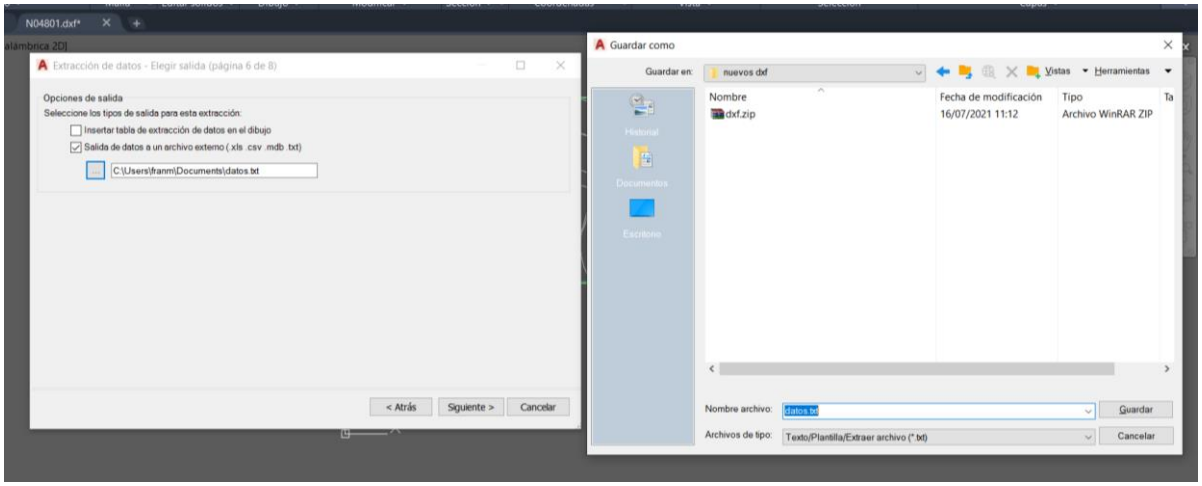



Figura 5.6: Paso 6, guardar en archivo externo

Finalmente, se pasa al paso 8, donde se finaliza el comando y son obtenidos los datos y guardados en el archivo de texto externo indicado.

## 5.2. Programación en Python

Python ha sido la forma de la que se ha podido comunicar RobotStudio y AutoCAD, hacer el proceso de extracción de datos y ser transmitidos a RobotStudio de la forma más rápida y directa, por lo que se ha conseguido el objetivo principal de esta parte. A continuación, se pasa a desarrollar el programa que se ha realizado en Python, el cual se puede encontrar en el anexo 9.3.4.

Lo primero que se ha definido han sido las diferentes librerías de Python a usar durante el programa. Siendo la conexión entre Python y RobotStudio lo siguiente que se ha programado, definiendo el socket con un host y un port, los cuales deben de ser los mismos en el programa de RAPID para poder crear la conexión. Tras haber creado la conexión, el programa espera a obtener una respuesta por parte de RobotStudio que confirme la conexión.

Lo siguiente que hace el programa, es pedir al usuario que introduzca por pantalla el nombre del modelo del que se quieren obtener los datos y que se va a realizar el proceso de Pick&Place, almacenando lo que se introduce en una variable llamada “dxf”. Con ella se crea una lista de datos, en la que se define el script de AutoCAD que realiza el proceso de abrir el archivo, ejecutar el comando EXTRACDAT definido en el anterior apartado pero ahora de forma que no aparece la venta emergente del comando y al que solo se debe pasar la ruta donde se encuentra la plantilla de extracción de datos explicada y creada en el apartado 5.1.1.

Una vez que la lista con el script está definida con los datos necesarios, se debe de crear el script. Para ello, se realiza de la misma forma que si se abriera un fichero de texto, pero la extensión será la de un script de AutoCAD (.scr), y en el que es escrito la lista “dat” que contiene los pasos a realizar en AutoCAD.

El siguiente paso que realiza el programa es abrir AutoCAD, haciendo uso de la librería subprocess de Python y con la que podemos ejecutar cualquier programa del ordenador si indicamos la ruta del ejecutable de dicho programa. Una vez que abre AutoCAD, ejecuta el script que creado anteriormente.

El programa de Python no continúa ejecutándose hasta que no finaliza el proceso de extracción de datos de AutoCAD, así que una vez que los ha extraído, cierra AutoCAD. La siguiente parte del código, lo que hace es abrir el archivo de texto con los datos para almacenarlos en una lista llamada “datos”, a excepción de la primera línea, en la que no hay ningún dato necesario y se salta.

A continuación, el programa pasa a crear un nuevo archivo de texto. Este será creado en la carpeta llamada “HOME”, que se encuentra dentro del controlador de RobotStudio, ya que es el único lugar del que se puede leer datos desde RAPID.
En este nuevo archivo de texto llamado “data.txt” se escribirán los datos que hemos extraído de AutoCAD y a los que se han realizado algún cambio, como eliminar los espacios o paréntesis que haya en el nombre del modelo o eliminar los decimales ya que no son necesarios de las coordenadas. Posteriormente, se escribe en cada línea el nombre, la coordenada X, la coordenada Y, y la rotación de cada una de las plantillas.

Una vez que se ha creado este último fichero, se vuelve a enviar una mensaje desde Python a RobotStudio con el que se le dice al robot que puede comenzar a leer los datos y comenzar el proceso de Pick&Place.

Finalmente, los ficheros “datos.txt” y “get_data.scr” son eliminados para que no haya problema al crearlos la siguiente vez que se ejecute el programa.

## 5.3. RobotStudio

### *5.3.1. Estación de RobotStudio*

La estación que se ha utilizado para esta parte del proyecto se trata de la misma estación utilizada en la primera parte del proyecto, pero ahora solamente se ha programado el proceso de Pick&Place con dos modelos de plantillas ( N04801 y N02027 ), ya que el objetivo principal de esta parte del proyecto era conseguir la obtención de los datos desde AutoCAD para que luego fueran pasados al robot automáticamente y pudiera clasificarlos de forma correcta en una de las mesas laterales.

Las planchas utilizadas han sido exportadas desde AutoCAD e importadas a RobotStudio para comprobar que la extracción de los datos se realiza correctamente. En las figuras 5.7 podemos ver ambos modelos que han sido utilizados para comprobar el funcionamiento del programa.

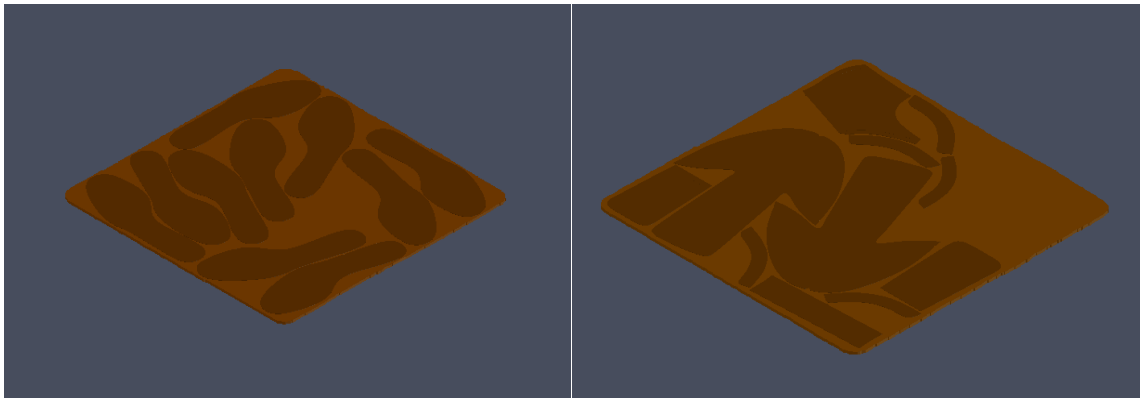


Figura 5.7: Modelo plancha N04801.dxf (izquierda) y N02027.dxf (derecha)

El resto de la estación se puede ver en la figura 5.8, diseñada de la misma forma que en la primera parte del presente proyecto.

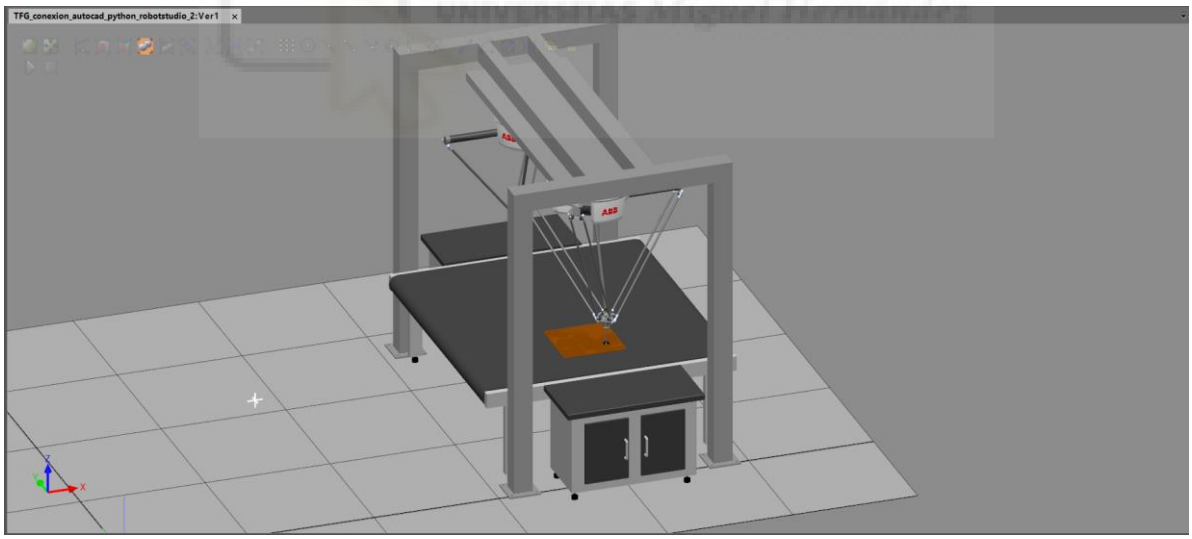



Figura 5.8: Estación conexión AutoCAD-Python-RobotStudio

### *5.3.2. Lógica de la Estación*

En esta parte del proyecto solamente ha sido programado uno de los robots, en este caso ha sido el robot situado a la izquierda del pórtico.

De esta forma, solamente será utilizado el Smart Component de la ventosa izquierda. En la figura 5.9 podemos observar cómo está conectado, explicado en la primera parte del proyecto.

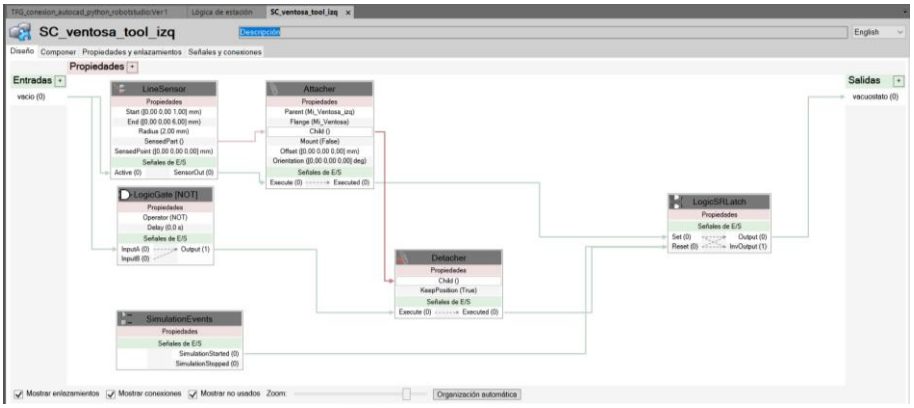



Figura 5.9: SC_ventosa_tool_izq

En este proceso no se va a utilizar ni las cintas transportadoras ni la ventosa del otro robot, por lo que sus respectivos Smart Componentes no hacen falta en la lógica de la estación, en la que solo se conecta el controlador de la estación con el Smart Component de la ventosa izquierda. En la figura 5.10 vemos la lógica completa de la estación de esta parte del proyecto.

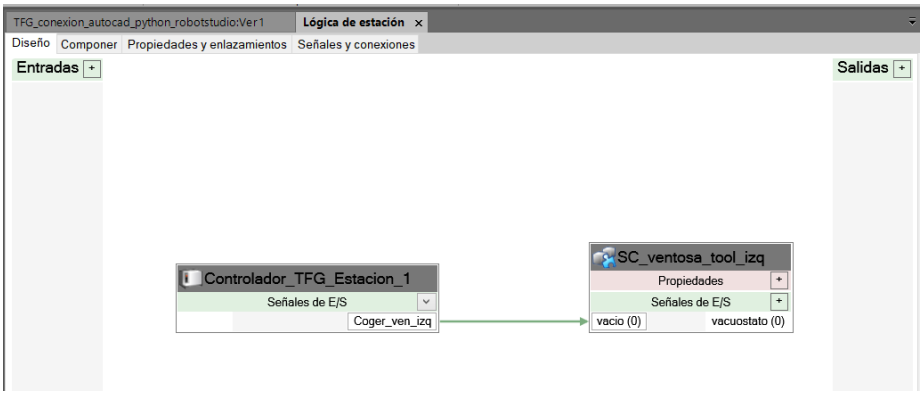



Figura 5.10: Lógica de la estación Conexión AutoCAD-Python-RobotStudio

Como se puede observar en la figura 5.10, el controlador solamente tiene una salida digital ya que es la única necesaria para poder realizar la tarea de Pick&Place y activar la ventosa izquierda.

### *5.3.3. Programación en RAPID*

La última parte de la conexión AutoCAD-Python-RobotStudio requiere de un programa en RAPID en la estación de RobotStudio que sea capaz de comunicarse mediante un socket con Python y que pueda lea los datos de un fichero. En el anexo 9.3.5 se encuentra el código de RAPID del contralor Module1 y a continuación se va a realizar una explicación de las distintas partes del programa.

Al principio del programa, se han realizado todas las declaraciones de variables necesarias que van a ser usadas durante el programa. Seguidamente, el primer proceso que hay se trata del proceso principal llamado “Main()”. Este proceso llama al resto de subprocesos, donde están definidos todas las acciones que se deben de realizar. El primero que se llama, es el subproceso “conexión”, seguido de un movimiento del robot a la posicion de Inicio. Después de situarse el robot en esa posición, tenemos un “WaitUntil” en el que se espera a que la señal “signal” tenga el valor correcto (adelante). Se desactiva la señal de salida del controlador conectada a la ventosa izquierda y se llama en orden a los subprocesos “Read”, “Calculate_offset”, “Generate” y “Secuence”, los cuales se van a desarrollar más detenidamente a continuación.

El primer subproceso, llamado “conexión”, se encarga de crear el socket (SocketCreate) y de conectarlo de tal forma que se pueda comunicar con Python indicando el mismo host y puerto que se había indicado en Python. Si la conexión se ha producido con éxito, Python espera a que desde RobotStudio se envíe un mensaje con “Conexión con ABB correcta.” Y a continuación, RobotStudio esperará recibir una señal de Python, esta señal será el indicativo de que se han extraído los datos desde AutoCAD y creado todos los ficheros necesarios.

El siguiente subproceso que hay definido se llama “Read” y es el encargado de leer los datos de las plantillas. Los datos han sido guardados en una archivo de texto llamado “data.txt” en la carpeta HOME del controlador de la estación, ya que se trata del lugar al que el controlador puede acceder para obtener datos externos a RobotStudio.

Una vez que se ha abierto el archivo con los datos, mediante un bucle FOR se leen todos los datos almacenando la primera palabra de una fila en una lista con el modelo de la plantilla, la segunda palabra es almacenada en las coordenadas X, la tercera en las coordenadas Y, y la última, en rotación. De esta forma tenemos distintas variables numéricas y string donde tenemos almacenas todos los datos de las plantillas ya en RobotStudio. Cuando se acaba de leer el fichero “data.txt”, se debe de cerrar.

A continuación, el subproceso “Calculate_offset” es el encargado de realizar una serie de cálculos, en los que se calcula la diferencia de distancia que hay entre la posición en la que se encuentran las plantillas según el centro de coordenadas de AutoCAD y el centro de coordenadas del Work Object de RobotStudio en el que se van a crear las posiciones.

El siguiente subprocess, llamado “Generate”, es el encargado de generar las posiciones de cada una de las plantillas. Para ello se ha programado con un bucle FOR recorriendo cada una de las posiciones donde se han almacenado los datos.
En este proceso se ha realizado una primera clasificación dependiendo del nombre de la plantilla que se esté leyendo en ese momento.

Donde se calcula, usando la rotación de la plantilla más una variación en el caso de algunas plantillas para que a la hora de situarlas en la mesa lateral se coloquen de la forma desea, y donde se calculan directamente los cuaternios del robot en esa posición.

También se calcula la posición de cada plantilla usando los datos de las coordenadas X e Y restándoles la diferencia de posición calculada en el subproceso “Calculate_offset”. En este proceso también se ha tenido que realizar una serie de ajuste en el caso de plantillas que eran muy pequeñas para que el robot las cogiese de forma correcta. Esto es debido a que para plantillas con forma curva, los datos obtenidos de AutoCAD no se encuentran sobre la pieza, por lo que el sensor de la ventosa no detecta la pieza y se debe de variar un poco la posición para que sean detectadas y recogidas por parte del robot.

Finalmente, se crea la posiciones en la variable “plantilla{j}”, con los datos calculados anteriormente.

El subproceso “Secuence” es donde se van sumando la cantidad de plantillas, al mismo tiempo que llama a dos subprocesos llamados “Pick” and “NewPlace”, que realizan, como su nombre indica, la tarea de coger una plantilla y depositarla mientras se clasifica en la mesa lateral.

En subproceso “Pick” está programado para que el robot vaya principalmente a la posición de Inicio por si no se encontrase en ella, y después, se mueva a la posición de la plantilla, donde se activa la salida “Coger_ven_izq” y que la plantilla sea cogida por la ventosa. El movimiento a la plantilla está condicionado al número de plantillas que lleve el robot recogidas, por lo que una vez que haya cogido la plantilla y la haya dejado, el robot pasará a recoger la siguiente plantilla que tenemos almacenada en los datos y que hemos creado las posiciones anteriormente.

El subproceso llamado “NewPlace” se basa en otra clasificación dependiendo del nombre de la plantilla que tenga el robot cogida con la ventosa. Al haber utilizado dos fichero que nos han mandado la empresa, el total de modelos distintos que hay son 13, dos en el modelo N04801 (pie derecho y pie izquierdo) y 11 en el modelos N02027 (distintas partes cortadas de un zapato).

Para las plantillas de pie derecho e izquierdo, cada una tiene un proceso independiente donde coloca las plantillas sobra uno montón con el mismo modelo, estos procesos son llamados “dejar_plantilla_derecha” y “dejar_plantilla_izquierda”.

Para el fichero N02027, al haber 11 modelos distintos se han clasificado en 5 grupos. El primero de ellos es el subproceso “dejar_plantilla_pieza_5”, en el que deja todos los modelos cuyo nombre empieza por Pieza_5 y los va colocando en fila, el siguiente subproceso es “dejar_pieza_3”, en este caso solamente hay una pieza de este modelo, y lo mismo ocurre con el subproceso “dejar_plantilla:otras_4000”.

Por último, los dos subprocesos que quedan, el primero es llamado “dejar_plantilla_Forro”, donde se clasifican las piezas para que queden bien colocadas en la mesa, y el último se llama “dejar_plantilla_Corte”, haciendo una clasificado también dependiendo del modelo.

Una vez que el robot está en la posición donde debe dejar la plantilla, se desactiva la señal de “Coger_ven_izq, para que el robot suelte la plantilla y vuelva a la posición de inicio para comenzar el siguiente proceso.

A continuación, se encuentra un diagrama de flujo en el que se resume el funcionamiento completo de la conexión AutoCAD-Python-RobotStudio.

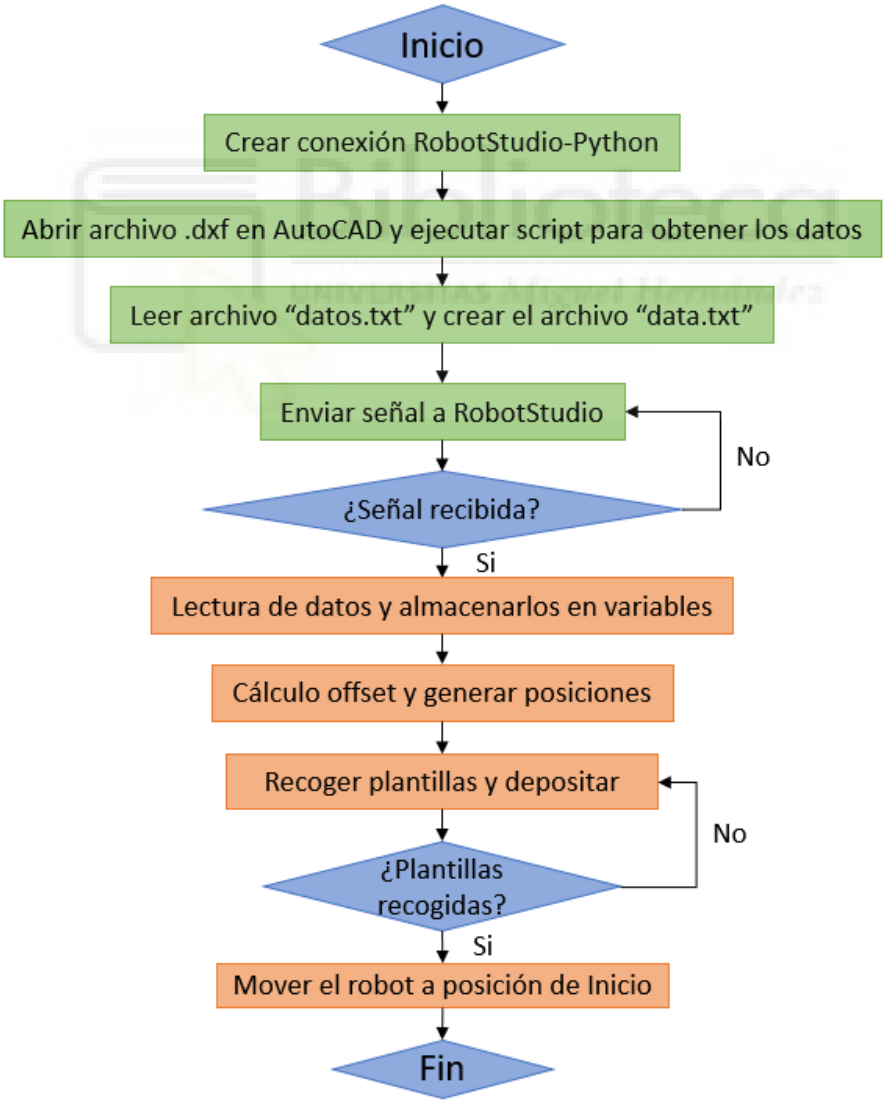



Figura 5.11: Diagrama de flujo Conexión AutoCAD-Python-RobotStudio

# 6. RESULTADOS

## 6.1. Simulación de la Estación de Recogida de plantillas

En este apartado se muestra la simulación de la estación de Pick&Place de plantillas, en la figura 6.1 se puede ver la estación completa antes de iniciarse la simulación.

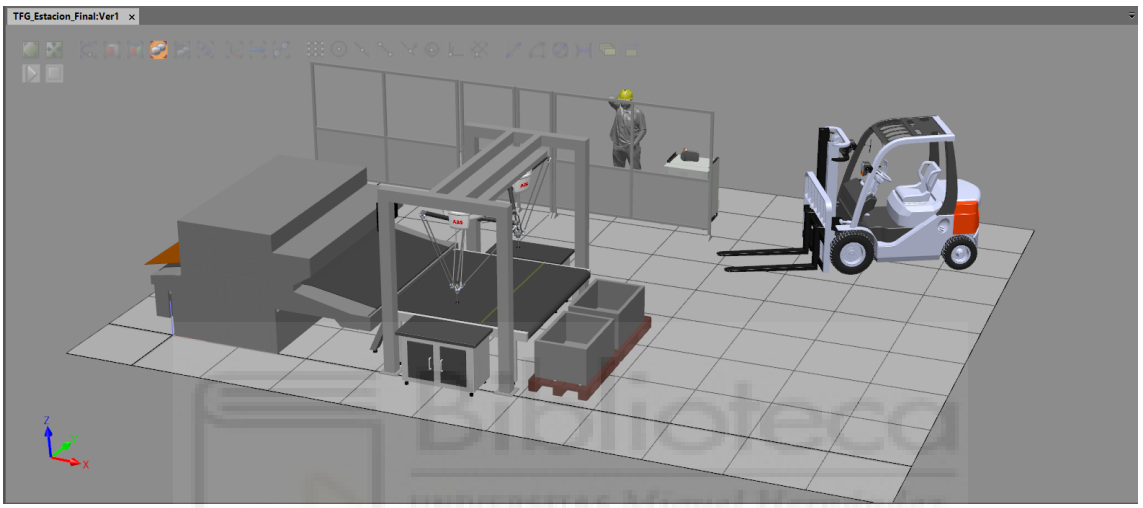



Figura 6.1: Estación Pick&Place

En primer lugar, se introducirán dos planchas sin cortar a la máquina CNC, cada una para uno robot, como se puede observar en la figura 6.2.

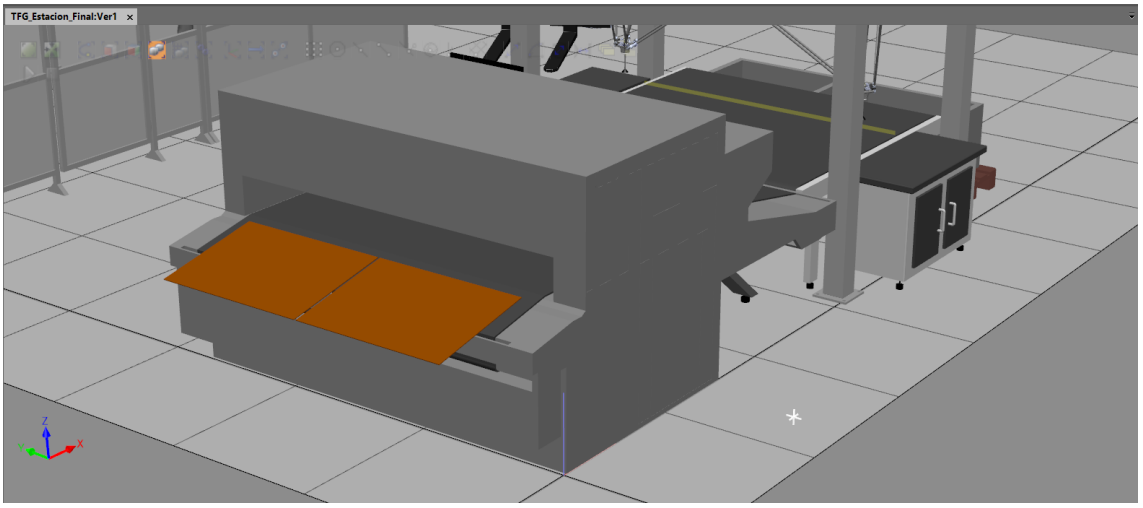



Figura 6.2: Introducción de dos plancha a la maquina CNC.

Una vez que las dos planchas alcanzan el lugar de corte dentro de la maquina CNC, con el sensor situado en la CNC se desactivarán las planchas sin cortar y se activarán las planchas cortadas que continuarán a lo largo de la cinta, como se puede ver en la figura 6.3.

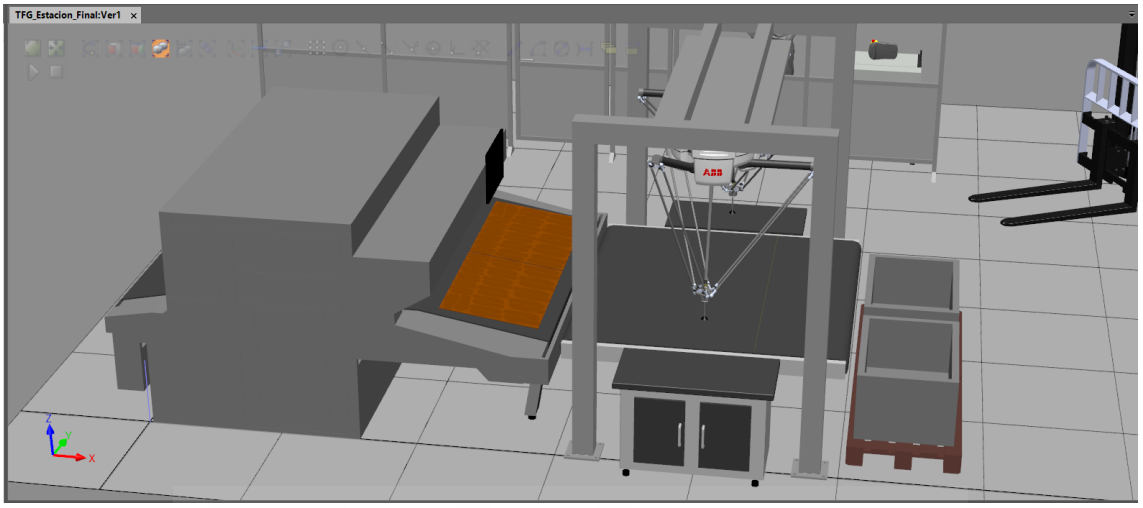



Figura 6.3: Salida de las planchas cortadas de la CNC

Las planchas serán transportadas por la cinta hasta que el sensor situado en la posición donde se deben recoger las plantillas por los robots, se parará la cinta e iniciará la tarea de Pick&Place por parte de los robots. En la figura 6.4 se muestra las planchas situadas en dicha posición.

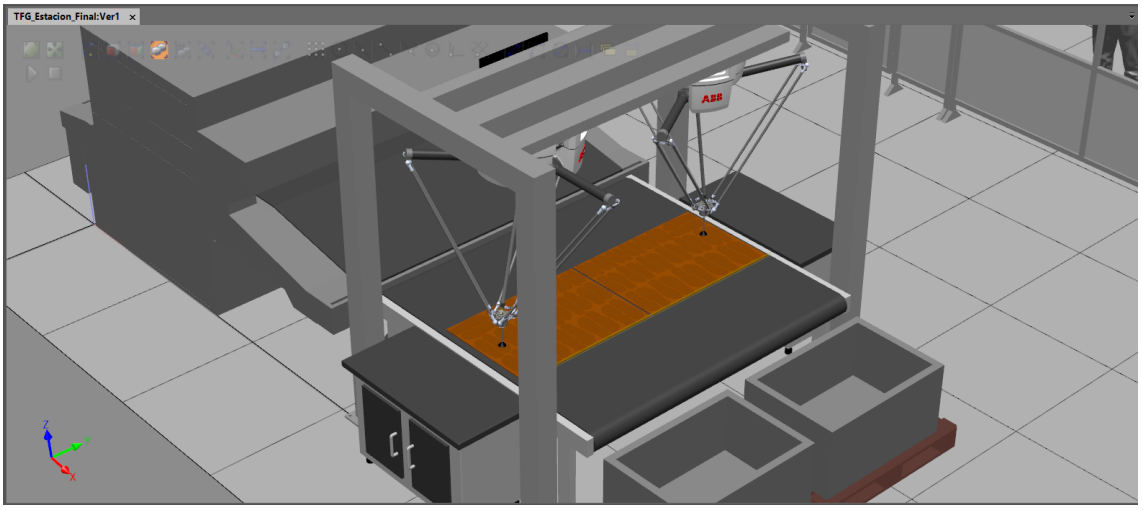



Figura 6.4: Paro de la cinta cuando el sensor se activa

En la figura 6.5 se puede ver como los robots se encargan de moverse hasta la posición de cada una de las plantillas para una vez que se encuentran en la posición para cogerla, se activa la señal del vacío desde el controlador para que la ventosa coja la plantilla.

También, podemos observar cómo ha sido programada la tarea, de forma que los dos robots comienzan desde el mismo lado de la plancha, para evitar así cualquier tipo de colisión entre ellos durante todos el proceso.

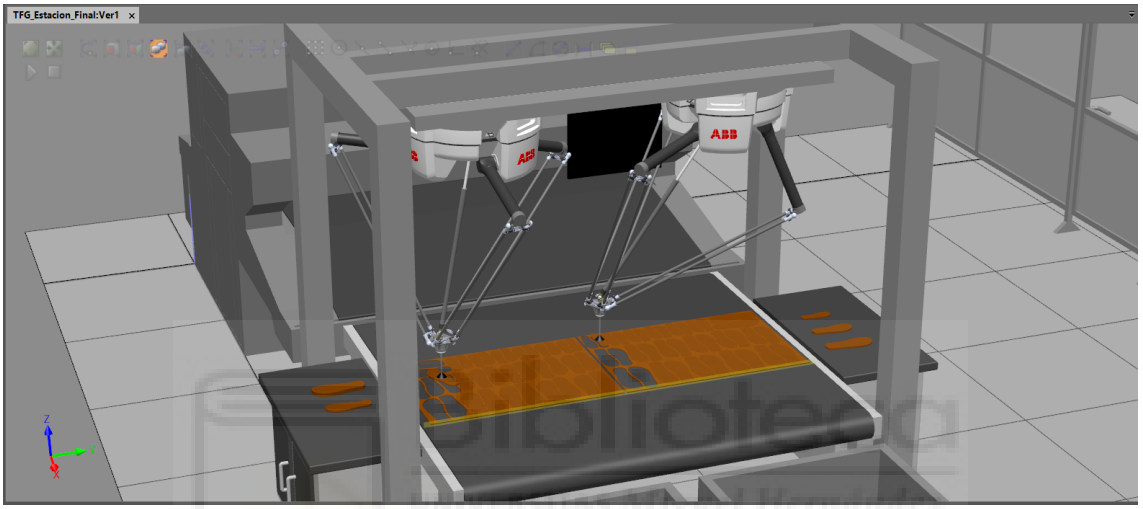



Figura 6.5: Recogida de las plantillas mediante vacío

Otro de los objetivos era conseguir que las plantillas se clasificasen dependiendo de su tamaño, en este caso tenemos tres tamaños distintos, grande, mediano y pequeño. Como se puede observar en la figura 6.6, los robots tienen programadas tres posiciones distintas en las que cada una corresponde a un tamaño distinto.

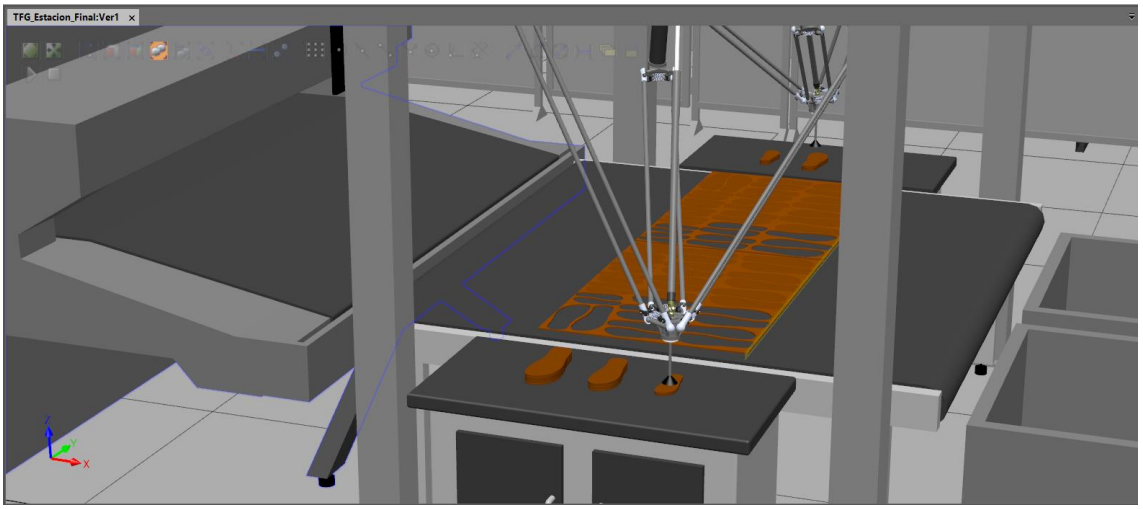



Figura 6.6: Clasificación de las plantillas por tamaños

Una vez que todas las plantillas se han recogido y clasificado, los robot vuelven a su posición de inicio, donde esperarán al próximo proceso. Como se puede ver en la figura 6.7, la tarea de Pick&Place ha finalizado, consiguiendo una perfecta y rápida recogida y clasificación de las plantillas en las mesas situados a los laterales de la cinta y desde donde un operario puede retirarlas.

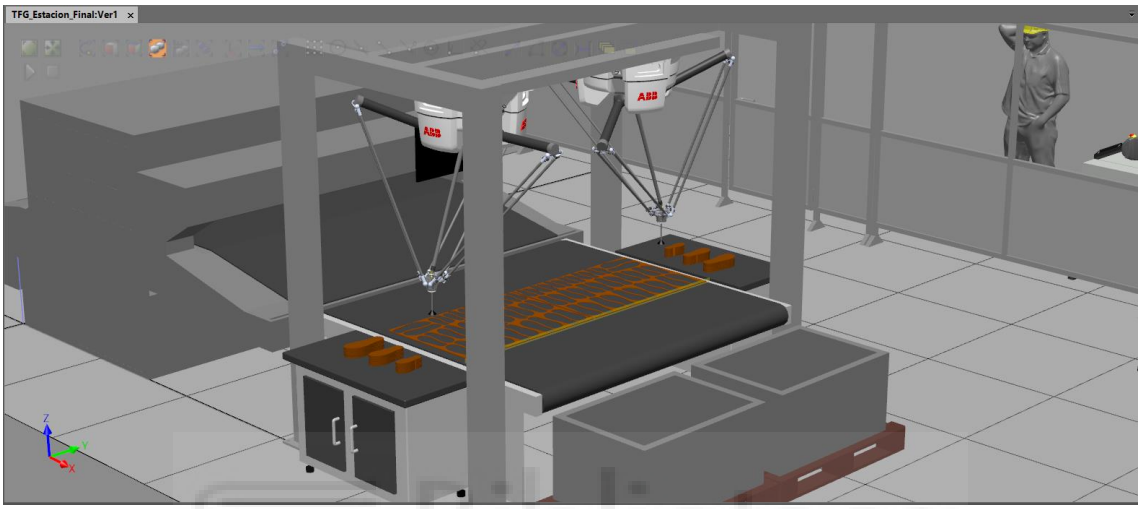



Figura 6.7: Finalización del proceso de Pick&Place

## 6.2. Comunicación AutoCAD-Python-RobotStudio

En este apartado se van a mostrar los resultados de la simulación de la estación Conexión AutoCAD-Python-RobotStudio, donde se han obtenido los datos directamente de los ficheros N04801.dxf y N02027.dxf en AutoCAD y se han pasado a RobotStudio mediante Python.

Primero, como se puede observar en la figura 6.8, hemos realizado la simulación para la estación del modelo N04801, donde se ve el robot situado sobre la cinta transportadora donde se encuentra la plancha cortada en forma de plantillas.

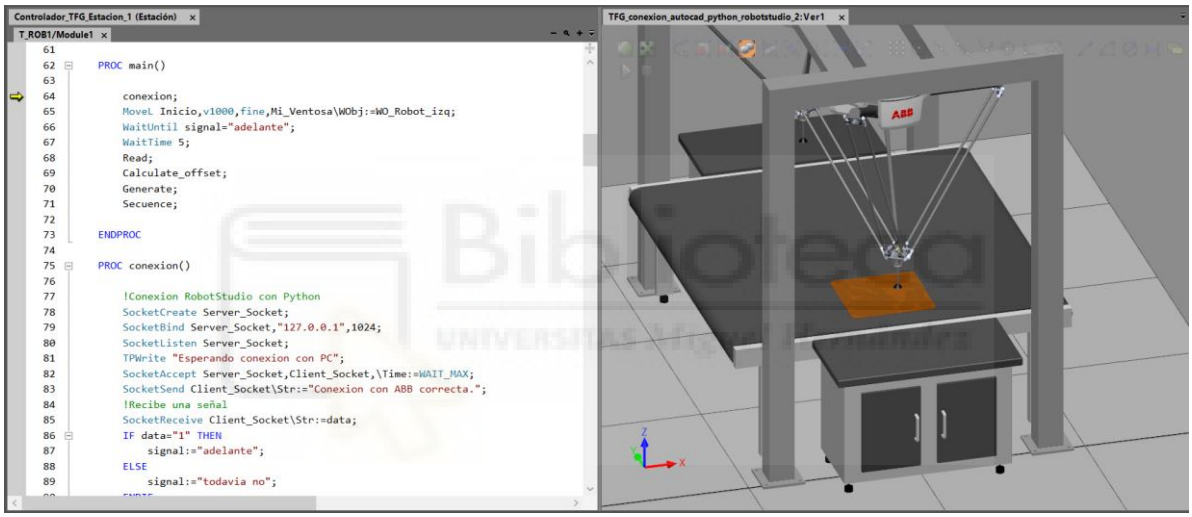



Figura 6.8: Estación conexión AutoCAD-Python-RobotStudio (N04801)

Para iniciar la simulación, primero se inicia la simulación en RobotStudio para crear el socket y a continuación, se inicia el programa de Python. De esta forma, en Python se puede ver que aparece un mensaje en el que dice que la conexión con ABB se ha producido correctamente. Seguidamente, el usuario debe introducir el nombre del archivo de AutoCAD del que se van a obtener los datos, en este caso, el archivo N04801. En la figura 6.9 se muestra cómo se recibe el mensaje de correcta conexión creada mediante el socket entre ambos programas y el nombre introducido por el usuario.


Figura 6.9: Mensaje recibido en Python e introducción nombre archivo

A continuación, el programa de Python creará el script de AutoCAD, el cual se puede ver en la figura 6.10 siguiente, este abrirá el archivo .dxf con el nombre que se ha introducido, obtendrá los datos con el comando EXTRACDAT y se cerrará.

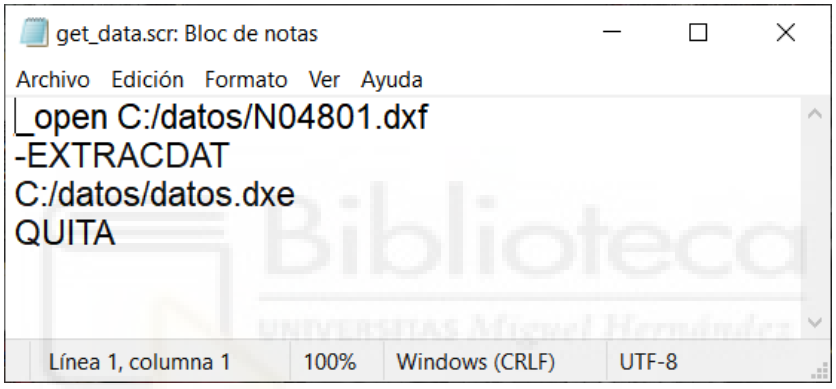



Figura 6.10: Script get_data.scr de AutoCAD creado con Python (N04801)

Una vez creado el script, Python abrirá la aplicación de AutoCAD 2020 en el ordenador y ejecutará el script para obtener los datos del archivo. En la siguiente figura 6.11 se puede ver el proceso de extracción de datos cuando está abriendo AutoCAD.

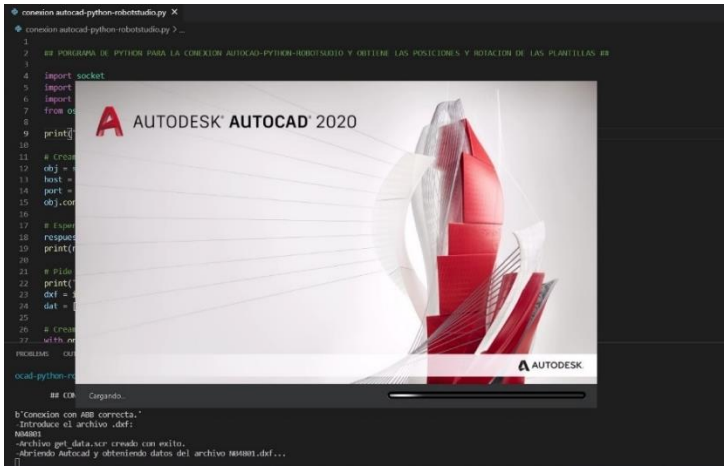



Figura 6.11: Abriendo AutoCAD para extraer datos

Una vez que el script ha terminado de realizar todos los comandos, AutoCAD se cerrará y se muestra por pantalla que el fichero datos.txt con los datos de las plantillas se ha creado correctamente. En el siguiente paso, Python lee el archivo que se acaba de obtener con los datos de las plantillas y los copia en un fichero nuevo llamado data.txt situado en la carpeta HOME del controlador de RobotStudio y desde el cual se van a leer los datos en RAPID. En la figura 6.12, podemos ver los mensaje que aparecen en pantalla, y en la figura 6.13, podemos ver el archivo de texto datos.txt con los datos extraídos de AutoCAD y el archivo data.txt que se ha creado desde Python.

```
-Abriendo Autocad y obteniendo datos del archivo N04801.dxf...
-Fichero datos.txt creado correctamente.
-Lectura del fichero datos.txt
-Creando fichero data.txt en la carpeta HOME del Controlador Virtual de RobotStudio
-Fichero data.txt creado con exito
```

Figura 6.12: Mensajes en pantalla sobre ficheros de texto (N04801)

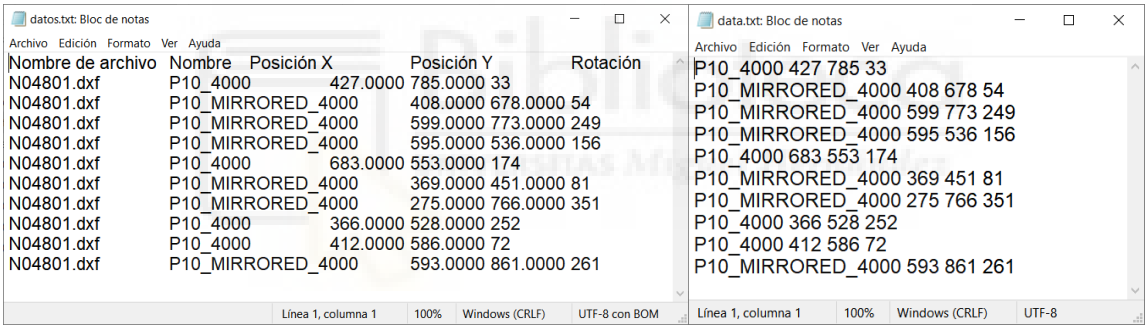



Figura 6.13: Ficheros de texto datos.txt obtenido y data.txt creado (N04801)

Al crear el archivo de texto data.txt en la carpeta del controlador, Python envía una señal a RobotStudio para que continúe con su programa y pase a la siguiente instrucción. A continuación, se leen todos los datos almacenados en el fichero data.txt y son almacenar en variables que permitirán crear las posiciones de cada una de las plantillas.

En la figura 6.14, se puede ver las variables de RobotStudio con los datos ya leídos de data.txt, incluida la señal que se ha enviado desde Python y que se ha dado el valor de “adelante” a la variable “signal”, la cual se comprobará para que continúe la simulación.

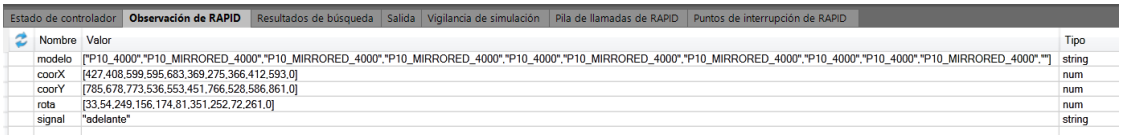



Figura 6.14: Valor de las variables en RobotStudio

Una vez leídos los datos y creadas las posiciones, el robot comenzará a recoger las plantillas y clasificarlas según sean de pie derecho o pie izquierdo en la mesa lateral. En la figura 6.15, se pueden ver algunas de las imágenes del proceso de Pick&Place de las plantillas y su posterior clasificación en la mesa lateral.

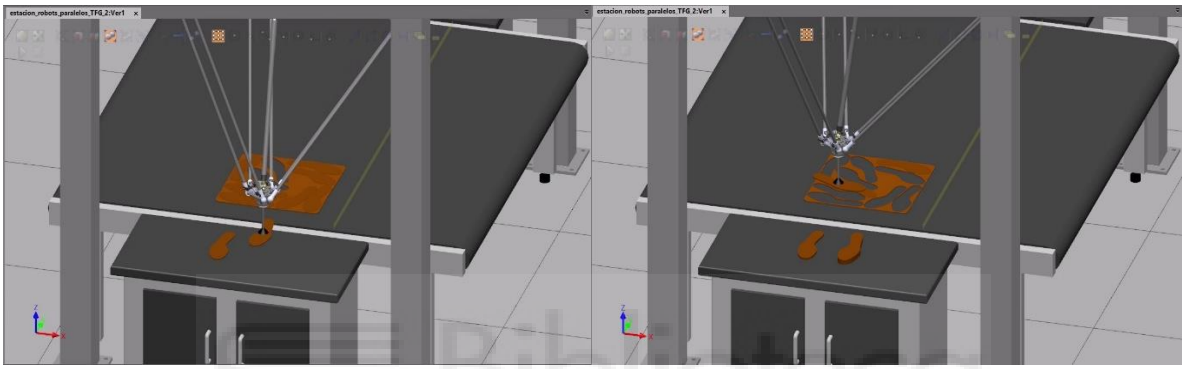


Figura 6.15: Proceso de Pick&Place (N04801)

Finalmente, una vez que todas las plantillas han sido recogidas, el robot volverá a la posición de inicio a la espera de que el siguiente proceso comience.

En la figura 6.16, podemos ver el final de la simulación, donde las plantillas han sido recogidas de forma correcta y depositadas en la mesa lateral, al mismo tiempo que han sido clasificadas dependiendo del pie al que pertenezcan.

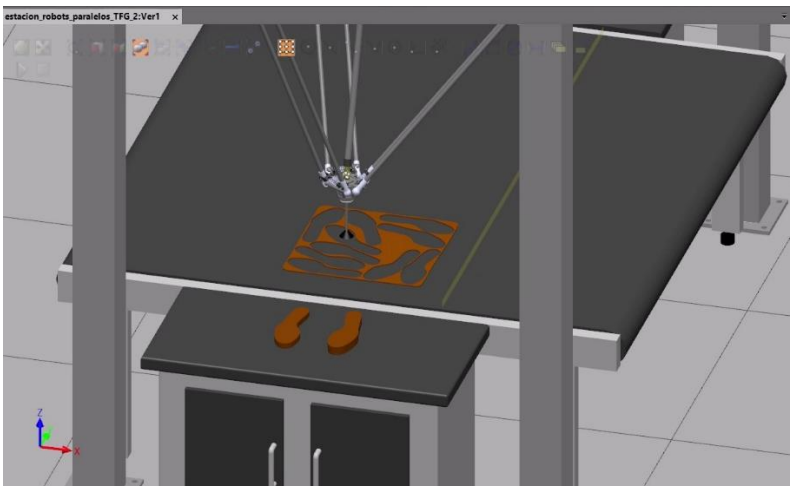


Figura 6.16: Fin Pick&Place conexión AutoCAD-Python-RobotStudio (N04801)

Una vez que se ha simulado el proceso para el fichero N04801.dxf y en el cual se han clasificado solamente un tipo de modelo se ha simulado el mismo proceso para el fichero N02027.dxf, en el cual encontramos más variedad de modelos cortados. En la figura 6.17, podemos ver la estación al inicio de la simulación con el modelo de plantillas N02027.dxf.

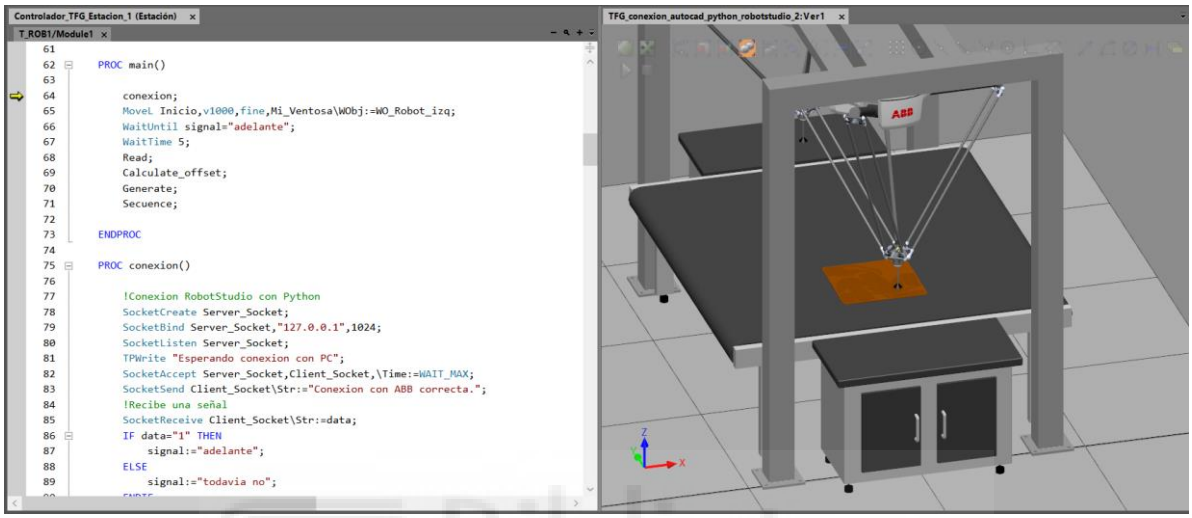



Figura 6.17: Estación conexión AutoCAD-Python-RobotStudio (N02027)

El proceso es el mismo que el realizado para el modelo anterior, por lo que los pasos que realiza el código son iguales. En la figura 6.18 se puede ver como se ha creado un script para AutoCAD distinto al anterior en el que, tras haber introducido el nuevo nombre del fichero, la ruta para abrir el archivo es distinta con el nombre correspondiente.

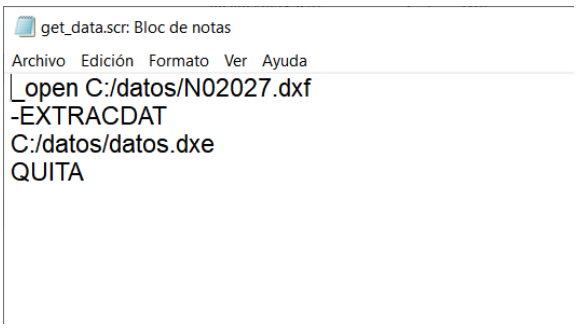



Figura 6.18: Script get_data.scr de AutoCAD creado con Python (N02027)

En la figura 6.19 se pueden observar los mensajes mostrados por pantalla una vez finalizado el proceso de extracción de datos de AutoCAD. También, en la figura 6.20, vemos los datos extraídos del fichero N02027.dxf en el fichero datos.txt, así como el nuevo fichero data.txt creado en la carpeta HOME del controlador.

```
55          datos[i][0] = datos[i][0].replace(' ','_')
56          datos[i][0] = datos[i][0].replace('(','')

PROBLEMS   OUTPUT   TERMINAL   CONSOLA DE DEPURACIÓN

-Abriendo Autocad y obteniendo datos del archivo N02027.dxf...
-Fichero datos.txt creado correctamente.
-Lectura del fichero datos.txt
-Creando fichero data.txt en la carpeta HOME del Controlador Virtual de RobotStudio
-Fichero data.txt creado con exito
-Enviando señal a RobotStudio
-Señal enviada con exito
-Archivo datos.txt eliminado con exito.
-Archivo get_data.scr eliminado con exito.
```

Figura 6.19: Mensajes en pantalla sobre ficheros de texto (N02027)

datos.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

| Nombre | Posición X | Posición Y | Rotación | |
|---|---|---|---|---|
| Pieza (5) Otras_4000 | | 1594.0000 | 658.0000 | 180.00 |
| Pieza (5) Otras_mirrored_4000 | | 1555.0000 | 973.0000 | 150.00 |
| Pieza (5) Otras_mirrored_3900 | | 1789.0000 | 697.0000 | 160.00 |
| Pieza (5) Otras_mirrored_3500 | | 1649.0000 | 964.0000 | 70.00 |
| Pieza (5) Otras_4200 | | 1550.0000 | 1052.0000 | 200.00 |
| Pieza Otras_4000 | 1747.0000 | 632.0000 | 0.00 | |
| Pieza Forro_mirrored_4100 | 1642.0000 | 812.0000 | 110.00 | |
| Pieza Forro_4100 | 1483.0000 | 768.0000 | 100.00 | |
| Pieza (3)_mirrored_4100 | | 1466.0000 | 1017.0000 | 170.00 |
| Corte_mirrored_3500 | | 1405.0000 | 674.0000 | 90.00 |
| Corte_4200 | 1808.0000 | 812.0000 | 90.00 | |

Línea 1, columna 1 100% Windows (CRLF)

data.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Pieza_5_Otras_4000 1594 658 180.00
Pieza_5_Otras_mirrored_4000 1555 973 150.00
Pieza_5_Otras_mirrored_3900 1789 697 160.00
Pieza_5_Otras_mirrored_3500 1649 964 70.00
Pieza_5_Otras_4200 1550 1052 200.00
Pieza_Otras_4000 1747 632 0.00
Pieza_Forro_mirrored_4100 1642 812 110.00
Pieza_Forro_4100 1483 768 100.00
Pieza_3_mirrored_4100 1466 1017 170.00
Corte_mirrored_3500 1405 674 90.00
Corte_4200 1808 812 90.00
```

Línea 1, co

Figura 6.20: Ficheros de texto datos.txt obtenido y data.txt creado (N02027)

Una vez pasado todos los datos del fichero N02027.dxf a RobotStudio, comienza el proceso de leerlos, crear las posiciones y todas las instrucciones necesarias para realizar la tarea de Pick&Place con los nuevos datos, y su clasificación en la mesa situada en el lateral de la cinta.

En la siguiente figura 6.21 se puede ver imágenes del proceso de Pick&Place durante la simulación realizada para este modelo.

Figura 6.21: Proceso de Pick&Place (N02027)

Una vez que el proceso de Pick&Place finaliza, el robot vuelve a la posición de inicio a la espera del siguiente proceso. En la figura 6.22, se puede ver el resultado final de la estación una vez que las piezas que han sido cortadas esta depositadas en la mesa lateral clasificadas por modelos de forma correcta.

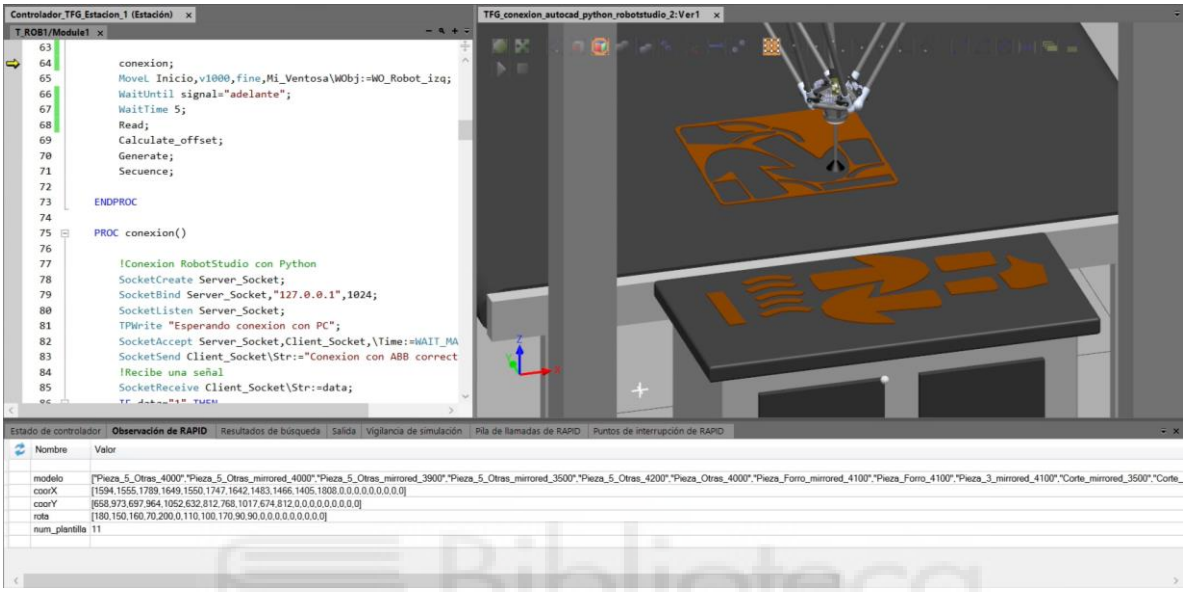



Figura 6.22: Fin Pick&Place conexión AutoCAD-Python-RobotStudio (N02027)

# 7. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

## 7.1. Conclusiones

Una vez finalizado el proyecto de Fin de Grado, se puede concluir que se ha conseguido cumplir con todos los objetivos marcados y planteados inicialmente.

El software de programación RobotStudio 6.11.01 es uno de los más utilizados en la programación de estaciones robotizas en la industria, así como de manera “offline”, permitiendo realizar simulaciones cercanas a la realidad de manera virtual.

Para cumplir con el aprendizaje del programa se han tenido que seguir previamente una serie de videotutoriales de aprendizaje y con los que se han podido adquirir gran cantidad de conocimientos sobre el mismo. El programa nos ha permitido realizar la programación de dos estaciones robotizadas, las cuales son de gran uso en la industria, como son las tareas de Pick&Place.

Para el diseño de las estaciones, se han utilizado diferentes objetos propios del RobotStudio, así como los sensores, las cintas, las vallas, etc. Pero también se ha tenido que diseñar otros elementos en AutoCAD 2020, como el pórtico para los robots, las mesas laterales, las planchas de las plantillas o la ventosa de los robots. Consiguiendo la simulación de un estación industrial con maquina CNC realizando el proceso de Pick&Place de plantillas, al mismo tiempo que son clasificadas de manera correcta.

Otro de los objetivos era automatizar aún más el proceso diseñado, partiendo de un archivo CAD con los datos de las plantillas. Para desarrollarlo se ha tenido que aprender a programar con Python y AutoLISP, haciendo uso también de diversos videotutoriales en los que a base de prueba-error se ha conseguido el resultado deseado.

El diseño que la segunda parte del proyecto se ha basado en la creación de un código en Python el cual conseguía extraer los datos necesarios de un fichero .dxf de AutoCAD, y su posterior comunicación mediante un socket con RobotStudio. Desde RAPID, se ha conseguido leer estos datos para crear todas las posiciones y movimientos necesarios para realizar las áreas de Pick&Place de los distintos modelos de plantillas que aparecen en los dos modelos utilizados, así como su clasificación en la mesa lateral.

Esta método aumenta la productividad del proceso, realizándolo de manera más autónoma y rápida, donde cabe destacar su gran flexibilidad para diversos modelos de corte de plantillas, como se ha podido observar.

## 7.2. Líneas futuras de trabajo

Por último, se van a enunciar las principales líneas futuras de trabajo derivadas de este trabajo, ya que no se ha podido desarrollar todas las posibilidades que ofrece.

La primera de ellas sería la implementación del proceso desarrollado en el presente trabajo a una línea industrial automatizada real. Donde se debería de tener en cuenta los tiempos y métodos que se fija en dicha línea para poder introducirla.

Sería conveniente un estudio profundo de los diferentes lenguajes de programación utilizados, ya que tienen un gran margen de mejora para una mayor optimización del proceso.

En cuanto a la extracción de datos, sería conveniente la creación de una plantilla la cual pueda ser utilizada con todos los modelos que sean diseñados en una empresa.

Finalmente, la introducción de una cámara en el proceso de comunicación AutoCAD-Python-RobotStudio, así como el uso de visión artificial, introduciría una gran variedad de mejoras al diseño a la hora de la localización de las planchas cortadas en las cintas transportadoras.

## 8. BIBLIOGRAFÍA

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

[2] ABB Robotics. (2019) IRB 120 Industrial Robot. 2 p.

[3] ABB Robotics. (2020) IRB 360 Industrial Robot. 2 p.

[4] ABB Robotics. (2018) Manual de referencia técnica. Instrucciones, funciones y tipos de datos de RAPID. RobotWare 6.08. 1876 p.

[5] Antonio Barrientos, Luís Felipe Peñín, Carlos Balaguer y Rafael Aracil. (1997) Fundamentos de Robótica. Universidad Politécnica de Madrid. 342 p.

[6] T. Huang, P. F. Wang, J.P. Mei, X. M. Zhao, D. G. Chetwynd. (2007) Time Minimum Trajectory Planning of a 2-DOF Translational Parallel Robot for Pick-and-place Operations. School of Mechanical Engineering, Tianjin University, China and School of Engineering, The University of Warwick, Coventry, UK. 4p.

[7] Amin Khorasani, Soheil Gholami, and Hamid D. Taghirad, Senior Member, IEEE. Advanced Robotics and Automated Systems (ARAS), Industrial Control Centre of Excellence (ICCE), Faculty of Electrical Engineering, K.N. Toosi University of Technology, Tehran, Iran. (2015) Optimization of KNTU Delta Robot for Pick and Place Application. 6p.

[8] V. Poppeová, J. Uríček, V. Bulej, P. Šindler. (2011). DELTA ROBOTS – ROBOTS FOR HIGH SPEED MANIPULATION. 11p.

**Páginas web consultadas:**

[9] https://www.edsrobotics.com/blog/evolucion-robotica-industrial/

[10] https://www.edsrobotics.com/blog/tipos-robots-industriales-usos/

[11] https://es.wikipedia.org/wiki/Robot_Delta

[12] https://guide.directindustry.com/es/que-robot-industrial-elegir/

[13] https://revistaderobots.com/robots-y-robotica/robot-delta-aplicaciones-y-precios/

[14] https://revistaderobots.com/robots-y-robotica/robot-delta-aplicaciones-y-precios/

[15] https://new.abb.com/products/robotics

[16] https://www.fanuc.eu/es/es/robots

[17] https://industrial.omron.es/es/products/robotics

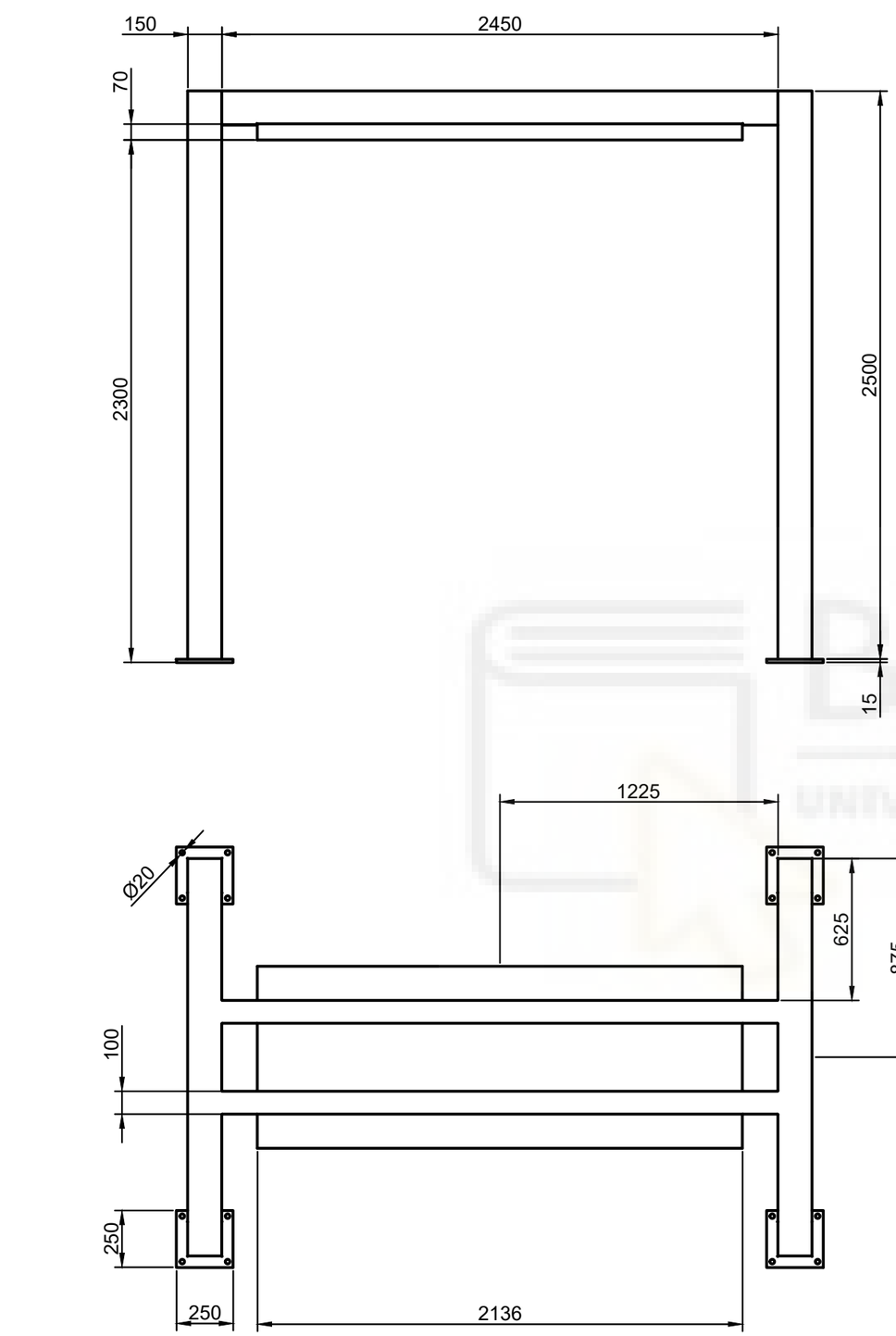
**Videotutoriales:**

https://new.abb.com/products/robotics/es/robotstudio/tutoriales
https://www.youtube.com/watch?v=4iHHf1-veJY&list=PLkEmFyPTnhOmGDFKSvOlJuuxh4c4sVpTy
https://www.youtube.com/watch?v=oDnHaVMZjp4&list=PLVdvHpsfqw1bX194j7Slup6ri5se2668p
https://www.youtube.com/playlist?list=PLgwW4GLuvZFGjfaZ90RTqxjYaw3DWJ9jI
https://www.youtube.com/watch?v=6Qks8TGY-_Y
https://www.youtube.com/watch?v=_1_uF7B5F80&list=PLv406soplrbEiggQnTOsNZaEsNxePjNM-&index=7
https://www.youtube.com/watch?v=vJiNoFfPfJw
https://www.youtube.com/watch?v=EE6Ut0k47j4
https://www.youtube.com/watch?v=8VvPgNnfgkU&list=PLv406soplrbEiggQnTOsNZaEsNxePjNM-&index=9
https://www.youtube.com/watch?v=Qsu259Nsd64&list=PLvimn1Ins-43WtzBU5281m6UwbNROArTB
https://www.youtube.com/watch?v=f6O2fyXF_5E&list=PLAzmBEJ4XYtitogcgaWsc5S3ImNpXkr30

# 9. ANEXOS

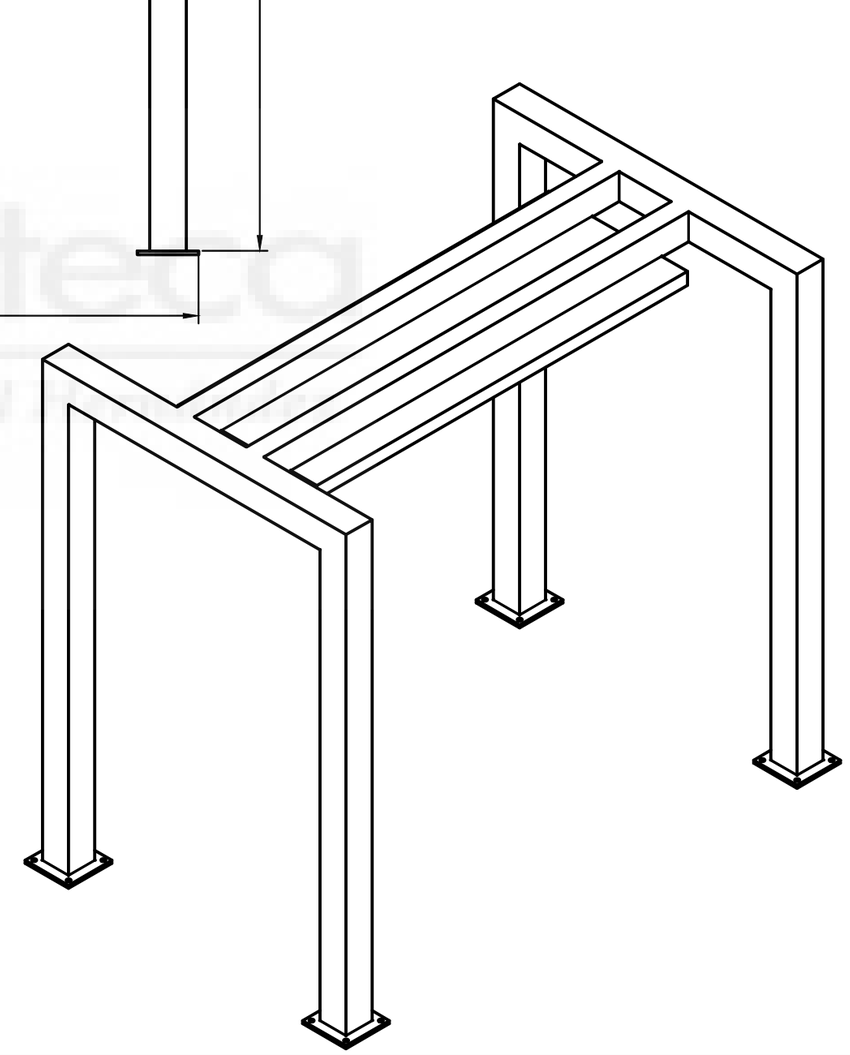
## 9.1. Planos

### *9.1.1. Planos de los sólidos de la estación de Pick&Place de plantillas*

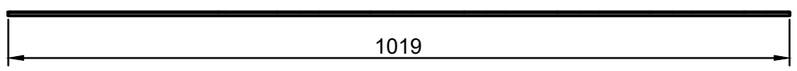




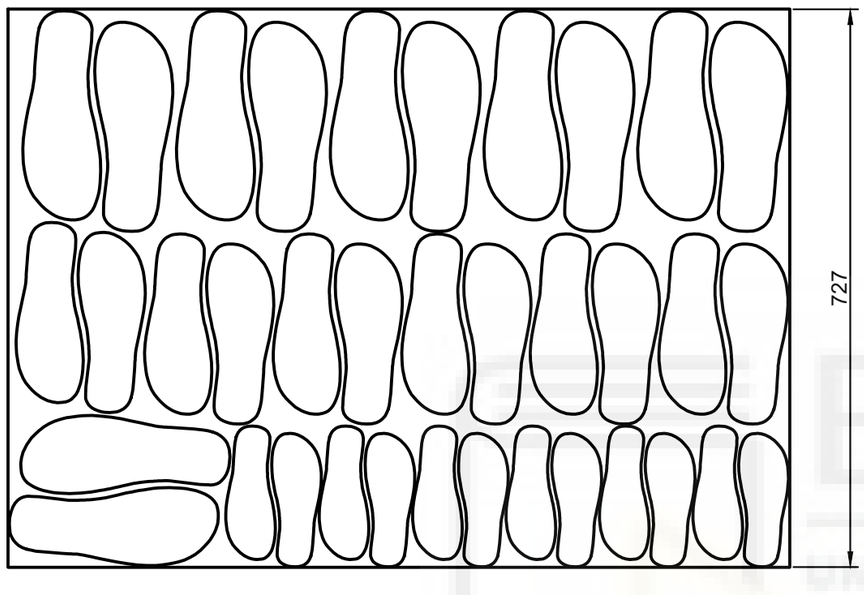


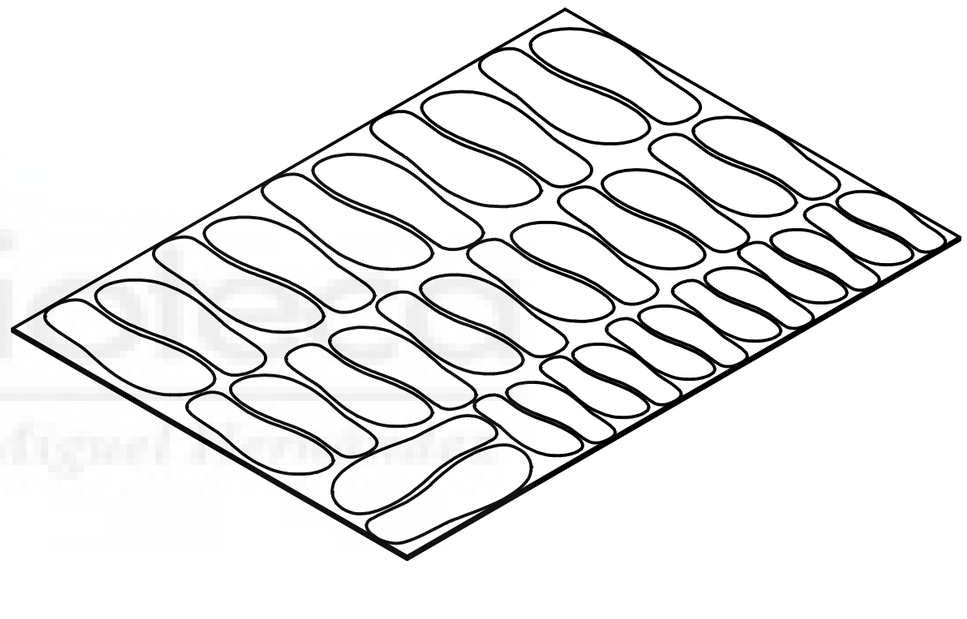


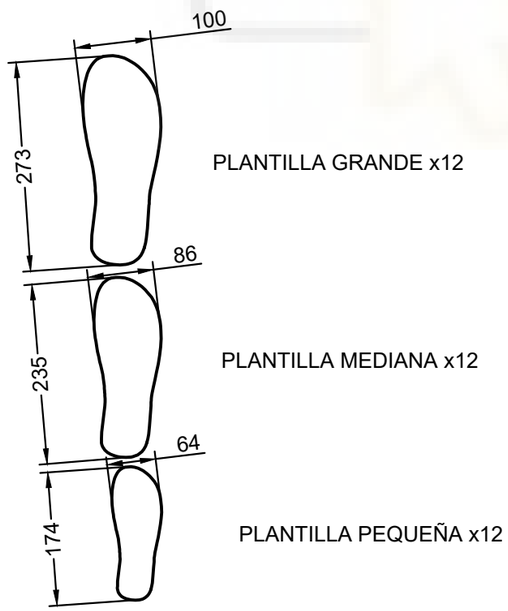
| Plano: PLANO 1. PÓRTICO | Proyecto: Trabajo Fin de Grado | Página: 96 |
|---|---|---|
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 1 / Formato: A3 |





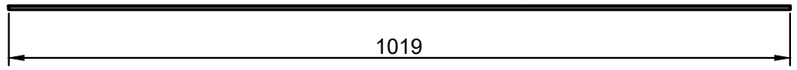


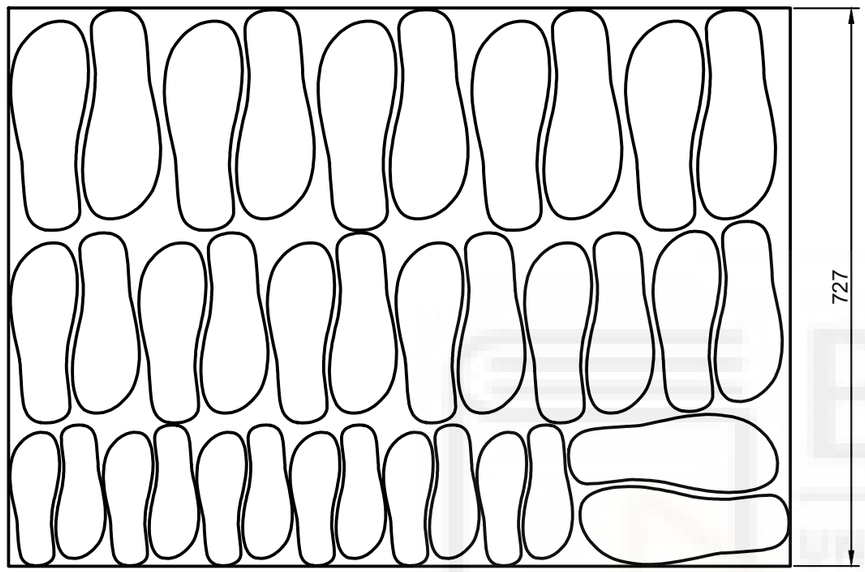


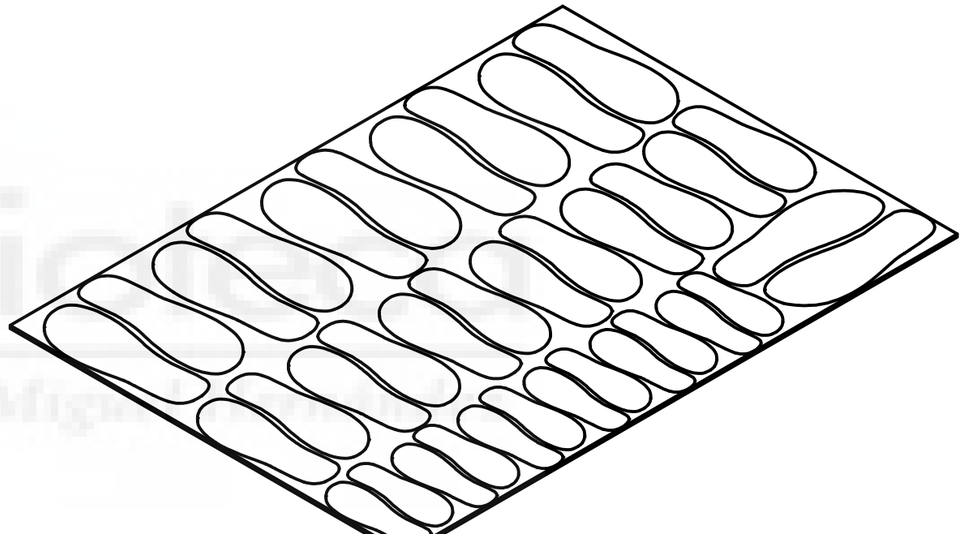

| | | | |
|---|---|---|---|
| Plano: PLANO 2. PLANCHA PLANTILLAS DER | Proyecto: Trabajo Fin de Grado | | Página: 97 |
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 2 | Formato: A3 |


CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK
CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK





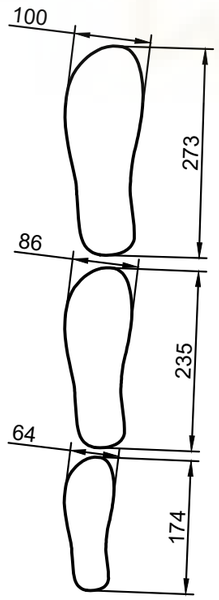


PLANTILLA GRANDE x12

PLANTILLA MEDIANA x12

PLANTILLA PEQUEÑA x12

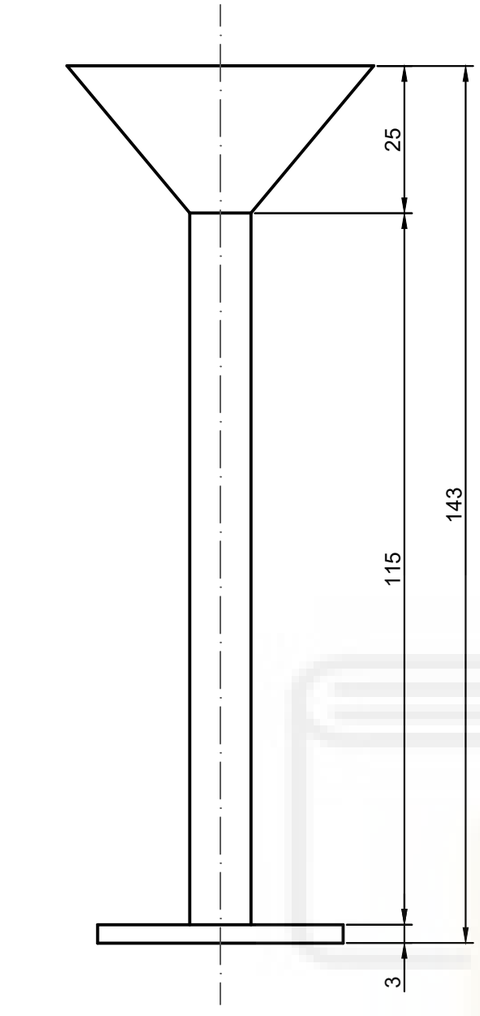


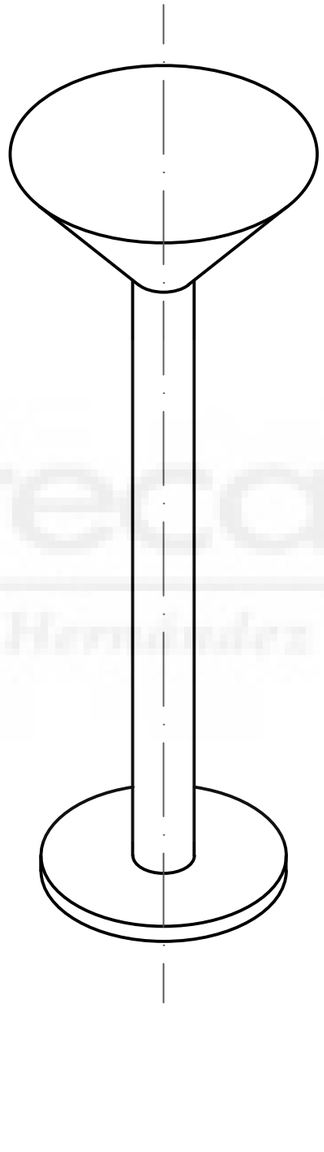

| | | | |
|---|---|---|---|
| Plano: PLANO 3. PLANCHA PLANTILLAS IZQ | Proyecto: Trabajo Fin de Grado | | Página: 98 |
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 3 | Formato: A3 |

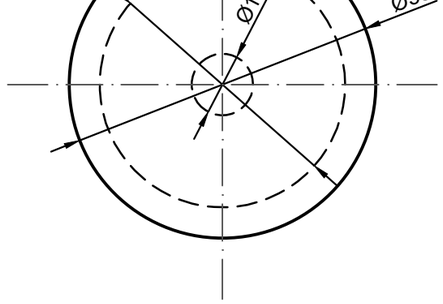


CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK

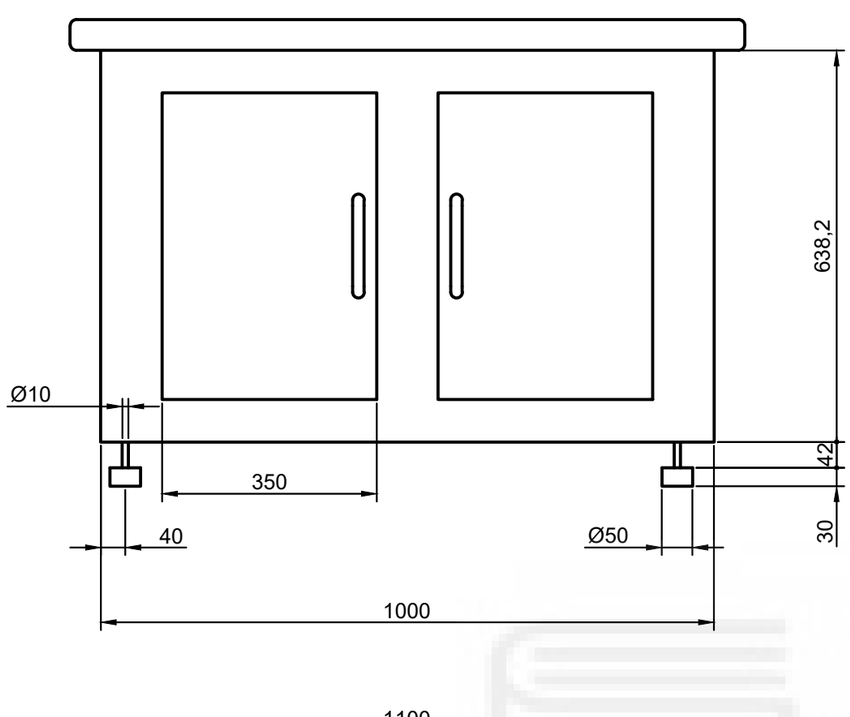
CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK

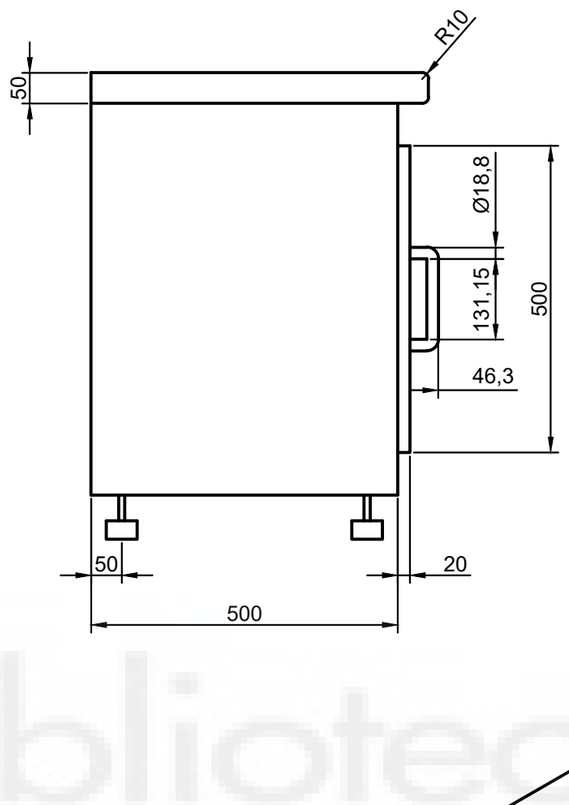


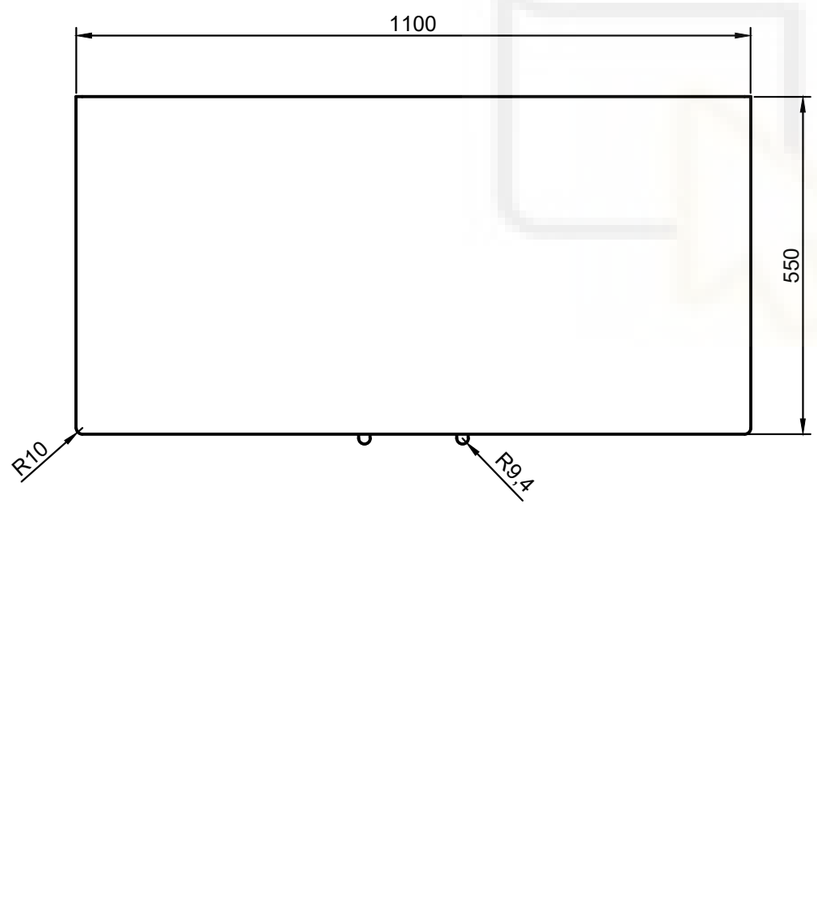



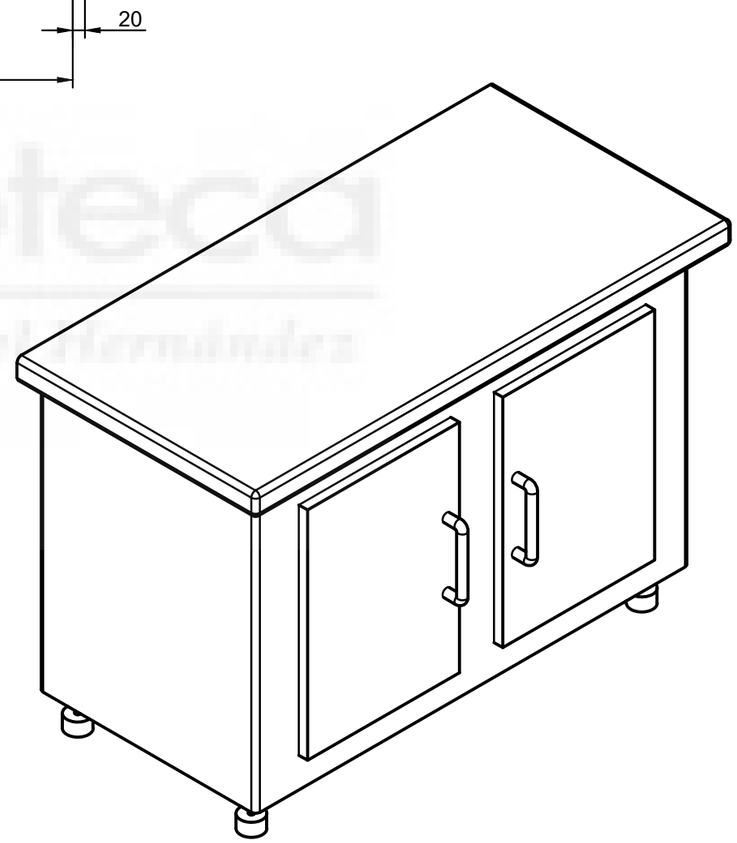



| | | | |
|---|---|---|---|
| Plano: PLANO 4. VENTOSA | Proyecto: Trabajo Fin de Grado | | Página: 99 |
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 4 | Formato: A3 |



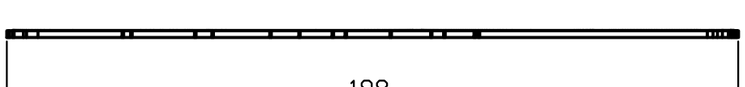



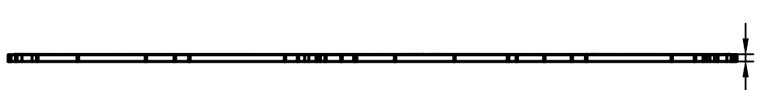



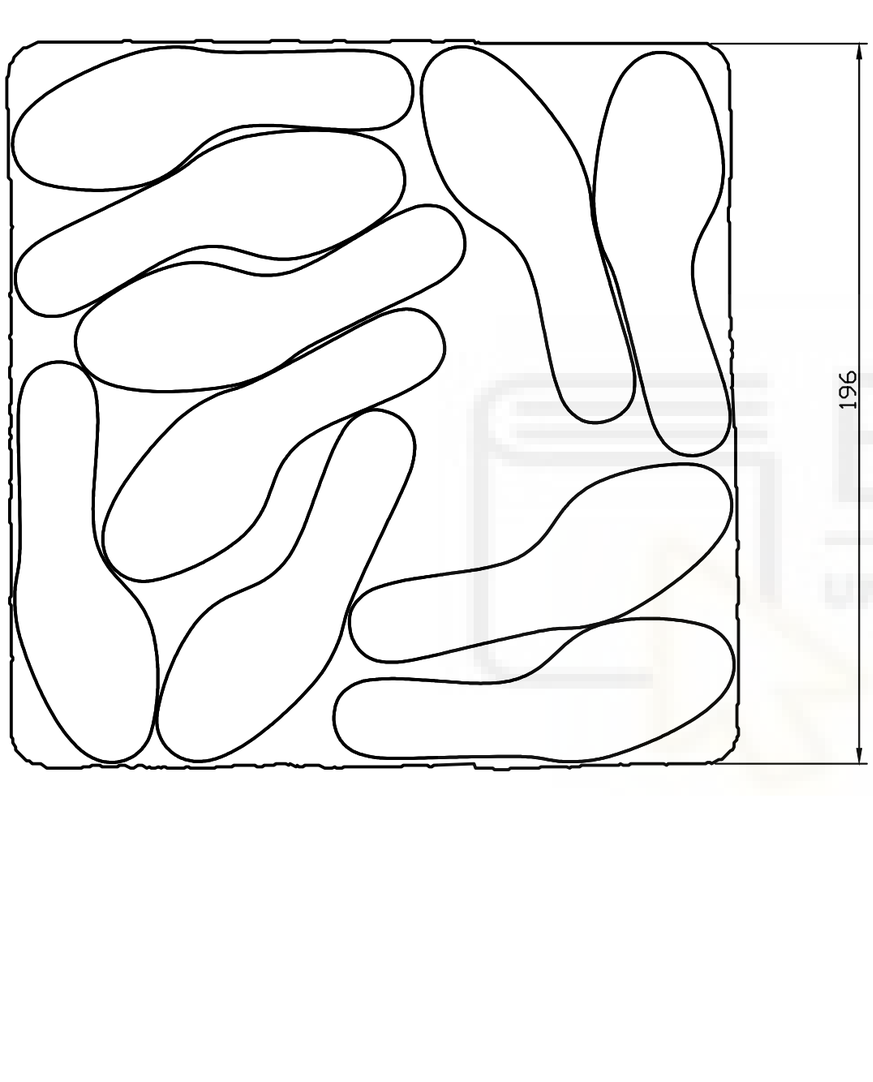



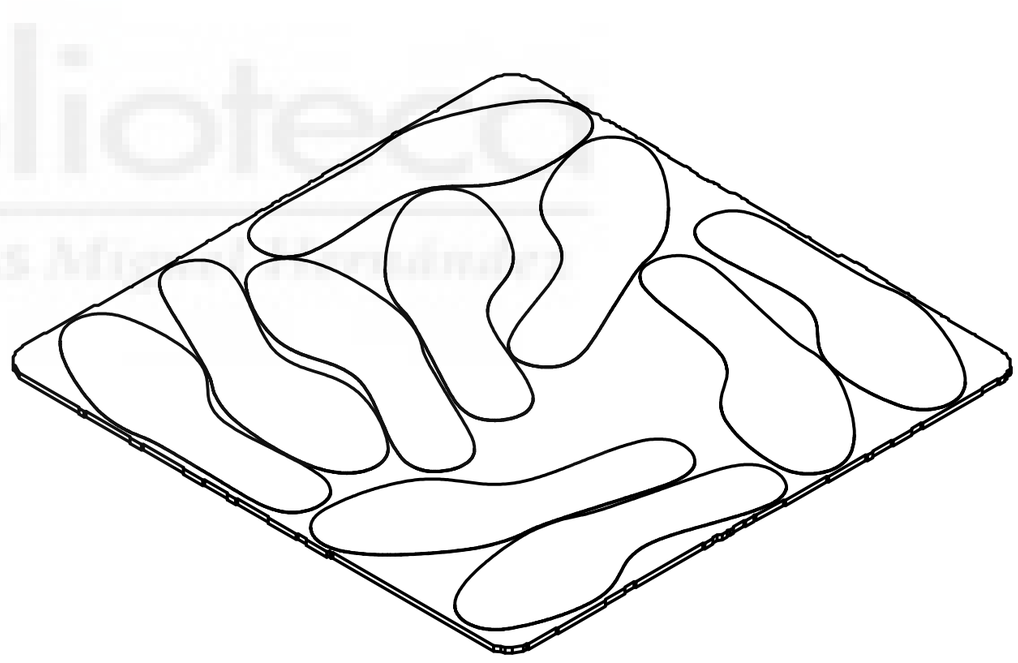
| | | | |
|---|---|---|---|
| Plano: PLANO 5. MESA LATERAL | Proyecto: Trabajo Fin de Grado | | Página: 100 |
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 5 | Formato: A3 |

### *9.1.2. Planos de la estación conexión AutoCAD-Python-RobotStudio*



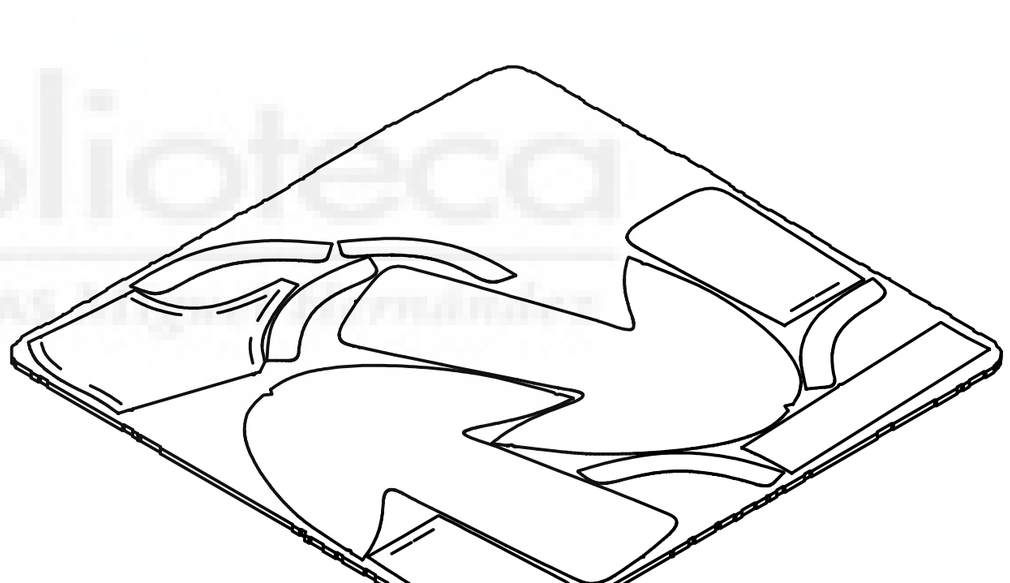



| | | | |
|---|---|---|---|
| Plano: PLANO 6. PLANCHA MODELO N04801.dxf | Proyecto: Trabajo Fin de Grado | | Página: 102 |
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 6 | Formato: A3 |



| | | | |
|---|---|---|---|
| Plano: PLANO 7. PLANCHA MODELO N02027.dxf | Proyecto: Trabajo Fin de Grado | | Página: 103 |
| Diseño de: FRANCISCO JOSÉ MARTÍNEZ PERAL | Fecha: Septiembre 2021 | Plano Nº: 7 | Formato: A3 |

## 9.2. Señales

### *9.2.1. Señales de la estación de Pick&Place de plantillas*

| Módulo de E/S. Controlador de los robots IRB360_6_1600_der y IRB360_6_1600_izq. | | | |
|---|---|---|---|
| **Entradas digitales** | **Mapeo** | **Salidas digitales** | **Mapeo** |
| Sensor_cinta | 0 | Coger_ven_izq | 10 |
| Sensor_CNC | 1 | Coger_ven_der | 11 |

Tabla 9.1: Señales E/S estación Pick&Place plantillas

### *9.2.2. Señales de la estación conexión AutoCAD-Python-Robotstudio*

| Módulo de E/S. Controlador estación conexión AutoCAD-Python-RobotStudio | | | |
|---|---|---|---|
| **Entradas digitales** | **Mapeo** | **Salidas digitales** | **Mapeo** |
| | | Coger_ven_izq | 0 |
| | | Coger_ven_der | 1 |

Tabla 9.2: Señales E/S estación conexión AutoCAD-Python-Robotstudio

## 9.3. Programas

### *9.3.1. Código RAPID de la Estación tutorial*

```
MODULE Module1
    CONST robtarget
Inicio:=[[669.162742149,0,394.566524049],[0.199804039,0,0.979835877,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30:=[[100,0,0],[0,0,1,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[-0.498663434,99.998756666,0],[0,0,1,0],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50:=[[-99.995026696,-0.997314468,0],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60:=[[0.498663434,-99.998756666,0],[0,0,1,0],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_70:=[[100,0,0],[0,0,1,0],[0,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    PROC main()

      Path_10;
    ENDPROC

    PROC Path_10()
      MoveL Inicio,v1000,fine,Weldgun\WObj:=wobj0;
      MoveL Target_30,v1000,fine,Weldgun\WObj:=Workobject_1;
      MoveC Target_40,Target_50,v1000,fine,Weldgun\WObj:=Workobject_1;
      MoveC Target_60,Target_70,v1000,fine,Weldgun\WObj:=Workobject_1;
      MoveL Inicio,v1000,fine,Weldgun\WObj:=wobj0;
    ENDPROC
ENDMODULE
```

### *9.3.2. Código RAPID de la Estación de Pick&Place de plantillas (T_ROB1)*

```
MODULE Module1
    CONST robtarget Inicio:=[[0,140,200.005],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_aprox_pequeña:=[[200,-220,90],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_aprox_grande:=[[-200,-220,90],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_aprox_mediana:=[[0,-220,90],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_grande:=[[-200,-220,0],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_mediana:=[[0,-220,0],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_pequeña:=[[200,-220,0],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget
Planta_1:=[[218.516,172.048,20.628],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_2:=[[-
37.635,156.114,19.208],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
```

```
    CONST robtarget
Planta_3:=[[216.753,263.851,20.619],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_4:=[[-
39.153,235.165,19.199],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
CONST robtarget Planta_5:=[[-218.045,247.252,18.207],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Planta_6:=[[-
309.847,245.489,17.698],[0,0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,
9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Planta_7:=[[-
52.634,323.788,19.124],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget
Planta_8:=[[218.516,372.577,20.628],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];

   CONST robtarget Planta_9:=[[-
54.152,402.838,19.116],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_10:=[[-
271.047,417.912,17.913],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_11:=[[216.753,464.38,20.619],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_12:=[[-
52.634,491.461,19.124],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_13:=[[-
272.171,476.443,17.907],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Planta_14:=[[-
271.047,539.764,17.913],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_15:=[[218.516,573.107,20.628],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget Planta_16:=[[-
54.152,570.512,19.116],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_17:=[[-
272.171,598.294,17.907],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_18:=[[216.753,664.91,20.619],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
```

```
    CONST robtarget Planta_19:=[[-
52.634,659.135,19.124],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_20:=[[-
271.047,661.616,17.913],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Planta_21:=[[-
272.171,720.146,17.907],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Planta_22:=[[-
54.152,738.185,19.116],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget
Planta_23:=[[218.516,773.637,20.628],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
   CONST robtarget Planta_24:=[[-
271.047,783.467,17.913],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Planta_25:=[[-
52.634,826.809,19.124],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_26:=[[-
272.171,841.998,17.907],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_27:=[[216.753,865.44,20.619],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_28:=[[-
54.152,905.859,19.116],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_29:=[[-
271.047,905.319,17.913],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_30:=[[218.516,974.166,20.628],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget Planta_31:=[[-
272.171,963.85,17.907],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_32:=[[-
52.634,994.482,19.124],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget Planta_33:=[[-
271.047,1027.171,17.913],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    CONST robtarget
Planta_34:=[[216.753,1065.969,20.619],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E
+09,9E+09,9E+09]];
```

```
    CONST robtarget Planta_35:=[[-
54.152,1073.533,19.116],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget Planta_36:=[[-
272.171,1085.702,17.907],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
    VAR num num_plantillas_grandes:=0;
    VAR num num_plantillas_medianas:=0;
    VAR num num_plantillas_pequeñas:=0;
    !***********************************************************
    !
    ! Módulo:  Module1
    !
    ! Descripción:
    !   <Introduzca la descripción aquí>
    !

    ! Autor: FRANCISCO JOSE MARTINEZ PERAL
    !
    ! Versión: 3.0
    !
    !***********************************************************
    PROC main()
        num_plantillas_grandes:=0;
        num_plantillas_medianas:=0;
        num_plantillas_pequeñas:=0;
        SetDO Coger_ven_izq,0;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        WaitDI Sensor_Cinta,1;
        Planta1;
        Dejar_plantilla_grande;
        Planta2;
        Dejar_plantilla_mediana;
        Planta3;
        Dejar_plantilla_grande;
        Planta4;
        Dejar_plantilla_mediana;
        Planta5;
        Dejar_plantilla_grande;
        Planta6;
        Dejar_plantilla_grande;
        Planta7;
        Dejar_plantilla_mediana;
        Planta8;
        Dejar_plantilla_grande;
        Planta9;
        Dejar_plantilla_mediana;
        Planta10;
        Dejar_plantilla_pequeña;
```

Planta11;
Dejar_plantilla_grande;
Planta12;
Dejar_plantilla_mediana;
Planta13;
Dejar_plantilla_pequeña;
Planta14;
Dejar_plantilla_pequeña;
Planta15;
Dejar_plantilla_grande;
Planta16;
Dejar_plantilla_mediana;
Planta17;
Dejar_plantilla_pequeña;
Planta18;
Dejar_plantilla_grande;
Planta19;
Dejar_plantilla_mediana;
Planta20;
Dejar_plantilla_pequeña;
Planta21;
Dejar_plantilla_pequeña;
Planta22;
Dejar_plantilla_mediana;
Planta23;
Dejar_plantilla_grande;
Planta24;
Dejar_plantilla_pequeña;
Planta25;
Dejar_plantilla_mediana;
Planta26;
Dejar_plantilla_pequeña;
Planta27;
Dejar_plantilla_grande;
Planta28;
Dejar_plantilla_mediana;
Planta29;
Dejar_plantilla_pequeña;
Planta30;
Dejar_plantilla_grande;
Planta31;
Dejar_plantilla_pequeña;
Planta32;
Dejar_plantilla_mediana;
Planta33;
Dejar_plantilla_pequeña;
Planta34;
Dejar_plantilla_grande;
Planta35;

```
        Dejar_plantilla_mediana;
        Planta36;
        Dejar_plantilla_pequeña;
    ENDPROC

    PROC Dejar_plantilla_grande()
        num_plantillas_grandes:=num_plantillas_grandes+1;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
Pos_planta_aprox_grande,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
Offs(Pos_planta_grande,0,0,num_plantillas_grandes*5.1),v3000,fine,Mi_Ventosa\
WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        MoveL
Pos_planta_aprox_grande,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    ENDPROC

    PROC Dejar_plantilla_mediana()
        num_plantillas_medianas:=num_plantillas_medianas+1;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
Pos_planta_aprox_mediana,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
Offs(Pos_planta_mediana,0,0,num_plantillas_medianas*5.1),v3000,fine,Mi_Ventos
a\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        MoveL
Pos_planta_aprox_mediana,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    ENDPROC

    PROC Dejar_plantilla_pequeña()
        num_plantillas_pequeñas:=num_plantillas_pequeñas+1;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
Pos_planta_aprox_pequeña,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
Offs(Pos_planta_pequeña,0,0,num_plantillas_pequeñas*5.1),v3000,fine,Mi_Ventos
a\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        MoveL
Pos_planta_aprox_pequeña,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    ENDPROC

    PROC Planta1()
        MoveL Planta_1,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
```

```
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_1,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta2()
    MoveL Planta_2,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_2,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta3()
    MoveL Planta_3,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_3,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta4()
    MoveL Planta_4,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_4,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta5()
    MoveL Planta_5,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_5,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta6()
    MoveL Planta_6,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_6,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta7()
    MoveL Planta_7,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_7,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta8()
    MoveL Planta_8,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_8,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta9()
    MoveL Planta_9,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
```

```
    MoveL Offs(Planta_9,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta10()
    MoveL Planta_10,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_10,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta11()
    MoveL Planta_11,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_11,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta12()
    MoveL Planta_12,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_12,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta13()
    MoveL Planta_13,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_13,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta14()
    MoveL Planta_14,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_14,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta15()
    MoveL Planta_15,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_15,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta16()
    MoveL Planta_16,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_16,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta17()
    MoveL Planta_17,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_17,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
```

```
ENDPROC

PROC Planta18()
    MoveL Planta_18,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_18,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta19()
    MoveL Planta_19,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_19,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta20()
    MoveL Planta_20,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_20,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta21()
    MoveL Planta_21,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_21,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta22()
    MoveL Planta_22,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_22,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta23()
    MoveL Planta_23,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_23,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta24()
    MoveL Planta_24,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_24,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta25()
    MoveL Planta_25,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_25,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC
```

```
PROC Planta26()
    MoveL Planta_26,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_26,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta27()
    MoveL Planta_27,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_27,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta28()
    MoveL Planta_28,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_28,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta29()
    MoveL Planta_29,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_29,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta30()
    MoveL Planta_30,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_30,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta31()
    MoveL Planta_31,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_31,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta32()
    MoveL Planta_32,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_32,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta33()
    MoveL Planta_33,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    SetDO Coger_ven_izq,1;
    MoveL Offs(Planta_33,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
ENDPROC

PROC Planta34()
    MoveL Planta_34,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
```

```
        SetDO Coger_ven_izq,1;
        MoveL Offs(Planta_34,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    ENDPROC

    PROC Planta35()
        MoveL Planta_35,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,1;
        MoveL Offs(Planta_35,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    ENDPROC

    PROC Planta36()
        MoveL Planta_36,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,1;
        MoveL Offs(Planta_36,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    ENDPROC
ENDMODULE
```

### *9.3.3. Código RAPID de la Estación de Pick&Place de plantillas (T_ROB2)*

```
MODULE Module1
    CONST robtarget Inicio_der:=[[0,140.78,200.005],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Inicio:=[[0,140.78,200.005],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_aprox_grande:=[[-200,-220,80],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_aprox_mediana:=[[0,-220,80],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_aprox_pequeña:=[[200,-220,80],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_grande:=[[-200,-220,0],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_mediana:=[[0,-220,0],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Pos_planta_pequeña:=[[200,-220,0],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget Planta_1:=[[-
216.752,1066.972,20.619],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9
E+09]];
```

```
    CONST robtarget
Planta_2:=[[54.153,1074.536,19.116],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget
Planta_3:=[[272.172,1086.705,17.907],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget Planta_4:=[[-
218.515,975.169,20.628],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_5:=[[52.635,995.485,19.124],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09
,9E+09,9E+09]];
    CONST robtarget
Planta_6:=[[271.048,1028.174,17.913],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget
Planta_7:=[[272.172,964.853,17.907],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget
Planta_8:=[[54.153,906.862,19.116],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09
,9E+09,9E+09]];
    CONST robtarget
Planta_9:=[[271.048,906.322,17.913],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_10:=[[-
216.752,866.443,20.619],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_11:=[[52.635,827.811,19.124],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget
Planta_12:=[[272.172,843.001,17.907],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget
Planta_13:=[[271.048,784.47,17.913],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_14:=[[-
218.515,774.64,20.628],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget
Planta_15:=[[54.153,739.188,19.116],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget
Planta_16:=[[272.172,721.149,17.907],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget Planta_17:=[[-
216.752,665.913,20.619],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
```

```
    CONST robtarget
Planta_18:=[[52.635,660.138,19.124],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget
Planta_19:=[[271.048,662.618,17.913],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget
Planta_20:=[[272.172,599.297,17.907],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget Planta_21:=[[-
218.515,574.11,20.628],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget
Planta_22:=[[54.153,571.515,19.116],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget
Planta_23:=[[271.048,540.766,17.913],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget
Planta_24:=[[52.635,492.464,19.124],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_25:=[[-
216.752,465.383,20.619],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
    CONST robtarget
Planta_26:=[[272.172,477.445,17.907],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget
Planta_27:=[[271.048,418.914,17.913],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+
09,9E+09,9E+09]];
    CONST robtarget
Planta_28:=[[54.153,403.841,19.116],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_29:=[[-
218.515,373.58,20.628],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget
Planta_30:=[[52.635,324.791,19.124],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_31:=[[218.046,248.255,18.207],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget
Planta_32:=[[309.848,246.492,17.698],[0,0.707106781,0.707106781,0],[0,0,0,0],[9
E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Planta_33:=[[-
216.752,264.853,20.619],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];
```

```
    CONST robtarget
Planta_34:=[[39.154,236.167,19.199],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    CONST robtarget Planta_35:=[[-
218.515,173.05,20.628],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+
09]];
    CONST robtarget
Planta_36:=[[37.636,157.117,19.208],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+0
9,9E+09,9E+09]];
    VAR num num_plantas_grandes:=0;
    VAR num num_plantas_medianas:=0;
    VAR num num_plantas_pequeñas:=0;


    !***********************************************************
    !
    ! Módulo:  Module1
    !
    ! Descripción:
    !   <Introduzca la descripción aquí>

    !
    ! Autor: FRANCISCO JOSE MARTINEZ PERAL
    !
    ! Versión: 3.0
    !
    !***********************************************************

    PROC main()
        num_plantas_grandes:=0;
        num_plantas_medianas:=0;
        num_plantas_pequeñas:=0;
        SetDO Coger_ven_der,0;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        WaitDI Sensor_Cinta,1;
        Planta1;
        Dejar_planta_grande_der;
        Planta2;
        Dejar_planta_mediana_der;
        Planta3;
        Dejar_planta_pequeña_der;
        Planta4;
        Dejar_planta_grande_der;
        Planta5;
        Dejar_planta_mediana_der;
        Planta6;
        Dejar_planta_pequeña_der;
        Planta7;
        Dejar_planta_pequeña_der;
```

Planta8;
Dejar_planta_mediana_der;
Planta9;
Dejar_planta_pequeña_der;
Planta10;
Dejar_planta_grande_der;
Planta11;
Dejar_planta_mediana_der;
Planta12;
Dejar_planta_pequeña_der;
Planta13;
Dejar_planta_pequeña_der;
Planta14;
Dejar_planta_grande_der;
Planta15;
Dejar_planta_mediana_der;
Planta16;
Dejar_planta_pequeña_der;
Planta17;
Dejar_planta_grande_der;
Planta18;
Dejar_planta_mediana_der;
Planta19;
Dejar_planta_pequeña_der;
Planta20;
Dejar_planta_pequeña_der;
Planta21;
Dejar_planta_grande_der;
Planta22;
Dejar_planta_mediana_der;
Planta23;
Dejar_planta_pequeña_der;
Planta24;
Dejar_planta_mediana_der;
Planta25;
Dejar_planta_grande_der;
Planta26;
Dejar_planta_pequeña_der;
Planta27;
Dejar_planta_pequeña_der;
Planta28;
Dejar_planta_mediana_der;
Planta29;
Dejar_planta_grande_der;
Planta30;
Dejar_planta_mediana_der;
Planta31;
Dejar_planta_grande_der;
Planta32;

```
        Dejar_planta_grande_der;
        Planta33;
        Dejar_planta_grande_der;
        Planta34;
        Dejar_planta_mediana_der;
        Planta35;
        Dejar_planta_grande_der;
        Planta36;
        Dejar_planta_mediana_der;
    ENDPROC

    PROC Dejar_planta_grande_der()
        num_plantas_grandes:=num_plantas_grandes+1;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL
Pos_planta_aprox_grande,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL
Offs(Pos_planta_grande,0,0,num_plantas_grandes*5.1),v3000,fine,Mi_Ventosa\W
Obj:=WO_Robot_der;
        SetDO Coger_ven_der,0;
        MoveL
Pos_planta_aprox_grande,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Dejar_planta_mediana_der()
        num_plantas_medianas:=num_plantas_medianas+1;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL
Pos_planta_aprox_mediana,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL
Offs(Pos_planta_mediana,0,0,num_plantas_medianas*5.1),v3000,fine,Mi_Ventosa
\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,0;
        MoveL
Pos_planta_aprox_mediana,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Dejar_planta_pequeña_der()
        num_plantas_pequeñas:=num_plantas_pequeñas+1;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL
Pos_planta_aprox_pequeña,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL
Offs(Pos_planta_pequeña,0,0,num_plantas_pequeñas*5.1),v3000,fine,Mi_Ventosa
\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,0;
```

```
        MoveL
Pos_planta_aprox_pequeña,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        MoveL Inicio,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta1()
        MoveL Planta_1,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_1,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta2()
        MoveL Planta_2,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_2,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta3()
        MoveL Planta_3,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_3,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;

    ENDPROC

    PROC Planta4()
        MoveL Planta_4,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_4,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta5()
        MoveL Planta_5,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_5,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta6()
        MoveL Planta_6,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_6,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta7()
        MoveL Planta_7,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
        SetDO Coger_ven_der,1;
        MoveL Offs(Planta_7,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    ENDPROC

    PROC Planta8()
```

```
    MoveL Planta_8,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_8,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta9()
    MoveL Planta_9,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_9,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta10()
    MoveL Planta_10,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_10,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta11()
    MoveL Planta_11,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_11,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta12()
    MoveL Planta_12,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_12,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta13()
    MoveL Planta_13,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_13,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta14()
    MoveL Planta_14,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_14,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta15()
    MoveL Planta_15,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_15,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta16()
    MoveL Planta_16,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
```

```
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_16,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta17()
    MoveL Planta_17,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_17,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta18()
    MoveL Planta_18,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_18,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta19()
    MoveL Planta_19,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_19,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta20()
    MoveL Planta_20,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_20,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta21()
    MoveL Planta_21,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_21,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta22()
    MoveL Planta_22,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_22,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta23()
    MoveL Planta_23,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_23,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta24()
    MoveL Planta_24,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
```

```
    MoveL Offs(Planta_24,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta25()
    MoveL Planta_25,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_25,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta26()
    MoveL Planta_26,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_26,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta27()
    MoveL Planta_27,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_27,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta28()

    MoveL Planta_28,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_28,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta29()
    MoveL Planta_29,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_29,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta30()
    MoveL Planta_30,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_30,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta31()
    MoveL Planta_31,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_31,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
ENDPROC

PROC Planta32()
    MoveL Planta_32,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
```

```
    MoveL Offs(Planta_32,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
  ENDPROC

  PROC Planta33()
    MoveL Planta_33,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_33,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
  ENDPROC

  PROC Planta34()
    MoveL Planta_34,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_34,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
  ENDPROC

  PROC Planta35()
    MoveL Planta_35,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_35,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
  ENDPROC

  PROC Planta36()
    MoveL Planta_36,v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;

    SetDO Coger_ven_der,1;
    MoveL Offs(Planta_36,0,0,10),v3000,fine,Mi_Ventosa\WObj:=WO_Robot_der;
  ENDPROC
ENDMODULE
```

### *9.3.4. Código de Python para la conexión AutoCAD-Python-RobotStudio*

```
## PORGRAMA DE PYTHON PARA LA CONEXION AUTOCAD-PYTHON-
ROBOTSUDIO Y OBTIENE LAS POSICIONES Y ROTACION DE LAS PLANTILLA
S ##

import socket
import os
import subprocess
from os import remove

print("\n\t## CONEXION AUTOCAD-PYTHON-ROBOTSTUDIO ##\n ")

# Creamos la conexion con RobotStudio
obj = socket.socket()
host = '127.0.0.1'
port = 1024
obj.connect((host,port))
```

```
# Esperamos respuesta de RobotStudio
respuesta = obj.recv(1024)
print(respuesta)

# Pide por pantalla el archivo del que se van a obtener los datos (modelo de plantill
a)
print("-Introduce el archivo .dxf: ")
dxf = input()
dat = ['_open C:/datos/' + dxf + '.dxf \n', '-
EXTRACDAT \n', 'C:/datos/datos.dxe \n', 'QUITA\n']

# Creamos el script get_data.scr con el archivo dxf que se debe de abrir para obte
ner los datos
with open('C:/datos/get_data.scr','w+') as fich:
    fich.writelines(dat)

print("-Archivo get_data.scr creado con exito.")
print("-Abriendo Autocad y obteniendo datos del archivo " + dxf + ".dxf...")

# Abrimos AutoCAD y ejecutamos el script get_data.scr para obtener los datos sub
process.run(['C:/Program Files/Autodesk/AutoCAD 2020/acad.exe','/b','C:/datos/get
_data.scr'])
print ("-Fichero datos.txt creado correctamente.")

# Leemos el archivo generado con los datos (datos.txt)
print("-Lectura del fichero datos.txt")
fich = open('C:/datos/datos.txt','r')
read_fich = fich.readlines()
fich.close()

# Se almacenan los datos en una lista de Python (datos)
datos=[]
for i in range(len(read_fich)):
    if i == 0:
        pass
    else:
        datos.append(read_fich[i].rstrip().split('\t'))

# Creamos un fichero data.txt en la carpeta HOME del controlador de RobtoStudio
y escribimos los datos que serán leidos desde RobotStudio
print("-
Creando fichero data.txt en la carpeta HOME del Controlador Virtual de RobotStudi
o")
with open('C:/Users/franm/Documents/RobotStudio/Solutions/estacion_robots_para
lelos_TFG/Virtual Controllers/Controlador_TFG_Estacion_1/HOME/data.txt','w+') a
s fich2:
```

```
    for i in range(len(datos)):
        datos[i][0] = datos[i][0].replace(' ','_')
        datos[i][0] = datos[i][0].replace('(','')
        datos[i][0] = datos[i][0].replace(')','')
        datos[i][1] = datos[i][1][0:len(datos[i][1])-5]
        datos[i][2] = datos[i][2][0:len(datos[i][2])-5]
        fich2.write(datos[i][0] + " " + datos[i][1] + " " + datos[i][2] + " " + datos[i][3] + "\n"
)
print("-Fichero data.txt creado con exito")

# Enviamos una señal a RobotStudio para que el robot lea el fichero data.txt
signal = b'1'
print("-Enviando señal a RobotStudio")
obj.send(signal)
print("-Señal enviada con exito")

# Eliminamos los archivos datos.txt y get_data.scr para no tener porblemas la sigui
ente vez que queramos obtener datos desde otro modelo de plantila.
remove('C:/datos/datos.txt')
print("-Archivo datos.txt eliminado con exito.")
remove('C:/datos/get_data.scr')
print("-Archivo get_data.scr eliminado con exito.")
```

### *9.3.5. Código de RAPID para la conexión AutoCAD-Python-RobotStudio*

```
MODULE Module1

    VAR socketdev Server_Socket;
    VAR socketdev Client_Socket;
    VAR string client_ip;
    VAR string data;
    VAR string signal;

    CONST robtarget Inicio:=[[0,140,200],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    ! Modelo N04801
    CONST robtarget pos_plantilla_der:=[[-100,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget pos_plantilla_izq:=[[100,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
```

```
    ! Modelo N02027
    CONST robtarget pos_plantilla_Pieza_5:=[[-250,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget pos_plantilla_Pieza_3:=[[420,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget pos_plantilla_otras_4000:=[[-410,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget pos_plantilla_Forro:=[[0,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];
    CONST robtarget pos_plantilla_Corte:=[[265,-220,5],[0,-
0.707106781,0.707106781,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]
];


    VAR iodev simfile;
    VAR iodev datafile;
    CONST string datadirectorio:="HOME:/data.txt";

    VAR string modelo{20};
    VAR num coorX{20};
    VAR num coorY{20};
    VAR num rota{20};
    VAR num num_piezas;
    VAR num altura:=6;
    VAR robtarget plantilla{20};
    VAR num num_plantilla;
    VAR num num_plantilla_der;
    VAR num num_plantilla_izq;
    VAR num num_pieza_5;

    VAR pose pose1;

    VAR num xmax;
    VAR num xmin;
    VAR num ymax;
    VAR num ymin;
    VAR num xoffset;
    VAR num yoffset;



    !***********************************************************
    !
    ! Módulo:  Module1
    !
```

```
! Descripción:
!   <Introduzca la descripción aquí>
!
! Autor: Fran
!
! Versión: 1.0
!
!*********************************************************

PROC main()

    conexion;
    MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    WaitUntil signal="adelante";
    WaitTime 5;
    Read;
    Calculate_offset;
    Generate;
    Secuence;

ENDPROC

PROC conexion()

    !Conexion RobotStudio con Python
    SocketCreate Server_Socket;
    SocketBind Server_Socket,"127.0.0.1",1024;
    SocketListen Server_Socket;
    TPWrite "Esperando conexion con PC";
    SocketAccept Server_Socket,Client_Socket,\Time:=WAIT_MAX;
    SocketSend Client_Socket\Str:="Conexion con ABB correcta.";
    !Recibe una señal
    SocketReceive Client_Socket\Str:=data;
    IF data="1" THEN
        signal:="adelante";
    ELSE
        signal:="todavia no";
    ENDIF

ENDPROC

PROC Read()

    num_piezas:=11;
    !       num_piezas:=10;
    !Abrimos el fichero que vamos a leer
    Open datadirectorio,datafile\Read;
```

```
    !Lectura del fichero data.txt
    FOR i FROM 1 TO num_piezas DO
        modelo{i}:=ReadStr(datafile\Delim:=" ");
        coorX{i}:=ReadNum(datafile\Delim:=" ");
        coorY{i}:=ReadNum(datafile\Delim:=" ");
        rota{i}:=ReadNum(datafile\Delim:="\09");
    ENDFOR
    !Cerramos el fichero que hemos leido
    close datafile;

ENDPROC

PROC Calculate_offset()
    ! Cálculo del lugar donde está centrada la pieza
    xmax:=0;
    xmin:=5000;
    ymax:=0;
    ymin:=5000;
    xoffset:=0;
    yoffset:=0;
    FOR i FROM 1 TO num_piezas DO
        IF coorX{i}>xmax THEN
            xmax:=coorX{i};
        ELSE
            xmax:=xmax;
        ENDIF
        IF coorX{i}<xmin THEN
            xmin:=coorX{i};
        ELSE
            xmin:=xmin;
        ENDIF

        IF coorY{i}>ymax THEN
            ymax:=coorY{i};
        ELSE
            ymax:=ymax;
        ENDIF
        IF coorY{i}<ymin THEN
            ymin:=coorY{i};
        ELSE
            ymin:=ymin;
        ENDIF
    ENDFOR
    xoffset:=((xmax-xmin)/2)+xmin;
    yoffset:=(ymax-ymin)/2;
ENDPROC
```

```
    PROC Generate()

        !Generamos la posicion donde se encuentran las plantillas
        FOR j FROM 1 TO num_piezas DO

            IF modelo{j}="P10_4000" THEN
                pose1.rot:=OrientZYX(rota{j}+90,0,180);
                pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset,altura];
            ELSEIF modelo{j}="P10_MIRRORED_4000" THEN
                pose1.rot:=OrientZYX(rota{j}+90,0,180);
                pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset,altura];
            ELSEIF modelo{j}="Pieza_5_Otras_4000" OR
modelo{j}="Pieza_5_Otras_mirrored_4000" OR
modelo{j}="Pieza_5_Otras_mirrored_3900" OR
modelo{j}="Pieza_5_Otras_mirrored_3500" OR modelo{j}="Pieza_5_Otras_4200"
THEN
                IF modelo{j}="Pieza_5_Otras_4000" THEN
                    pose1.rot:=OrientZYX(rota{j}+45,0,180);
                    pose1.trans:=[coorX{j}-xoffset+10,coorY{j}-yoffset+10,altura];
                ELSEIF modelo{j}="Pieza_5_Otras_mirrored_4000" THEN
                    pose1.rot:=OrientZYX(rota{j}+135,0,180);
                    pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset+10,altura];
                ELSEIF modelo{j}="Pieza_5_Otras_mirrored_3900" THEN
                    pose1.rot:=OrientZYX(rota{j}+135,0,180);
                    pose1.trans:=[coorX{j}-xoffset-10,coorY{j}-yoffset+10,altura];
                ELSEIF modelo{j}="Pieza_5_Otras_mirrored_3500" THEN
                    pose1.rot:=OrientZYX(rota{j}+135,0,180);
                    pose1.trans:=[coorX{j}-xoffset+10,coorY{j}-yoffset,altura];
                ELSEIF modelo{j}="Pieza_5_Otras_4200" THEN
                    pose1.rot:=OrientZYX(rota{j}+45,0,180);
                    pose1.trans:=[coorX{j}-xoffset+10,coorY{j}-yoffset+10,altura];
                ENDIF
            ELSEIF modelo{j}="Pieza_Otras_4000" THEN
                pose1.rot:=OrientZYX(rota{j},0,180);
                pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset,altura];
            ELSEIF modelo{j}="Pieza_Forro_mirrored_4100" OR
modelo{j}="Pieza_Forro_4100" THEN
                pose1.rot:=OrientZYX(rota{j}+90,0,180);
                pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset,altura];
            ELSEIF modelo{j}="Pieza_3_mirrored_4100" THEN
                pose1.rot:=OrientZYX(rota{j},0,180);
                pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset,altura];
            ELSEIF modelo{j}="Corte_mirrored_3500" OR modelo{j}="Corte_4200"
THEN
                pose1.rot:=OrientZYX(rota{j}+90,0,180);
                pose1.trans:=[coorX{j}-xoffset,coorY{j}-yoffset,altura];
            ENDIF
```

```
plantilla{j}:=[pose1.trans,pose1.rot,[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E
+09]];

        ENDFOR
    ENDPROC

    PROC Secuence()

        num_pieza_5:=0;
        num_plantilla:=0;
        num_plantilla_der:=0;
        num_plantilla_izq:=0;
        FOR k FROM 1 TO num_piezas DO
            num_plantilla:=num_plantilla+1;
            Pick;
            NewPlace;
            SetDO Coger_ven_izq,0;
        ENDFOR

    ENDPROC

    PROC Pick()

        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL
offs(plantilla{num_plantilla},0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq
;
        MoveL plantilla{num_plantilla},v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,1;
        WaitTime 0.5;
        MoveL
offs(plantilla{num_plantilla},0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq
;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ENDPROC

    PROC NewPlace()

        IF modelo{num_plantilla}="P10_4000" THEN
            num_plantilla_der:=num_plantilla_der+1;
            dejar_plantilla_derecha;
        ELSEIF modelo{num_plantilla}="P10_MIRRORED_4000" THEN
            num_plantilla_izq:=num_plantilla_izq+1;
            dejar_plantilla_izquierda;
```

```
        ELSEIF modelo{num_plantilla}="Pieza_5_Otras_4000" OR
modelo{num_plantilla}="Pieza_5_Otras_mirrored_4000" OR
modelo{num_plantilla}="Pieza_5_Otras_mirrored_3900" OR
modelo{num_plantilla}="Pieza_5_Otras_mirrored_3500" OR
modelo{num_plantilla}="Pieza_5_Otras_4200" THEN
            num_pieza_5:=num_pieza_5+1;
            dejar_plantilla_Pieza_5;
        ELSEIF modelo{num_plantilla}="Pieza_Otras_4000" THEN
            dejar_plantilla_otras_4000;
        ELSEIF modelo{num_plantilla}="Pieza_Forro_mirrored_4100" OR
modelo{num_plantilla}="Pieza_Forro_4100" THEN
            dejar_plantilla_Forro;
        ELSEIF modelo{num_plantilla}="Pieza_3_mirrored_4100" THEN
            dejar_plantilla_Pieza_3;
        ELSEIF modelo{num_plantilla}="Corte_mirrored_3500" OR
modelo{num_plantilla}="Corte_4200" THEN
            dejar_plantilla_Corte;
        ENDIF

    ENDPROC

    PROC dejar_plantilla_Pieza_5()

        MoveL offs(pos_plantilla_Pieza_5,-5,200-
50*num_pieza_5,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL offs(pos_plantilla_Pieza_5,-5,200-
50*num_pieza_5,0),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        WaitTime 0.5;
        MoveL offs(pos_plantilla_Pieza_5,-5,200-
50*num_pieza_5,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ENDPROC

    PROC dejar_plantilla_Pieza_3()

        MoveL
offs(pos_plantilla_Pieza_3,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL pos_plantilla_Pieza_3,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        WaitTime 0.5;
        MoveL
offs(pos_plantilla_Pieza_3,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ENDPROC

    PROC dejar_plantilla_otras_4000()
```

```
        MoveL
offs(pos_plantilla_otras_4000,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_i
zq;
        MoveL
pos_plantilla_otras_4000,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        WaitTime 0.5;
        MoveL
offs(pos_plantilla_otras_4000,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_i
zq;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ENDPROC

    PROC dejar_plantilla_Forro()

        IF modelo{num_plantilla}="Pieza_Forro_mirrored_4100" THEN

            MoveL
offs(pos_plantilla_Forro,20,90,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
            MoveL
offs(pos_plantilla_Forro,20,90,0),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
            SetDO Coger_ven_izq,0;
            WaitTime 0.5;
            MoveL
offs(pos_plantilla_Forro,20,90,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
            MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

        ELSEIF modelo{num_plantilla}="Pieza_Forro_4100" THEN

            MoveL offs(pos_plantilla_Forro,0,-
90,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
            MoveL offs(pos_plantilla_Forro,0,-
90,0),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
            SetDO Coger_ven_izq,0;
            WaitTime 0.5;
            MoveL offs(pos_plantilla_Forro,0,-
90,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
            MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

        ENDIF

    ENDPROC

    PROC dejar_plantilla_Corte()

        IF modelo{num_plantilla}="Corte_mirrored_3500" THEN
```

```
        MoveL offs(pos_plantilla_Corte,0,-
50,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL offs(pos_plantilla_Corte,0,-
50,0),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        WaitTime 0.5;
        MoveL offs(pos_plantilla_Corte,0,-
50,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ELSEIF modelo{num_plantilla}="Corte_4200" THEN

        MoveL
offs(pos_plantilla_Corte,25,100,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_iz
q;
        MoveL
offs(pos_plantilla_Corte,25,100,0),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        SetDO Coger_ven_izq,0;
        WaitTime 0.5;
        MoveL
offs(pos_plantilla_Corte,25,100,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_iz
q;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ENDIF

  ENDPROC

  PROC dejar_plantilla_derecha()

    MoveL
offs(pos_plantilla_der,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    MoveL
Offs(pos_plantilla_der,0,0,num_plantilla_der*5.1),v1000,fine,Mi_Ventosa\WObj:=W
O_Robot_izq;
    SetDO Coger_ven_izq,0;
    WaitTime 0.1;
    MoveL
offs(pos_plantilla_der,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
    MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

  ENDPROC

  PROC dejar_plantilla_izquierda()

    MoveL
offs(pos_plantilla_izq,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
```

```
        MoveL
Offs(pos_plantilla_izq,0,0,num_plantilla_izq*5.1),v1000,fine,Mi_Ventosa\WObj:=W
O_Robot_izq;
        SetDO Coger_ven_izq,0;
        WaitTime 0.1;
        MoveL
offs(pos_plantilla_izq,0,0,100),v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;
        MoveL Inicio,v1000,fine,Mi_Ventosa\WObj:=WO_Robot_izq;

    ENDPROC

    PROC Place()
        dejar_plantilla_derecha;
        dejar_plantilla_izquierda;
    ENDPROC

ENDMODULE
```

## 9.4. Hojas de características de los Robots

### *9.4.1. Hoja de características del Robot IRB120 [2]*

ROBOTICS

# IRB 120

## ABB's 6 axis robot – for flexible and compact production

The IRB 120 robot is the latest addition to ABB's new fourth-generation of robotic technology. It is ideal for material handling and assembly applications and provides an agile, compact and lightweight solution with superior control and path accuracy.



**Compact and lightweight**
IRB 120's compact design enables it to be mounted virtually anywhere at any angle without any restriction - for example inside a cell, on top of a machine or close to other robots.

IRB 120 is also the most portable and easy to integrate on the market with its 25 kg weight. The smooth surfaces are easy to clean and the cables for air and customer signals are internally routed, all the way from the foot to the wrist, ensuring that integration is effortless.

**Multipurpose**
IRB 120 is ideal for a wide range of industries including the electronic, food and beverage, machinery, solar, pharmaceutical, medical and research sectors.

The Food Grade Lubrication (NSF H1) option includes Clean Room ISO Class 5, which ensures uncompromising safety and hygiene for food and beverage applications.

**Optimized working range**
IRB 120 has a horizontal reach of 580 mm, the best in class stroke, the ability to reach 112 mm below its base and a very compact turning radius.

**Fast, accurate and agile**
Designed with a light, aluminum structure, the motors ensure the robot is enabled with a fast acceleration, and can deliver accuracy and agility in any application.

**IRC5 Compact controller – optimized for small robots**
ABB's new IRC5 Compact controller presents the capabilities of the IRC5 controller in a compact format. It brings accuracy and motion control to applications which have been exclusive to large installations and enables easy commissioning through one phase power input, external connectors for all signals and a built-in expandable 16 in, 16 out, I/O system.

RobotStudio for offline programming enables manufacturers to simulate a production cell to find the optimal position for the robot, and provide offline programming to prevent costly downtime and delays to production.

**Reduced footprint**
The combination of the new lightweight architecture of the IRB 120 with the new IRC5 Compact controller introduces a significantly reduced footprint.

## Specification

| Robot version | Reach (m) | Handling capacity (kg) | Armload (kg) |
|---|---|---|---|
| IRB 120-3/0.6 | 0.58 | 3* | 0.30 |
| Number of axes | 6 | | |
| Protection | IP30 | | |
| Mounting | Any angle | | |
| Controller | IRC5 Compact/IRC5 Single Cabinet | | |
| Integrated signal supply | 10 signals on wrist | | |
| Integrated air supply | 4 air on wrist (5 bar) | | |

* 4 with vertical wrist

## Performance (according to ISO 9283)

| | IRB 120 |
|---|---|
| 1 kg picking cycle | |
| 25 x 300 x 25 mm | 0.58 s |
| 25 x 300 x 25 with 180° axis 6 reorientation | 0.92 s |
| Acceleration time 0-1 m/s | 0.07 s |
| Position repeatability | 0.01 mm |

## Technical information

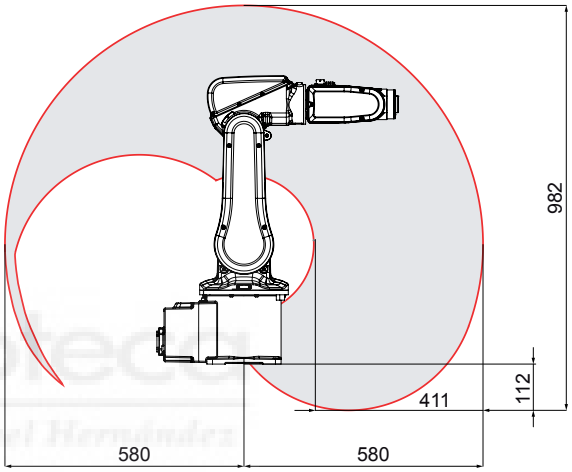
| | |
|---|---|
| **Electrical Connections** | |
| Supply voltage | 200-600 V, 50/60 Hz |
| Rated power transformer rating | 3.0 kVA |
| Power consumption | 0.24 kW |
| **Physical** | |
| Robot base | 180 x 180 mm |
| Robot height | 700 mm |
| Robot weight | 25 kg |
| **Environment** | |
| Ambient temperature for robot manipulator: | |
| During operation | +5°C (41°F) to +45°C (113°F) |
| During transportation and storage | -25°C (-13°F) to +55°C (131°F) |
| During short periods (max. 24 h) | up to +70°C (158°F) |
| Relative humidity | Max. 95% |
| Noise level | Max. 70 dB (A) |
| Safety | Safety and emergency stops 2-channel safety circuits supervision, 3-position enabling device |
| Emission | EMC/EMI-shielded |
| Options | Clean Room ISO class 5 (certified by IPA)** |

** ISO class 4 can be reached under certain conditions.
Data and dimensions may be changed without notice.

## Movement

| Axis movement | Working range | Velocity IRB 120 |
|---|---|---|
| Axis 1 rotation | +165° to -165° | 250°/s |
| Axis 2 arm | +110° to -110° | 250°/s |
| Axis 3 arm | +70° to -110° | 250°/s |
| Axis 4 wrist | +160° to -160° | 320°/s |
| Axis 5 bend | +120° to -120° | 320°/s |
| Axis 6 turn | Default: +400° to -400° Max. rev: +242 to -242 | 420°/s |

## Working range



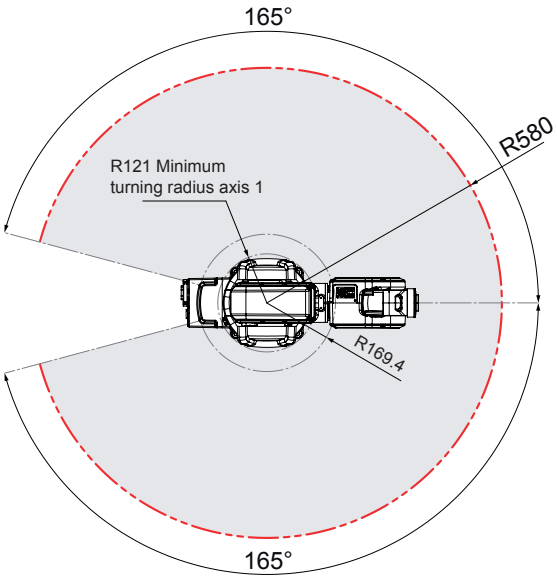




**abb.com/robotics**



ROBO149EN_D Rev. J November 2019

### *9.4.2. Hoja de características del Robot IRB360 [3]*

ROBOTICS

# IRB 360 FlexPicker®

## Greater flexibility in a compact footprint


ABB's IRB 360 FlexPicker® has been the leader in state-of-the-art high speed robotic picking and packing technology. Compared to conventional hard automation, the IRB 360 offers much greater flexibility in a compact footprint while maintaining accuracy and high payloads.



ABB's IRB 360 family of robots, more commonly known as the FlexPicker, are capable of the fastest picking applications and have been optimized for packing applications. The robot has outstanding motion performance with the shortest cycle times, precision accuracy, and high payloads.

The IRB 360 family includes variants with payloads of 1 kg, 3 kg, 6 kg, and 8 kg and reaches of 1130 mm and 1600 mm—meaning there is an IRB 360 for almost every need.

Featuring outstanding motion control, short cycle times, and precision accuracy, an IRB 360 can operate at very high speeds in anything from narrow to wide spaces with very tight tolerances. Every FlexPicker also benefits from a re-engineered tool flange which can accommodate larger grippers, allowing for efficient handling of flow wrapped products at high speeds from an indexing belt.

**Food handling applications**
A stainless option for food handling applications is IP69K validated so that it can be washed down with industrial detergents and high pressure hot water. This variant is also designed with smooth and easy to rinse-off surfaces and lubricant free joints that are resistant to most corrosives.

**Software and controls**
Setting up an application becomes easy using PickMaster® software which has evolved into an invaluable help for integrators and users. It simplifies vision config-uration and offers the application tools needed for efficient high speed picking.

The reliable, market leading IRC 5 controller is also an integral part of the FlexPicker robot family. The IRC 5 with TrueMove™ and QuickMove™ guarantees high speeds and path following facilities which allows these robots to track fast moving conveyor belts with extreme accuracy.

The IRC 5 also is available in a panel-mounted version that offers substantial space savings and easy integration into machines and production lines.

**Features**
- High speed flexibility
- High capacity – up to 8 kg payload
- Hygienic design for washdown applications
- Superior tracking performance
- Integrated vision software
- Integrated control of indexing belts

## Specifications

| Robot version | Handling capacity | Diameter | No. Axes |
|---|---|---|---|
| IRB 360-1/1130* | 1 kg | 1130 mm | 3/4 |
| IRB 360-3/1130 | 3 kg | 1130 mm | 3/4 |
| IRB 360-1/1600 | 1 kg | 1600 mm | 4 |
| IRB 360-6/1600 | 6 kg | 1600 mm | 4 |
| IRB 360-8/1130 | 8 kg | 1130 mm | 4 |
| Supplementary load | on upper arm | | 350 g |
| | on lower arm | | 350 g |
| Integrated signal supply | 12 poles 50V, 250mA | | |
| Integrated vacuum supply | Max. 7 bar/max vacuum 0.75 bar | | |

* Tested by IPA, all axes

## Physical

| | | |
|---|---|---|
| Robot mounting | Inverted | |
| Weight | 120 kg (Standard & wash down) | |
| | 145 kg (Stainless wash down) | |

## Performance

| | | |
|---|---|---|
| Position repeatability | 0.1 mm | |
| Angular repeatability | Standard & stainless axis 4 | 0.4° |
| | Wash-down axis 4 | 1.5° |

## Cycle times

| | 0.1 kg | 1 kg | 3 kg | 6 kg | 8 kg |
|---|---|---|---|---|---|
| 25/305/25 (mm) | | | | | |
| IRB 360-1/1130 | 0.30 | 0.36 | | | |
| IRB 360-3/1130 | 0.40 | 0.40 | 0.52 | | |
| IRB 360-8/1130 | | 0.38 | 0.42 | | 0.60 |
| IRB 360-1/1600 | 0.35 | 0.40 | | | |
| IRB 360-6/1600 | | 0.43 | 0.48 | 0.60 | |
| 90/400/90 (mm) | | | | | |
| IRB 360-1/1130 | 0.44 | 0.51 | | | |
| IRB 360-3/1130 | 0.60 | 0.60 | 0.75 | | |
| IRB 360-8/1130 | | 0.55 | 0.65 | | 0.92 |
| IRB 360-1/1600 | 0.50 | 0.54 | | | |
| IRB 360-6/1600 | | 0.57 | 0.63 | 0.80 | |

** The cycle times are measured under real conditions, but may vary depending on the actual application. Please use RobotStudio or real cycle tests to verify actual cycle time.

## Electrical connections

| | | |
|---|---|---|
| Supply voltage | 200-600 V, 60 Hz | |
| Transformer rating | 7.2 kVA | |
| Power consumption | type of movement | IRB 360/1 |
| | typical pick & place cycle (1 kg) | 0.477 kW |

## Conveyor tracking

| | speed (mm/s) | repeatability |
|---|---|---|
| Constant conveyor | 200 | 1.0 mm |
| | 350-750 | 1.5 mm |
| | 800-1400 | 5.0 mm |
| Start/stop conveyor | 500 (start/stop in 0.2 sec) | 3.5 mm |
| Indexing conveyor control | 3.5 g acceleration/ deceleration | 2.0 mm |

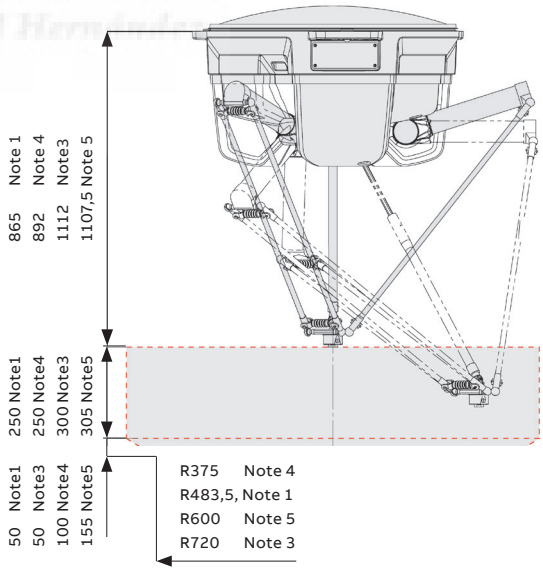
**The tracking is measured under real conditions with IRB 360-1/1130 & PickMaster. The figures may vary depending on the actual robot max. speed & acceleration in relation to the application demands.

## Environment

| | | |
|---|---|---|
| Ambient temperature | IRB 360 manipulator | ±0°C to +45°C |
| | Relative humidity | Max. 95 % |
| Noise level | < 70 dB(A) | |
| Safety | Double circuits with supervision, emergency stops & safety functions, 3 position enabling device | |
| Emission | EMC/EMI shielded | |
| Clean room options axis 4 | Standard | Clean room 7 |
| | Wash down | Clean room 5* |
| | Stainless | Clean room 5* |
| | Option | Collision detection |

* Certified by IPA
** Data and dimensions may be changed without notice

—
Working range




Note 1: IRB 360-1/1130 and IRB 360-3/1130
Note 3: IRB 360-1/1600
Note 4: IRB 360-8/1130
Note 5: IR B360-6/1600

—
**abb.com/robotics**

—


ROB0082EN Rev.J May 2020