UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

AGRADECIMIENTOS

"A mi familia
A mi tutor
A mis compañeros
…"

RESUMEN

En este proyecto se realiza la optimización de una línea de producción en una fábrica de discos de frenos, para distintos modelos de coches de la marca Volkswagen, mediante la implementación de un robot industrial ABB IRB4600. Se busca sustituir la introducción manual de discos de freno en la línea mediante un proceso automatizado utilizando un robot industrial. Se realiza una simulación de la celda en RobotStudio para visualizar el proceso, gestionar y controlar tiempos de ciclo del mismo y analizar así la viabilidad del proyecto antes de llevarlo a cabo.

Se va a realizar la programación del robot mediante la escritura del código del programa en RobotStudio, utilizando lenguaje de programación RAPID, para posteriormente sincronizarlo con la estación. Se pretende realizar la comunicación entre robot y PLC utilizando componentes inteligentes para gestionar las entradas y salidas, la detección de piezas mediante sensores, animaciones de la estación etc.

En una herramienta de CAD (Inventor) se crean todas las geometrías necesarias para llevar a cabo la simulación de la estación, como piezas, blisters, herramientas del robot, soportes de herramientas y demás modelos necesarios.

Tareas a realizar:

1. Programación del robot ABB modelo IRB4600 con RobotStudio.

2. Simulación del robot en el entorno de una línea de producción de discos de freno del sector de la automoción.

3. Diseño en Inventor de herramientas y útiles necesarios en el proyecto.

4. Optimización de la línea mediante la reducción de tiempos de ciclo.

Palabras Clave: RobotStudio, ABB, Simulación, Automatización Industrial, Robot, RAPID, Componentes Inteligentes, Inventor.

ABSTRACT

This project optimizes a production line in a brake disk factory for different Volkswagen car models through the implementation of an ABB IRB4600_60_205 industrial robot. It seeks to replace the manual introduction of brake disks in the line by means of an automated process using an industrial robot. A simulation of the cell is carried out in RobotStudio to visualize the process, manage and control cycle times to finally analyze the viability of the project before carrying it out.

The robot will be programmed by writing the program code in RobotStudio using RAPID language synchronized with the station. It is intended to carry out the communication between robot and PLC using smart components to manage inputs and outputs, part present with sensors, station animation etc.

In a CAD software (Inventor) all necessary geometries are created to carry out the simulation of the station such as parts, blisters, robot tools, tool stands and other used objects.

Tasks to perform:

1. Programming of the ABB robot model IRB4600 in RobotStudio.

2. Robot simulation within the production of the brake disk environment production line in the automotive sector.

3. Design in Inventor of tools and supplies needed in the project.

4. Optimization of the line by reducing cycle times.

Keywords: RobotStudio, ABB, Simulation, Automatization Industrial, Robot, RAPID, Smart Components, Inventor.

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL PROYECTO

ÍNDICE DOCUMENTO Nº1: MEMORIA

ÍNDICE DOCUMENTO Nº2: PLANOS

ÍNDICE DOCUMENTO Nº3: PLIEGO DE CONDICIONES

ÍNDICE DOCUMENTO Nº4: PRESUPUESTO

ÍNDICE DOCUMENTO Nº5: ANEXO I: DATASHEETS

ÍNDICE DOCUMENTO Nº6: ANEXO II: CÓDIGO DE PROGRAMA

ÍNDICE DOCUMENTO Nº7: ANEXO III: COMPONENTES INTELIGENTES

ÍNDICE DOCUMENTO Nº8: ANEXO IV: MANUAL DEL USUARIO DE LOS APARTADOS EN LA SIMULACIÓN DE UNA ESTACIÓN

ÍNDICE FIGURAS Y TABLAS

MEMORIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº1: MEMORIA

1. INTRODUCCIÓN

1.1 OBJETIVO DEL PROYECTO

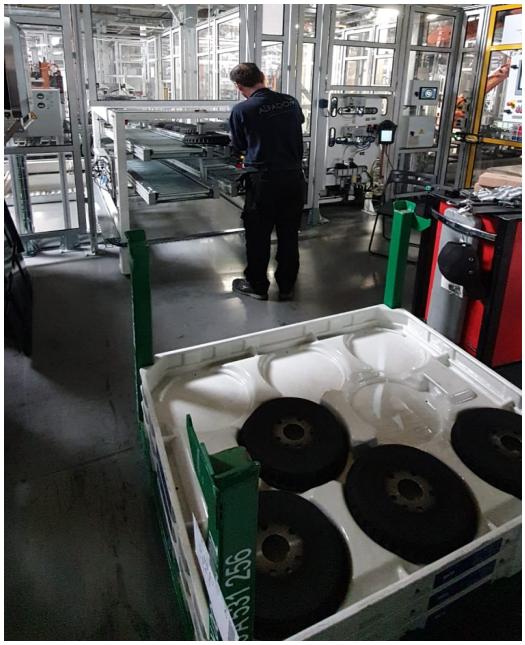


Figura 1 Operario realizando carga manual de discos en la planta de Volkswagen en Braunschweig. Elaboración propia.

En este proyecto se pretende evaluar la viabilidad de una mejora en una línea de producción de discos de frenos en una factoría de Volkswagen en Alemania. Dicha mejora consta de la implementación de un robot industrial para la introducción de piezas en el interior de una línea de producción, ya en funcionamiento, sustituyendo las labores de los operarios en ese puesto.

Mediante la simulación y la programación offline (OLP) podemos analizar, optimizar y comprobar la factibilidad de las mejoras propuestas para el proyecto. Utilizando componentes inteligentes se simula la comunicación y funciones del PLC (Autómata Programable), así como la disposición de la celda para una correcta distribución de los elementos externos. La programación offline del robot facilita y reduce el tiempo del trabajo en planta, la posibilidad de errores en el código, la comprobación de alcance de todas las posiciones del robot, las posiciones y trayectorias óptimas, el control de tiempos de ciclos etc.

Utilizando Inventor como herramienta CAD en 3D se diseñan y modelan las herramientas del robot y los útiles necesarios para el desarrollo del proyecto, incluyendo los respectivos planos y medidas de cada uno de ellos.

Con la automatización del proceso anteriormente explicado se pretende aumentar la producción de piezas, abaratar costes, mejorar la autonomía de producción, y dotar a la línea de mayor seguridad y precisión.

1.2 ANTECEDENTES

Este proyecto se basa en la línea de producción de discos de frenos de la fábrica de Volkswagen en Braunschweig, Alemania.
En 1938 se construyó la primera planta Volkswagen AG en Braunschweig, que sirvió como planta piloto para la posterior planta principal en Wolfsburgo y fue usada para formar a los empleados. Actualmente la fábrica mide 642.000 m^2 y cuenta con aproximadamente 7000 empleados. En ella se producen piezas para el grupo Volkswagen tales como discos de freno, cojinetes, amortiguadores, sistemas de dirección, productos de tecnología plástica, ejes traseros y delanteros y sistemas de batería para futuros modelos híbridos y eléctricos.

La línea de producción producto de estudio cuenta con 24 robots de la marca ABB y aproximadamente 12 operarios por turno para el funcionamiento de la misma. Las principales tareas que desempeñan los operarios son la introducción y gestión de las piezas en la línea, control de calidad, aplicación de accesorios y acabados de plástico en ellas y otras funciones genéricas.

El aumento del volumen de piezas que se fabrican es el objetivo principal que todas las empresas quieren conseguir para ser más competitivas respecto a la competencia. Por ello se cree conveniente aumentar el número de robots de la línea en tareas repetitivas y monótonas como puede ser la carga de piezas, para disminuir tiempos de producción y optimizar la línea.

1.3 MOTIVACIÓN

Después de haber trabajado en un proyecto en esta línea como programador de robots industriales he visto la posibilidad de estudiar la optimización de la misma con la introducción de otro robot industrial.

Gracias a este proyecto consigo afianzar mis conceptos en el campo de la simulación industrial para poder aplicarlos más adelante en mi trabajo actual. En este sector la simulación juega un papel vital tanto para los estudios de viabilidad previos de los proyectos, como para la programación de los robots.
Cada vez más vemos como este sector está creciendo de manera exponencial con la implementación de la Industria 4.0 y los nuevos avances en el mundo de la inteligencia artificial y la robótica. Es por ello de suma importancia para los ingenieros industriales, formarse y especializarse en estos sectores, conociendo los diversos lenguajes de programación, softwares utilizados y nuevos avances que vayan desarrollándose para estar preparados para un futuro no muy lejano.

2. ASPECTOS A CONSIDERAR

2.1 EL ROBOT INDUSTRIAL. APLICACIONES

El robot industrial aparece por primera vez a partir de la tercera revolución industrial a mediados del siglo XX. Fue George Charles Devol, un inventor estadounidense, con la intención de crear una maquina flexible, adaptable al entorno y de fácil manejo, quien patentó el primer manipulador programable en 1948, estableciendo así las bases para los futuros robots industriales.
Existen múltiples definiciones establecidas para un robot industrial pero posiblemente la más utilizada sea la de la Asociación de Industrias Robóticas (RIA) según la cual:
"Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias primas, herramientas, o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas."

En el siguiente enlace se puede ampliar el conocimiento de la historia del robot industrial. https://es.wikipedia.org/wiki/Robot_industrial

2.1.1. Tipos de robots industriales

Actualmente existen diversos tipos de robots industriales cuyas diferenciaciones los hacen esenciales para diferentes tareas. Entre estas características podemos destacar las siguientes:

- Grados de libertad o movimientos
- Capacidad de carga
- Velocidad
- Accesibilidad

ROBOTS CARTESIANOS

Robots con movimientos lineales y perpendiculares con respecto a los ejes cartesianos (x,y,z). Son robots normalmente fáciles de programar y de bajo coste económico que ofrecen soluciones para tareas simples y repetitivas. Control Numérico (CNC), fresado o dibujo son ejemplos de aplicaciones de este tipo de robots.

Figura 2.1 Robot cartesiano

ROBOTS CILÍNDRICOS

Figura 2.2 Robot cilíndrico

Dispone de 3 grados de libertad ya que puede realizar dos movimientos lineales (y,z) y un movimiento rotacional. Este tipo de robots se emplea en operaciones de ensamblaje, vaciado y moldeado de metales, soldadura por puntos y en el manejo de máquinas o herramientas.

ROBOTS PARALELOS

También conocidos como robots de tipo DELTA, disponen de tres grados de libertad con articulaciones prismáticas y rotatorias. Están formados por dos bases paralelas, una móvil y otra fija, unidas mediante cadenas cinemáticas. Su campo de trabajo suele ser limitado pero se caracterizan por altas velocidades de trabajo siendo útiles en aplicaciones de empaquetado o en impresión 3D.

Figura 2.3 Robot paralelo FlexPicker ABB

ROBOTS DE CONFIGURACIÓN MEZCLADA O SCARA

Figura 2.4 Robot SCARA

Los robots de tipo SCARA (Selective Compliant Articulated Robot Arm) tienen cuatro grados de libertad con posicionamiento horizontal. Se diferencian del cartesiano por disponer de un movimiento rotacional al final del eje z que hace posible girar el final del brazo. Esto los hace habituales en procesos de ensamblaje, sellado o manejo de herramientas. Producen un amplio campo de trabajo con gran capacidad de carga y altas velocidades.

ROBOTS ANTROPOMÓRFICOS

Su estructura se asemeja a las distintas articulaciones del brazo de una persona desde el hombro hasta la muñeca, destacando así por su alto número de grados de libertad (6 grados de libertad). Son utilizados en una gran variedad de aplicaciones industriales, especialmente en aquellas de difícil accesibilidad. Son muy comunes en el sector de la automoción para soldadura, pintura, sellado, remachado y manipulación de piezas como puede ser en aplicaciones de paletizado. Dispone de una gran gama y variedad de marcas y modelos.

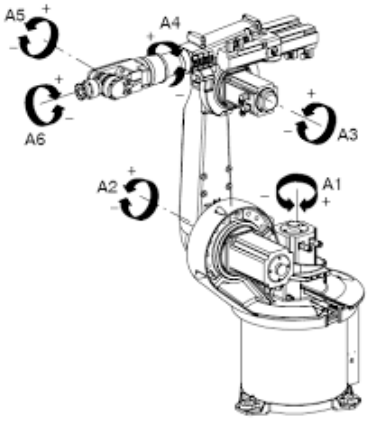


Figura 2.5 Robot Antropomórfico

ROBOTS COLABORATIVOS O COBOTS

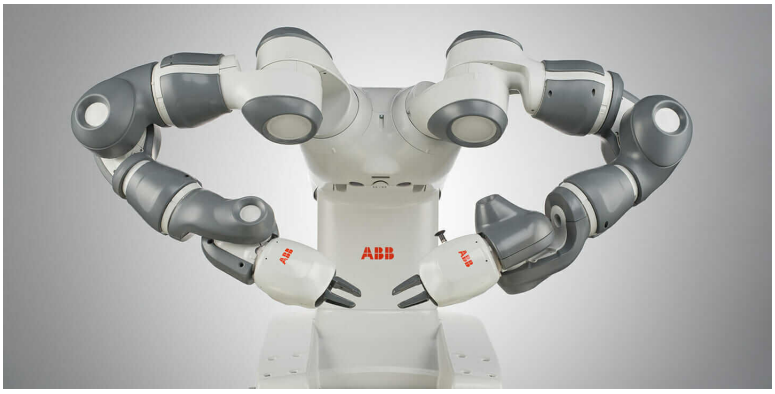


Figura 2.6 Robot colaborativo YuMi ABB

Estos robots han sido creados para trabajar mano a mano con los operarios de una forma colaborativa ayudando a éstos en tareas pesadas o poco seguras. Estos robots ofrecen 6 grados de libertad, gran flexibilidad, fácil integración, seguridad, comodidad de trabajo y bajo consumo energético. Han sido los últimos en desarrollarse y se consideran una excelente solución para el trabajo del futuro.

Existen múltiples páginas webs de las que obtener información adicional acerca de los distintos tipos de robots. Se aconsejan las siguientes, de las cuales se ha obtenido parte de la información proporcionada anteriormente.

https://www.edsrobotics.com/blog/tipos-robots-industriales-usos/
https://www.vld-eng.com/blog/tipos-de-robots-industriales/
https://www.gruasyaparejos.com/gruas-industriales/tipos-de-robots-industriales/#robot_cartesiano

2.1.2. Estudio y elección del robot

De entre todos los tipos anteriormente mencionados, para nuestro caso de estudio usaremos robots de tipo antropomórficos ya que son los más habituales en el sector por sus características.

Vamos a utilizar robots de la marca ABB ya que han sido los elegidos por el cliente para trabajar en la línea de producción. ABB es el proveedor líder en la industrial de la automoción por contar con un rango muy amplio de soluciones para dicho sector por todo el mundo. Dispone de una gran variedad de modelos de robots lo que permite elegir el que más se adapte a las necesidades requeridas según la aplicación, carga y alcance.
Analizando nuestro campo de trabajo, nuestra aplicación es puramente la de manipular piezas, necesitando gran rango de trabajo y no excesivo manejo de carga.

ABB INICIO > OFERTAS > ABB ROBÓTICA > ROBOTS INDUSTRIALES > SELECTOR DE ROBOTS — GLOBAL SITE

Selector de robots

Escoja su robot por:

Aplicaciones	Carga	Alcance
Any	Any	Any

Products

Name	Payload	Reach
IRB 120	3 kg	0.58 m
IRB 1200	5.7 kg	0.7, 0.9 m
IRB 140	6 kg	0.81 m
IRB 1410	5 kg	1.44 m
IRB 14000 YuMi	0.5 kg	0.5 m
IRB 1520ID	4 kg	1.5 m
IRB 1600	6 kg;10 kg	1.2 m;1.45 m
IRB 1600ID	4 kg	1.5 m
IRB 2400	12 kg;20 kg	1.55 m
IRB 260	30 kg	1.52 m
IRB 2600	12 kg;20 kg	1.65 m;1.85 m
IRB 2600ID	8 kg;15 kg	1.85 m;2.00 m
IRB 360	1 kg;8 kg	Ø1.13 (m)
IRB 4400	10 kg;60 kg	1.95 m;2.55 m
IRB 460	110 kg	2.40 m
IRB 4600	20 kg;40 kg;45 kg;60 kg	2.05 m;2.51 m;2.55 m
IRB 52	7 kg	1.2 m, 1.45
IRB 5350	7 kg	1.35 m
IRB 5400	25 kg	3.1 m;15 m
IRB 5500-22 - FlexPainter	13 kg	2.7
IRB 5500-25 - Elevated rail (English)	13 kg	2.7
IRB 580	10 kg	2.2 m;2.6 m
IRB 660	180 kg;250 kg	3.15 m
IRB 6620	150 kg	2.2 m
IRB 6620LX	150 kg	1.9 m
IRB 6640	130 - 235 kg	2.55 - 3.2 m
IRB 6650S	125 kg;200 kg	3.0 m;3.5 m
IRB 6660 for pre machining	205 kg	1.9 m
IRB 6660 for press tending	130 kg	3.10 m
IRB 6660FX (English)	40 kg	3.10 m + 1.40 m
IRB 6700	235 kg	2.65 m
IRB 760	450 kg	3.18 m
IRB 7600	150 - 500 kg	2.3 - 3.5 m
IRB 8700 (English)	550-800 kg	3.5 m, 4.2 m
IRB 910SC SCARA (English)	6 kg	0.45, 0.55, 0.65 m

Figura 3.1 Selector de robots ABB

Usando la opción de filtrado de la que dispone ABB podemos encontrar los modelos que mejor se adaptan a nuestras necesidades como podemos ver a continuación:

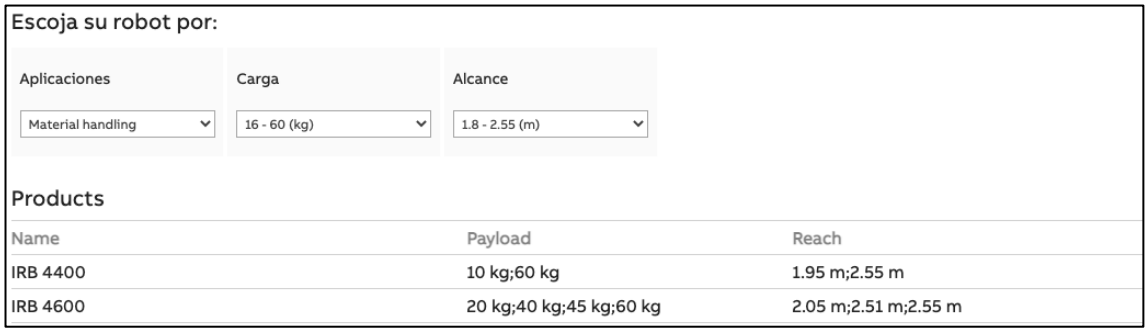


Figura 3.2 Selector robot ABB

Como se puede observar en la figura anterior, la selección del robot se ha hecho utilizando como aplicación la de manejo de material, carga entre 16-60 kg y alcance de hasta 2.55 metros.

El robot finalmente seleccionado es el **IRB 4600.**

Robot Articulado	Carga(kg)	Alcance (m)	Pos. Rep.(mm)	Anclaje	Protección	Ejes	Controlador
 IRB 4600	20/40/ 45/60	2.5/2.55/ 2.05/2.05	0.05 - 0.06	Floor, inverted, tilted, shelf	Std: IP67 Option: Foundry Plus 2, Foundry Prime 2	6	IRC5

Figura 3.3 Selección ABB IRB 4600

En la Bibliografía apartado *[1.1] de Documentación técnica IRB4600 y Controlador IRC5* se encuentra el link de la web oficial de ABB a partir del cual se puede acceder a toda la variedad y características de robots de la que dispone la marca y de donde se han *obtenido Figura 3.1, Figura 3.2 y Figura 3.3.*

2.1.3. Sistema del robot industrial. Componentes y Hardware

El controlador es el conjunto de elementos computacionales que gestionan el comportamiento global del robot. Contiene el hardware y software necesarios para el funcionamiento del robot, así como elementos para las seguridades y de comunicación.

El controlador industrial que vamos a utilizar para nuestro robot es el IRC5. La quinta generación de controladores de la marca ofrece la tecnología de control de movimiento TrueMove y QuickMove que son esenciales para el rendimiento del robot en términos de precisión, rapidez, tiempos de ciclo, programación y sincronización con dispositivos externos. Al igual que anteriormente, se puede consultar la información de los controladores en la página web del fabricante accesible a través del apartado *[1.1]* de la Bibliografía.

Figura 4.1 Controlador ABB IRC5

Figura 4.2 ABB FlexPendant

El Teach Pendant es una unidad manual de programación. Se conecta al controlador mediante el cable rojo, lo cual nos permite poder controlar y programar el robot desde cualquier ubicación. Dispone de pantalla táctil a color y joystick para el manejo del robot. Es una herramienta imprescindible para programar cualquier robot.

2.2 SOFTWARE UTILIZADO

Figura 5.1 Logo ABB RobotStudio

Para la creación de la estación se ha utilizado el software de programación y simulación oficial de la marca ABB, RobotStudio. Es una herramienta muy potente con la que se pueden realizar múltiples tareas tanto online (con robots reales en el centro de producción) como offline (con robots virtuales en un PC). En nuestro campo de aplicación utilizaremos RobotStudio para programar y gestionar nuestro sistema robot de forma indirecta y paralela a la línea sin interrumpir la producción. El código del programa del robot se programa mediante lenguaje RAPID.

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

Figura 5.2 Logo Autodesk Inventor

Se ha usado Inventor, una herramienta de Autodesk para el modelado de sólidos en 3D, para el diseño y creación de las herramientas del robot, útiles y demás elementos necesarios para el proyecto, así como para realizar los planos de dichos elementos.

https://www.autodesk.es/products/inventor/overview?term=1-YEAR&support=null

2.3 ELEMENTOS EXTERNOS

Los elementos externos utilizados en la estación son principalmente sensores y barreras de seguridad. En la simulación aparecen en forma de componentes inteligentes para asemejar su comportamiento al de la realidad. Por petición del cliente, los elementos externos utilizados deben ser de los proveedores Wenglor y SICK, referenciados en la Bibliografía apartados *[2.1] Documentación técnica elementos externos de SICK y [2.2] Documentación técnica elementos externos de WENGLOR GmbH.*

C4P-SA13531A00, C4P-EA13531D00

Figura 6.1 Barrera de seguridad SICK C4P-SA13531A00

Estas barreras de luz de seguridad se utilizan para proteger el acceso a puntos o áreas peligrosas. Existe una gran variedad dependiendo de la altura de protección, rango y otras propiedades.
En nuestra simulación las utilizamos en el soporte para los racks tanto en la entrada como por la parte posterior. De esta manera si se corta la barrera estando el robot dentro de la zona, la estación se parará por seguridad.

OCP242X0135

Figura 6.2 Sensor Wenglor OCP242X0135

Sensor tipo optoelectrónico para detección a distancia sin necesidad de contacto. Además sirven para medir distancias con gran exactitud, reconocer colores y luminiscencia. Disponen de gran variedad de productos según sus características. Regulando el rango de lectura se pueden detectar objetos a distancias concretas. Es uno de los sensores más utilizados en la línea de producción.

VT12T-2P410

Figura 6.3 Sensor inductivo SICK VT12T-2P410

Sensores inductivos de proximidad utilizados para detección de superficies metálicas, normalmente próximas al contacto. Son sensores con bastante robustez y fiabilidad y bastante utilizados en cualquier industria. Los distintos modelos varían según la métrica, rango de detección, tipo de conexión etc.

LMS4000 2D LiDAR SENSOR

Figura 6.4 Escáner SICK LMS4000 2D

Es el escáner utilizado para detectar las piezas en los racks. Se utiliza para mediciones precisas de objetos en movimiento independientemente de la forma, color o calidad de la superficie.

La programación del escáner está efectuada por una empresa externa ya que el código es específico y complejo. Es necesario para recibir datos concretos de las piezas, altura de los blisters, centro del rack, centro del blíster etc.

Nota: Se puede acceder de forma directa a las webs de los fabricantes, a través de la cual se han obtenido tanto las *Figuras 6.1, 6.2, 6.3 y 6.4* así como los documentos técnicos adjuntos en el Documento Nº5 , mediante los siguientes enlaces:

https://www.sick.com/es/en/
https://www.wenglor.com/es/

Referenciados también en los apartados *[2.1] y [2.2]* de la Bibliografía.

3. MODELADO DE GEOMETRÍAS EN CAD

3.1 MODELADO DE PIEZAS EN 3D MEDIANTE INVENTOR

Para el correcto desarrollo del proyecto y estudio de la simulación es necesario disponer previamente de todos los elementos con los que se va a trabajar. Algunos elementos son de nuevo diseño, específicos para el proyecto a plantear y otros existen ya en producción actualmente.

Lo primero que debemos hacer es crear un proyecto para almacenar y asociar todas las geometrías correctamente. Haciendo clic en la pestaña *Proyecto>Nuevo proyecto de Vault.*

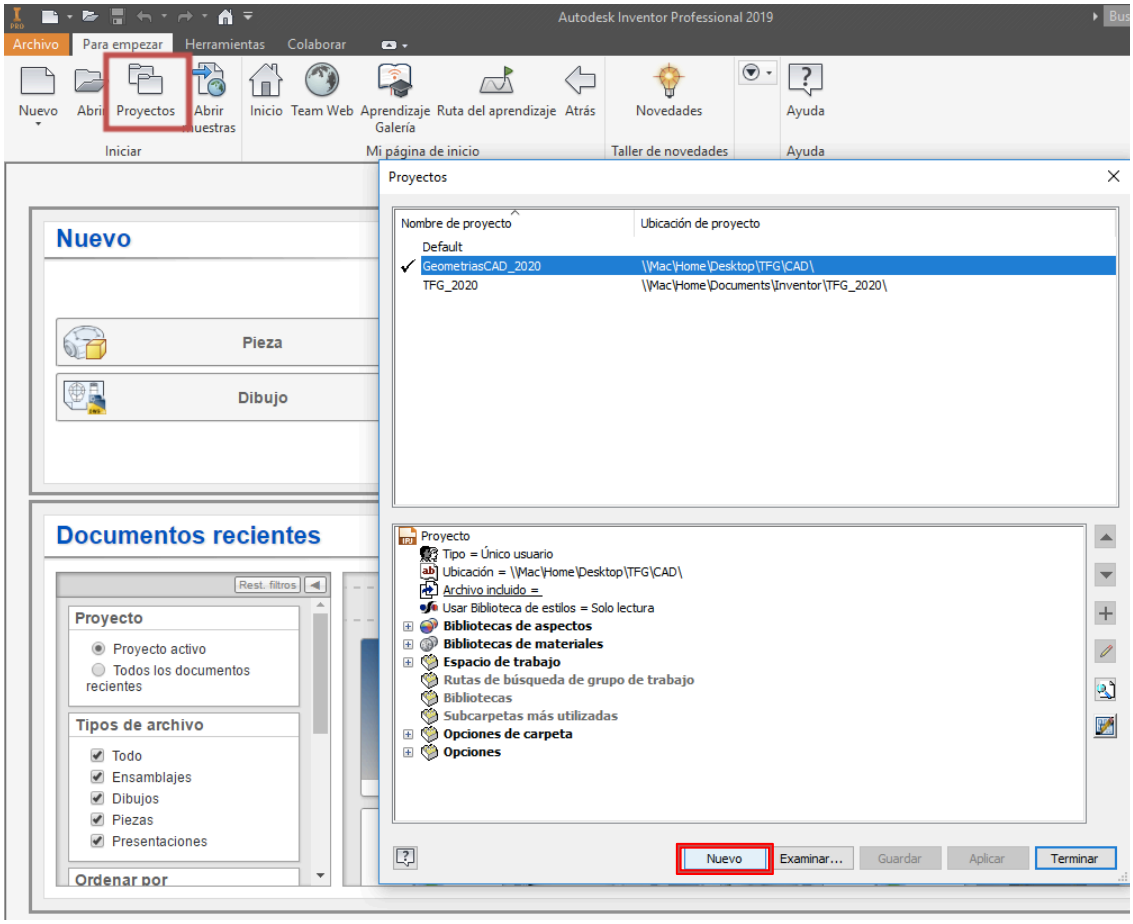


Figura 7.1 Crear nuevo proyecto en Inventor

Posteriormente seleccionamos *Pieza>Normal.ipt* y establecemos correctamente el sistema de coordenadas para iniciar el boceto 2D adecuadamente.

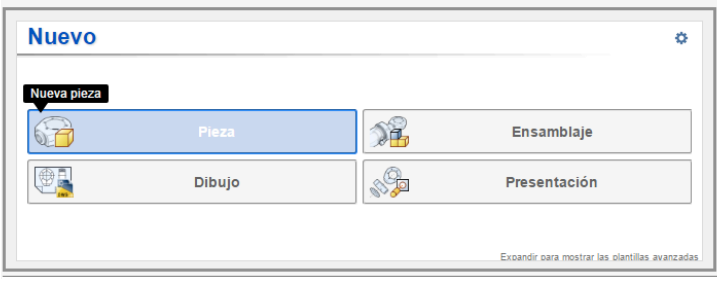


Figura 7.2 Crear nueva pieza Inventor

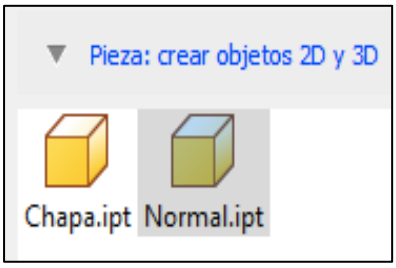


Figura 7.3 Tipos de piezas 2D Inventor

Se puede encontrar más información sobre el proceso de modelado de cada pieza en la "*Guía del Usuario"* al final del proyecto.

ÚTIL DE DESCARGA (FIXTURE)

Útil formado por dos raíles con cilindros rotatorios (rodillos). Las piezas de cada modelo son depositadas por el robot en uno de los extremos y caen por gravedad hasta el otro extremo donde son introducidas en la línea de producción. El útil tiene unas dimensiones de 2270x800x500mm con un 4% de inclinación desde la parte superior.

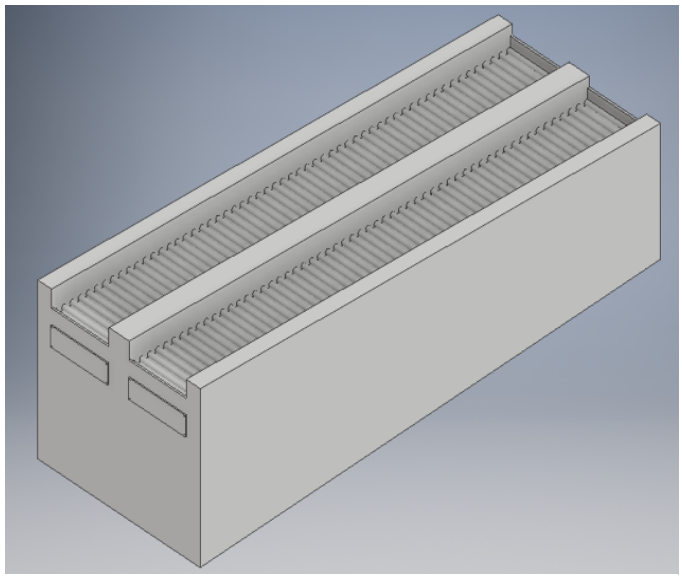


Figura 8.1 Útil de descarga modelado en Inventor

Se ha diseñado siguiendo las dimensiones y geometría del útil existente y en uso actualmente, por petición del cliente. Sirve como buffer para almacenar hasta 8 discos por modelo, lo cual favorece al tiempo de ciclo de la estación en caso de problemas o parones durante la producción.

En la *Figura 8.2* se puede observar el modelo actual el cual se utiliza para 4 variantes distintas de los dos modelos de disco producidos.

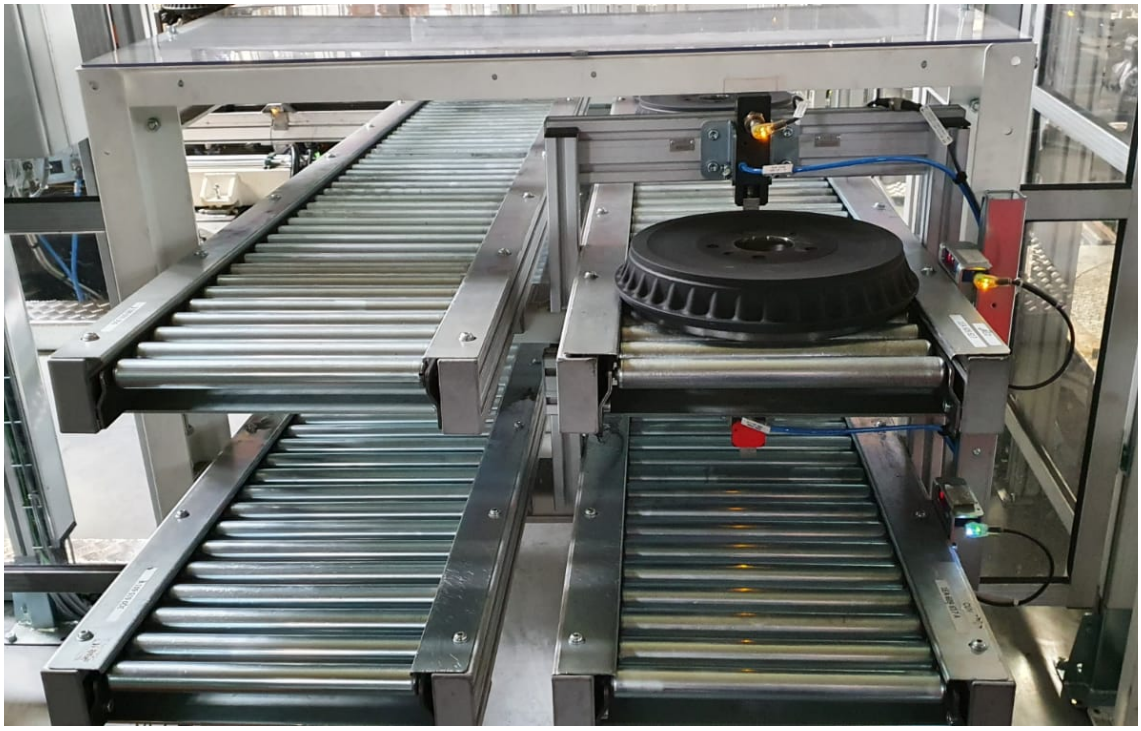


Figura 8.2 Útil de descarga en la planta de VW Braunschweig. Elaboración propia.

SOPORTE PARA RACKS

Ahora que es el robot el que coge las piezas es necesario disponer de un espacio cerrado ya que los racks quedan expuestos a los operarios siendo un problema de seguridad. Por ello se utilizan unas barreras fotoeléctricas (SICK) que detienen el automático de la estación cuando algo las rebasa. De esta forma desde fuera no se podría acceder al interior mientras el robot esta en funcionamiento tal y como queda indicado en la norma ISO 13857 para distancias de seguridad. Las dimensiones de este soporte para dos racks son de 3400x1210x640mm.

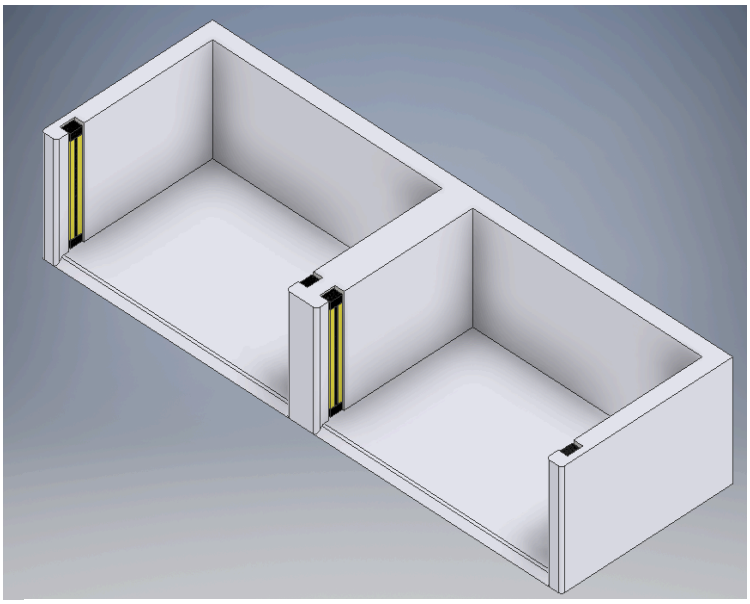


Figura 8.3 Soporte para racks modelado en Inventor

RACK

Los racks se utilizan para almacenar y distribuir los blisters de piezas. Los racks se almacenan uno encima del otro para disminuir el espacio ocupado y se transportan mediante carretillas elevadoras ya que son bastante pesados. Cada rack tiene unas dimensiones de 1500x920x590mm.

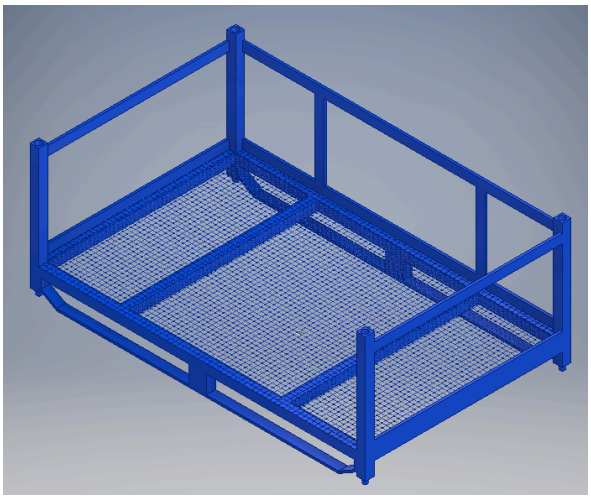


Figura 8.4 Rack modelado en Inventor

Figura 8.5 Pila de racks en planta de VW Braunschweig. Elaboración propia.

En cada rack se pueden almacenar 8 blisters.

Se ha modelado para la simulación, aunque se están utilizando de manera estandarizada por toda la planta para las distintas líneas como se observa en la *Figura 8.5.*

BLÍSTER

Al igual que los racks, los blisters se utilizan para transportar y almacenar las piezas. También se utilizan actualmente por lo que únicamente se ha diseñado para poder realizar la simulación correctamente. En la *Figura 8.7* podemos observar como cada blíster puede albergar 8 piezas.

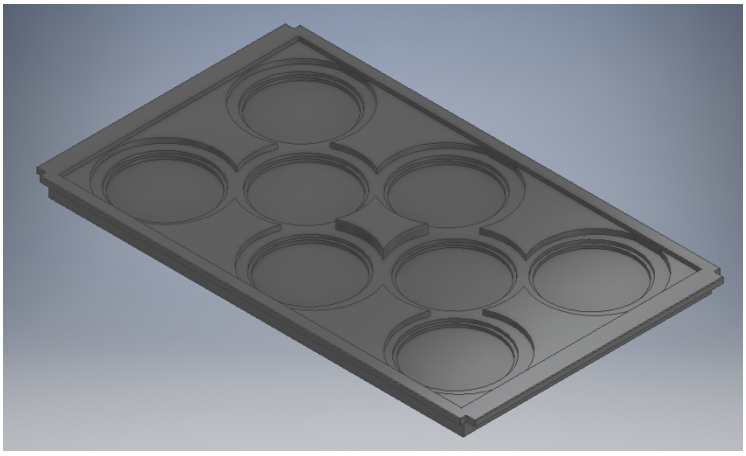


Figura 8.6 Blíster modelado en Inventor

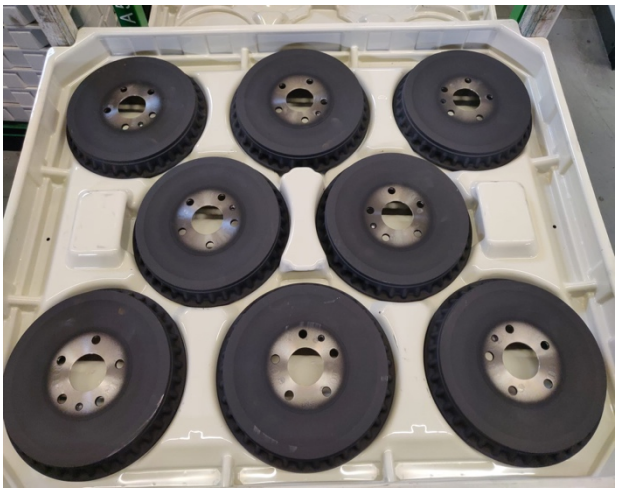


Figura 8.7 Blíster con piezas en VW Braunschweig. Elaboración propia.

Una vez todas las piezas de un blíster son descargadas, el robot lo depositará en un rack de descarga mediante una herramienta de succión.

Las dimensiones de cada blíster son de 1500x920x70mm.

HERRAMIENTA DE SUCCIÓN

La herramienta de succión queda anclada al robot mediante la herramienta de manipulación por la parte inferior. Mediante unos conectores cilíndricos el robot entrega energía a la herramienta para que pueda succionar el blíster. La conexión entre ambos se hace mediante direcciones IP y sus correspondientes módulos de FESTO para vincular las entradas y salidas.

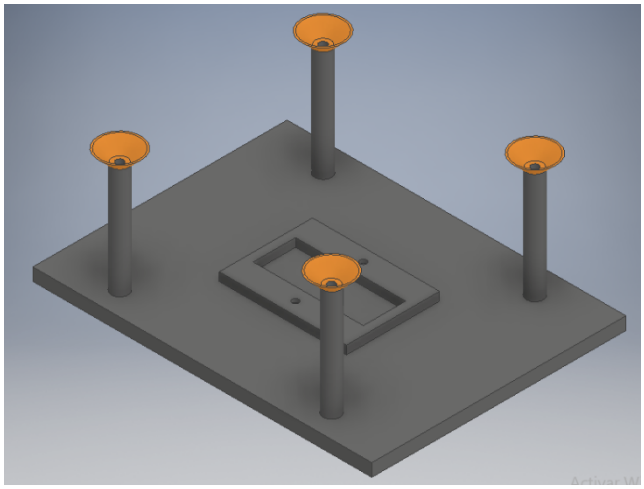


Figura 8.8 Ventosa modelada en Inventor

Con las 4 ventosas naranjas la herramienta puede succionar el blíster y transportarlo al rack de descarga correspondiente. Mediante el hueco de la parte central la herramienta se mantiene en su soporte.

Las dimensiones de la herramienta son de 800x600x320mm.

Es importante conocer los datos de la herramienta para ajustar movimientos de supervisión calculando el centro de gravedad y momentos de inercia con el peso.

SOPORTE DE LA HERRAMIENTA DE SUCCIÓN

Este soporte va anclado al suelo y sirve para depositar la herramienta de succión cuando no se está utilizando. Por medidas de seguridad, un sensor detecta la presencia de la herramienta en el soporte. Tiene unas dimensiones de 500x500x1060mm.

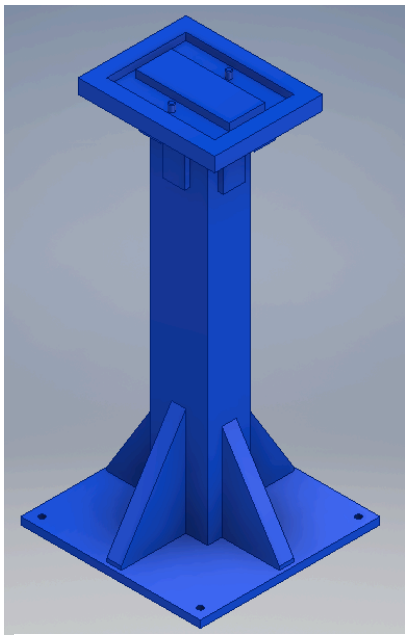


Figura 8.9 Soporte de ventosa modelado en Inventor

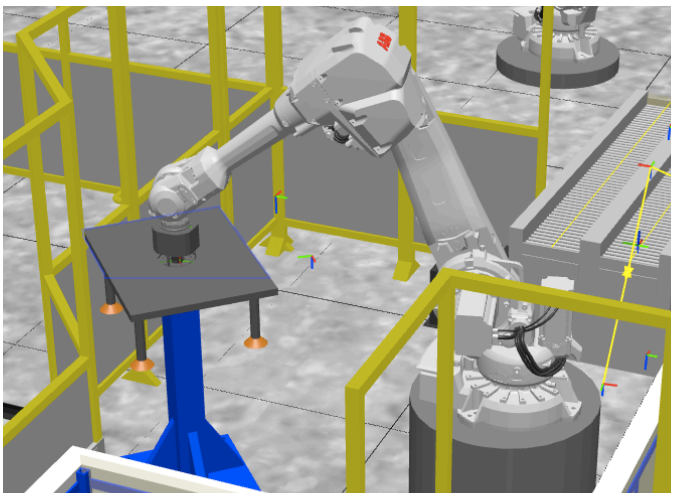


Figura 8.10 Captura de pantalla del robot cogiendo la ventosa en el soporte en RobotStudio

MANIPULADOR

Es la herramienta principal del robot. Está formada por el ensamblaje de las bridas y la base de la herramienta. Con ella se cogen las piezas y la herramienta de succión. Tiene tres bridas en la parte posterior que se expanden o contraen para fijar los objetos a manipular. El cilindro superior sirve como referencia para coger el disco y opcionalmente se podría utilizar un sensor analógico que se activa cuando se ejerce mucha presión sobre dicho cilindro, por ejemplo, cuando hay una colisión. Diseño propio basado en distintas herramientas de la línea.

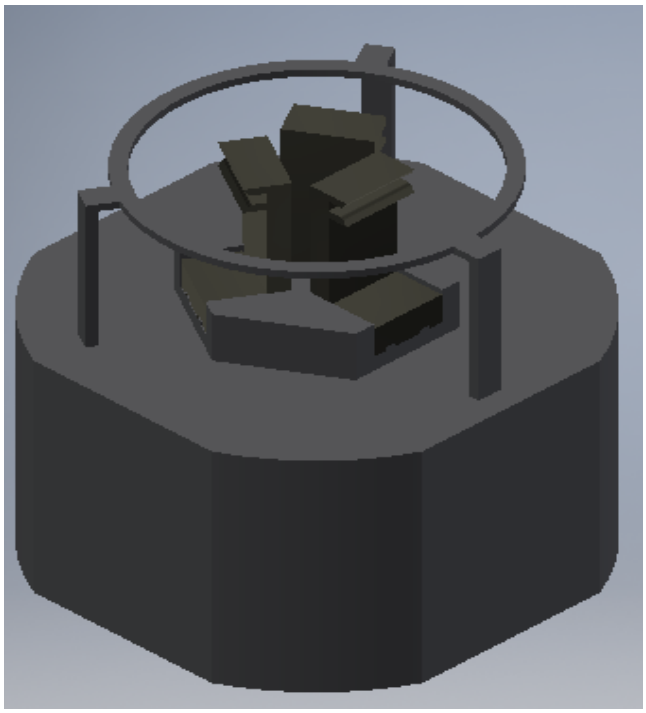


Figura 8.11 Ensamblaje Manipulador modelado en Inventor

Por la parte inferior se ancla de forma fija a la brida del robot (robot flange) que se encuentra en su extremo. La conexión se realiza mediante dirección IP y con el correcto mapeo de las entradas y salidas, correspondientes a la conexión eléctrica en el módulo de FESTO mediante un EPlan (Plano Eléctrico). Se pueden activar o desactivar las bridas desde el Teach del robot.

Las dimensiones de la herramienta son de 171x171x170mm, con los bordes de la base redondeados. Las dimensiones de la brida son de 28x50x30mm.

DISCOS DE FRENO

En la línea de producción se producen 4 modelos de piezas correspondientes a dos plataformas diferentes. Por simplicidad en la simulación se utiliza un modelo de cada una de las plataformas.

En la plataforma **MEB** (Plataforma de Accionamiento Eléctrico Modular) fue diseñada específicamente para coches eléctricos por el Grupo Volkswagen. Los modelos CUV (Vehículo Utilitario Compacto) y NEO forman parte de esta plataforma.

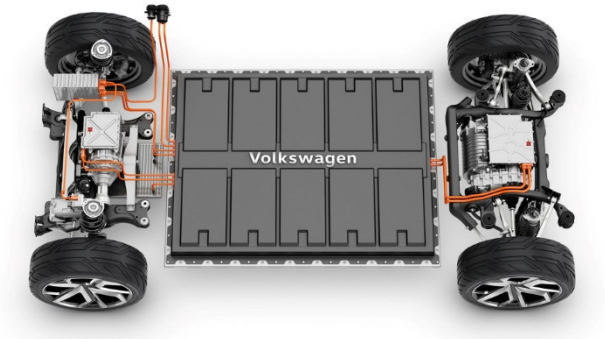


Figura 8.12 Modelo de la plataforma VW MEB

Sorprendentemente los frenos de los coches eléctricos de esta plataforma son de tambor. Tienen un coste inferior de producción, bajo mantenimiento, diseño simple y vida útil más larga.

Figura 8.13 Tambor de freno MEB modelado en Inventor

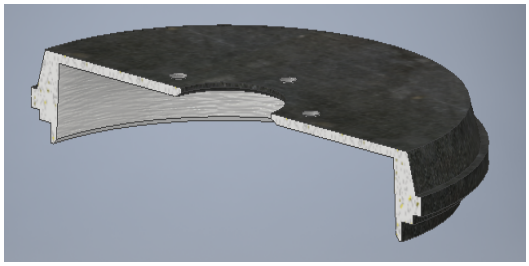


Figura 8.14 Sección tambor de freno MEB modelado en Inventor

Ambos modelos tienen las mismas dimensiones, siendo estas de 280mm de diámetro y 36,5mm de altura. El cilindro interior es de 72mm de diámetro.

La plataforma **MQB** (Plataforma de Construcción Modular Transversal) forma parte de un gran rango de los vehículos del grupo. Para esta plataforma los dos tipos de discos de freno que se producen se distinguen por las pulgadas del disco (16” y 17”) siendo el más grande de ambos ventilado, como se aprecia en las figuras siguientes.

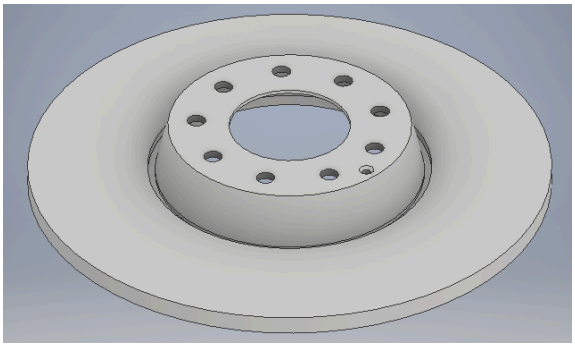


Figura 8.15 Disco de freno MQB 16" modelado en Inventor

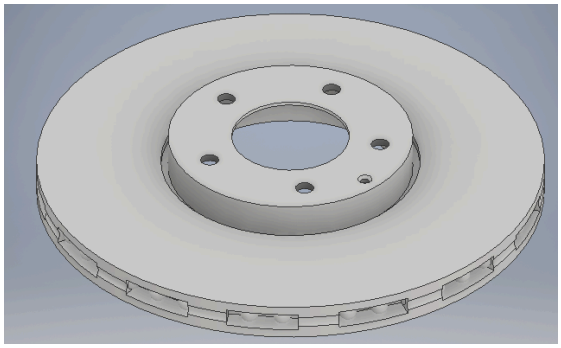


Figura 8.16 Disco de freno MQB 17" modelado en Inventor

Figura 8.17 Disco de freno MQB 17" montado en un coche

Los conocimientos adquiridos en la asignatura de Ing. Gráfica de la carrera, [5.1] *Ferrán Naya (2018)*, así como los apuntes de la misma, han servido de base para la realización del modelado de las geometrías y los respectivos planos. Apartado *[5.2] Inventor – Generador de Planos. (2018). Nuria Aleixos.* de la Bibliografía.

3.2 ESTUDIO DE DISTRIBUCIÓN DE LA CELDA

Como hemos comentado anteriormente y teniendo en cuenta las limitaciones exigidas por el cliente (*Figura 9.1*), se ha realizado un estudio para distintas distribuciones de la celda con el objetivo de encontrar la más adecuada. Lo que se ha pretendido es encontrar un *layout* en el que se redujeran tiempos de ciclo del robot, existiera una armonía de movimientos durante la producción y fuera apropiada para realizar labores de mantenimiento dentro de ella.

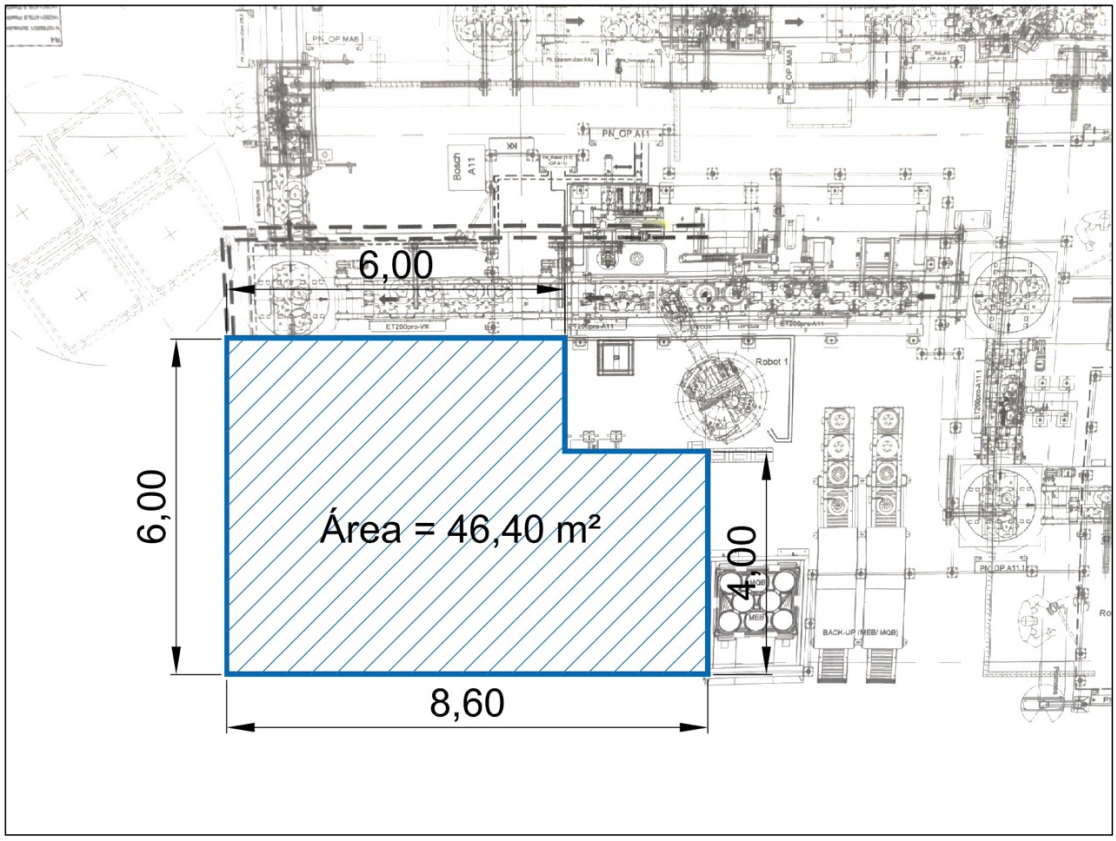


Figura 9.1 Plano con medidas de planta en VW Braunschweig

Según los planos de la estación proporcionados, se establece el área de trabajo necesaria para la elección de la celda nueva. Se han estudiado varias distribuciones, cada una de ellas con sus respectivas ventajas e inconvenientes que se explican a continuación.

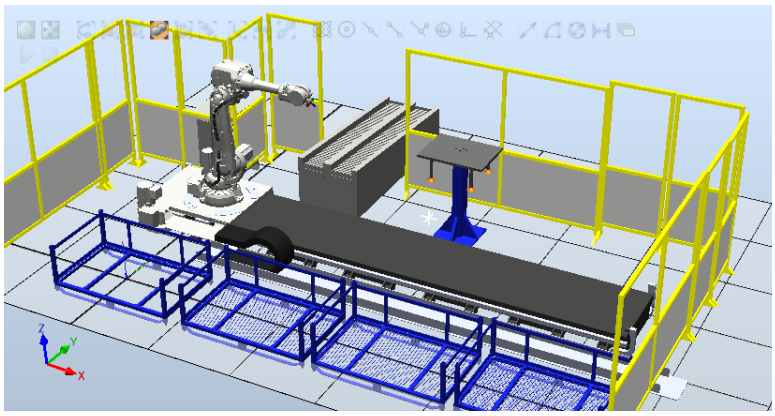


Figura 9.2 Distribución celda RobotStudio 1

Distribución 1: Como podemos observar en esta distribución disponemos de 4 racks dispuestos en paralelo, lo que facilitaría el manejo de éstos durante la producción. La idea de esta celda es disponer de mucha versatilidad a la hora de trabajar, pudiendo introducir en cualquier rack el modelo necesario o incluso dejar cualquiera vacío como rack de descarga. El escáner sería el encargado de establecer la funcionalidad de cada rack en ese momento. Sería necesario en este caso disponer de un track (carril sobre el que se desplaza el robot) para poder alcanzar todas las posiciones de trabajo sin problemas. Esto supone una distancia longitudinal excesiva y un aumento en el precio de la celda y el tiempo de ciclo de la misma. Este aumento de distancia supone a su vez mayor espacio interior y mejor acceso a la celda, pudiendo ponerse dos puertas de entrada a la celda para facilitar labores de mantenimiento.

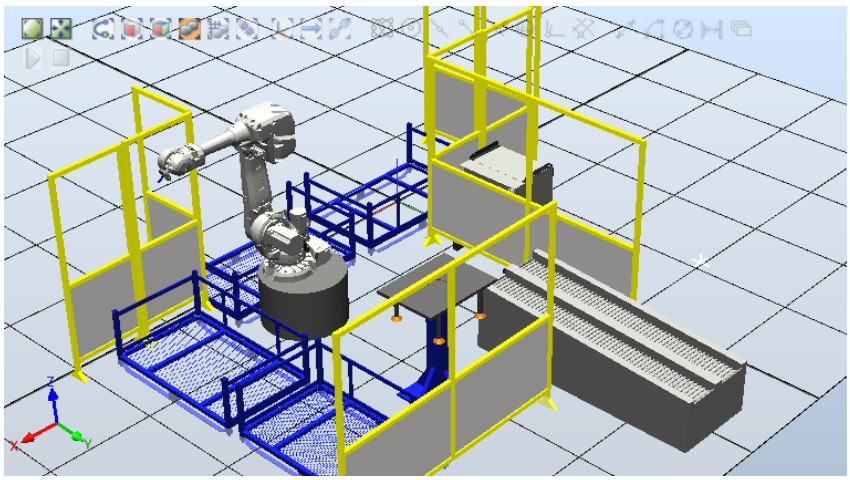


Figura 9.3 Distribución celda RobotStudio 2

Distribución 2: Siguiendo con la dinámica de los 4 racks, existe la posibilidad de esta otra distribución con los racks en forma de L, en la que se podrían disponer los dos modelos de pieza (MEB y MQB) separados de forma perpendicular. Esto supone una celda más compacta, sin la necesidad de un track, ya que el robot podría alcanzar todas las posiciones anclado a un soporte en altura, reduciendo el coste de la misma. Es posible que para este caso el modelo de robot tenga que ser distinto al seleccionado previamente para aumentar su alcance y llegar a ambos racks situados en los extremos. El acceso a la celda sería más limitado por la falta de

espacio y la dificultad para realizar mantenimiento. El trabajo de carga y descarga de racks sería más costoso debido a la separación de ellos.

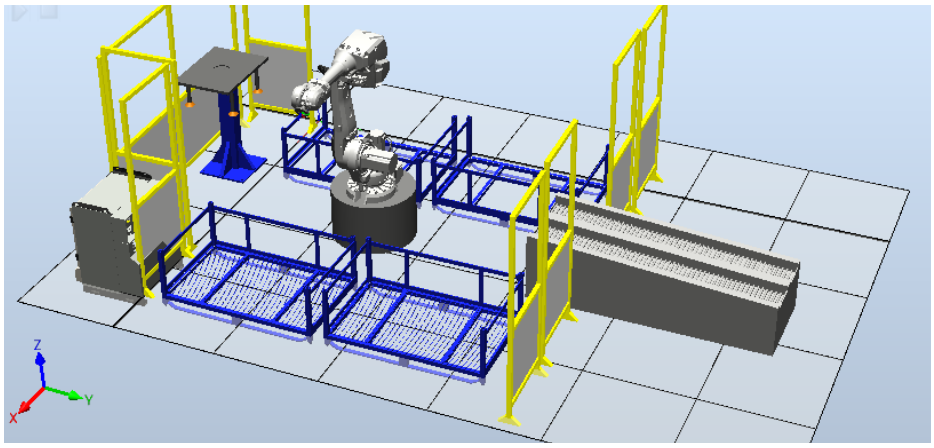


Figura 9.4 Distribución celda RobotStudio 3

Distribución 3: La mayor ventaja de esta distribución con respecto a las demás es la reducción del tiempo de ciclo del robot ya que los racks de ambos modelos se encuentran próximo al útil de descarga. El espacio es reducido y el acceso al robot es relativamente bueno por la parte izquierda de la imagen, para facilitar labores de mantenimiento. En este caso el manejo de los racks resulta más complejo ya que ambos modelos se encuentran en lados opuestos de la celda. Habría que modificar el útil de descarga para adaptarlo al área de trabajo establecida.

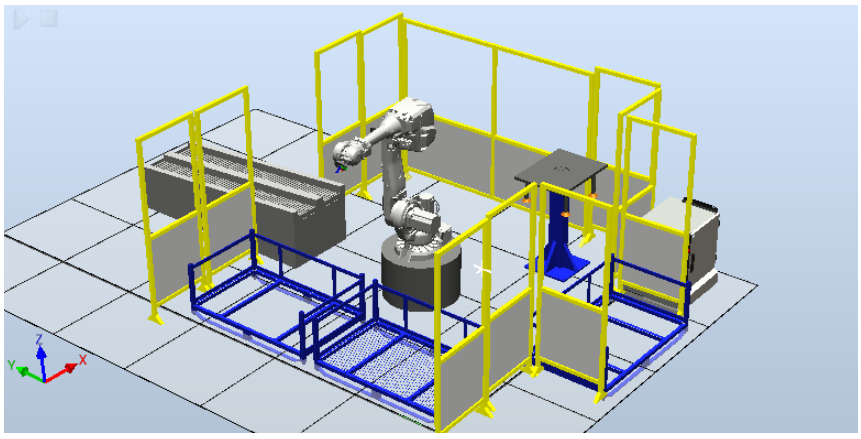


Figura 9.5 Distribución celda RobotStudio 4

Distribución 4: Esta distribución es relativamente distinta a las demás ya que existe un rack para cada modelo pero únicamente uno de descarga que es común a ambos. Esto reduce aún más el espacio de la celda y facilita el trabajo del operario para el manejo de racks, aunque éstos se encuentran dispuestos en perpendicular. Sigue habiendo espacio en el interior para realizar trabajos de mantenimiento con una entrada por la parte posterior de la imagen. El flujo de racks será mayor, lo que podría aumentar el tiempo de ciclo, aunque se podría compensar introduciendo en ambos racks el mismo modelo de piezas a producir.

Una vez establecidas las diferentes opciones para el layout de la estación y antes de su elección, es necesario un estudio en mayor profundidad para comprobar su viabilidad teniendo en cuenta las limitaciones y condiciones. Es importante que el diseño seleccionado tenga total accesibilidad por parte del robot a todas las posiciones, se encuentre comprendido dentro del espacio disponible para la estación, sea factible la programación para el tiempo de ciclo establecido y demás consideraciones.

Mediante un simple boceto en AutoCAD, que se observa en la *Figura 9.6,* se muestran las dimensiones de la celda basada en la distribución 4 (*Figura 9.5*), elegida para el diseño y simulación de la estación.

Se puede observar como la manipulación de los racks para la carga y descarga de los mismos, queda por las partes accesibles de la línea y que el útil de descarga se sitúa de forma perpendicular al utilizado para la carga manual, conectado con la otra celda para que le robot pueda coger discos de ambos útiles. Las dimensiones de la celda se adaptan a las establecidas por el cliente.

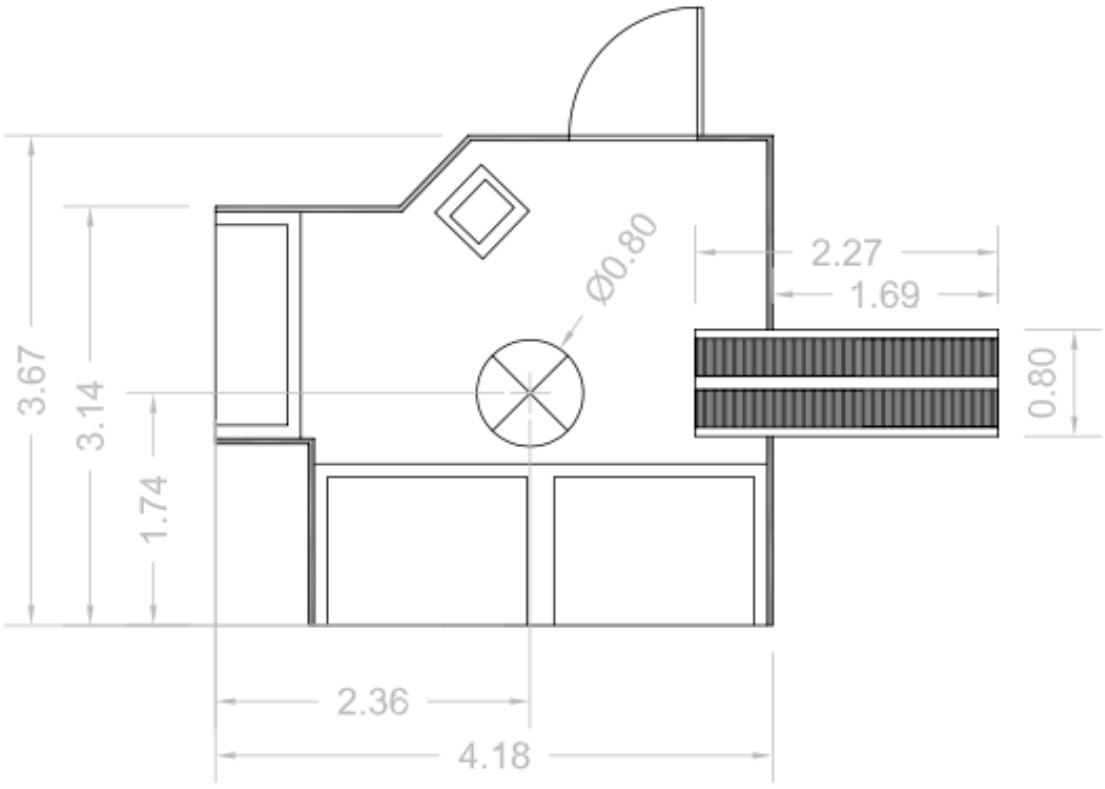


Figura 9.6 Plano de la Distribución 4 seleccionada en AutoCAD

4. IMPLEMENTACIÓN DE LA ESTACIÓN EN ROBOTSTUDIO

4.1 CREACIÓN DE LA ESTACIÓN

4.1.1. Creación de una estación vacía

Antes de empezar deberemos crear una estación vacía sobre la que poder trabajar. Para ello hacemos clic en Archivo>Nuevo>Estación vacía>Crear, tal y como se muestra en la *figura 10.1*

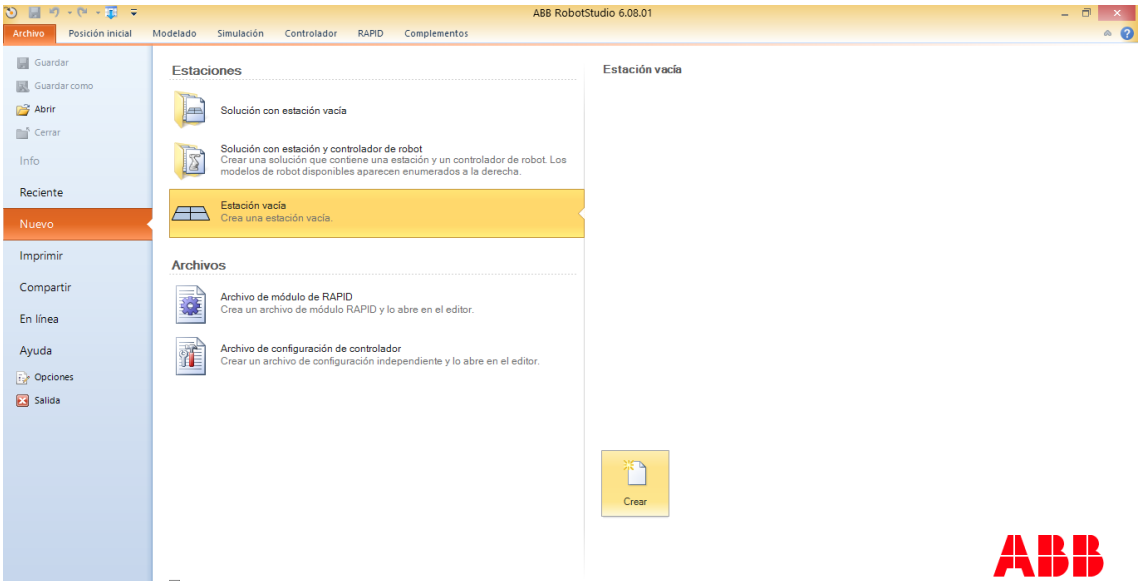


Figura 10.1 Creación de una estación vacía en RobotStudio

Una vez creada nos aparecerá la vista principal de RobotStudio en la que se muestran las distintas ventanas, pestañas y comandos disponibles.

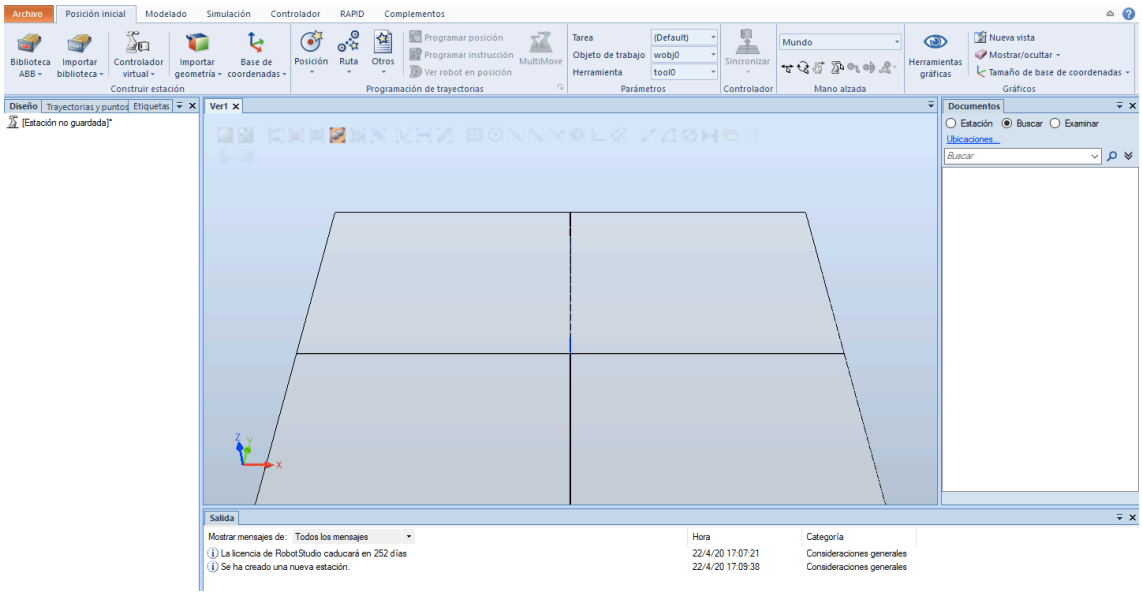


Figura 10.2 Vista principal en RobotStudio

En la parte central encontramos una cuadricula, que nos hace de base de la estación con su respectivo sistema de coordenadas, y algunos comandos de modelado. En la parte inferior encontramos las distintas ventanas de seguimiento donde se muestran los mensajes de salida. En la parte izquierda tenemos las ventanas de diseño, trayectorias y etiquetas. Desde esta sección podemos visualizar todos los elementos de la estación y las trayectorias del robot. En la parte de la derecha se encuentran los documentos, aunque esta sección se suele usar para el control de la simulación de la estación. En la parte superior se encuentran desplegados todos los comandos de las distintas ventanas del programa. De ellas destacamos las de simulación, controlador y RAPID ya que son las más utilizadas para nuestro caso de estudio.

4.1.2. **Diseño de geometrías de la estación**

A partir del plano realizado con anterioridad en AutoCAD de la distribución de la estación con las dimensiones de las distintas geometrías de la misma, procedemos a diseñarla en RobotStudio.

Empezamos creando la estructura de la celda, insertando el vallado que la rodea. Las vallas que vamos a utilizar las podemos encontrar en la biblioteca proporcionada por ABB desde Posición Inicial>Importar biblioteca>Equipamiento>Otros. Entre los tres tipos de vallado que encontramos elegiremos el más conveniente para cumplir con las dimensiones normalizadas establecidas. Para las dimensiones del vallado tenemos valla individual de 740mm, valla doble de 2500 y valla con puerta de 1500mm.

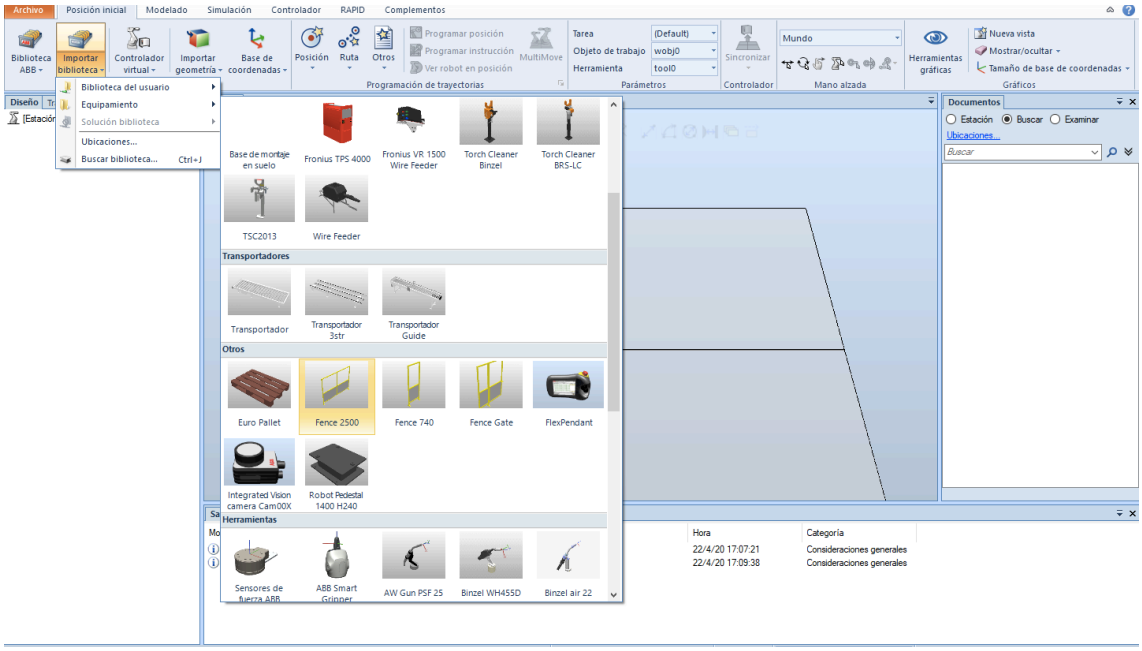


Figura 10.3 Biblioteca de geometrías en RobotStudio

Es de gran utilidad la comunidad GrabCAD para encontrar geometrías que poder añadir a la estación, referenciada en la Bibliografía apartado *[6] Geometrías.*

Para colocar las distintas geometrías podemos definir su posición haciendo clic con el botón derecho sobre el componente importado>Posición>Fijar posición. Introducimos la posición y orientación de la geometría según el sistema de referencia indicado. Utilizaremos sistema de referencia Mundo, siendo éste el correspondiente a la estación, tal y como se muestra en la pantalla central. También podemos colocar los elementos a mano alzada con los comandos pertinentes.

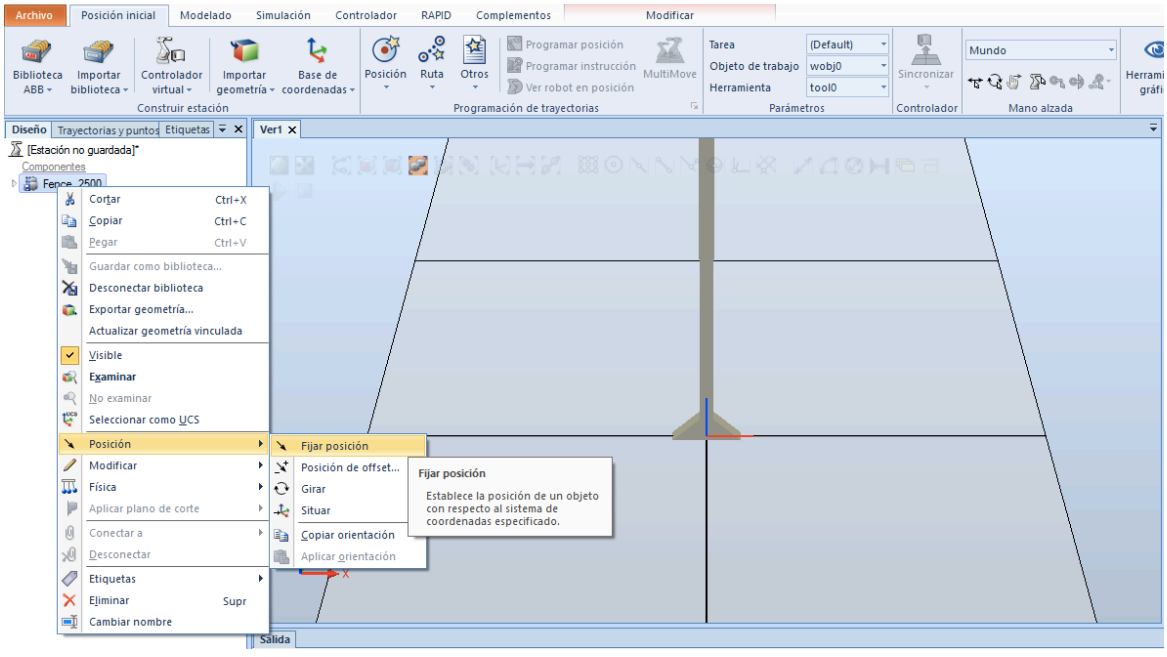


Figura 10.4 Fijar posiciones de geometrías en RobotStudio

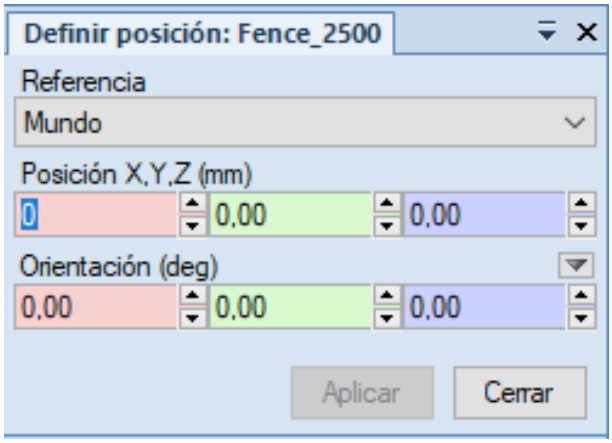


Figura 10.5 Definir posición en RobotStudio

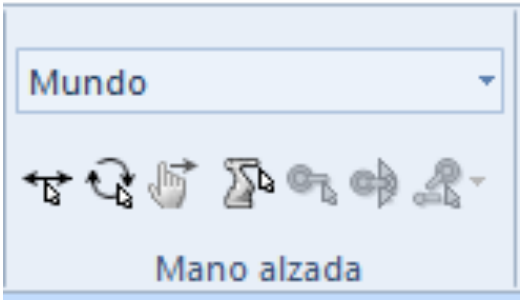


Figura 10.6 Movimientos a mano alzada en RobotStudio

Ahora podemos añadir las herramientas y útiles modelados en Inventor de la misma forma. En este caso deberemos importar las geometrías con el correspondiente formato (.ipt). Para ello hacemos clic sobre Posición inicial>Importar geometría>Buscar geometría y seleccionamos el archivo a importar. De igual manera podemos también fijar las posiciones.

Antes de colocar el robot, debemos crear el soporte para el mismo a través de la ventana de Modelado>Sólido>Cilindro. Elegimos las dimensiones, la posición donde vamos a colocarlo y el sistema de referencia para crearlo. El robot irá colocado encima del soporte, que tiene un diámetro de 800mm y una altura de 500mm. El soporte es necesario para lograr un correcto alcance del robot a todas las posiciones.

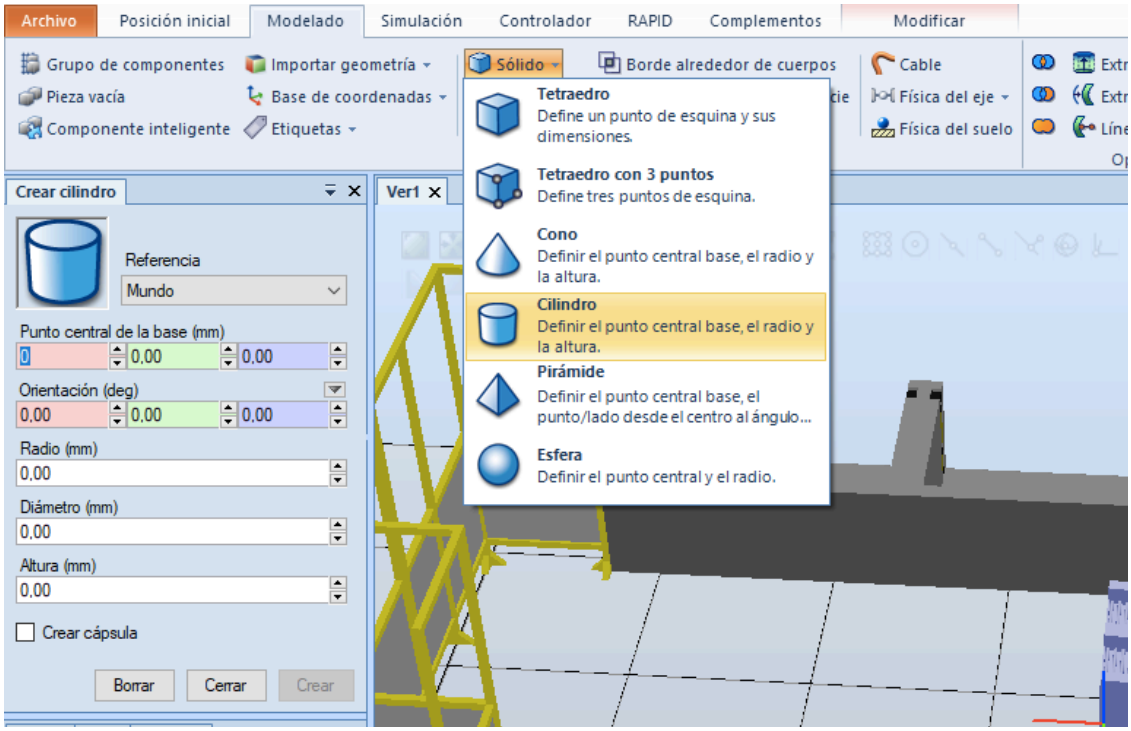


Figura 10.7 Crear cilindro en RobotStudio

Mediante un proceso de iteración se calcula que 500mm es la altura óptima que asegura que el robot llega a todos los puntos necesarios, minimizando la cantidad de material necesario para su fabricación.

El modelo de robot que hemos seleccionado podemos introducirlo desde la biblioteca ABB en la que podemos encontrar los distintos modelos de robots, posicionadores y tracks necesarios. Posición Inicial>Biblioteca ABB>Robots>IRB4600. Una vez seleccionado podemos elegir la versión que vamos a utilizar según su capacidad de carga y alcance. En nuestro caso hemos elegido IRB 4600-60/2.05, siendo 60 kg la carga máxima que puede soportar en el extremo de su brazo y 2.05m el alcance máximo.

Figura 10.8 Biblioteca de robots en RobotStudio

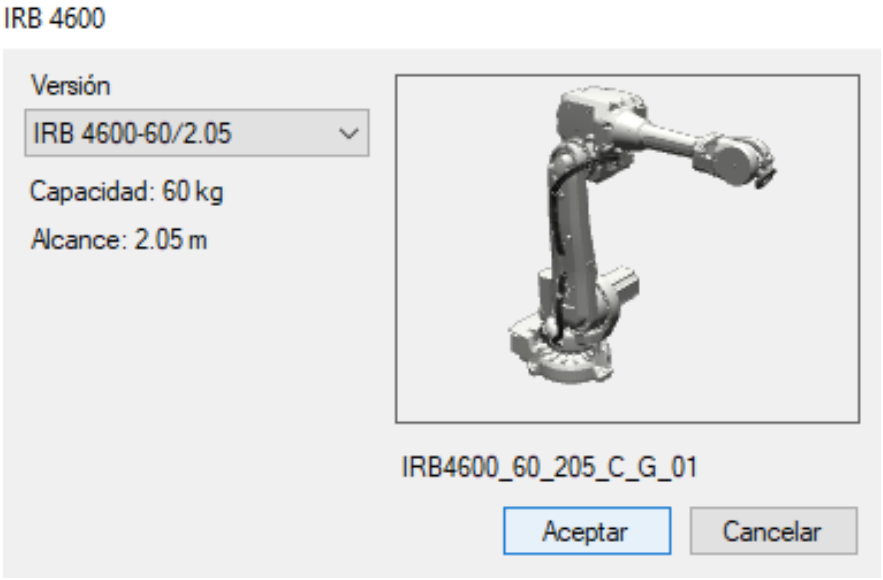


Figura 10.9 Selección del robot IRB4600 en RobotStudio

Por último y una vez importado, la herramienta principal del robot se ancla a la brida del robot (flanje). Para ello simplemente arrastramos la herramienta encima del robot y aceptamos la vinculación de ambas de forma automática.

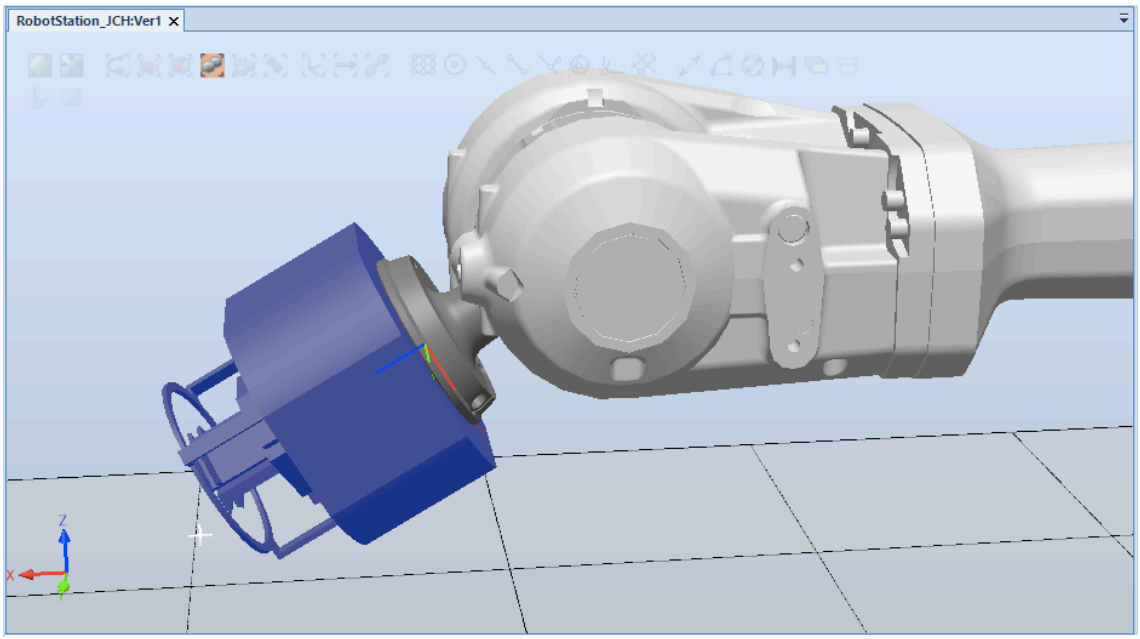


Figura 10.10 Herramienta anclada al flanje del robot en RobotStudio

Algunos elementos se han importado de bibliotecas externas como por ejemplo, la carretilla elevadora, los operarios y los elementos estéticos para la estación.

Una vez terminado de colocar todos los elementos de la celda, necesitamos crear un grupo de componentes formados por el rack, los blisters y las piezas correspondientes a cada grupo de carga. De esta manera se facilita el trabajo a la hora de realizar la simulación con los componentes inteligentes. Cuando tenemos todos los componentes ensamblados correctamente, los arrastramos dentro del grupo creado desde Modelado>Grupo de Componentes. Ahora todo el ensamblaje se puede mover de forma conjunta.

Figura 10.11 Crear un grupo de componentes en RobotStudio

Las geometrías importadas a la estación se pueden encontrar en *[5.1]* y *[5.2]* del apartado **Geometrías** de la Bibliografía.

4.2 CONTROLADOR VIRTUAL

Es necesario instalar un sistema de software para controlar el robot desde un ordenador. Este sistema emula a un controlador IRC5 en la vida real mediante el software RobotWare (RobotWare 6.08) que contiene la configuración y funciones necesarias para controlar los robots y equipamientos periféricos.

Se puede crear el sistema para el robot desde Posición Inicial>Controlador virtual>Desde diseño, para que lo haga respecto al diseño de la estación que ya tenemos.

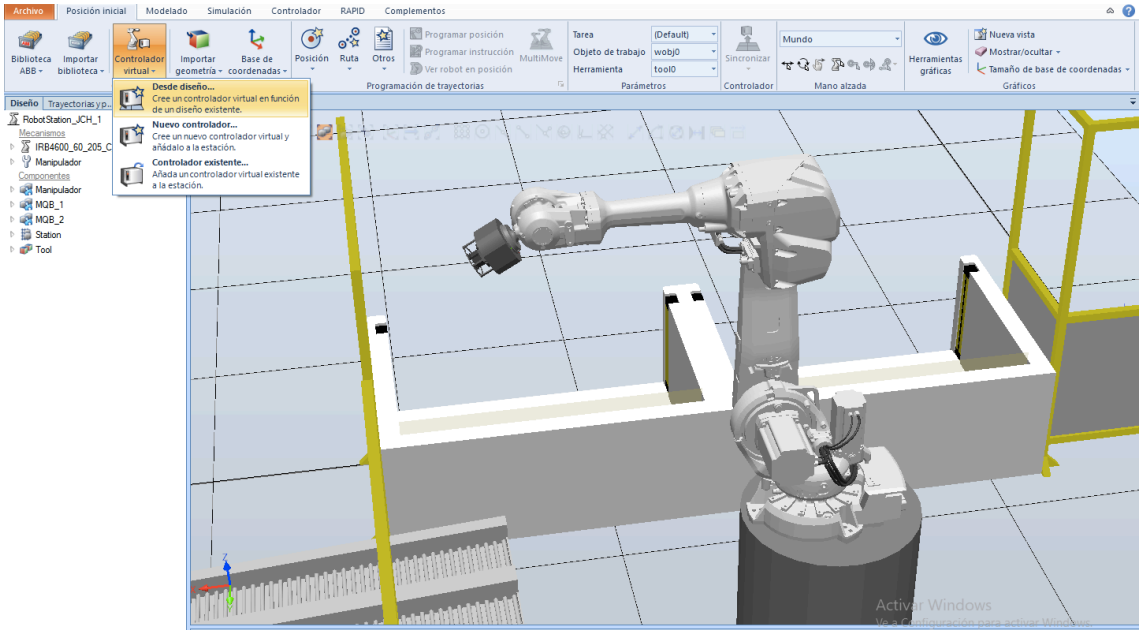


Figura 11.1 Crear controlador virtual en RobotStudio

A continuación, le damos nombre al controlador virtual, elegimos la versión de RobotWare que vamos a utilizar y seleccionamos los mecanismos para el controlador, en este caso, el robot seleccionado.

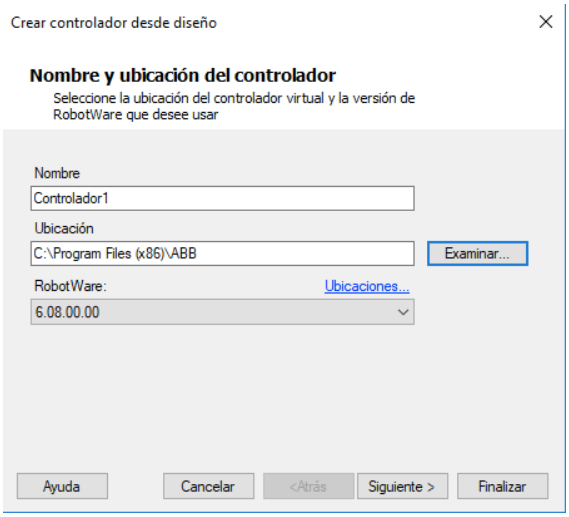


Figura 11.2 Crear controlador virtual desde diseño

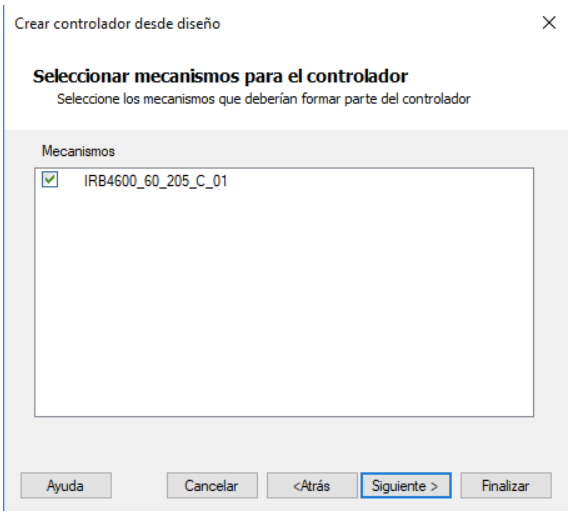


Figura 11.3 Seleccionar mecanismos para el controlador

4.3 TRAYECTORIAS Y PUNTOS

La pestaña correspondiente a trayectorias y puntos nos permite t*eachear* (establecer/modificar) posiciones y objetos de trabajo que se utilizan en el programa. Por defecto se usa el objeto de trabajo ***wobj0*** situado en la base del robot y ***tool0*** situado en el flanje del robot (extremo donde va anclada la herramienta).

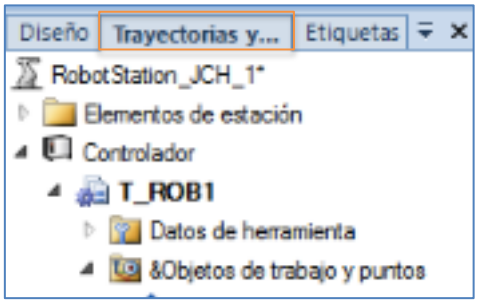


Figura 11.4 Pestaña de Trayectorias y Puntos en RobotStudio

De forma rápida y sencilla se puede colocar el robot en cualquier posición ya sea mediante movimientos independientes de los ejes, movimientos lineales según el sistema de coordenadas de la herramienta seleccionada o de forma automática. Ésta última forma se realiza mediante comandos predefinidos del programa los cuales permiten situar posiciones en círculos concéntricos, esquinas de objetos, punto medio de líneas rectas etc.

Existen dos posiciones genéricas en prácticamente todos los estándares que establecen la base e inicio de las rutinas y movimientos:

j_Home es una posición establecida según los ejes del robot ya que no depende de los elementos de la estación. Establece el inicio y final de los programas del robot y suele ser una posición en la que el robot se encuentre libre del resto de elementos, para facilitar el acceso por la estación, y centrada con respecto a los elementos principales. El PLC comprueba siempre que el robot se encuentre en Home como condición imprescindible para enviar programa al robot.
p_Pounce es una posición activa del robot, la cual puede estar establecida por ejes del robot o coordenadas en función del estándar. El robot avanza a dicha posición al iniciar el programa, donde comprueba y espera permisos para ejecutarlo. Esta posición es especialmente útil para la reducción de tiempos de ciclo y como nexo de unión de rutinas.

Una trayectoria es una sucesión de movimientos que se ejecutan uno detrás del otro. Las posiciones se convierten en instrucciones de movimiento al introducirlas en una trayectoria. Es importante tener en cuenta los argumentos de la instrucción ya que esto condiciona en gran medida la fluidez, precisión y tiempo de ciclo del robot.

$$\overbrace{\text{MoveJ}}^{Instrucción}\ \overbrace{\text{pCentroRack\_1, vmax, z200, t\_Manipulador\textbackslash WObj:= w\_Rack\_1;}}^{Argumentos}$$

Existen muchos factores, argumentos opcionales y propiedades, que afectan al movimiento del robot. En el ejemplo anterior podemos destacar el tipo de movimiento, velocidad y el recorte. En este caso es un movimiento ***MoveJ*** rápido basado en coordenadas con respecto a la herramienta ***t_Manipulador*** y el objeto de trabajo ***w_Rack_1***. La ubicación de ambos afectará al movimiento ejecutado. En cuanto a la velocidad ***vmax*** hace referencia a la velocidad del robot. Cada robot tiene una velocidad máxima que suele rondar los 8000mm/s. El recorte ***z200*** hace referencia a la precisión con la que el robot pasa por un punto. En este caso el robot pasa

a 200mm de la posición ***pCentroRack_1*** sin detenerse. Con el argumento ***fine*** el robot pasa por el punto exacto, deteniéndose en dicho punto.

En la "*Guía del Usuario"* al final del documento se puede encontrar un ejemplo de cómo crear posiciones y trayectorias, así como la sincronización con el código RAPID de forma detallada.

4.4 SINCRONIZACIÓN DE LA ESTACIÓN

Es necesario sincronizar con el controlador virtual para crear programas de RAPID a partir de los puntos y trayectorias creadas en la estación. La sincronización actualiza el programa RAPID del controlador virtual con las últimas modificaciones creadas en la estación.

Este procedimiento se puede realizar tanto desde RobotStudio hacia el controlador o viceversa.

Para ello simplemente seleccionamos Posición inicial>Sincronizar>Sincronizar con RAPID y seguimos los pasos establecidos seleccionando los componentes a sincronizar.

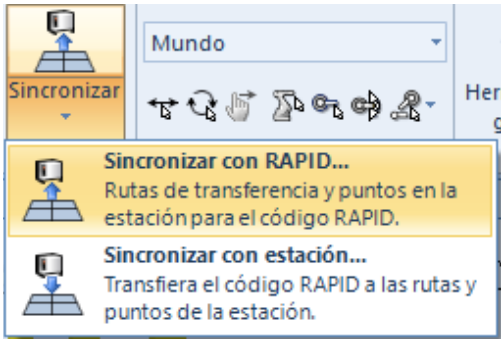


Figura 12.1 Sincronización en RobotStudio

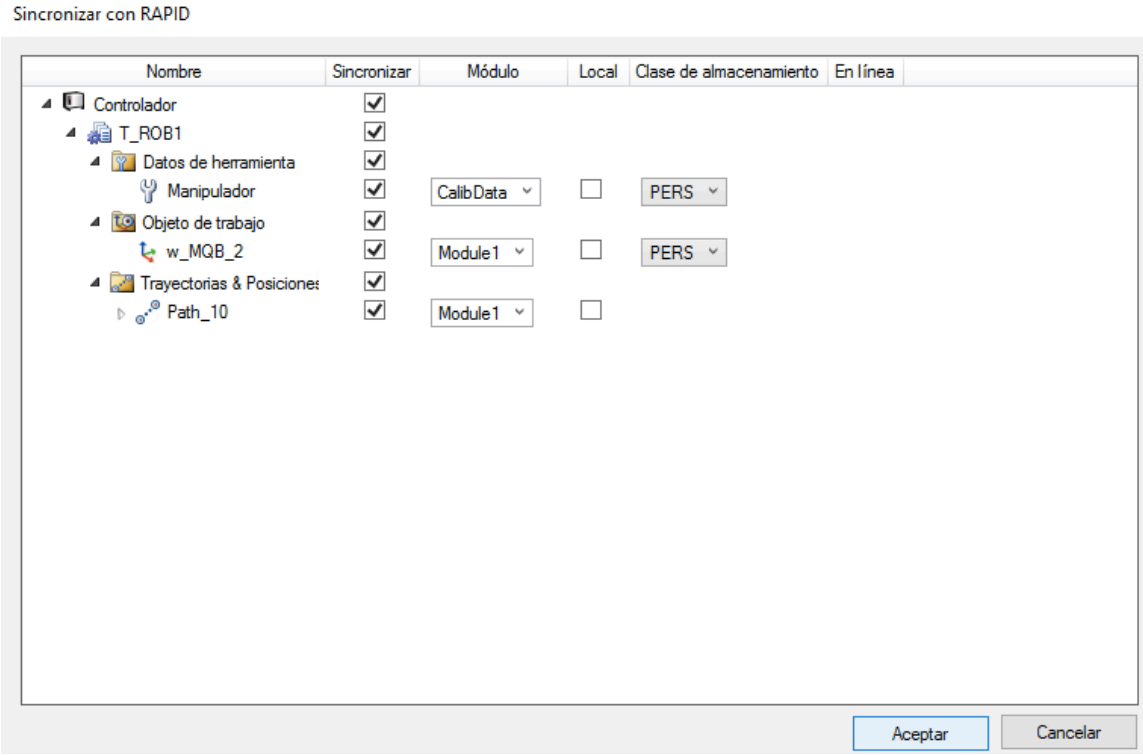


Figura 12.2 Sincronización con RAPID

Desde la ventana de RAPID podemos acceder al módulo de programa con la definición de los puntos y la trayectoria que acabamos de crear ahora en lenguaje RAPID.

4.5 MECANISMO MANIPULADOR

Para que las bridas de la herramienta de manipulación puedan abrirse y cerrarse para coger las piezas debemos crear un mecanismo a partir del ensamblaje creado previamente en Inventor. Mediante componentes inteligentes se podrá controlar las bridas de la herramienta de forma individual otorgándoles movimiento.

Seleccionando Modelado>Crear mecanismo, podemos editar las distintas configuraciones del mecanismo desde la ventana derecha que se muestra.

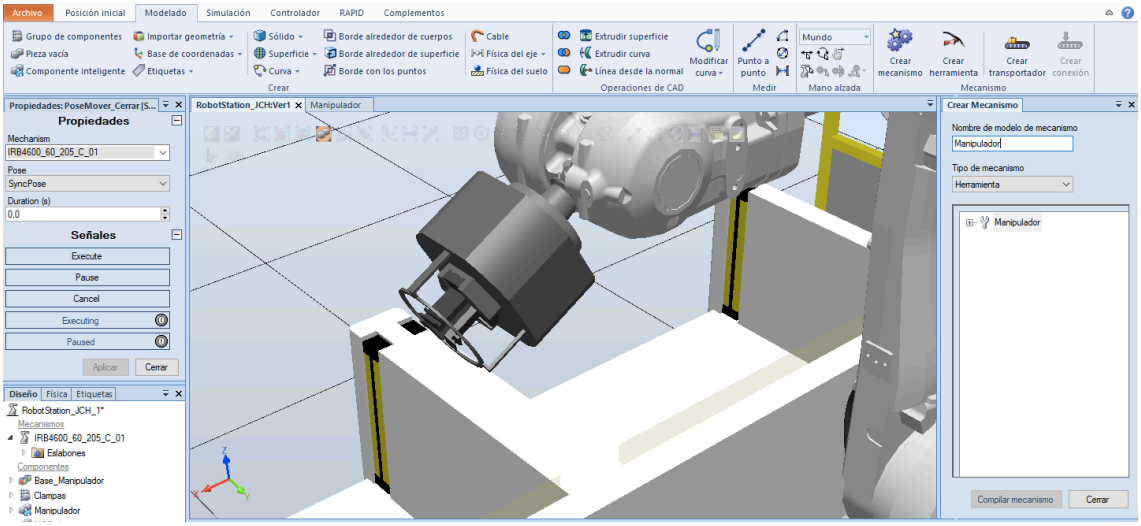


Figura 13.1 Crear mecanismo en RobotStudio

Una vez elegidos el nombre y el tipo de mecanismo a crear, podemos añadir los distintos eslabones haciendo clic con el botón derecho sobre Eslabones. Una vez en dicha ventana añadimos Base_Manipulador y lo establecemos como eslabón base. Del mismo modo añadimos el resto de bridas como eslabones independientes.

Figura 13.2 Crear eslabones del mecanismo

Ahora tenemos que añadir los distintos ejes para los mecanismos. En nuestro caso solamente deberemos crear un eje prismático para cada brida en la dirección del movimiento. Seleccionamos dos puntos para crear el eje en la dirección deseada y establecemos los límites de movimiento como se muestra en la *Figura 13.3.*

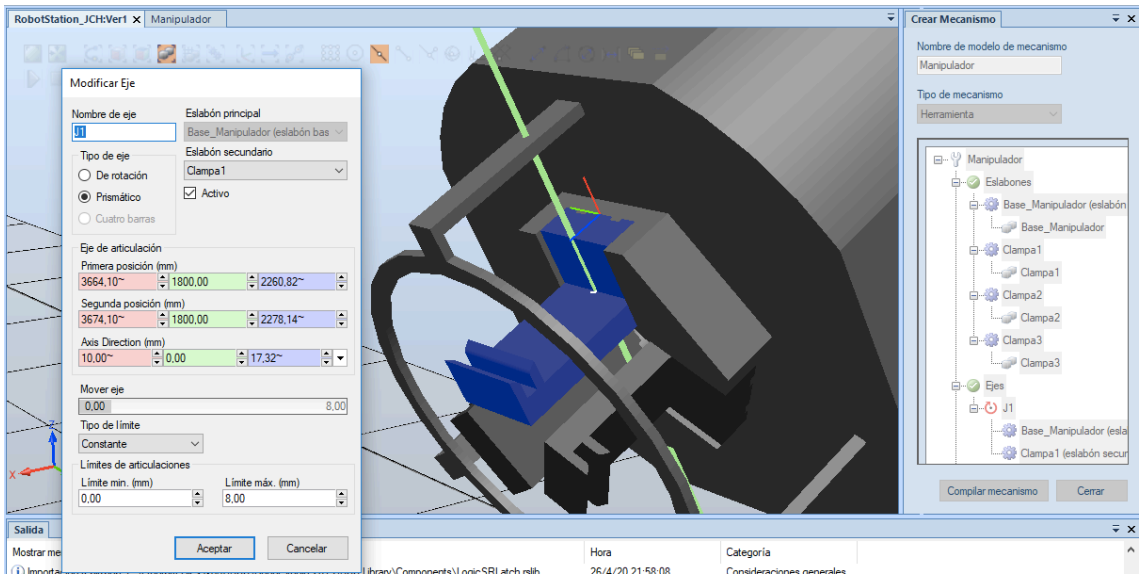


Figura 13.3 Crear eje del mecanismo

De la misma forma creamos los datos del mecanismo y las dependencias. Los ejes de las bridas 2 y 3 serán dependientes del eje de la brida 1 con factor la unidad. Con esto lo que conseguimos es que moviendo únicamente el eje 1 de la brida se muevan al unísono el resto. Una vez tenemos preparados todos los datos y configuraciones del mecanismo podemos compilarlo.

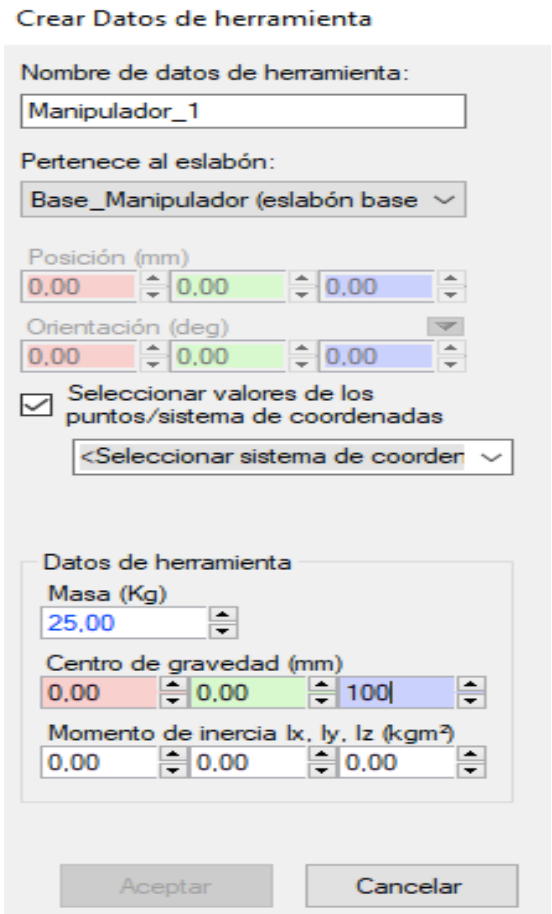


Figura 13.4 Crear datos de herramienta para el mecanismo

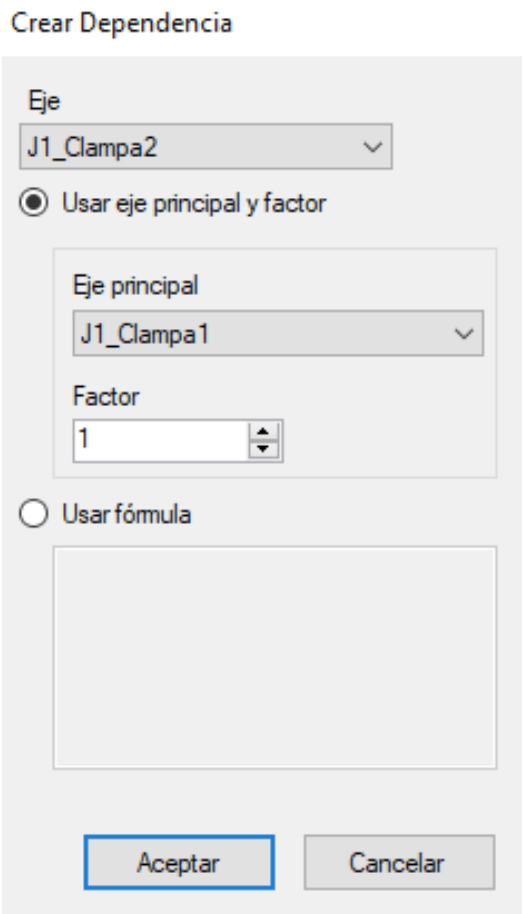


Figura 13.5 Crear dependencias del mecanismo

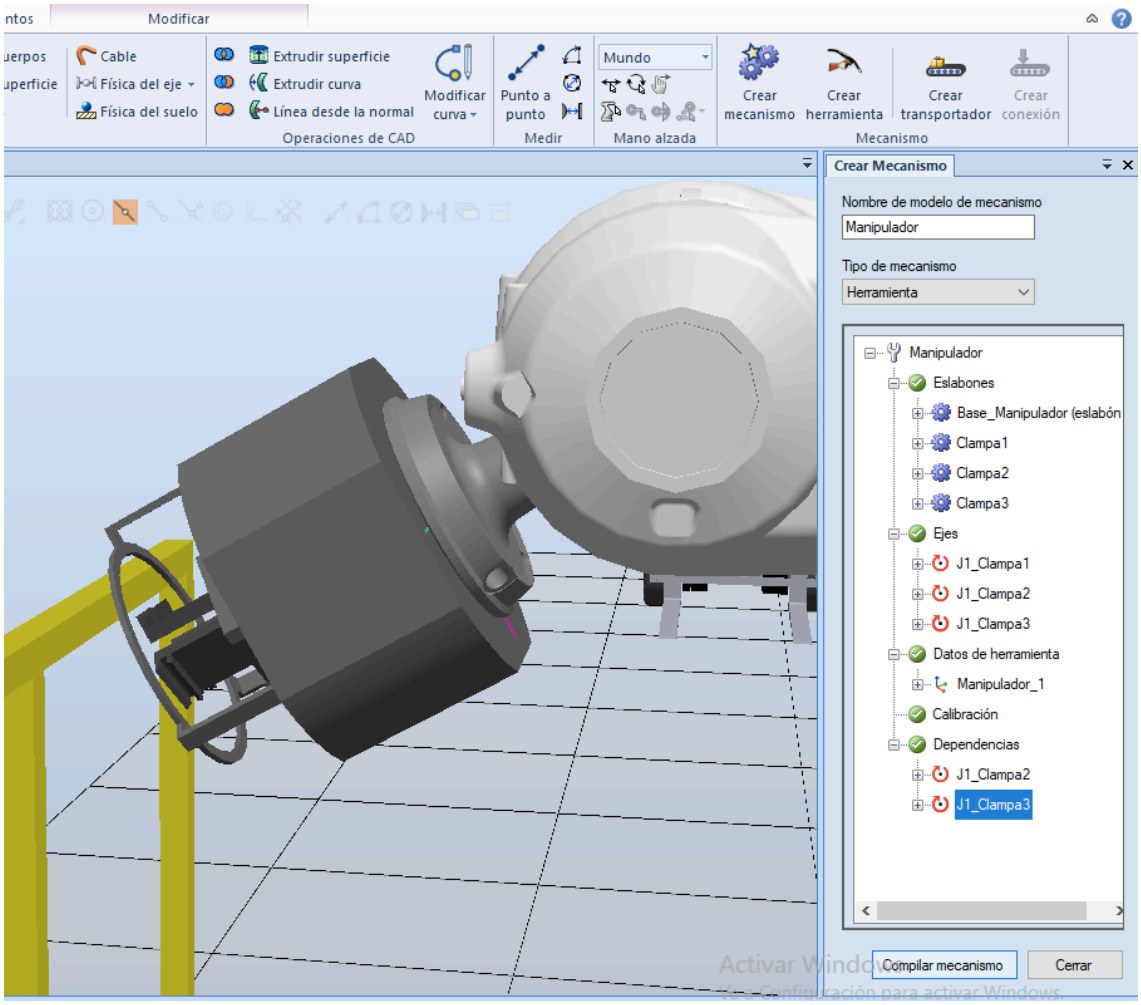


Figura 13.6 Compilación del mecanismo

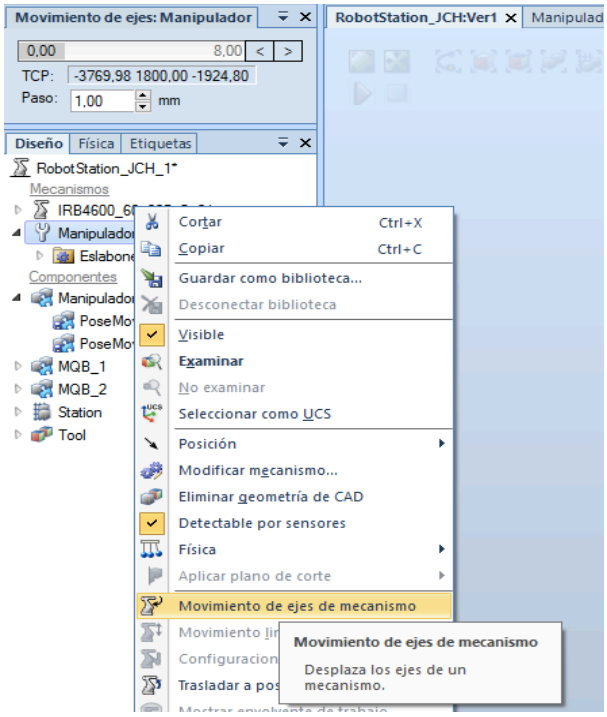


Figura 13.7 Movimiento de los ejes del mecanismo

Podemos comprobar el funcionamiento del mecanismo moviendo los ejes de forma manual seleccionando Movimiento de ejes de mecanismo dentro de nuestro mecanismo como se muestra en la *Figura 13.7.*

En el apartado **Simulación de la estación** de la Bibliografía se encuentran referencias de videos que son de gran utilidad como base y apoyo para la creación y simulación de estaciones en RobotStudio como *[4.1]* Tutorial del Proyecto Final de curso ROBOT-STUDIO.

Recomendable el curso de RobotStudio de *Martín Castillo, J. (2021)* (REEA), al que se puede acceder a través del apartado *[4.2]* de la Bibliografía o del siguiente enlace.

http://reea-blog.blogspot.com/p/robotstudio-abb.html

5. COMPONENTES INTELIGENTES

5.1 INTRODUCCIÓN A LOS COMPONENTES INTELIGENTES

Los componentes inteligentes son objetos de RobotStudio y sirven para simular el comportamiento de los componentes que no forman parte del controlador. Tienen propiedades y lógica integrada para realizar acciones de forma virtual que hacen la simulación más real y dinámica. En nuestro caso, los componentes inteligentes simulan la parte de la comunicación entre el PLC y el robot, lo que nos permite poder controlar entradas y salidas, habilitar acciones con movimiento de componentes, actuación con sensores y otras funciones.

En este apartado se van a explicar las características de los componentes inteligentes básicos utilizados para la simulación de la célula robotizada y establecer así una base sobre dichos componentes.

Existen varias categorías donde se distribuyen todos los componentes inteligentes que podemos gestionar a través de RobotStudio. A continuación se muestran los más relevantes:

- Señales y propiedades » Puertas Lógicas, Contadores…
- Primitivos paramétricos »
- Sensores » Colisión, Lineales, Planos…
- Acciones » Attacher, Detacher, Source…
- Manipuladores » Movimiento Lineal, Posicionador, Movimiento Ejes…
- Controlador »
- Física »
- PLC »
- Otros » Cola

Se puede consultar la *"Guía del usuario"* para mayor información sobre las propiedades de los componentes inteligentes.

5.2 PROGRAMACIÓN Y DISEÑO DE LOS COMPONENTES INTELIGENTES

Nuestra estación se compone de varios componentes inteligentes (Smart Components - SC) necesarios para realizar la simulación. Existe un componente inteligente para cada parte concreta de la estación cuyo procedimiento genérico y diseño se explican a continuación.

Para entender un poco sobre los componentes inteligentes cabe explicar que éstos funcionan mediante señales de entradas y salidas. Dichas señales se pueden controlar de forma independiente desde la simulación o vincularlas directamente con el código Rapid a través del controlador. Las señales se enlazan mediante flechas con los componentes inteligentes internos de manera concreta para activarlos o desactivarlos. Las propiedades de cada componente se pueden modificar según las necesidades.

Es importante comprender que un componente inteligente es un bloque, formado por otros componentes inteligentes subordinados ya creados y existentes en Robot Studio.

Cabe destacar que dichos componentes son bastante complejos y elaborados, por lo que resulta imposible explicar con detalle cada uno de ellos. Existe un laborioso trabajo de fondo para la creación y funcionamiento de éstos. En caso de necesitar más información sobre cómo crear componentes inteligentes se puede consultar en la guía de usuario que se encuentra en el anexo.

5.2.1. Componente inteligente RACK

Este componente hace referencia a los dos racks de carga. Ambos racks tienen componentes inteligentes homólogos por lo que únicamente se explicará el correspondiente al rack 1. A su vez cada rack está compuesto por dos componentes inteligentes independientes.

Para los casos que la nomenclatura sea del tipo MQB_1_x, la “x” hace referencia a un rack lleno (1) o vacío (0).

Rack_1:

Con este componente se controla el funcionamiento de las carretillas para conseguir cargar y descargar los racks de piezas, según el modelo seleccionado, dentro del soporte correspondiente para el mismo. Está formado por un total de 5 entradas y 6 salidas.

ENTRADAS	SALIDAS
MQB_1_1	MQB_1_inpos
MEB_1_1	MEB_1_inpos
Carretilla_HOME	Carretilla_1_inpos
Rack_OUT_1	Rack_Libre_1
CheckRackLleno	RackLleno
	Rack_1_Attach

Tabla 1.1 E/S del componente Rack_1

La carretilla introducirá en el soporte del rack 1 el modelo de pieza seleccionado según la señal de entrada seleccionada (MQB_1_1 o MEB_1_1). Cuando se activen las salidas RackLleno (detecta si hay blisters en el rack), Rack_Libre_1 (cuando la carretilla no corta el sensor de entrada al rack) y MEB_1_inpos o MQB_1_inpos (indicar que el rack de dicho modelo está en posición) y siempre y cuando el rack esté activo, el robot recibirá programa para descargar los discos en el lugar de descarga correspondiente.

Sensores_Rack_1:

Formado mayoritariamente por sensores, este componente conecta cada sensor con una salida detectando cada uno de los discos y blisters que se encuentran presentes. Compuesto de 2 entradas y 54 salidas.

ENTRADAS	SALIDAS
SetCont_Blisters_1	HayPiezas_1_1 (x6)
ResetCont_Blisters_1	HayPiezas_1_1_1 (x48)

Tabla 1.2 E/S componente Sensores_Rack_1

Al activar la entrada SetCont_Blisters_1 se establece el número de blisters que hay en el rack incrementando un contador con cada blíster detectado mediante el sensor. El número obtenido en el contador se almacena en una variable de tipo numérico en RAPID. A su vez se activan los sensores correspondientes a los discos junto con su salida correspondiente HayPiezas_1_x_y (donde "y" hace referencia al número de pieza en blíster situado en el nivel "x"). Cuando al menos se encuentra presente un disco en cada blíster se activan las salidas HayPiezas_1_1. En el programa del robot se establecen unas booleanas a "TRUE" cuando se detecta disco y se almacenan en una matriz 6x8 tipo bool. El programa recorre la matriz para la altura de blíster almacenada en la variable y coge la pieza exacta para cada posición en la que la booleana está en TRUE.

5.2.2. Componente inteligente MANIPULADOR

Para que las acciones del manipulador se asemejen a la realidad se ha creado un componente inteligente del mecanismo Manipulador. La idea es poder controlar el accionamiento de las bridas (abrir/cerrar) según sea necesario a través del código de programa. A su vez se acopla o desacopla el objeto en contacto según la acción efectuada. Está formado por 2 señales de entradas y 2 señales de salida.

ENTRADAS	SALIDAS
ABRIR_Clampa	Abierta
CERRAR_Clampa	Cerrada

Tabla 1.3 E/S componente Manipulador

Al activar la señal ABRIR_Clampas se setea (pone a 1) un componente inteligente de movimiento de ejes para abrir físicamente las bridas y resetea a su vez el componente inteligente inverso utilizado para cerrar las mismas. Cuando se completa el movimiento de las bridas, se activan las señales de salida correspondientes a la acción ejecutada. A la vez se activa un sensor de colisión detectando la pieza en contacto para acoplarla a la herramienta utilizando otro componente (Attacher). Esta acción se utiliza tanto para acoplar los discos como la herramienta ventosa. En esta última se establece como herramienta principal al conectarla con el manipulador. Las señales se conectan a través de la lógica de la estación con el controlador para ser gestionadas de forma apropiada desde el código de programa.

5.2.3. Componente inteligente ESCÁNER

Este componente inteligente simula el funcionamiento de una cámara láser situada en la parte posterior de los racks con el objetivo de escanear los elementos presentes en cada rack. En la realidad el PLC mediante código puede decodificar los datos recibidos por el escáner para enviarle valores concretos al robot. Los datos que recibe el robot son las posiciones exactas de las piezas presentes (coordenadas X e Y con respecto al objeto común de trabajo) y la altura a la que se encuentra el blíster (coordenada Z). De esta forma el robot sabe las piezas que tiene que coger o si tiene que vaciar el blíster en caso de estar vacío. Dispone de 4 entradas y 4 salidas.

ENTRADAS	SALIDAS
Start_Escáner_Rack_1	EscánerON_Rack_1
Start_Escáner_Rack_2	EscánerON_Rack_2
HIDE	EscánerFianlizado_1
Escáner_HOME	EscánerFianlizado_2

Tabla 1.4 E/S componente Escáner

Desde el código de programa se puede activar las entradas de Start_Escáner para iniciar el escaneado del rack seleccionado. Mientras se está ejecutando el componente inteligente de movimiento lineal de la cámara, se activa la salida EscánerON correspondiente, y cuando se completa la ejecución del mismo se activan las salidas de EscánerFinalizado. De esta forma el robot espera a recibir ésta última salida para iniciar la cogida de las piezas.

5.2.4. Componente inteligente VENTOSA

Una vez que el manipulador tiene conectada la herramienta ventosa, se activa o desactiva la succión de la ventosa mediante una señal de entrada para coger y dejar los blisters vacíos. Dispone de 2 entradas y 2 salidas.

ENTRADAS	SALIDAS
Ventosa	VacíoVentosa
Ventosa_HOME	VentosaHomePos

Tabla 1.5 E/S componente Ventosa

Cuando el robot se encuentra en la posición exacta para coger el blíster se activa el vacío de la ventosa mediante la entrada Ventosa para acoplar el blíster a la misma. Cuando la señal de entrada se resetea, la ventosa desacopla el blíster para depositarlo en el rack de descarga. En este caso mediante una única señal se acopla (alto, 1) o desacopla (bajo, 0) el objeto. Cuando se está efectuando el vacío, la señal de salida VacíoVentosa cambia a 1.

5.2.5. Componente inteligente RACK_DESCARGA

Hace referencia al rack de descarga donde se depositan los blisters vacíos. De igual forma que pasaba en el rack de carga, está compuesto por dos componentes inteligentes independientes.

Rack_Descarga:

Controla las carretillas que introducen los racks vacíos y se llevan los llenos de blisters. Está formado por 4 señales de entrada y 5 señales de salida.

ENTRADAS	SALIDAS
MarchaCarretilla	ÁreaLibreDescarga
MarchaCarretilla1	RackDescInPos
CarretillaDesc_HOME	CarretillaDescInPos
PrimerRackDesc	CarretillaAct
	RackLibre

Tabla 1.6 E/S componente Rack_Descarga

Cuando se selecciona uno de los racks de carga activo y mientras no se encuentre ningún rack en el soporte de descarga, se activa la entrada PrimerRackDesc para introducir un único rack vacío. Cuando se encuentra un rack en posición de descarga se activa la salida RackDescInPos y cuando la carretilla no está cortando el sensor de entrada, la salida ÁreaLibreDescarga habilita al robot a efectuar la dejada de blisters.

Cuando el primer rack está lleno y el robot deja el quinto blíster, una carretilla acerca un rack vacío mediante MarchaCarretilla. Cuando el robot libera el área de dejada tras haber depositado el último blíster, la carretilla cambia el rack lleno por uno vacío con la entrada MarchaCarretilla1.

Sensores_Rack_Descarga:

Los sensores que controlan el número de blisters que se encuentran en el rack de descarga y desde donde se activan las entradas del componente inteligente anterior para efectuar la carga o descarga de los racks mediante las carretillas. 4 señales de entrada y 2 señales de salida forman este componente inteligente.

ENTRADAS	SALIDAS
SetCont_Blisters_3	RackDescargaLleno
ResetCont_Blisters_3	InicioDescarga
RobFueraArea	
RackDescLibre	

Tabla 1.7 E/S componente Sensores_Rack_Descarga

Un contador recibe el número de blisters detectado por los sensores, que se almacena en una variable para conocer la altura de dejada. Cuando dicho contador detecta 5 blisters se activa la salida InicioDescarga que está conectada con la entrada MarchaCarretilla del componente anterior para iniciar el proceso de descarga. Con todos los blisters detectados por los sensores

se activa la salida RackDescargaLleno para efectuar el cambio de los racks, ya que de igual manera, la salida está conectada con MarchaCarretilla1 en la lógica de la estación.

5.2.6. Componente inteligente ÚTIL_DESCARGA

Existe un componente inteligente independiente para el útil de descarga según el modelo de disco. Por ser componentes homólogos analizaremos únicamente uno de ellos.

Para cada modelo el disco se deposita en el lado de descarga correspondiente. Este componente detecta el disco una vez depositado por el robot y lo desplaza hasta el final del útil donde, en la realidad, sería cogido por otro robot e introducido en la línea de producción. Se encuentra formado por 2 entradas y 1 salida.

ENTRADAS	SALIDAS
RobOutArea	MQB/MEB_Dejada
Sink	

Tabla 1.8 E/S componente Útil_Descarga

Cuando no se encuentra ningún disco en la posición de descarga, la señal de salida MQB_Dejada indica que el robot tiene permiso para efectuar la dejada del disco MQB. Cuando el robot libera el área de dejada después de depositar el disco, la entrada RobOutArea hace actuar un componente para desplazar el disco hasta el final del útil. Cuando el próximo disco se deposita, éste desaparece (sink) simulando la cogida por el robot de la otra celda.

5.3 LÓGICA DE ESTACIÓN

La lógica de la estación conecta las entradas/salidas de la estación, los componentes inteligentes y el sistema del robot, mediante el controlador virtual, entre sí. El correcto funcionamiento entre todos los elementos de la estación está gestionado desde la lógica de la estación. Desde la pestaña Simulación>Lógica de estación podemos acceder a la configuración de la misma. Mediante flechas podemos crear las conexiones correspondientes e incluso añadir puertas lógicas u otros componentes para el correcto enlazamiento.

En la *Figura 14* en la siguiente página, se visualiza el diseño completo de la lógica de la estación en RobotStudio.

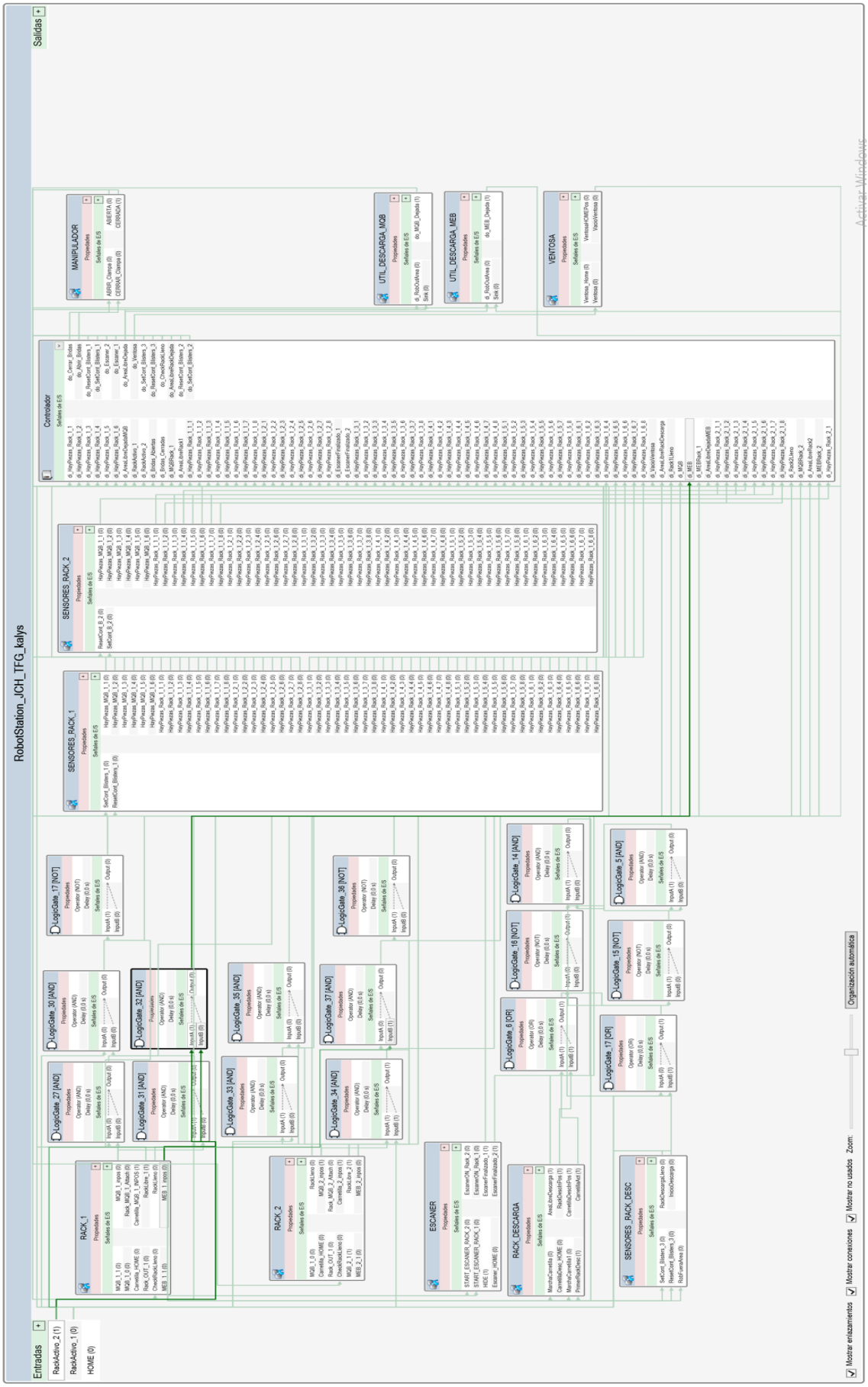


Figura 14 Lógica de la estación desde RobotStudio

6. PROGRAMACIÓN DEL ROBOT INDUSTRIAL

6.1 INTRODUCCION AL LENGUAJE RAPID

Robotics Application Programming Interactive Dialogue es el lenguaje de programación desarrollado por ABB para controlar el robot mediante una secuencia de instrucciones que forman un programa. Consta de una rutina principal (MAIN), unas subrutinas subyacentes a la rutina principal y los datos del programa. Se pueden crear diferentes estructuras mediante módulos dentro del programa. Las subrutinas facilitan la ejecución del código al estar estructurado en distintas partes más compactas.

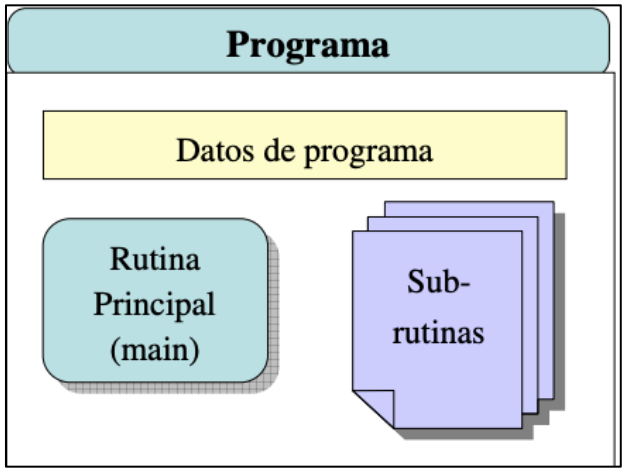


Figura 15.1 Partes de un programa en lenguaje RAPID

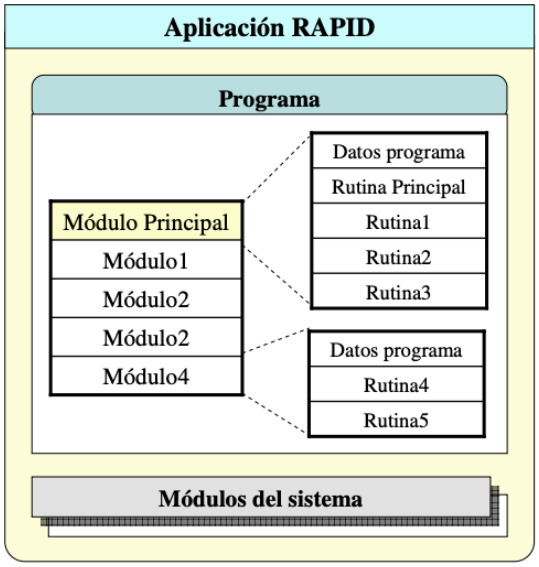


Figura 15.2 Aplicación lenguaje RAPID

Por la rutina principal (main) empieza a ejecutarse el código, vinculado con los datos del programa, enlazando con las distintas rutinas según avanza el puntero del programa.

En los datos del programa se declaran todas las variables a utilizar, las posiciones, objetos de trabajo, herramientas u otros datos de tecnologías específicas.

El programa que se ha creado para este proyecto está formado por 3 módulos, los cuales se explican a continuación.

Se puede consultar la información previamente proporcionada sobre el lenguaje Rapid a través de *[3.2] Lenguaje RAPID.*

MÓDULO CALIBDATA: Alberga todos los datos genéricos del programa, lo que facilita la organización y la búsqueda de información. Se puede encontrar la definición de posiciones, objetos de trabajo, datos de las herramientas etc.

MÓDULO MAIN: Es el módulo principal del programa donde encontramos el procedimiento "*Main*" y por el que empieza el programa. Además, se encuentran otras rutinas genéricas, normalmente sin movimientos o trayectorias. Al inicio del módulo se declaran todas las variables de distintos tipos (num, bool etc.) utilizadas en dicho módulo.

MÓDULO PROGRAMAS: En este módulo se encuentran todas las rutinas subyacentes a la rutina principal como pueden ser los programas principales y las rutinas de movimiento necesarias.

6.2 FUNCIONES Y COMANDOS BÁSCIOS

La programación del código se basa en lenguaje Rapid, pero las funciones y comandos básicos son específicos de cada marca, en este caso ABB. Por ello es necesario explicar la estructura de las funciones y comandos más importantes, así como algunos conceptos genéricos de la robótica.

OBJETOS DE TRABAJO (WORKOBJECTS):

Los objetos de trabajo están referenciados mediante un sistema de coordenadas definido por el usuario, y vinculado con la base del robot (wobj0). La base del robot queda unívocamente establecida con respecto al sistema de coordenadas mundo. En la realidad éste coincide con la base del robot.

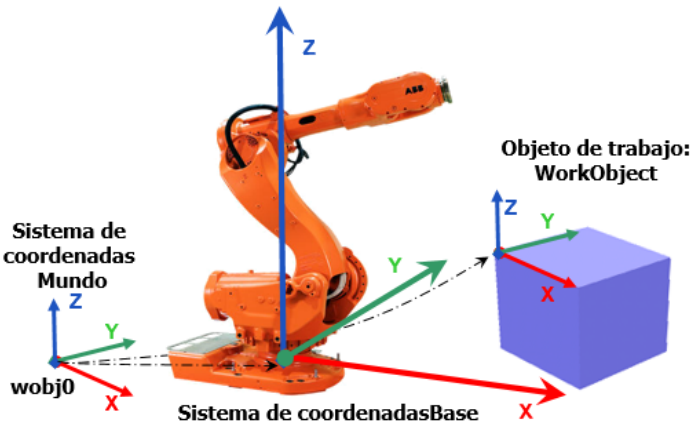


Figura 16.1 Sistemas de coordenadas robot

Definimos un objeto de trabajo mediante una variable wobjdata de la siguiente forma:

```
PERS wobjdata wobj1 :=[ FALSE, TRUE, "", [ [30, 60, 20], [1, 0, 0 ,0] ], [ [0, 20, 30], [1, 0, 0 ,0] ] ];
```

El objeto de trabajo de la figura anterior se describe utilizando los valores siguientes:

- TRUE/FALSE según si el robot sostiene el objeto de trabajo.
- TRUE/FALSE si se utiliza el sistema fijo de coordenadas del usuario.
- La posición del origen del sistema de coordenadas del usuario (x, y, z) en mm.
- La rotación del sistema de coordenadas del usuario, expresada como un cuaternio (q1, q2, q3, q4).
- La posición del origen del sistema de coordenadas del objeto (x, y, z) en mm.
- La rotación del sistema de coordenadas del objeto, expresada como un cuaternio (q1, q2, q3, q4).

Todas las posiciones asociadas a un objeto de trabajo quedan referenciadas a dicho sistema de coordenadas por lo que al desplazar por cualquier motivo el objeto de trabajo, simplemente será necesario calcularlo de nuevo y las posiciones y trayectorias se actualizarán con respecto nuevo objeto de trabajo automáticamente.

En nuestro programa se han definido 5 objetos de trabajo correspondientes a cada elemento principal de la estación. Se definen en el módulo ***CalibData*** que aparece completo en el *Anexo I.*

POSICIONES Y MOVIMIENTOS:

Una posición puede estar definida mediante coordenadas cartesianas (x,y,z) a través del TCP (Punto Central de la Herramienta) con respecto al workobject establecido. Se define en una variable *robtarget* como se muestra a continuación:

```
CONST robtarget p1 := [[100, 500, 200], [1, 0, 0, 0], [1, 1, 0, 0], [ 9E9, 9E9, 9E9, 9E9, 9E9, 9E9]];
```

Se define la posición p1 de la forma siguiente:

- La posición del robot: x = 100, y = 500 y z = 200 mm en el sistema de coordenadas de objeto.
- La orientación de la herramienta representada en cuaternios.
- La configuración de ejes del robot.
- La posición de los ejes externos expresada en grados o mm (en función del tipo de eje).

También se pueden definir posiciones según los ejes del robot (en grados) ya que cada uno de ellos se puede mover de manera independiente. De esta forma se definen con una variable del tipo *jointtarget*.

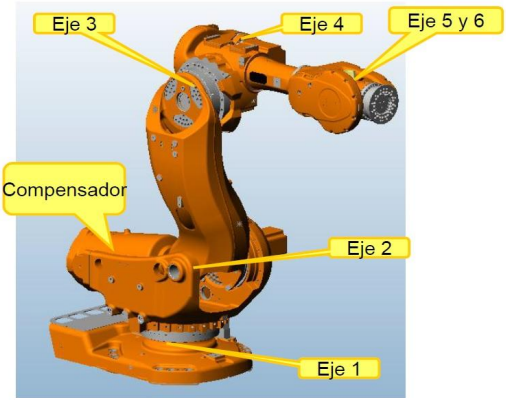


Figura 16.2 Ejes de un robot antropomórfico

```
CONST jointtarget j_Home := [ [ 0, 0, 0, 0, 0, 0], [ 0, 9E9, 9E9, 9E9, 9E9, 9E9] ];
```

Se define la posición j_Home de la forma siguiente:

- La rotación de cada eje del robot en grados, positivo o negativo, a partir de la posición de calibración del eje.
- La posición de los ejes externos expresada en grados o mm (en función del tipo de eje).

De forma análoga a lo anteriormente explicado, se realizan los distintos movimientos del robot. Los movimientos manuales del robot se pueden realizar, por tanto, por ejes o de forma lineal y según algún sistema de coordenadas. En este último se puede reorientar la herramienta sobre su TCP con respecto al workobject seleccionado.

Para movimientos del robot según instrucciones del programa y con la misma idea arriba explicada, se describen los más comunes.

MoveJ/MoveL/MoveC: Instrucciones para posiciones del tipo robtarget referenciadas a un sistema de coordenadas. En un MoveJ el robot se desplazada al punto seleccionado siguiendo un movimiento no lineal de la forma más rápida posible. En un MoveL el robot describe una trayectoria siguiendo una línea recta sin importar la configuración óptima de los ejes. Es un movimiento más lento que el anterior. MoveC hace referencia a un movimiento circular, el cual queda definido por un punto inicial, otro final y un punto intermedio.

MoveAbsJ/MoveAbsL: Movimientos para posiciones definidas según jointtargets en los que se realiza un movimiento rápido por ejes. No se tiene en cuenta los sistemas de coordenadas ni el TCP de la herramienta.

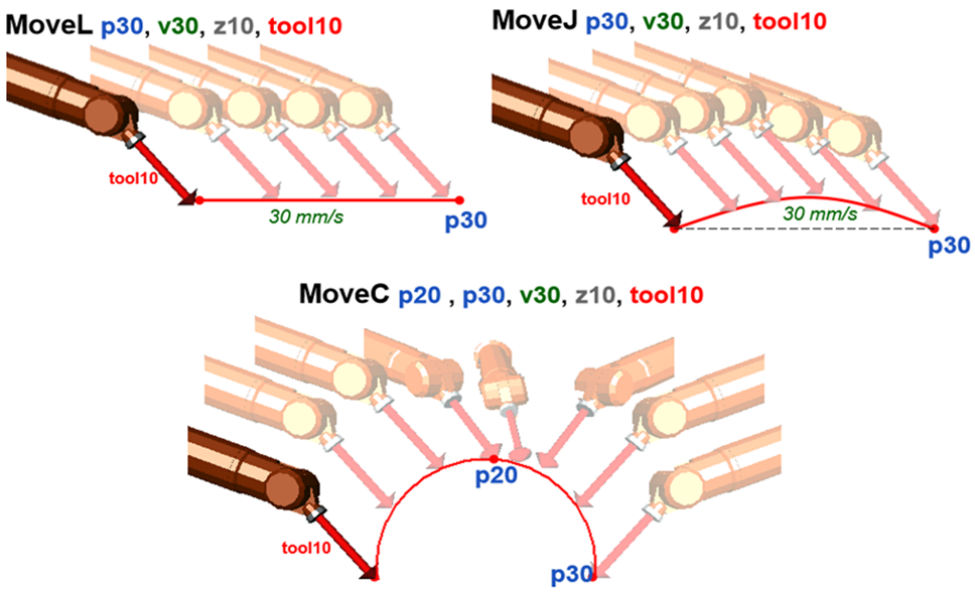


Figura 16.3 Tipos de instrucciones de movimiento

Otros comandos importantes que se han utilizado en el programa son los siguientes:
Set/Reset: Para activación y desactivación de señales digitales de salida. Están conectadas con las entradas de los componentes inteligentes mediante el controlador en la lógica de la estación.
WaitUntil: Instrucción de espera hasta que se cumple una condición establecida. En la mayoría de los casos es una señal digital de entrada, pero se puede utilizar también con booleanas (True/False), variables numéricas, etc.
WaitTime: El puntero del programa espera un tiempo indicado en la instrucción sin continuar recorriendo el programa. Se utiliza para garantizar una lectura correcta del código en puntos concretos, aunque no es una instrucción que deba utilizarse con asiduidad ya que empeora el tiempo de ciclo.

A través de *[3.3] Manual del Operador RobotStudio* o de la Librería de ABB se puede ampliar conocimientos acerca de funciones y herramientas de RobotStudio. Se puede acceder directamente a ambos mediante los enlaces proporcionados en la Bibliografía en el apartado **Programación y Código RAPID** o desde RobotStudio en la pestaña de Ayuda *[3.1] Código y funciones RAPID.*

6.3 ESQUEMA DE PROGRAMACIÓN

Como hemos comentado anteriormente el código se ha programado en 3 módulos claramente diferenciados. Todos ellos están vinculados entre sí por lo que se puede llamar a rutinas que no se encuentren en el módulo del cursor actual. Para un mayor entendimiento de la estructura de programación utilizada se presenta la siguiente figura.

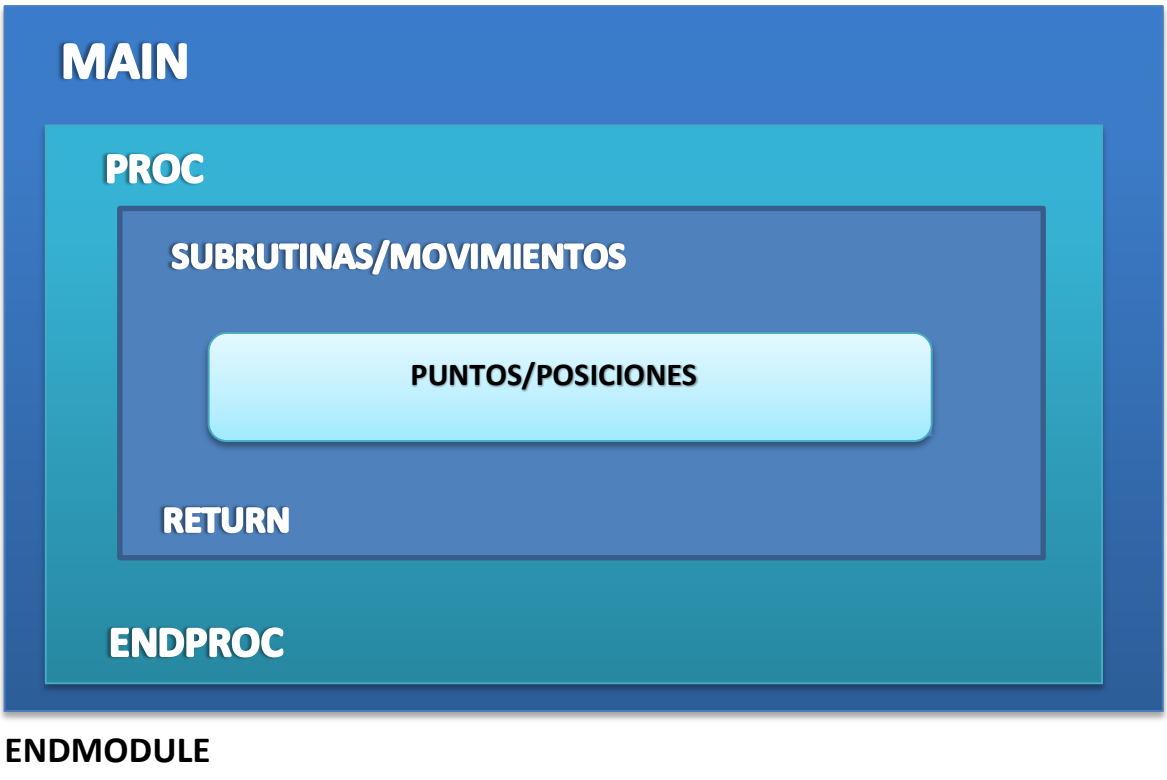


Figura 17 Esquema de programación

6.4 CÓDIGO DEL PROGRAMA

El programa empieza en el procedimiento principal o *MAIN*, donde se inicializan las variables y se comprueba que el robot se encuentra en la posición de inicio (*HOME*) con el manipulador en correcto estado. Posteriormente y dentro de un bucle *WHILE* se efectúa la elección del programa según el rack activo y el modelo de disco introducido. A través de un *TEST* con sus respectivos *CASE* se ejecuta el programa indicado. Existen dos programas dependiendo cada uno del rack donde se va a trabajar.

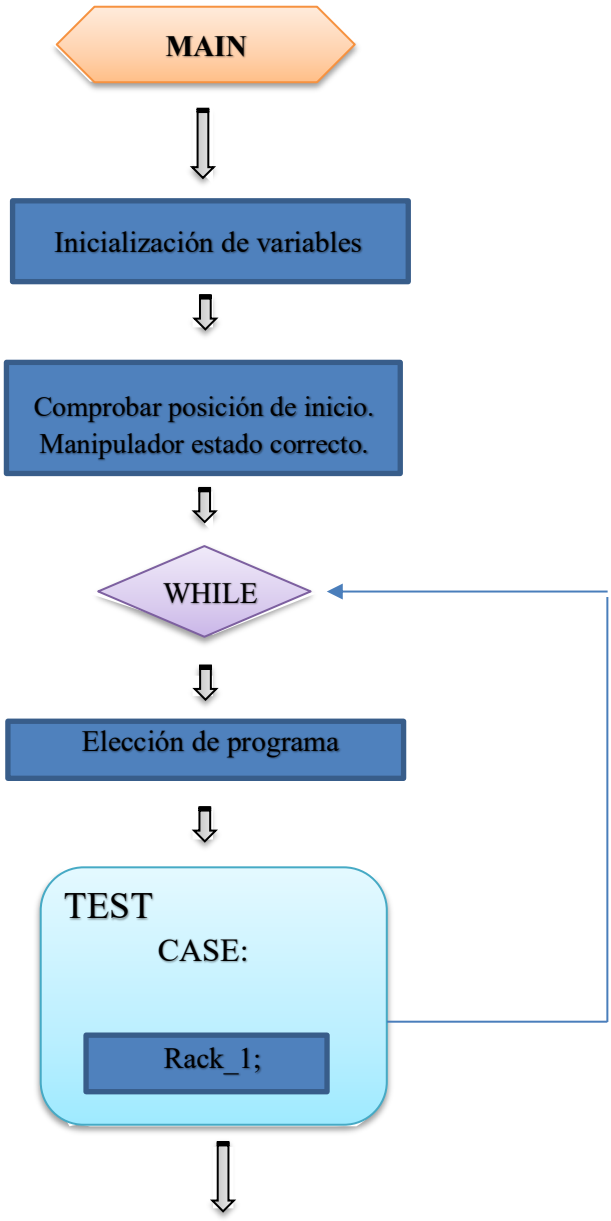


Figura 18.1 Diagrama 2D del procedimiento Main

Una vez se han comprobado todas las condiciones necesarias para el inicio del programa ***RACK_1*** el robot se desplaza desde la posición de inicio a una posición activa, próxima al objeto de trabajo. Esta posición activa (***Pounce***) se utiliza para reducir tiempos de ciclo y como punto de conexión entre rutinas. En dicha posición se espera a que el escáner se haya comprobado, el área de trabajo esté libre y se hayan recibido y actualizado las posiciones de las piezas a coger que detecta y envía el escáner. Mediante un bucle ***FOR*** se recorre una matriz de booleanas asociadas a los discos presentes, cogiendo todos los discos cuya booleana sea **TRUE.** Se almacenan los valores de las posiciones (*x,y,z*) en variables numéricas obtenidos de una matriz de posiciones. Estas variables numéricas se suman a la posición de referencia y establecen una nueva posición para cada disco. Una vez en esa posición se activan las bridas del manipulador para coger la pieza que posteriormente se deposita en el útil de dejada apropiado al modelo.

Una vez se han vaciado todos los discos de un blíster y saliendo del bucle, el robot ancla la herramienta ventosa, que se encuentra en su soporte, al manipulador para succionar el blíster vacío. La altura del blíster se obtiene de manera análoga a los discos. Se deposita posteriormente en el rack de descarga en la posición correspondiente al siguiente nivel donde se detecta un blíster. Cuando el rack de descarga está lleno se sustituye por uno vacío utilizando carretillas.

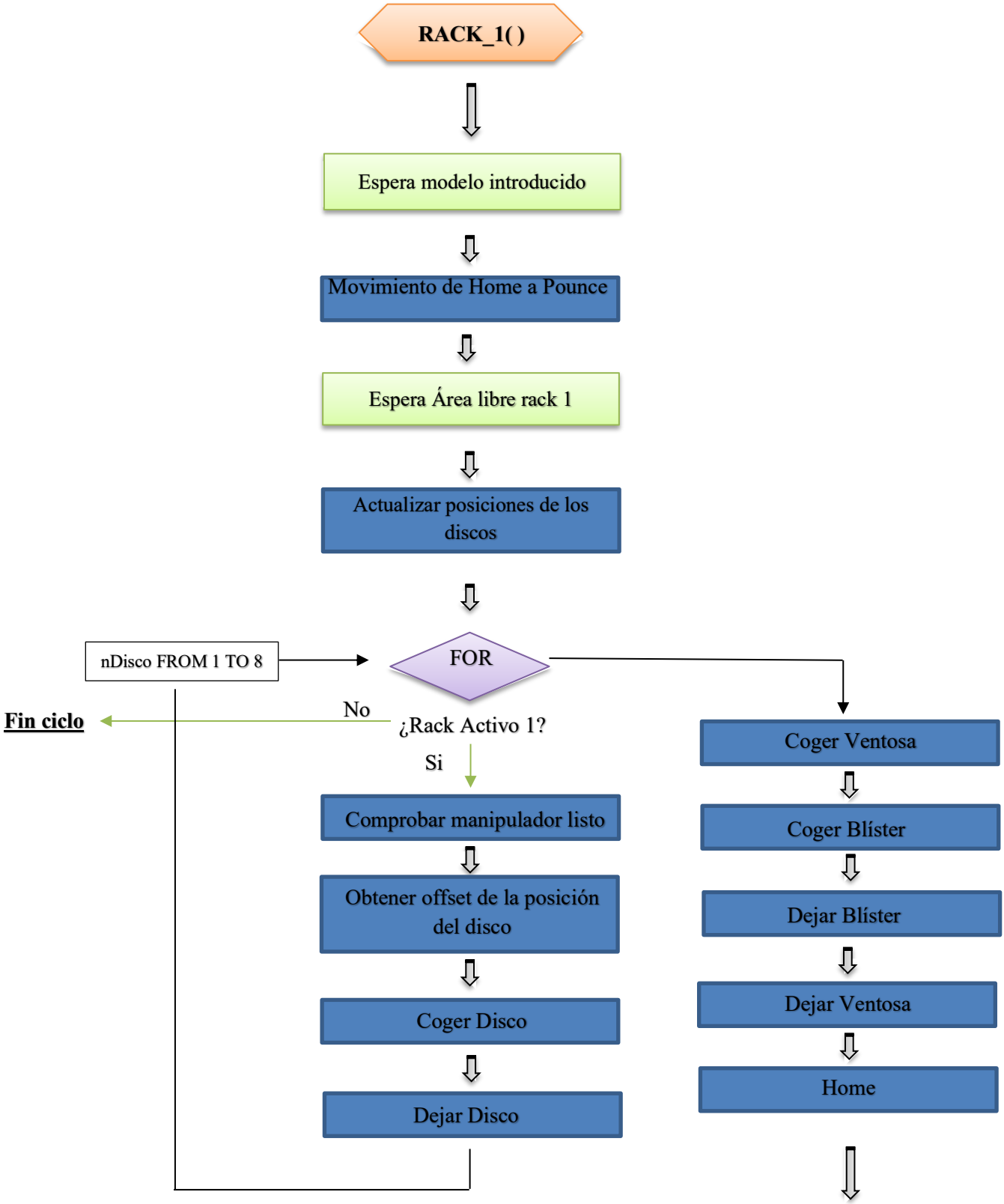


Figura 18.2 Diagrama 2D del procedimiento Rack_1

6.5 SEÑALES DE ENTRADAS Y SALIDAS

Las señales de entradas y salidas son necesarias para una correcta interacción entre la estación y el código RAPID. Por ello se conectan mediante el controlador en la lógica de la estación cada una ellas.

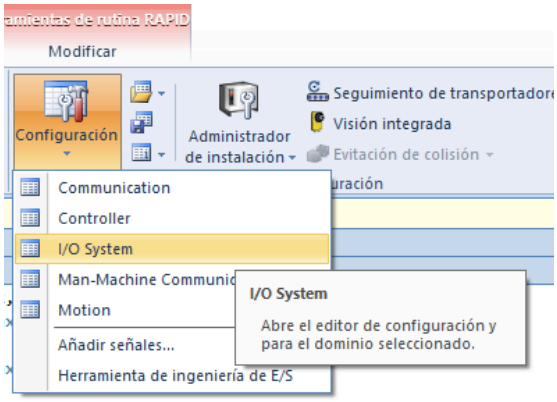


Figura 19.1 E/S del sistema en RobotStudio

Para crear una entrada o salida digital en el programa debemos hacerlo desde la pestaña Controlador>Configuración>I/O System>Signal.

Haciendo clic secundario sobre el apartado de Signal podemos crear nuevas entradas o salidas tal y como se muestra en la imagen de la instancia del editor

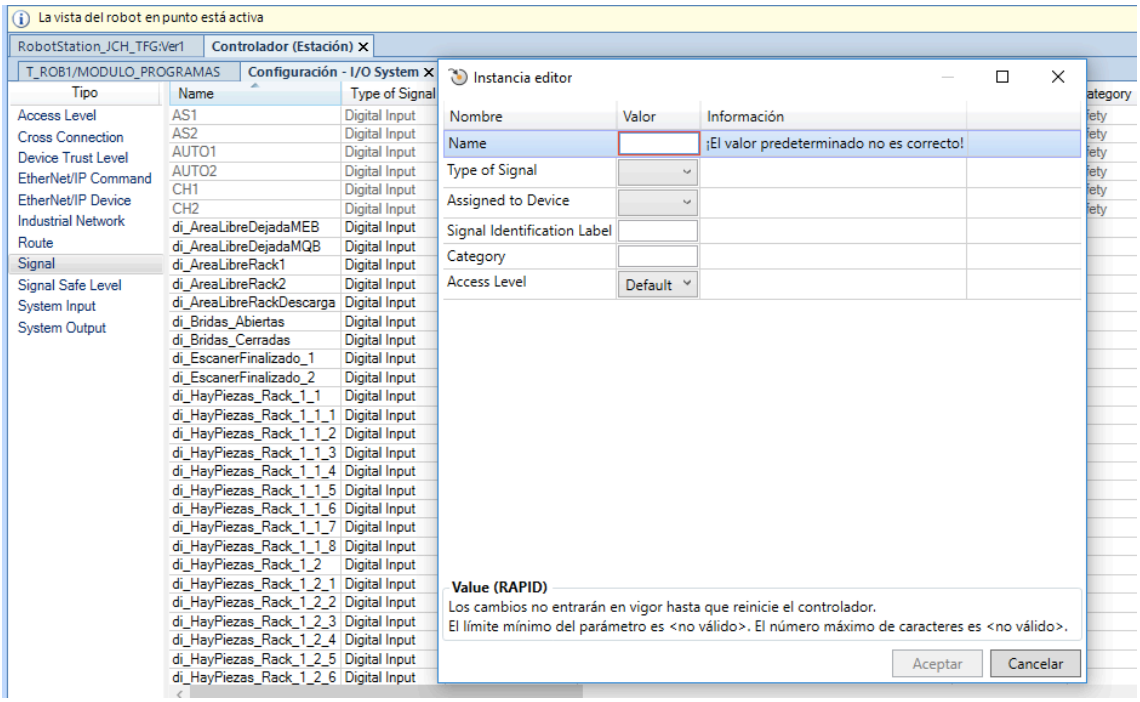


Figura 19.2 Crear nueva señal en el controlador

Se pueden crear distintos tipos de entradas y salidas, pero todas ellas irán mapeadas en el puerto correspondiente tanto en el controlador del robot como en el módulo físico (FESTO), siguiendo el **EPLAN** (Plano Eléctrico). En la simulación las señales no están mapeadas ya que esto depende del estándar del cliente. Todas las E/S se almacenan en el robot en un módulo de sistema llamado **EIO.**

Cuando se activa una salida, se manda un impulso eléctrico al puerto del módulo correspondiente para hacer actuar cualquier mecanismo (ej. bridas) mediante hidráulica, ya que se conectan con tubos por donde circulan 6 bar de aire a presión.

Figura 19.3 Módulo FESTO de E/S

Los sensores del robot activan las entradas digitales del programa ya que éstas también se encuentran conectadas a través del módulo FESTO de entradas en el puerto correspondiente.

Hay un total de **128** entradas digitales y **13** salidas digitales en el programa. En la *Tabla 2* se muestran todas ellas:

ENTRADAS DIGITALES (DI)	SALIDAS DIGITALES (DO)
Di_AreaLibreDejadaMEB	Do_Abrir_bridas
Di_AreaLibreDejadaMQB	Do_AreaLibreDejda
Di_AreaLibreRack1	Do_AreaLibreRackDejda
Di_AreaLibreRack2	Do_Cerrar_Bridas
Di_AreaLibreRackDescarga	Do_CheckRackLleno
Di_Bridas_Abiertas	Do_Escaner_1
Di_Bridas_Cerradas	Do_Escaner_2
Di_EscanerFinalizado_1	Do_ResetCont_Blisters_1
Di_EscanerFinalizado_2	Do_ResetCont_Blisters_2
Di_HayPiezas_Rack_1_x (x6)	Do_ResetCont_Blisters_3
Di_HayPiezas_Rack_2_x (x6)	Do_SetCont_Blisters_1
Di_HayPiezas_Rack_1_x_x (x48)	Do_SetCont_Blisters_2
Di_HayPiezas_Rack_2_x_x (x48)	Do_SetCont_Blisters_3
Di_MEB	
Di_MEBRack_1	
Di_MEBRack_2	
Di_MQB	
Di_MQBRack_1	
Di_MQBRack_2	
Di_Rack1Lleno	
Di_Rack2Lleno	
Di_RackActivo_1	
Di_RackActivo_2	
Di_VacioVentosa	

Tabla 2 E/S del programa

6.6 TIEMPO DE CICLO

Este es el apartado más importante y más complicado en la programación de un robot. Todas las empresas quieren reducir los tiempos de producción para producir el máximo número de piezas posible en el menor tiempo, siendo más competitivos y generando mayores ingresos. En algunos casos será necesario reducir en decimas de segundo el tiempo de ciclo para conseguir los objetivos establecidos, lo cual no es tarea nada fácil.

Es por ello que una de las condiciones del cliente para la línea de producción es que se produzca un set de piezas cada 24 segundos. Los pallets circulan por el *conveyor* (cinta transportadora) dando vueltas por la línea infinitamente. Cada pallet desplaza dos piezas, correspondientes a la parte izquierda y derecha del coche. Por lo que para el cálculo del tiempo de ciclo de nuestra estación se tendrá que calcular por cada dos discos de freno.

Para ello se utiliza una rutina creada en un módulo de sistema que escribe y almacena los datos de tiempo de ciclo de programa en un archivo (***EscribirTCEnArchivo***). Se utilizan contadores de tiempo en el programa para establecer su duración de forma precisa. La comunicación con el PLC con respecto a los SC, funcionamiento de elementos externos y el ajuste más preciso de trayectorias y velocidades del programa en planta hace que los tiempos sean distintos a los de la simulación. Por ello esta tarea se deberá realizar de nuevo en planta cuando se haya montado la estación para contrastar con los tiempos obtenidos en la simulación.

Para calcular el tiempo de ciclo de nuestra estación se tendrá en cuenta tanto el programa como el modelo de discos. En cada programa el robot vacía un rack entero con sus respectivos cambios de herramienta para depositar los blisters vacíos. Se tienen en cuenta las condiciones más desfavorables para realizar el cálculo ya que existen muchos factores y condiciones que afectan. Se supone que las peores condiciones serán las correspondientes al último blíster de cada rack para el modelo MEB con el rack de descarga vacío ya que las posiciones y trayectorias son las más lejanas.

$$\boldsymbol{TCM\ RACK1} = \frac{\sum_{i=1}^{n} \frac{\boldsymbol{TCRack1\_n}}{4}}{n} + \text{TCInit}\ (\boldsymbol{S}) \quad \text{n=6 (nivel por blíster)}$$

Ecuación 1 Tiempo de Ciclo Medio para el Rack1 (siendo "n" el nivel por blíster)

$$\boldsymbol{TC\ ESTACIÓN} = \frac{\mathbf{TCM\ RACK\_1 + TCM\ RACK\_2}}{\mathbf{2}}\ (\boldsymbol{S})$$

Ecuación 2 Tiempo de Ciclo de la Estación

Teóricamente el tiempo de ciclo de la estación se calcula como la media de los tiempos de ciclo medios de cada rack según la *Ecuación 2* aunque no se asume dicha hipótesis para el cálculo ya que por versatilidad de la estación no se tiene por que alternar entre ambos. Por ello se utiliza el tiempo de ciclo de cada rack por separado o en su caso, la media de los tiempos más extremos de cada rack.

Nota: Tener en cuenta que cuando se calcula el TCM de ambos racks, la posición de cogida de cada blíster es inversa a la posición de dejada. Esto se ve reflejado en el cálculo del TCM.

La programación del código del programa se ha ido optimizando y adaptando para reducir lo máximo posible el tiempo de ciclo.

- Se han ajustado las velocidades y los recortes de los puntos para crear movimientos más fluidos y rápidos sin incumplir las condiciones básicas de estándar de la línea. Por ello para las velocidades con blíster, en puntos de aproximación y en posiciones concretas, se han mantenido las velocidades fijadas por estándar.
- Para el escáner se utilizan rutinas con condiciones de "no checkeo" de señales para evitar la detención del robot y su posterior comprobación durante los movimientos de vuelo, reduciendo el tiempo de ciclo de forma considerable. Además, el escaneo del blíster vacío se realiza durante la cogida de la ventosa y se comprueba de forma inmediata antes de realizar la cogida.
- Se utilizan posiciones activas (Pounce) para aproximar el robot al punto de acción hasta que se cumplan todas las condiciones necesarias para iniciar el ciclo.
- En el programa, debido a limitaciones en la ejecución de la simulación, se utilizan esperas de tiempo (Waittime) que aumentan el tiempo de ciclo del programa, aunque dichas esperas se deberán eliminar en planta en caso de no ser necesarios u optimizarlos según necesidad de comunicación con el PLC.

Los resultados obtenidos se muestran en la hoja de cálculo generada por la rutina implementada.

TFG_JOAN_2021-03-08

NoPrograma	TipoPrograma	TipoDisco	NoBlister	TiempoCiclo	TiempocicloInit	
1	Rack 1	MEB	6	65.45	4.05	
1	Rack 1	MEB	5	62.12	0.00	
1	Rack 1	MEB	4	62.75	0.00	
1	Rack 1	MEB	3	63.83	0.00	
1	Rack 1	MEB	2	63.07	0.00	
1	Rack 1	MEB	1	62.80	0.00	
2	Rack 2	MEB	1	77.40	3.49	
1	Rack 1	MEB	1	67.94	3.02	

Tabla 3 Tiempo de ciclo de distintos programas obtenidos mediante la rutina *EscribirTCEnArchivo* desde RobotStudio en Numbers (MacOS)

Como se puede observar en la tabla obtenida se diferencian los tiempos de ciclo enmarcados con el corchete verde (correspondientes a TCM RACK 1) y los enmarcados con los corchetes azules (correspondientes a los TC más desfavorables de cada rack individualmente).

Mediante la *Ecuación 1* y los datos obtenidos (*Tabla 3*) enmarcados en verde, se muestra como se obtendría el TCM del Rack1.

$$\boldsymbol{TCM\ RACK1} = \frac{\frac{65.45 + 62.12 + 62.75 + 63.83 + 63.07 + 62.80}{4}}{6} + 4.05 = 19.88\ (\boldsymbol{S})$$

De manera análoga se podría obtener el TC para las condiciones más desfavorables de cada rack como se muestra a continuación:

$$\mathbf{TC_{Desf}\ RACK1} = \frac{\frac{67.94}{4}}{1} + 4.05 = 16.985 + 4.05 = 21.04\ (\boldsymbol{S})$$

$$\mathbf{TC_{Desf}\ RACK2} = \frac{\frac{77.40}{4}}{1} + 4.05 = 19.35 + 4.05 = 23.40\ (\boldsymbol{S})$$

Nota: La variación entre los TCInit se debe al procesamiento de la simulación y se considera despreciable puesto que en condiciones reales debería ser mínima. Por ello se utiliza el peor de estos para todos los cálculos.

Según los resultados obtenidos se puede concluir que el tiempo de ciclo para las condiciones más desfavorables del Rack2 es superior al correspondiente al Rack1. Aún así se cumplen las expectativas de tiempo de ciclo (tiempo de ciclo inferior a 24s). Teniendo en cuenta lo anterior se pueden establecer, de forma orientativa, el valor medio de sets de piezas que se podrían producir para cada rack en esta estación según los resultados de la Ecuación *1* para cada rack:

$$\mathbf{Número\ Piezas\ Rack\ 1} = \frac{7 \times 3600}{19.88} \times 3 = 3802\ {}^{piezas}/_{día} = \mathbf{1267}\ {}^{\boldsymbol{piezas}}/_{\boldsymbol{turno}}$$

$$\mathbf{Número\ Piezas\ Rack\ 2} = \frac{7 \times 3600}{22.11} \times 3 = 3419\ {}^{piezas}/_{día} = \mathbf{1139}\ {}^{\boldsymbol{piezas}}/_{\boldsymbol{turno}}$$

Nota: Se consideran 3 turnos de producción al día con 7 horas de trabajo efectivo (3 descansos de 20 minutos por turno).

Recordar que el útil de descarga hace funciones de buffer, pudiendo almacenar hasta 8 discos de cada modelo, por lo que la celda tiene un cierto tiempo de recuperación en caso de una parada en la misma sin afecta

7. CONCLUSIONES

Es importante sacar conclusiones sobre del trabajo realizado, afianzar conocimientos y seguir progresando. Por ello tras la finalización del proyecto se puede afirmar que se han estudiado, probado y valorado distintas opciones para cada ámbito, con el objetivo de elegir la opción que mejor se adapta a las necesidades del cliente.

El código del programa, que es la parte más importante y compleja del proyecto, se ha podido realizar offline, modificando, adaptando y mejorando el programa conforme se iba programando, gracias a la simulación de la estación en RobotStudio. Esto ha supuesto un ahorro extremadamente importante en cuanto a recursos: dinero, materiales y seguridad. También hay que destacar que gracias a lo anteriormente comentado el cliente puede valorar previamente el proyecto, lo que conlleva menos problemas burocráticos durante y después del proyecto ya que todo quedaría establecido de antemano.

La puesta en marcha de la estación será mucha más simple y rápida, siguiendo la distribución y dimensiones de la simulación, así como el Reglamento Técnico de Baja Tensión indicado en el apartado [8.1] de la Bibliografía. Una vez realizado el montaje de la misma solamente habría que cargar la OLP en el robot y ajustar mínimamente los objetos de trabajo y trayectorias. Basándose en los componentes inteligentes habría que programar el PLC y configurar las señales y comunicación con la estación y el robot. Existe la posibilidad de realizar la programación del PLC de forma offline a medida que se realiza la simulación en RobotStudio.

A nivel personal se ha conseguido profundizar mucho más en todos los campos necesarios para la realización del proyecto, desde la parte de modelado de geometrías utilizando Inventor hasta la totalidad de la simulación. Es fácil darse cuenta de la complejidad de los componentes inteligentes y de la programación del código RAPID. Es verdad que RobotStudio es el software más potente actualmente para la programación de código de todas las marchas de robots del mercado, aunque se valora para futuros trabajos la posibilidad de realizar la simulación de la estación en algún programa diferente, específico para simulación como puede ser Process Simulate.

Económicamente es un proyecto viable pues el presupuesto es asumible en relación a los beneficios que conlleva, relacionados con el diseño, simulación y programación. Se puede ver con detalle dicho presupuesto en el Documento Nº4 y en el apartado *[7.1] Apuntes de Teoría de Proyectos* de la Bibliografía.

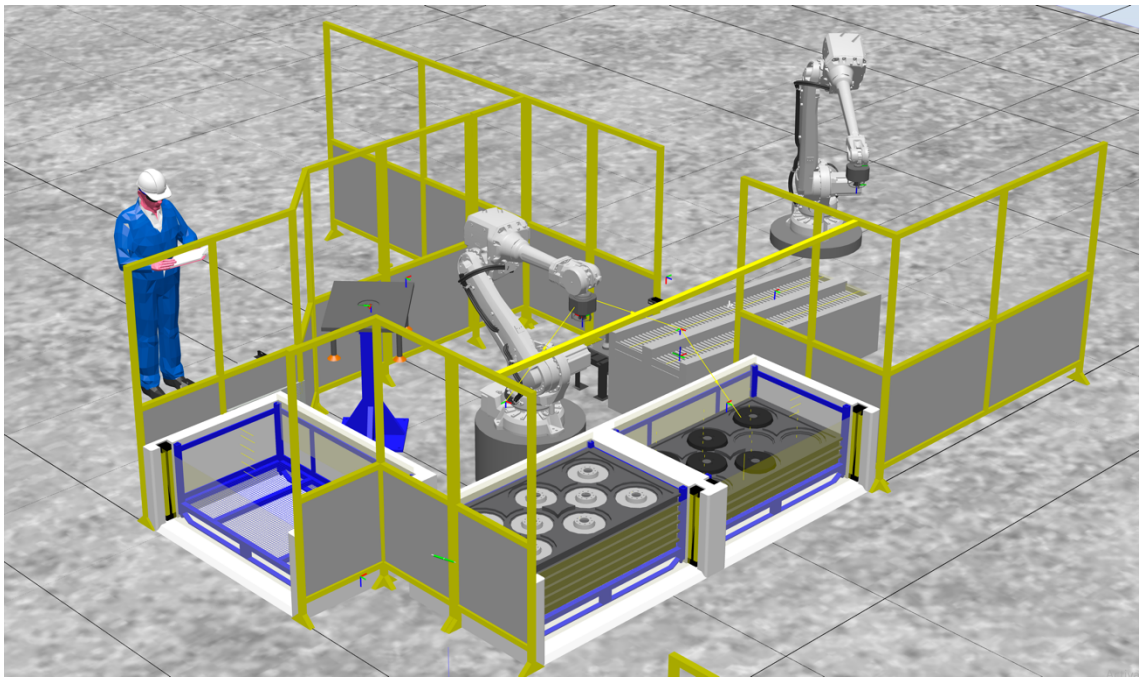


Figura 20.1 Captura de la celda en RobotStudio

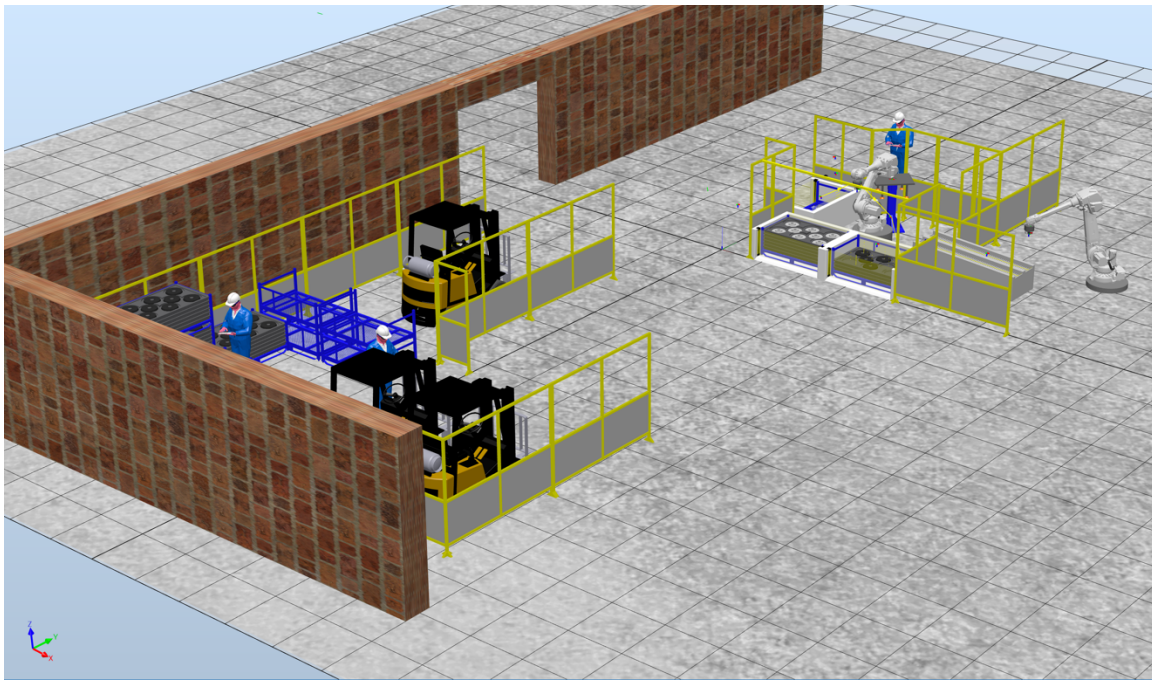


Figura 20.2 Captura de la estación en RobotStudio I

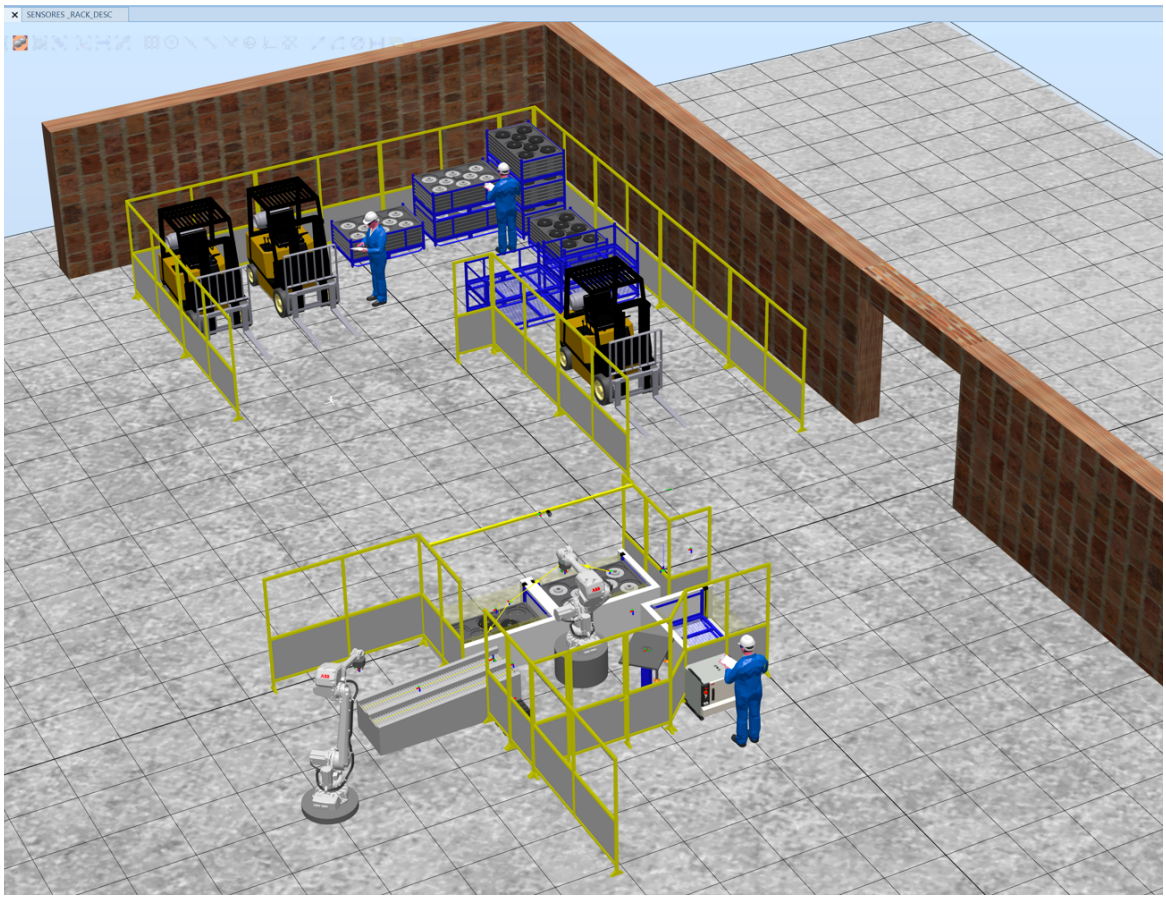


Figura 20.3 Captura de la estación en RobotStudio II

8. BIBLIOGRAFÍA

Documentación técnica IRB4600 y Controlador IRC5

[1.1] *ABB en España*. New.abb.com. (2021). Último acceso abril 2021, https://new.abb.com/es.

Documentación técnica elementos externos

[2.1] Documentación técnica elementos externos de SICK. *SICK Spain | SICK*. Sick.com. (2021). Último acceso enero 2021, https://www.sick.com/es/en/.

[2.2] Documentación técnica elementos externos de WENGLOR GmbH. *the innovative family*. Wenglor.com. (2021). Último acceso enero 2021, https://www.wenglor.com/es/.

Programación y Código RAPID

[3.1] Código y funciones RAPID - Manuales de Ayuda de RobotStudio. (2019). ABB España. RobotStudio.

[3.2] *Lenguaje RAPID*. Personal.biada.org. (2021). Último acceso enero 2021, http://personal.biada.org/~jhorrillo/INTRODUCCIO%20RAPID.pdf.

[3.3] *Manual del Operador RobotStudio*. Library.e.abb.com. (2021). Último acceso enero 2021, https://library.e.abb.com/public/6aeb483836740e11c1257b4b0052375b/3HAC032104-005_revE_es.pdf.

Simulación de la estación

[4.1] *Tutorial del Proyecto Final de curso ROBOT-STUDIO*. Youtube.com. (2021). Último acceso mayo 2020, https://www.youtube.com/watch?v=-T8YE9S8bHg&t=11672s.

[4.2] Martín Castillo, J. (2021). *Canal Youtube*. Youtube.com. Último acceso mayo 2020, https://www.youtube.com/user/jcmcastillo.

Modelado en Inventor

[5.1] Apuntes de Teoría de Ingeniería Gráfica 4º. (2018). Grupo V. Ferrán Naya.

[5.2] Inventor – Generador de Planos. (2018). Nuria Aleixos.

Geometrías

[6.1] *GrabCAD Community*. (2021). Último acceso mayo 2020, https://grabcad.com/.

[6.2] Librerías ABB de RobotStudio. (2019). ABB España. RobotStudio

Presupuesto

[7.1] Apuntes de Teoría de Proyectos de 4º. (2018). Grupo V.

Reglamento Técnico Baja Tensión

[8.1] *BOE.es - BOE-A-2002-18099 Real Decreto 842/2002, de 2 de agosto, por el que se aprueba el Reglamento electrotécnico para baja tensión.*. Boe.es. (2021). Último acceso marzo 2021, https://www.boe.es/eli/es/rd/2002/08/02/842.

PLANOS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

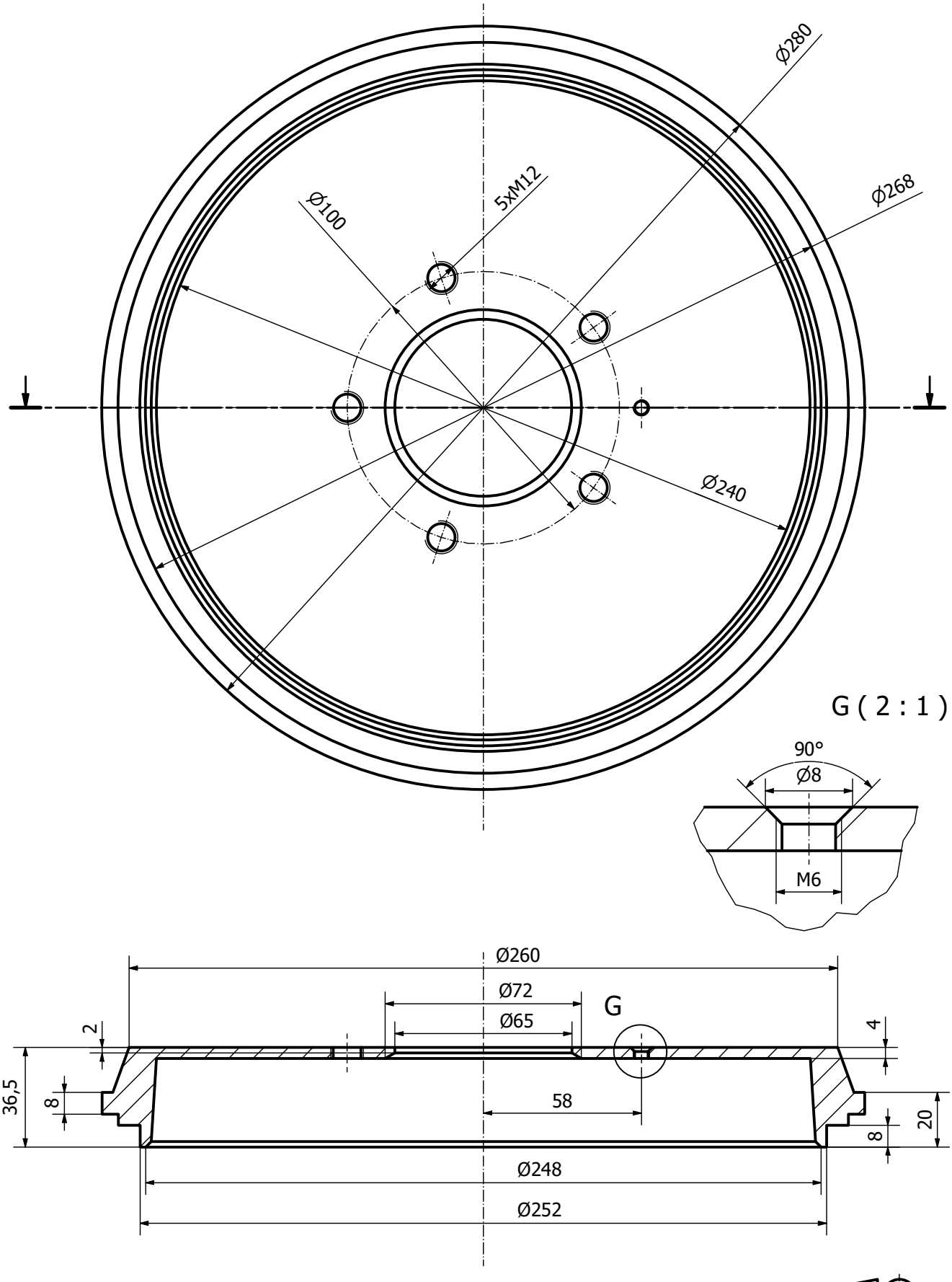
PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

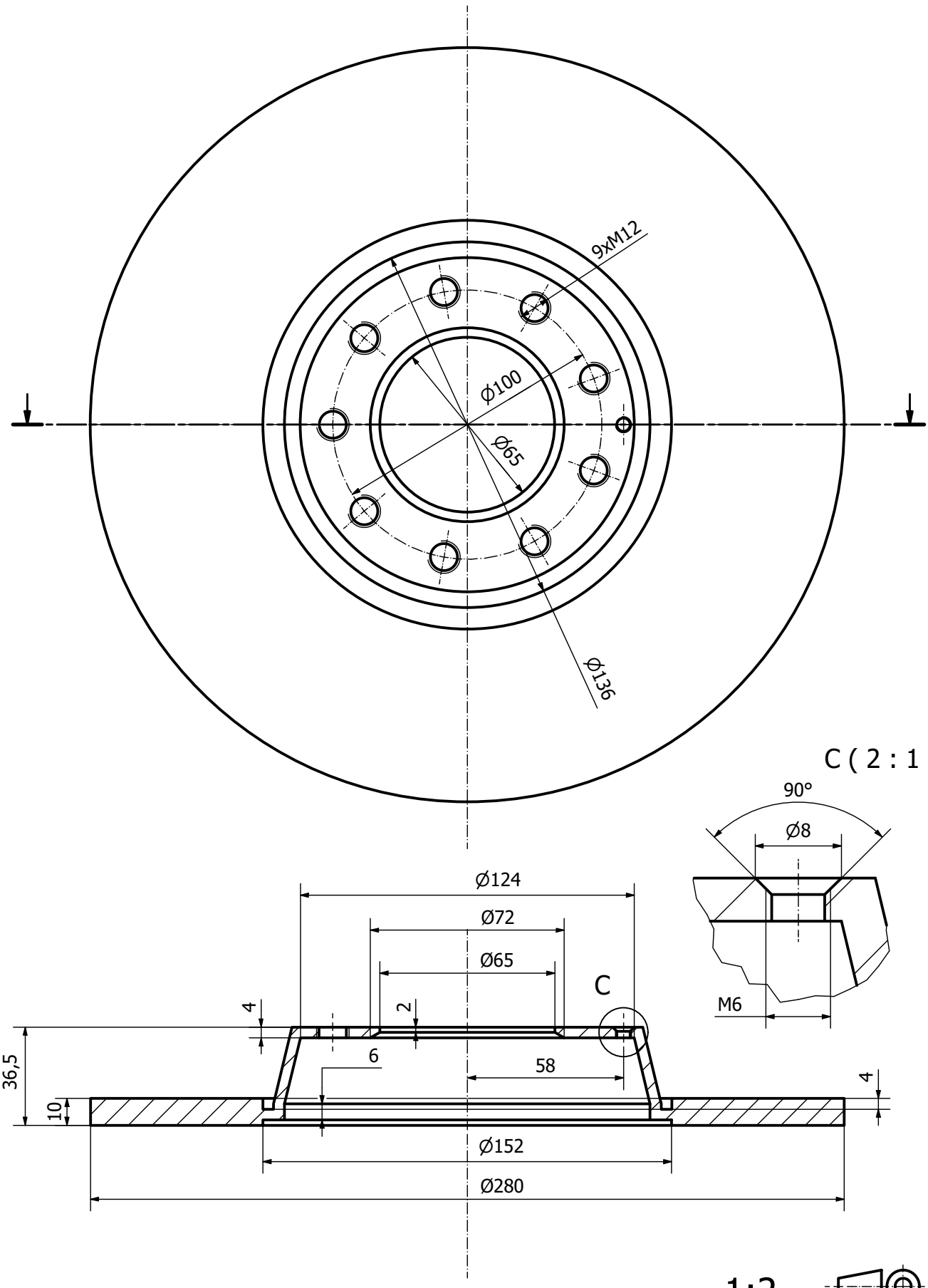
ÍNDICE DOCUMENTO Nº2: PLANOS



G (2 : 1)

Escala: 1:2

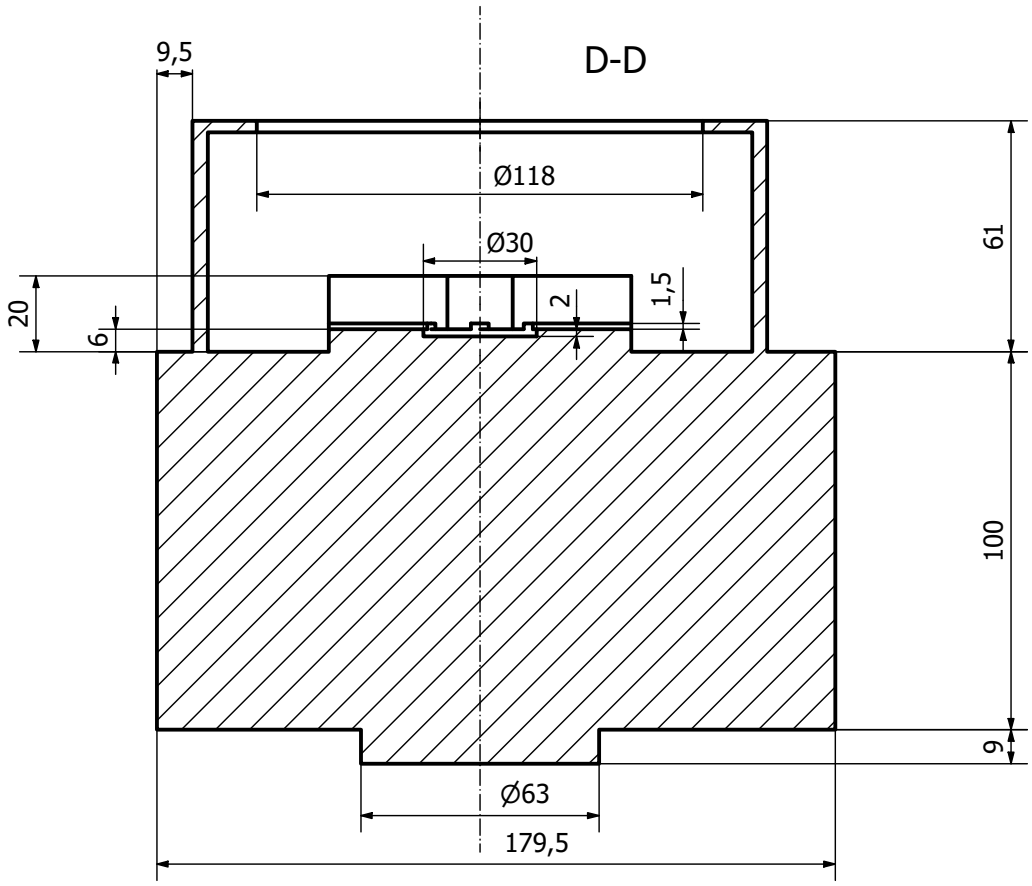
Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 1	Título del plano: MEB

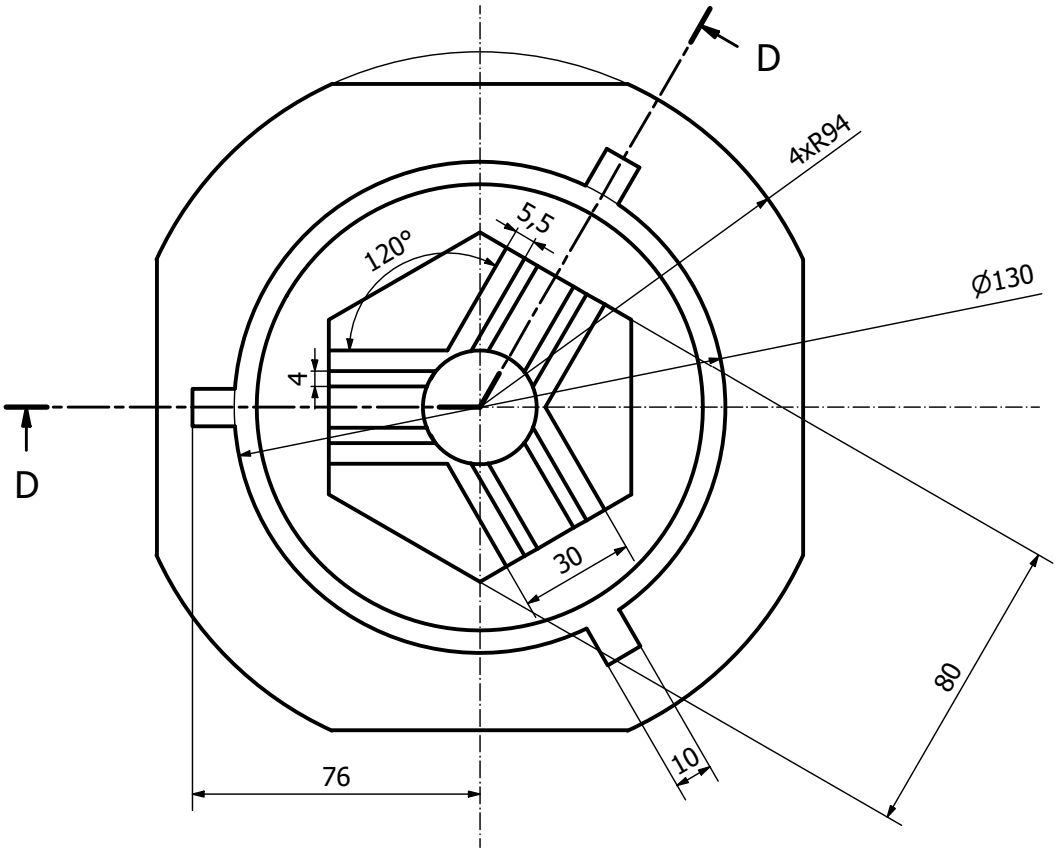


C (2 : 1)

Escala: 1:2

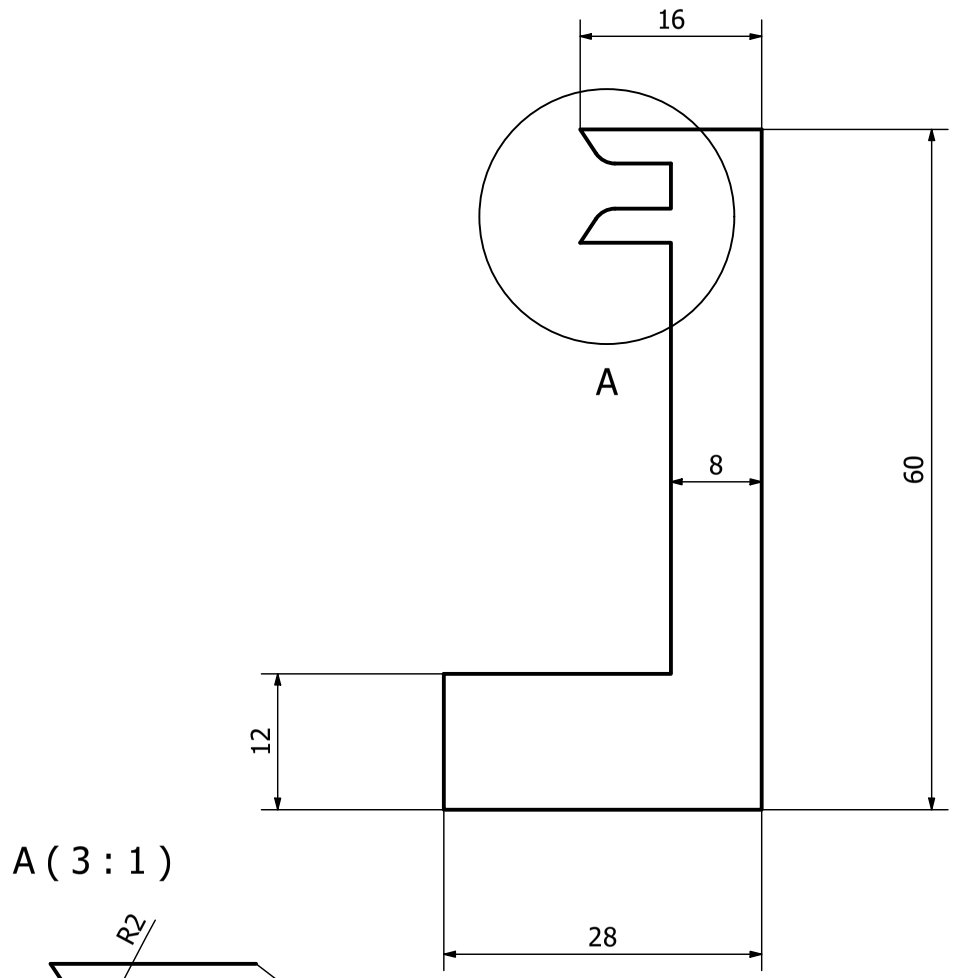
Creado por: Joan Calvo Herrero	Título: MQB	Tipo de documento: Plano de detalle
Aprobado por:		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de hoja: 2	Propietario legal: Joan Calvo Herrero

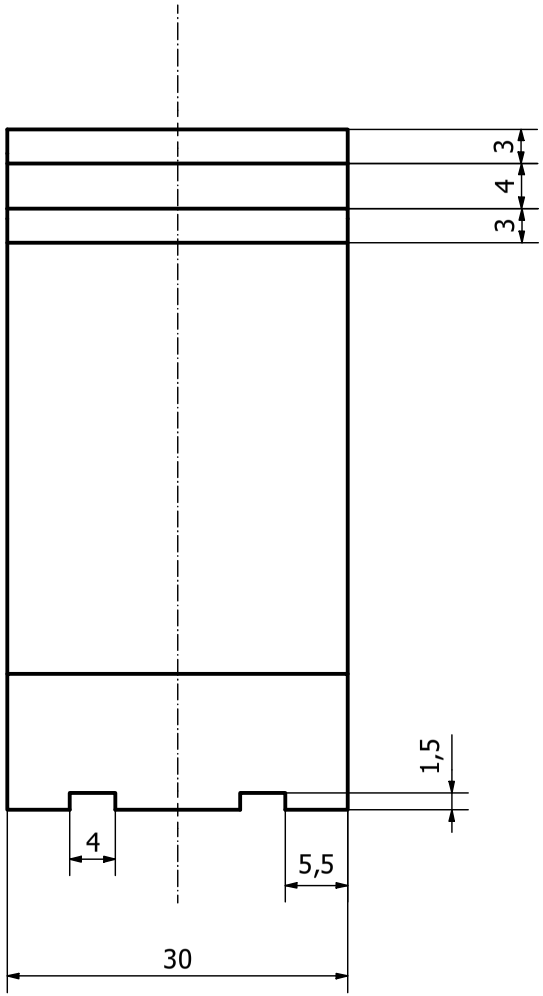




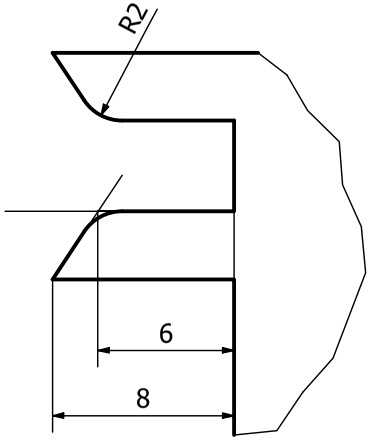
Escala: 1:2

Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 3	Título del plano: BASE MANIPULADOR



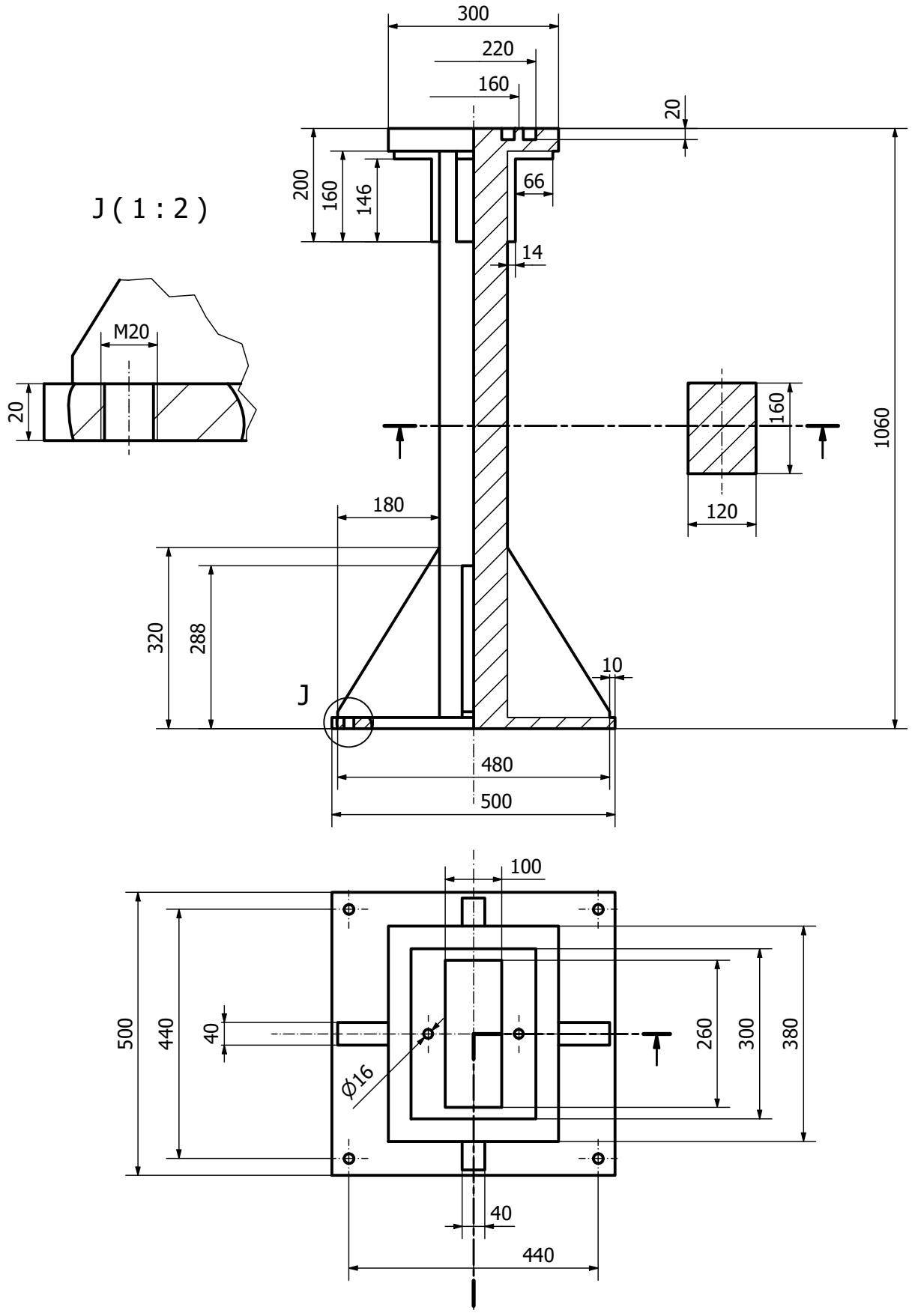


A (3 : 1)



Escala: 3:2

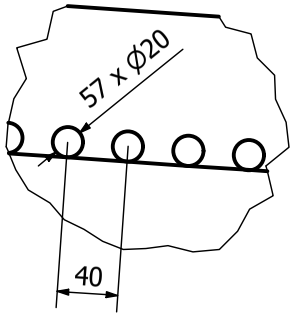
Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 4	Título del plano: BRIDA

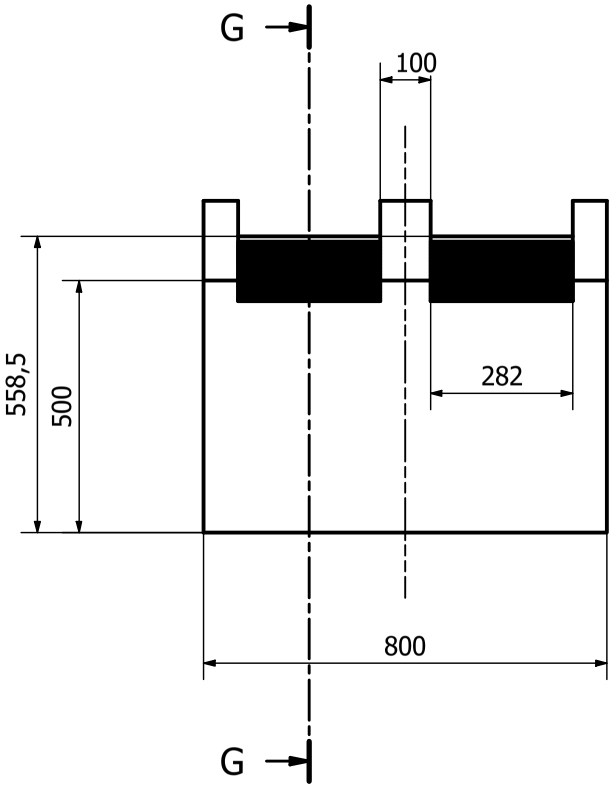


Escala: 1:10

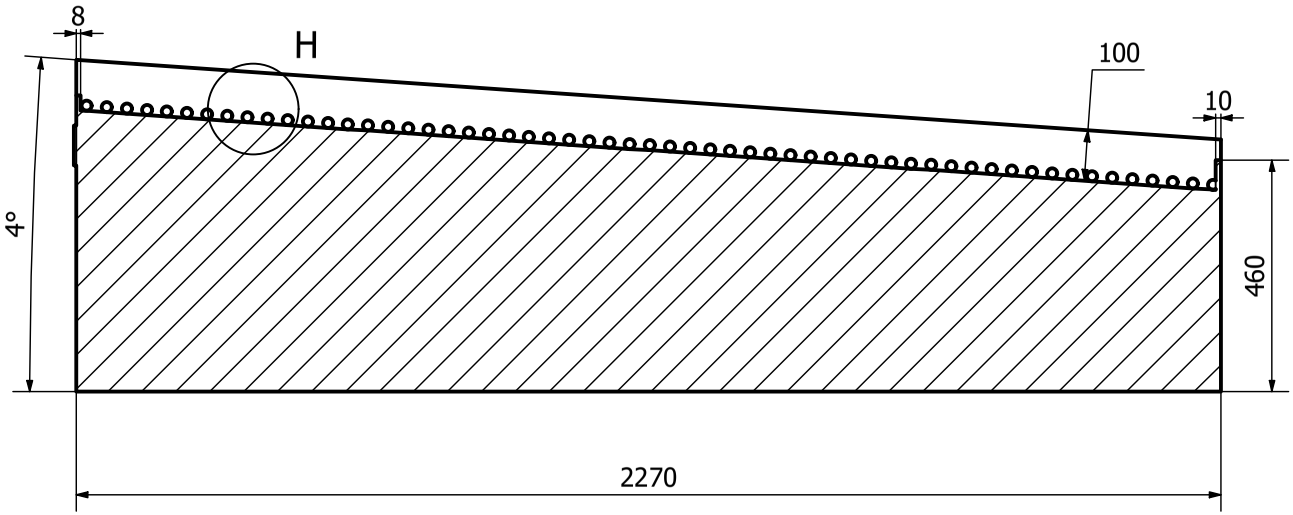
Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 5	Título del plano: TOOL STAND

H (1:5)





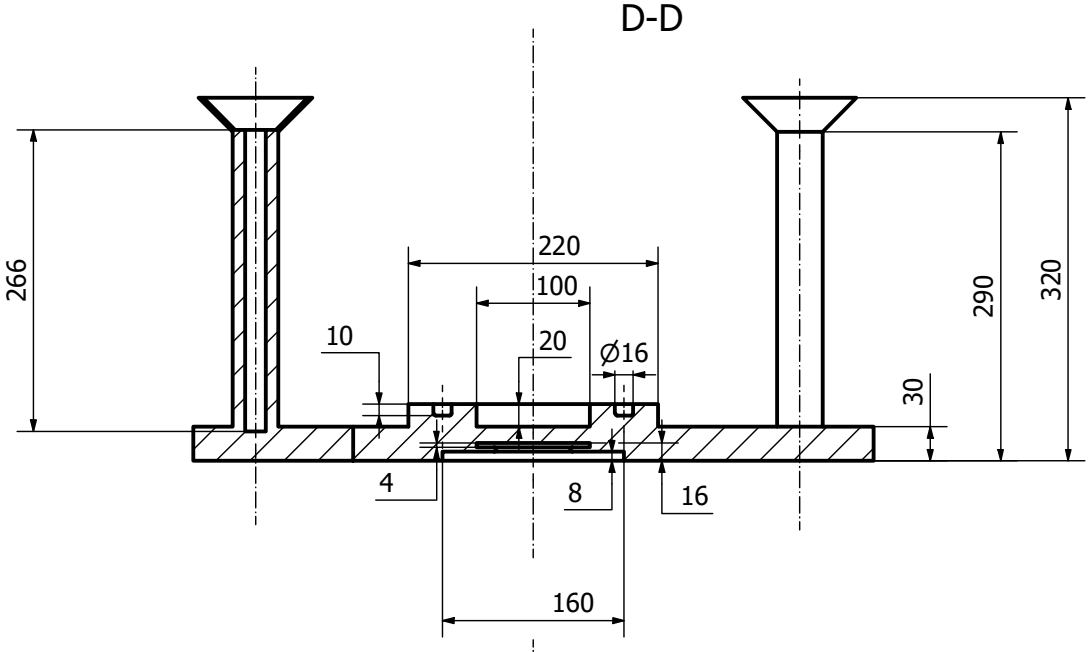
G-G

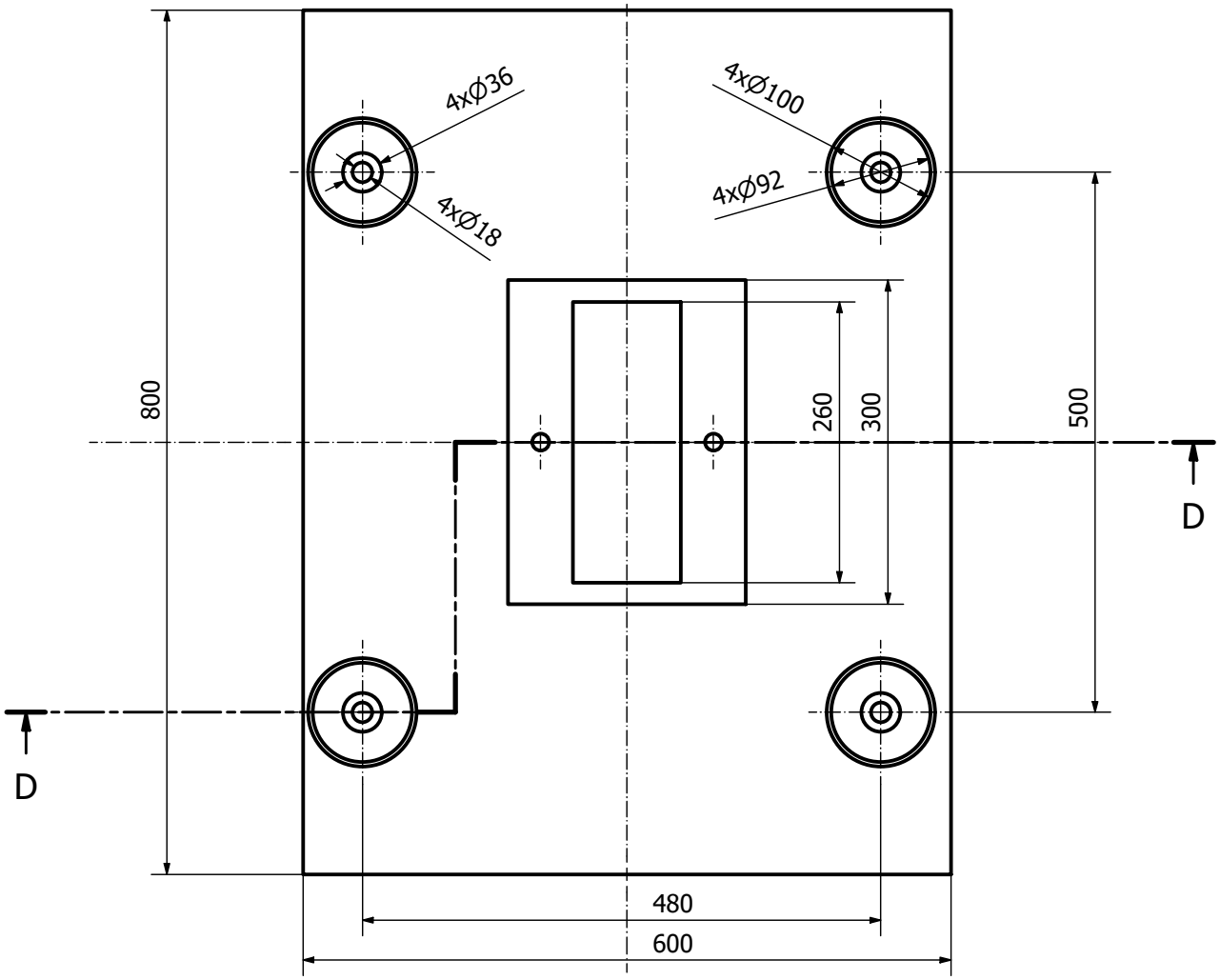


Escala: 1:15

Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 6	Título del plano: ÚTIL DESCARGA

D-D

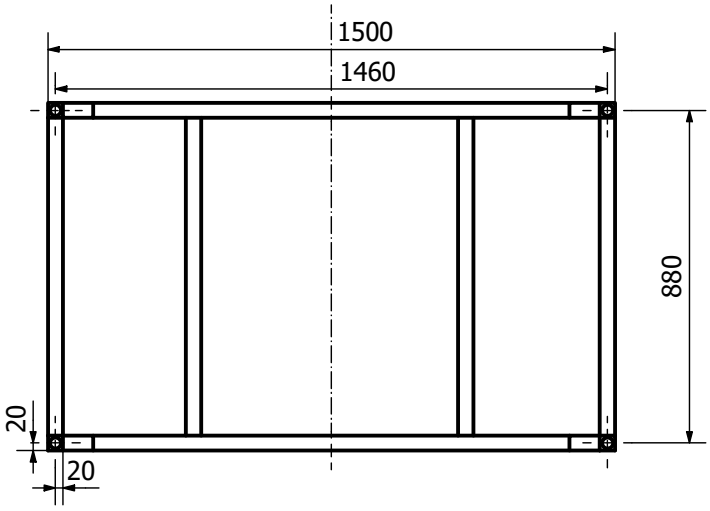




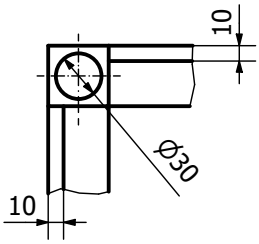
Escala: 3:20

Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 7	Título del plano: VENTOSA

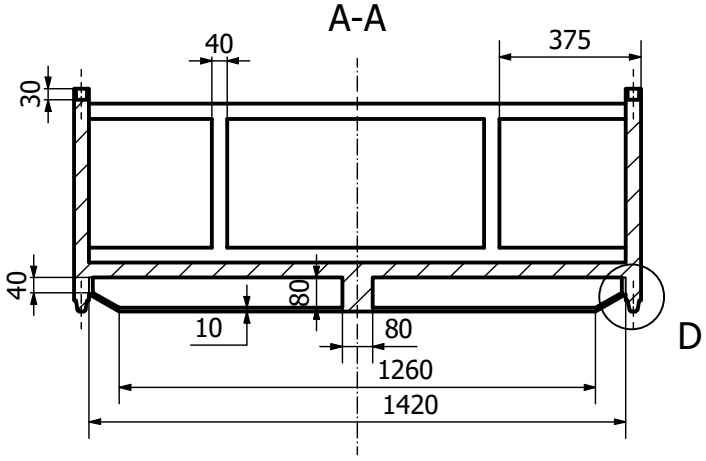
En el plano no se ha tenido en cuenta la rejilla de la base

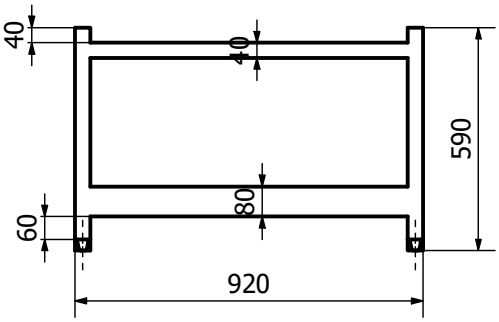


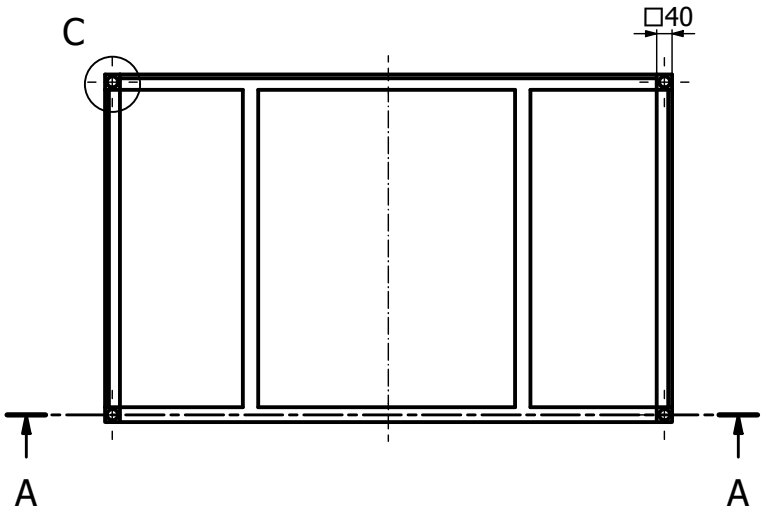
C (1 : 5)



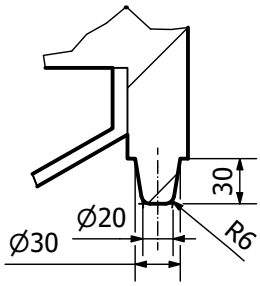
A-A





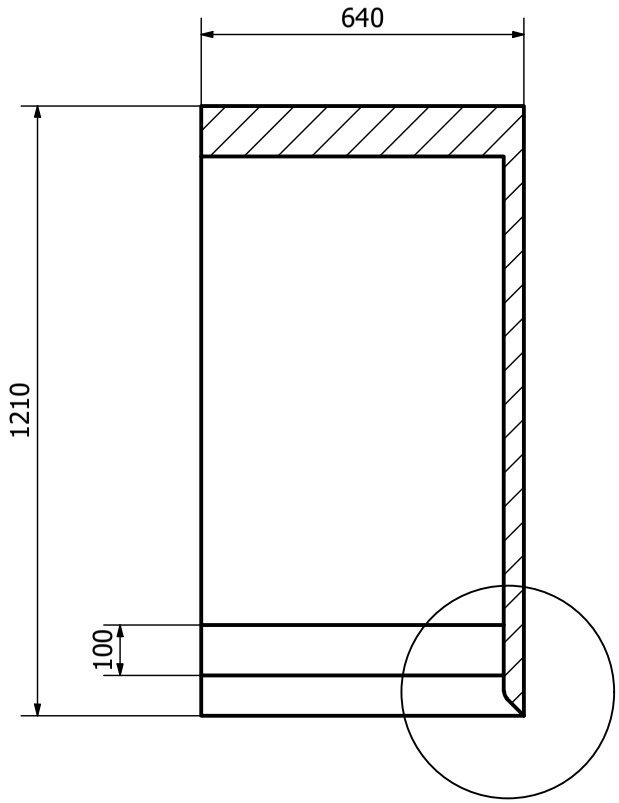


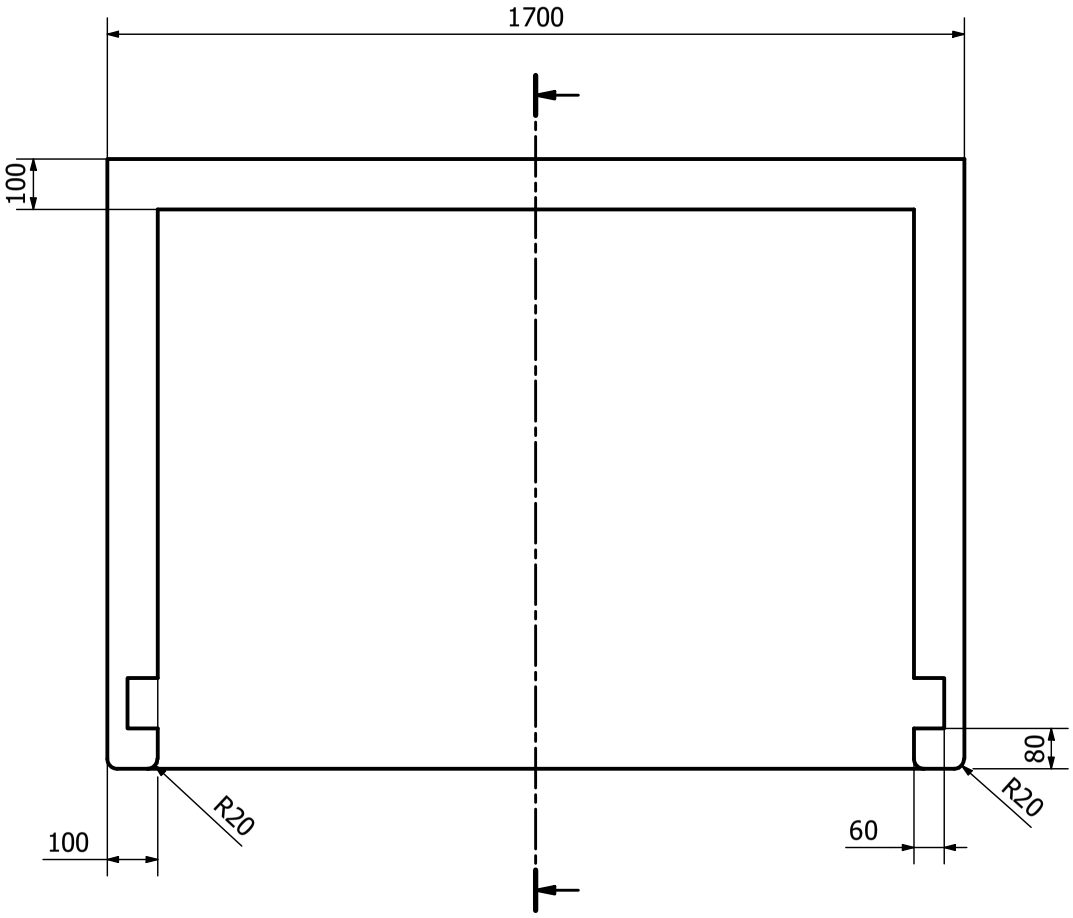
D (1 : 5)



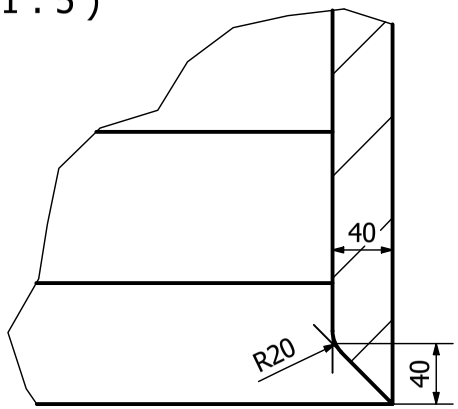
Escala: 1:20

Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 8	Título del plano: RACK





G (1 : 5)



G

Escala: 1:15

Autor proyecto: Joan Calvo Herrero	Proyecto: Proyecto de automatización de una línea de producción de discos de freno mediante la programación en RobotStudio de un robot industrial ABB modelo IRB4600_60_205.	Tipo de documento: Plano de detalle
Tutor proyecto: José Vicente Salcedo Romero de Ávila		Nº de identificación: 33571273-J
Fecha de edición: 11/02/2021	Nº de plano: 9	Título del plano: SOPORTE RACK

PLIEGO DE CONDICIONES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº3: PLIEGO DE CONDICIONES

1. CONDICIONES Y LIMITACIONES

La línea de producción sobre la cual vamos a realizar el proyecto se encuentra actualmente en producción por lo que el diseño de la celda que vamos a implementar para el robot debe adaptarse a la distribución actual sin alterar los elementos existentes en la línea. Además, por requerimiento del cliente, se debe mantener el útil de descarga manual ahora utilizado.

Los elementos utilizados tales como piezas, blisters, racks y soportes son proporcionados por el cliente, manteniendo las medidas actuales. Por ello el modelado de los sólidos se hace basándose en las piezas originales. No obstante, el resto de piezas como la ventosa, manipulador y útil de descarga se han diseñado para adaptarse a las necesidades del nuevo proyecto.

Según las medidas proporcionadas por el cliente contamos con un espacio total de 46,6 m^2 según se muestra en el plano de la *Figura 10.1*

Los elementos externos utilizados deberán estar disponibles dentro del catálogo de productos de las marcas subcontratadas para el proyecto.

Para continuar con el volumen de producción de la línea se debe estudiar el tiempo de ciclo de la nueva estación, siendo necesario mantener o reducir el mismo. El tiempo de ciclo de la estación debe ser inferior a 24s.

Hay que tener en cuenta que, aunque el proyecto es en una factoría de Volkswagen, no se utiliza el estándar VASS del grupo en esta línea. Por ello para realizar el código del programa se ha seguida la estructura del código del resto de robots de la línea ya programados.

Al tratarse de un anteproyecto de simulación no existen muchas normas que se deben cumplir durante la realización del trabajo, aunque sí se deberán en el montaje e instalación en planta.

En el diseño de la celda se tiene en cuenta que ésta quede totalmente cerrada e inaccesible por el vallado de seguridad y que se detenga inmediatamente el robot al traspasar cualquier barrera de seguridad incumpliendo las condiciones de correcto funcionamiento.

2. REGLAMENTO TÉCNICO DE BAJA TENSIÓN

Según el Real Decreto 842/2002, de 2 de agosto, por el que se aprueba el Reglamento Técnico para baja tensión, se deberán tener en cuenta las instrucciones técnicas complementarias (ITC) especificadas en el mismo y garantizar el total cumplimiento de las mismas. Para nuestro caso de estudio, se debe prestar especial atención a la instrucción ITC-BT-51 que hace referencia a las Instalaciones de Sistemas de Automatización.

Dicho reglamento tiene por objetivo el cumplimiento de las condiciones técnicas y garantías en instalaciones eléctricas conectadas a una fuente de suministro dentro de los límites establecidos de baja tensión para:

- Asegurar el correcto funcionamiento de instalaciones eléctricas sin que afecte o perturbe otras instalaciones o servicios.
- Garantizar la seguridad de las personas y bienes.
- Contribuir a la fiabilidad técnica y eficiencia económica de las instalaciones.

Se puede acceder al **Reglamento Técnico de Baja Tensión** a través del siguiente link:

https://www.boe.es/eli/es/rd/2002/08/02/842.

PRESUPUESTO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº4: PRESUPUESTO

1.INTRODUCCIÓN

En este capítulo se desarrolla el presupuesto del proyecto en el que se incluyen los costes de todos los elementos que intervienen, prorrateados según la duración del mismo.

Se debe tener en cuenta que el trabajo es un anteproyecto de adaptación de una línea de producción ya existente mediante el estudio por simulación. Es por ello que en el presupuesto no se reflejan los elementos que se incorporarán en el proyecto (robot, vallado, sensores, elementos eléctricos, honorarios de trabajadores en planta etc.) sino simplemente los costes de trabajo e investigación para el anteproyecto.

Se ha presupuestado como un proyecto de 3 meses de trabajo durante 8 horas diarias con una media de 21 días laborables mensual. El proyecto ha sido realizado por un único ingeniero industrial.

2.ESTRUCTURA DEL PRESUPUESTO

2.1 CUADRO Nº1: MANO DE OBRA

En este cuadro se tiene en cuenta el precio medio por hora de un **Ingeniero en Tecnologías Industriales**.

CÓDIGO	UNIDAD	DESCRIPCIÓN	IMPORTE (€/Unidad)
MO.ING	h	Ingeniero en Tecnologías Industriales	18

2.2 CUADRO Nº2: MATERIALES

En este cuadro se tienen en cuenta todos los materiales utilizados en el proyecto que incluyen elementos del hardware y software.

HARDWARE:

Para la realización del proyecto se ha utilizado un ordenador portátil ***MacBook Air 16GB*** al cual se le realiza un uso de aproximadamente 2016 horas anuales de media. Se estima una vida útil para este ordenador de unos 6 años.

$$Amortización\ M.OP = \frac{\text{Base Amortización (€)}}{\text{Vida Útil (h)}} = \frac{1859\ €}{6\ (\text{años}) \cdot 2016\ \frac{h}{años}} = 0{,}154\ €/h$$

También se ha utilizado un monitor portátil ***ASUS MB169B+*** valorado en 169€ y con una vida útil de aproximadamente 4 años.

$$Amortización\ M.MP = \frac{\text{Base Amortización (€)}}{\text{Vida Útil (h)}} = \frac{169\ €}{4\ (\text{años}) \cdot 2016\ \frac{h}{años}} = 0{,}021\ €/h$$

SOFTWARE:

En esta parte se tienen en cuenta el coste de las licencias de los programas utilizados calculado de forma prorrateada a la duración del proyecto.

Es necesario una máquina virtual para utilizar programas con sistema operativo Windows en Mac. En este caso se ha utilizado ***Parallels Desktops 16***. El coste de la licencia es de 79,99 € anuales.

$$Coste\ M.PD = \frac{\text{Coste Anual (€)}}{\text{Duración (h)}} = \frac{79{,}99\ (€)}{1\ (\text{años}) \cdot 2016\ (\frac{h}{años})} = 0.040\ €/h$$

El coste de la licencia de ***ABB RobotStudio 2019*** es de 1020€ al que habría que añadirle 504€ por cada paquete de convertidores ya que estos no se incluyen en la licencia. En nuestro caso necesitaremos un paquete para convertir archivos CAD.

$$Coste\ M.RS = \frac{\text{Coste Anual (€)}}{\text{Duración (h)}} = \frac{1020 + 504\ (€)}{1\ (\text{años}) \cdot 2016\ (\frac{h}{años})} = 0{,}756€/h$$

El coste de la licencia de ***Autodesk Inventor 2021*** es de 2747€ al año.

$$Coste\ M.INV = \frac{\text{Coste Anual (€)}}{\text{Duración (h)}} = \frac{2747\ (€)}{1\ (\text{años}) \cdot 2016\ (\frac{h}{años})} = 1{,}363\ €/h$$

CÓDIGO	UNIDAD	DESCRIPCIÓN	IMPORTE (€/Unidad)
M.RS	h	Licencia de RobotStudio	0,756
M.INV	h	Licencia de Inventor	1,363
M.OP	h	Ordenador Portátil MacBook Air	0,154
M.MP	h	Monitor portátil ASUS	0,021
M.PD	h	Parallels Desktops 16	0,040

2.3 CUADRO Nº3: PRECIOS UNITARIOS

CÓDIGO	UNIDAD	DESCRIPCIÓN	IMPORTE (€/Unidad)
UD.01	ud	Estudio y viabilidad del proyecto	290,10
UD.02	ud	Diseño y modelado de geometrías en Inventor	938,73
UD.03	ud	Implementación de la estación en RobotStudio	3035,36
UD.04	ud	Programación del código RAPID	2276,52
UD.05	ud	Redacción del proyecto	1860,68

2.4 CUADRO Nº4: PRECIOS DESCOMPUESTOS

CÓDIGO	UNIDAD	DESCRIPCIÓN	RENDIMIENTO	PRECIO	IMPORTE
MO.ING	h	Ingeniero en Tecnologías Industriales	16	18	288
M.OP	h	Ordenador portátil MacBook Air	12	0,154	1,85
M.MP	h	Monitor portátil ASUS	12	0,021	0,25
UD.01	**ud**	**Estudio y viabilidad del proyecto**			**290,1**

CÓDIGO	UNIDAD	DESCRIPCIÓN	RENDIMIENTO	PRECIO	IMPORTE
MO.ING	h	Ingeniero en Tecnologías Industriales	48	18	864
M.OP	h	Ordenador portátil	48	0,154	7,39
M.INV	h	Software Inventor	48	1,363	65,42
M.PD	h	Parallels Desktops 16	48	0,040	1,92
UD.02	**ud**	**Diseño y modelado en Inventor**			**938,73**

CÓDIGO	UNIDAD	DESCRIPCIÓN	RENDIMIENTO	PRECIO	IMPORTE
MO.ING	h	Ingeniero en Tecnologías Industriales	160	18	2880
M.OP	h	Ordenador portátil MacBook Air	160	0,154	24,64
M.MP	h	Monitor portátil ASUS	160	0,021	3,36
M.RS	h	Software RobotStudio	160	0,756	120,96
M.PD	h	Parallels Desktops 16	160	0,040	6,40
UD.03	**ud**	**Implementación en RobotStudio**			**3035,36**

CÓDIGO	UNIDAD	DESCRIPCIÓN	RENDIMIENTO	PRECIO	IMPORTE
MO.ING	h	Ingeniero en Tecnologías Industriales	120	18	2160
M.OP	h	Ordenador portátil MacBook Air	120	0,154	18,48
M.MP	h	Monitor portátil ASUS	120	0,021	2,52
M.RS	h	Software RobotStudio	120	0,756	90,72
M.PD	h	Parallels Desktops 16	120	0,040	4,80
UD.04	**ud**	**Programación del código RAPID**			**2276,52**

CÓDIGO	UNIDAD	DESCRIPCIÓN	RENDIMIENTO	PRECIO	IMPORTE
MO.ING	h	Ingeniero en Tecnologías Industriales	100	18	1800
M.OP	h	Ordenador portátil MacBook Air	100	0,154	15,40
M.MP	h	Monitor portátil ASUS	100	0,021	2,10
M.RS	h	Licencia RobotStudio	20	0,756	15,12
M.INV	h	Software Inventor	20	1,363	27,26
M.PD	h	Parallels Desktops 16	20	0,040	0,80
UD.05	**ud**	**Redacción del proyecto**			**1860,68**

3. PRESUPUESTO BASE DE LICITACIÓN

	DESCRIPCIÓN	IMPORTE
UD.01	Estudio y viabilidad del proyecto	290,10 €
UD.02	Diseño y modelado de geometrías en Inventor	938,73 €
UD.03	Implementación de la estación en RobotStudio	3035,36 €
UD.04	Programación del **c**ódigo RAPID	2276,52 €
UD.05	Redacción del proyecto	1860,68 €

PRESUPUESTO DE EJECUCION MATERIAL	8401,39 €
Gastos Generales (13%)	1092,18 €
Beneficio Industrial (6%)	504,08 €
PRESUPUESTO TOTAL	9997,65 €
I.V.A. (21%)	2099,51 €
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	**12097,16 €**

Asciende el presente presupuesto a la expresada cantidad de:
DOCE MIL NOVENTA Y SIETE EUROS CON DIECISÉIS CÉNTIMOS

ANEXO I: DATASHEETS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº5: ANEXO I: DATASHEETS

ROBOTICS

IRB 4600
Industrial Robot

IRB 4600 is a highly productive general purpose robot optimized for short cycle times where compact robots can help create high density cells. The IRB 4600 enables more compact manufacturing cells with increased production output and higher quality - and that means improved productivity.

Shortest cycle times
Thanks to the new compact and optimized design resulting in a low weight, the IRB 4600 can cut the cycle times of the industry benchmark by up to 25%. The maximum acceleration achievable is highest in its class, together with high maximum speeds. With the high acceleration it is possible to use to avoid obstacles or to follow the path. The benefit is increased production capacity and higher productivity.

Ultra-wide working range
You can position the IRB 4600 in the most favourable way with regard to reach, cycle time and auxiliary equipment. Flexible mounting with floor, tilted, semi-shelf or inverted mounting is very useful when you are simulating the best position for your application.

Compactness
The small footprint, the slim swing base radius around axis 1, the fine elbow behind axis 3, the small lower and upper arms, and the compact wrist all contribute to the most compact robot in its class. With the IRB 4600 you can create your production cell with reduced floorspace by placing the robot closer to the served machines, which also increases your output and your productivity.

Best protection available
ABB has the most comprehensive protection program on the market and it will be even further enhanced with the IRB 4600. Foundry Plus includes IP 67, resistant paint, rustprotected mounting flange and protection for molten metal spits on non-moving cables on the rear of the robot and extra protection plates over the floor cable connections on the foot.

Optimize and go sharp
To get the IRB 4600 ready for the targeted applications you have access to high performing workpiece positioners, track motions, and the motor and gear unit range.

To simulate your production cell to find the optimal position for the robot and program it offline, RobotStudio is available on subscription together with PowerPacs for several applications.

Learn more about how to use the IRB 4600 in your applications and environments - watch simulations on several applications at www.abb.com/robotics.

Main applications
- Arc Welding
- Assembly
- Material Handling
- Machine Tending
- Material Removal
- Cleaning/Spraying
- Dispensing
- Packing
- Laser Cutting
- Laser Welding

Specification

Robot version	Reach (m)	Payload (kg)	Armload (kg)
IRB 4600-60/2.05	2.05	60	20
IRB 4600-45/2.05	2.05	45	20
IRB 4600-40/2.55	2.55	40	20
IRB 4600-20/2.50	2.51	20	11
Number of axes	6+3 external (up to 36 with MultiMove)		
Protection	Standard IP67, as option Foundry Plus 2		
Mounting	Floor, shelf, inverted or tilted		
Controller	IRC5 Single cabinet		

Performance (according to ISO 9283)

	Position repeatability	Path repeatability*
IRB 4600-60/2.05	0.06 mm	0.46 mm
IRB 4600-45/2.05	0.05 mm	0.13 mm
IRB 4600-40/2.55	0.06 mm	0.28 mm
IRB 4600-20/2.50	0.05 mm	0.17 mm

*Measured at speed 250 mm/s.

Technical information

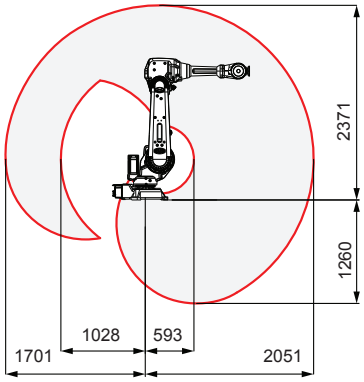
Electrical Connections	
Supply voltage	200-600 V, 50-60 Hz
Physical	
Robot base	512 x 676 mm
Height	
IRB 4600-60/2.05	1727 mm
IRB 4600-45/2.05	1727 mm
IRB 4600-40/2.55	1922 mm
IRB 4600-20/2.50	1922 mm
Robot weight	
IRB 4600-60/2.05	425 kg
IRB 4600-45/2.05	425 kg
IRB 4600-40/2.55	435 kg
IRB 4600-20/2.50	412 kg
Environment	
Ambient temperature for mechanical unit	
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%
Safety	Double circuits with supervisions, emergency stops and safety functions. 3-position enable device
Emission	EMC/EMI shielded

Data and dimensions may be changed without notice.

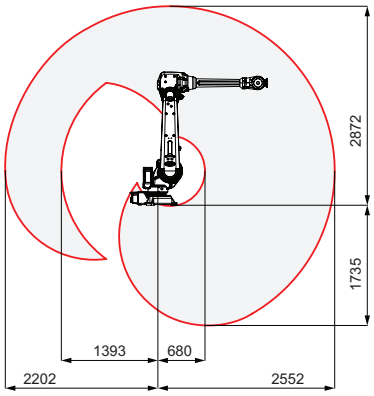
Movement

Axis movement	Working range	Axis max. speed
Axis 1 rotation	+180° to -180°	175°/s
Axis 2 arm	+150° to -90°	175°/s
Axis 3 arm	+75° to -180°	175°/s
Axis 4 wrist	+400° to -400°	250° (360° for IRB 4600-20/2.50)
Axis 5 bend	+120° to -125°	250° (360° for IRB 4600-20/2.50)
Axis 6 turn	+400° to -400°	360° (500° for IRB 4600-20/2.50)

Working range, IRB 4600-60/2.05, IRB 4600-45/2.05



Working range, IRB 4600-40/2.55



Working range, IRB 4600-20/2.50

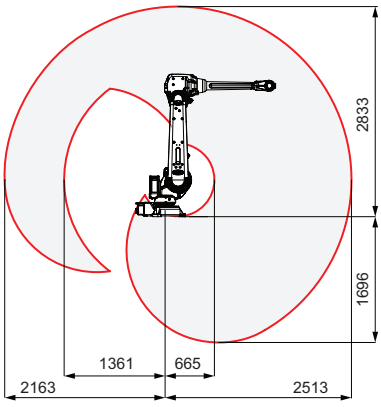


abb.com/robotics

ROB0109 EN Rev.K January 2019

ROBOTICS

IRC5
Industrial Robot Controller

Based on more than four decades of robotics experience, the IRC5 is the robotic industry's benchmark in robot controller technology. In addition to ABB's unique motion control it brings flexibility, safety, modularity, application interfaces, multi-robot control and PC tool support. The IRC5 comes in different variants to provide a cost-effective and optimized solutions for every need.

Fast and accurate
The IRC5 gives our robots the ability to perform their tasks in a highly efficient manner. Based on advanced dynamic modelling, the IRC5 automatically optimizes the performance of the robot by reducing cycle times (QuickMove®) and providing precise path accuracy (TrueMove®). Thanks to ABB's IRC5 technology, a robot's motion is predictable and its performance high, with no tuning required by the programmer. What you program is what you get.

Safe
Operator safety is a leading benefit of the IRC5. It fulfills all relevant regulations and is certified by third party inspectors worldwide.

SafeMove2 represent a new generation of safety, enabling more flexible robotic cell safety concepts, e.g. enabling floor space reduction and collaboration between robot and humans.

Compatible
No matter where in the world your robot is located, and regardless of what regulatory standards apply, the IRC5 is up to the task. ABB's controller is compatible with various types of main voltages and can handle a broad spectrum of environmental conditions. IRC5 communicates with other machines in a manufacturing environment; in a safe and predictable way. It supports the majority of all state-of-the-art industrial networks for I/O. Sensor interfaces, remote access and a rich set of programmable interfaces are examples of the IRC5's many powerful networking features.

Programmable
All ABB robot systems are programmed with RAPID™, ABB's flexible, high-level programming language. On the surface RAPID's basic features and functionality are easy to use, but dig deeper and you will find that this programming language allows you to create highly sophisticated solutions. It is a truly universal language on and off the shop floor which supports structured programs, and advanced features. It also incorporates powerful support for the most common robot process applications such as welding and assembly.

Reliable
The IRC5 is practically maintenance free, and its outstanding quality ensures unmatched up-time. Built-in diagnostic functions help ensure fast recovery and production restarts when operations are interrupted on the factory floor.

The IRC5 also comes equipped with remote monitoring technology, ABB Ability Connected Services. Advanced diagnostics allow quick investigation of failures as well as real-time monitoring of the robot condition throughout its lifecycle; all made to increase your productivity.

IRC5 Single Cabinet Controller

- Designed for high IP protection and full expandability.
- Provides a protected environment for axillary equipment in the robot system.
- Capable of control of up to four robots in a MultiMove® setup. Just add a compact drive module to each additional robot.
- MultiMove® opens up previously unthinkable operations, thanks to the perfect coordination of complex motion patterns.

—

Technical information

Electrical Connections	
Supply voltage	3 phase, 200-600 V, 50-60 Hz
Physical	
Dimensions	
Single cabinet	970 x 725 x 710 mm
MultiMove® drive modules	720 x 725 x 710 mm
Weight	
Single cabinet	150 kg
MultiMove® drive modules	130 kg
Environment	
Ambient temperature	0-45°C, optional: 0-52°C
Relative humidity	Max. 95% non condensing
Safety	SafeMove: Supervision of stand-still, speed, position and orientation (robot and additional axes): 8 safe inputs for function activation, 8 safe monitoring outputs
Extended safety	Electronic Position Switches: 5 safe outputs monitoring axis 1-7
Degree of protection	IP54 (cooling ducts IP33)

IRC5C Compact Controller

- Offers the capabilities of the powerful IRC5 controller in a compact format.
- Delivers space saving benefits and easy commissioning through one phase power input
- External connectors for all signals and a built in expandable 16 in, 16 out I/O system.

—

Technical information

Electrical Connections	
Supply voltage	Single phase 220/230 V, 50-60 Hz
Physical	
Dimensions	320 x 449 x 442 mm
Weight	28.5 kg
Environment	
Ambient temperature	0-45°C
Relative humidity	Max. 95% non condensing
Degree of protection	IP20

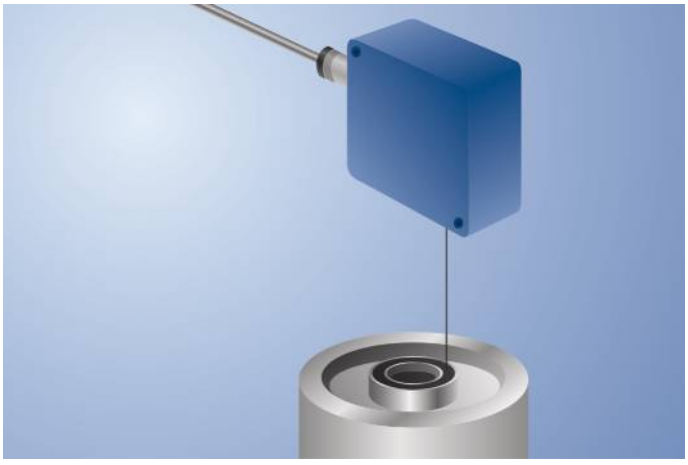
Sensor de distancia de alto rendimiento

OCP242X0135 LASER

Referencia

- **Gran exactitud en la distancia de conmutación**
- **Histéresis mínima de conmutación**
- **Punto de conmutación independiente del material, color o brillo**
- **Tecnología de fotodiodo CMOS**

Los sensores utilizan un fotodiodo CMOS de gran resolución y tecnología DSP y calculan la distancia a través de una medición de ángulo. Así, se elimina virtualmente el material, color y brillo relacionados con las diferencias de los puntos de conmutación.
Dispone de dos salidas de conmutación independientes, y en cada una de ellas pueden configurarse dos umbrales de conmutación y un retardo del tiempo de conexión o desconexión (en pasos de 10 ms). Las funciones del sensor pueden activarse mediante la interfaz RS-232 así como obtener los resultados de la lectura.

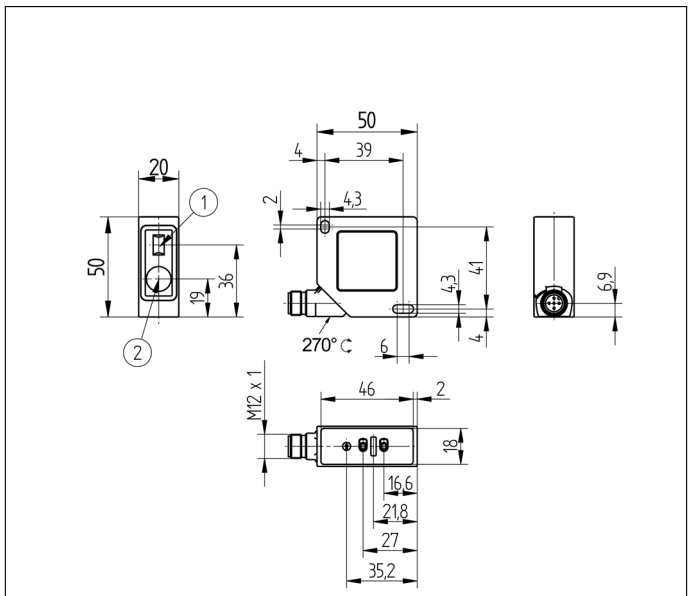


Datos técnicos

Datos ópticos	
Alcance	240 mm
Distancia de ajuste	40...240 mm
Histéresis de conmutación	< 0,5 %
Tipo de luz	Láser (rojo)
Longitud de onda	655 nm
Vida útil (Tu = +25 °C)	100000 h
Clase láser (EN 60825-1)	1
Lux externa máx. admisible	10000 Lux
Diámetro del punto luminoso	Ver tabla 1
Datos eléctricos	
Tensión de alimentación	10...30 V DC
Consumo de corriente (Ub = 24 V)	< 50 mA
Frecuencia de conmutación	300 Hz
Tiempo de reacción	< 1,7 ms
Retardo del tiempo de (des-)conexión RS-232	0...1 s
Temperatura de desvío	< 15 µm/K
Rango de temperatura	-25...60 °C
Número de salidas de conmutación	2
Caída de tensión salida de conmutación	< 1,5 V
Corriente de conmutación / salida de conmutación	200 mA
Protección cortocircuitos	sí
Protección cambio polaridad	sí
Modo Teach-In	HT, VT, FT, TP
Velocidad de transferencia	9600 Bd
Categoría de protección	III
FDA Accession Number	1120718-000
Datos mecánicos	
Tipo de ajustes	Teach-in
Carcasa	Plástico
Clase de protección	IP67
Conexión	M12 × 1; 4/5-pines
Salida de error	●
Configurable PNP/NPN/Push-Pull	●
NO/NC conmutable	●
RS-232 con caja adaptador	●
Entrada Teach-In externa	●
Nº Esquema de conexión	779
Nº Panel de control	P8
Nº Conector adecuado	2 35
Nº Montaje adecuado	380

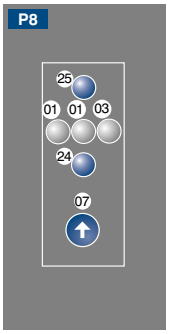
Productos Adicionales

Caja adaptador A232
Carcasa protectora ZSV-0x-01
Set Carcasa protectora ZSP-NN-02
Software

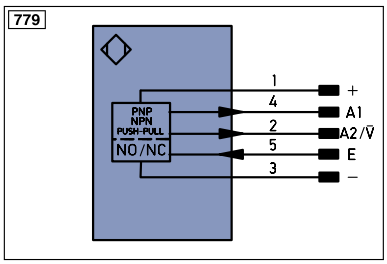


1 = Diodo emisor
2 = Diodo receptor
Tornillo M4 = 0,5 Nm
Todas las dimensiones en mm (1mm =0.03937 Inch)

Panel



01 = Display de estado de conmutación
03 = Display de error
07 = Interruptor selector
24 = Botón más
25 = Botón menos



Aclaración de símbolos

+	Tensión de alimentación +	PT	Resistencia de medición de platino	$EN_{A\overline{RS422}}$	Codificador A/Ā (TTL)
–	Tensión de alimentación 0 V	nc	no está conectado	$EN_{B\overline{RS422}}$	Codificador B/B̄ (TTL)
~	Tensión de alimentación (tensión alterna)	U	Test de entrada	EN_A	Codificador A
A	Salida de conmutación contacto de trabajo (NO)	Ū	Test de entrada inverso	EN_B	Codificador B
Ā	Salida de conmutación contacto de reposo (NC)	W	Entrada activadora	A_{MIN}	Saída digital MIN
V	Salida contaminación/error (NO)	W –	"Masa de referencia" entrada activadora	A_{MAX}	Saída digital MAX
V̄	Salida contaminación/error (NC)	O	Salida analógica	A_{OK}	Saída digital OK
E	Entrada (analógica o digital)	O –	"Masa de referencia" salida analógica	SY In	Sincronización In
T	Entrada de aprendizaje	BZ	Salida en bloque	SY OUT	Sincronización OUT
Z	Retardo temporal (activación)	$A_{\overline{W}}$	Salida electroválvula/motor	O_{LT}	Saída da intensidade luminosa
S	Apantallamiento	a	Salida control de válvula +	M	el mantenimiento
RxD	Receptor RS-232	b	Salida control de válvula 0 V	res	reservada
TxD	Emisor RS-232	SY	Sincronización	Color de los conductores según DIN IEC 757	
RDY	Listo	SY–	"Masa de referencia" sincronización	BK	negro
GND	Cadencia	E+	Conductor del receptor	BN	marrón
CL	Ritmo	S+	Conductor del emisor	RD	rojo
E/A	Entrada/Salida programable	⏚	Puesta a tierra	OG	naranja
	IO-Link	SnR	Reducción distancia de conmutación	YE	amarillo
PoE	Power over Ethernet	Rx+/–	Receptor Ethernet	GN	verde
IN	Entrada de seguridad	Tx+/–	Emisor Ethernet	BU	azul
OSSD	Salida de seguridad	Bus	Interfaz-Bus A(+)/B(–)	VT	violeta
Signal	Salida de señal	La	Luz emitida desconectable	GY	gris
BI_D+/-	Línea datos Ethernet Gigabit bidirecc. (A-D)	Mag	Control magnético	WH	blanco
$EN_{0\,RS422}$	Codificador 0-Impuls 0/0̄ (TTL)	RES	Entrada de confirmación	PK	rosa
		EDM	Comprobación de contactores	GNYE	verde/amarillo

Tabla 1

Alcance de detección	40 mm	240 mm
Tamaño del punto de luz	0,4 × 0,9 mm	1,1 × 2,3 mm

Hoja de datos de producto

C4P-SA13531A00, C4P-EA13531D00

deTec

CORTINAS FOTOELÉCTRICAS DE SEGURIDAD

SICK
Sensor Intelligence.

C4P-SA13531A00, C4P-EA13531D00 | deTec

CORTINAS FOTOELÉCTRICAS DE SEGURIDAD

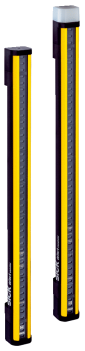


Imagen aproximada

Información sobre pedidos

Parte del sistema	Tipo	N.º de artículo
Emisor	C4P-SA13531A00	1220130
Receptor	C4P-EA13531D00	1220157

Volumen de suministro de emisor y receptor sin conector de sistema. Los conectores de sistema deben adquirirse por separado. Para obtener más detalles, consulte “Accesorios”.

Otros modelos del dispositivo y accesorios ➜ www.sick.com/deTec

Datos técnicos detallados

Características

Aplicación	Entorno industrial estándar
Parte del sistema	Par
Resolución	30 mm
Alcance	30 m
Altura del campo de protección	1.350 mm
Tiempo de respuesta	12 ms (Sin codificación) 21 ms (Código 1 o Código 2)
Ausencia de zonas ciegas	Sí
Sincronización	Sincronización óptica
Auxiliar de alineación láser integrado	✓
Tapa final con piloto señalizador integrado	✓
Elementos suministrados	Emisor Receptor Barra de comprobación con diámetro conforme a la resolución de la cortina fotoeléctrica de seguridad Indicacion de seguridad Instrucciones de montaje Instrucciones de uso para descargar

Características técnicas de seguridad

Tipo	Tipo 4 (IEC 61496-1)
Nivel de integridad de seguridad	SIL3 (IEC 61508) SILCL3 (IEC 62061)
Categoría	4 (ISO 13849-1)
Performance Level	PL e (ISO 13849-1)
PFH_D (probabilidad media de un potencial riesgo por fallo a la hora)	
Dispositivo individual	$15{,}3 \times 10^{-9}$
Cascada con un guest	$30{,}5 \times 10^{-9}$

Cascada con dos dispositivos guest	45,6 x 10^{-9}
TM (tiempo de uso)	20 años (ISO 13849-1)
Estado seguro en caso de fallo	Como mínimo una salida conmutada segura (OSSD) se encuentra en estado de desconexión.

Funciones

Diseñada para fines de protección	✓
Medición automática de la anchura del campo de protección	✓
Codificación de haces	✓
Bloqueo de rearme	✓
Control de contactor (EDM)	✓
En cascada	✓
Protección inteligente para detectar la presencia de personas entre la cortina fotoelectrica de seguridad y la zona peligrosa	✓
Resolución reducida	✓
Anchura del campo de protección dinámica durante el funcionamiento	✓
Diferencia hombre material (muting)	✓

Interfaz

Conexión de sistema	En función del conector de sistema (conector macho, M12, 5 polos u 8 polos)
Conexión de ampliación	En función del conector de sistema (sin conexión de ampliación o con conector hembra M12, 5 polos)
Tipo de configuración	Interruptores DIP en el conector de sistema
Elementos de indicación	LEDs
Indicación del estado de sincronización de los haces superior e inferior	✓
Indicadores luminosos	✓
Salida de aviso (ADO)	✓
IO-Link	✓
Near Field Communication (NFC)	✓
Bus de campo, red industrial	
Integración mediante controlador de seguridad Flexi Soft	CANopen [1)] DeviceNet™ EtherCAT® EtherNet/IP™ Modbus TCP PROFIBUS DP PROFINET

[1)] Más información sobre Flexi Soft -> www.sick.com/Flexi_Soft.

Datos eléctricos

Clase de protección	III (IEC 61140)
Tensión de alimentación V_S	24 V DC (19,2 V ... 28,8 V)
Ondulación	≤ 10 %
Consumo de energía habitual	3,69 W (DC) / 1,75 W (DC) (Según modelo)

[1)] Válido para las tensiones dentro de un rango de –30 V a +30 V.

Salidas conmutadas seguras (OSSD)	
Tipo de salida	2 semiconductores PNP, a prueba de cortocircuitos, con supervisión de cortocircuitos entre las salidas de conmutación [1]
Estado ON, tensión de conmutación HIGH	24 V CC (U_V – 2,25 V CC ... U_V)
Estado OFF, tensión de conmutación LOW	≤ 2 V DC
Capacidad de carga eléctrica de cada OSSD	≤ 500 mA
Salida de aviso (ADO)	
Tipo de salida	Semiconductor PNP, a prueba de cortocircuitos [1]
Tensión de salida HIGH (activa)	≥ V_S – 3 V
Tensión de salida LOW (inactiva)	Alta resistencia
Intensidad de salida HIGH (activa)	≤ 100 mA

[1] Válido para las tensiones dentro de un rango de –30 V a +30 V.

Datos mecánica

Dimensiones	Véase el dibujo acotado
Material de la carcasa	Perfil de aluminio extruido

Datos de ambiente

Grado de protección	IP65 (CEI 60529) IP67 (CEI 60529)
Temperatura ambiente de servicio	–30 °C ... +55 °C
Temperatura de almacenamiento	–30 °C ... +70 °C
Humedad del aire	15 % ... 95 %, sin condensación
Resistencia a la fatiga por vibraciones	5 g, 10 Hz ... 55 Hz (CEI 60068-2-6)
Resistencia contra choques	10 g, 16 ms (CEI 60068-2-27)

Otros datos

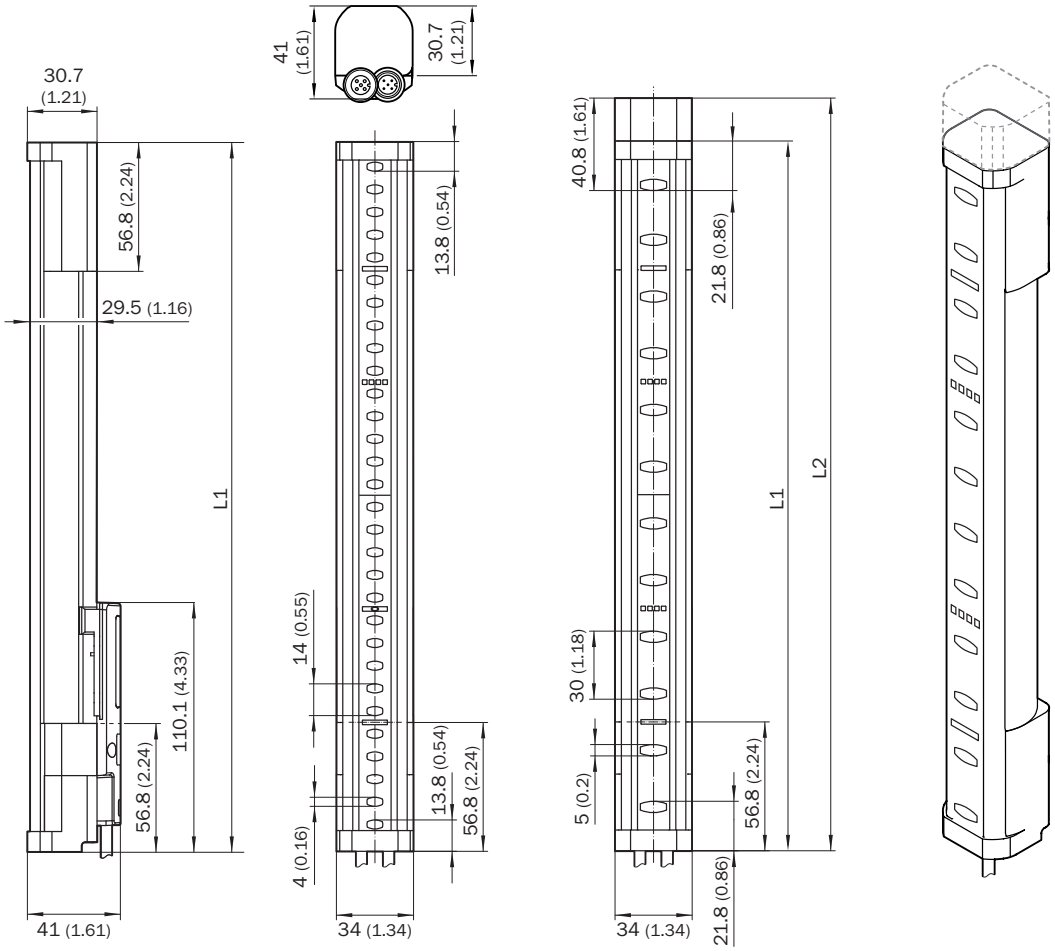
Longitud de onda	850 nm
Tipo de luz	Próximo al Infrarrojo (NIR), invisible
Auxiliar de alineación láser integrado	✓
Clase de láser	1
Longitud de onda	650 nm
Tipo de luz	Luz roja visible

Clasificaciones

ECl@ss 5.0	27272704
ECl@ss 5.1.4	27272704
ECl@ss 6.0	27272704
ECl@ss 6.2	27272704
ECl@ss 7.0	27272704
ECl@ss 8.0	27272704
ECl@ss 8.1	27272704
ECl@ss 9.0	27272704
ECl@ss 10.0	27272704
ECl@ss 11.0	27272704

ETIM 5.0	EC002549
ETIM 6.0	EC002549
ETIM 7.0	EC002549
UNSPSC 16.0901	46171620

Esquema de dimensiones (Medidas en mm)



Altura del campo de protección	L1	L2
300 (11.81)	313 (12.32)	332 (13.07)
450 (17.72)	463 (18.23)	482 (18.98)
600 (23.62)	613 (24.13)	632 (24.88)
750 (29.53)	763 (30.04)	782 (30.79)
900 (35.43)	913 (35.94)	932 (36.69)
1,050 (41.34)	1,063 (41.85)	1,082 (42.6)
1,200 (47.24)	1,213 (47.75)	1,232 (48.5)
1,350 (53.15)	1,362 (53.62)	1,381 (54.37)
1,500 (59.06)	1,512 (59.53)	1,531 (60.28)

Hoja de datos de producto

VT12T-2P410

V12-2

FOTOCÉLULAS CILÍNDRICAS

SICK
Sensor Intelligence.

Imagen aproximada

Información sobre pedidos

Tipo	N.º de artículo
VT12T-2P410	6026212

Otros modelos del dispositivo y accesorios ➜ www.sick.com/V12-2

Datos técnicos detallados

Características

Principio del sensor/ de detección	Fotocélula de detección sobre objeto, Energético
Dimensiones (An x Al x Pr)	12 mm x 12 mm x 65,5 mm
Forma de la carcasa (salida de luz)	Cilíndrico
Longitud de caja	65,5 mm
Diámetro de la rosca (carcasa)	Conector macho M12 x 1
Alcance de detección máx.	0 mm ... 115 mm [1]
Distancia de conmutación	2 mm ... 100 mm
Tipo de luz	Luz infrarroja
Fuente de luz	LED [2]
Tamaño del spot (separación)	Ø 20 mm (100 mm)
Ángulo de dispersión	Aprox. 11,4°
Longitud de onda	880 nm
Ajuste	Tecla teach-in simple (distancia de conmutación) [3] Opcional, por entrada de control C (distancia de conmutación) [4]

[1] Material con un 90% de reflectancia (sobre el blanco estándar según DIN 5033).

[2] Vida útil media de 100.000 h con T_U = 25 °C.

[3] Manual, mediante tecla teach-in.

[4] Electrónica, por entrada de control C (0 V).

Mecánica/Electrónica

Tensión de alimentación	10 V DC ... 30 V DC [1]
Ondulación	≤ 10 % [2]
Consumo de corriente	20 mA [3]
Salida conmutada	PNP
Modo de conmutación	Conmutación en claro/oscuro
Tipo de conmutación seleccionable	Opcional, por entrada de control C
Corriente de salida $I_{máx.}$	≤ 100 mA [3]
Tiempo de respuesta	≤ 1,25 ms [4]
Frecuencia de conmutación	400 Hz [5]
Tipo de conexión	Conector macho M12 de 4 polos
Protección de circuito	A [6] B [7] C [8] D [9]
Clase de protección	III
Peso	+ 18 g
Material de la carcasa	Metal, Latón niquelado/PA
Material de elementos ópticos	Plástico, PMMA
Grado de protección	IP67
Operación a temperatura ambiente	–25 °C ... +70 °C
Almacenamiento a temperatura ambiente	–25 °C ... +70 °C
N.º de archivo UL	E175606

[1] Valores límite.
[2] No se deben sobrepasar por exceso o por defecto las tolerancias de U_V.
[3] Sin carga.
[4] Duración de la señal con carga óhmica.
[5] Con una relación claro/oscuro de 1:1.
[6] A = Conexiones U_V protegidas contra polarización inversa.
[7] B = Entradas y salidas protegidas contra polarización incorrecta.
[8] C = Supresión de impulsos parásitos.
[9] D = Salidas a prueba de sobrecorriente y cortocircuitos.

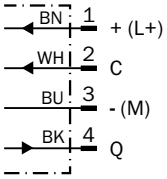
Clasificaciones

ECl@ss 5.0	27270904
ECl@ss 5.1.4	27270904
ECl@ss 6.0	27270904
ECl@ss 6.2	27270904
ECl@ss 7.0	27270904
ECl@ss 8.0	27270904
ECl@ss 8.1	27270904
ECl@ss 9.0	27270904
ECl@ss 10.0	27270904
ECl@ss 11.0	27270904

ETIM 5.0	EC002719
ETIM 6.0	EC002719
ETIM 7.0	EC002719
UNSPSC 16.0901	39121528

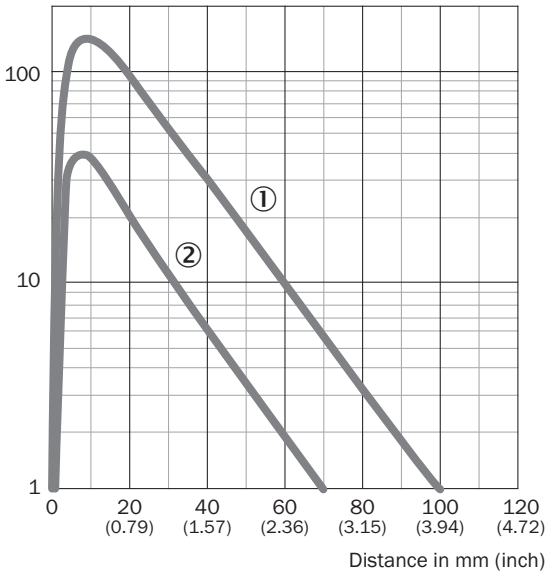
Esquema de conexión

Cd-099



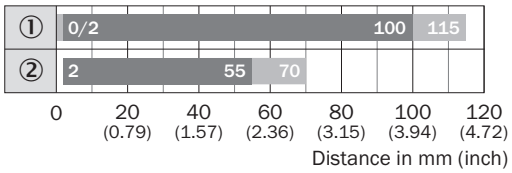
Curva característica

Operating reserve



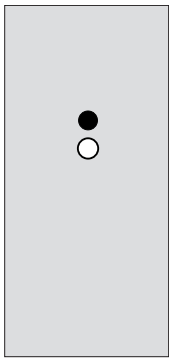
① Distancia de conmutación sobre blanco, reflexión 90%
② Distancia de conmutación sobre gris, reflexión 18%

Diagrama del rango de sensibilidad

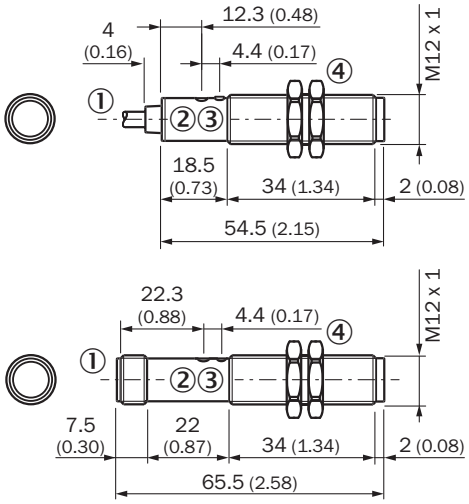


① Distancia de conmutación sobre blanco, reflexión 90%
② Distancia de conmutación sobre gris, reflexión 18%

Posibilidades de ajuste



Esquema de dimensiones (Medidas en mm)



① Cable o conector macho M12, 4 polos
② Ajustador de la sensibilidad: tecla teach-in simple
③ Indicador LED amarillo,- encendido fijo: señal de recepción > factor de reserva 2- parpadea: señal de recepción< factor de reserva 2, pero > umbral de conmutación 1
④ Tuerca de fijación (2 x); SW 17, metal

ANEXO II: CÓDIGO PROGRAMA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº6: ANEXO II: CÓDIGO DE PROGRAMA

1.INTRODUCCIÓN

Se presenta el código del programa de los 3 módulos que lo componen, así como la rutina del módulo de sistema que se utiliza para escribir el tiempo de ciclo en un archivo.

Para la programación del código se ha tenido en cuenta la estructura de programación del resto de robots de la línea y de la forma que se ha considera óptima para cada situación. Han servido de apoyo manuales de programación oficiales de ABB como los disponibles en el apartado de Ayuda de RobotStudio.

2.MÓDULOS DE PROGRAMA

2.1 MÓDULO CALIBDATA

```
MODULE CalibData

   !***********************************************************
   ! Descripción: Módulo para la declaración de variables y datos
   !***********************************************************

   !***********************************************************
   !               JOINTTARGETS
   !***********************************************************
   CONST jointtarget jHome:=[[0,-45,45,0,90,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST jointtarget jPounce_Init:=[[0,-10,0,0,70,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   CONST jointtarget
jPounce_Rack_1:=[[56,7.4,40,0,44,166.403],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   CONST jointtarget jPounce_Rack_2:=[[-
54,1,46,0,42,224.686],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   CONST jointtarget jPounce_Rack_Descarga:=[[-125.188,-8.45171,24.1207,3.42435E-
06,74.3309,-81.7595],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
   !
   VAR robtarget p_CogerDisco:=[[0.000126287,-0.000007003,-0.000067298],[1,-
0.000000016,0.000000026,0.000000022],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   !***********************************************************
   !               RACK 1
   !***********************************************************
   CONST robtarget p_PounceVentosa_Rack_1:=[[-651.62,-77.23,-1272.55],[0.37482,-
4.37924E-7,-1.71384E-7,-0.927098],[0,0,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   CONST robtarget p_CentroVentosa_Rack_1:=[[-495.43,222.55,-903.50],[0.376947,3.48128E-
7,8.71894E-8,-0.926235],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   PERS robtarget p_CogerBlister_Rack_1:=[[-495.43,222.55,-313.33],[0.376947,2.98226E-
7,6.49743E-8,-0.926235],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   CONST robtarget pCentroRack_1:=[[-526.00,203.00,-523.50],[0.952432,-6.06402E-8,-
2.9449E-8,0.304751],[0,-1,1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
   CONST robtarget p_RefVentosa_Rack_1:=[[-495.43,222.55,-313.33],[0.376947,2.98226E-
7,6.49743E-8,-0.926235],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
```

```
!*********************************************************
!               RACK 2
!*********************************************************
CONST robtarget p_PounceVentosa_Rack_2:=[[-651.62,-77.23,-1272.55],[0.37482,-
4.37924E-7,-1.71384E-7,-0.927098],[0,0,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
CONST robtarget p_CentroVentosa_Rack_2:=[[-495.43,222.55,-903.50],[0.376947,3.48128E-
7,8.71894E-8,-0.926235],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
PERS robtarget p_CogerBlister_Rack_2:=[[-495.43,222.55,-313.33],[0.376947,2.98226E-
7,6.49743E-8,-0.926235],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
CONST robtarget pCentroRack_2:=[[-486.00,203.00,-523.50],[0.21259,-9.39037E-
8,4.16156E-8,0.977141],[-1,-1,2,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
CONST robtarget p_RefVentosa_Rack_2:=[[-495.43,222.55,-313.33],[0.376947,2.98226E-
7,6.49743E-8,-0.926235],[0,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];

!*********************************************************
!               RACK DESCARGA
!*********************************************************
CONST robtarget p_CentroVentosa_Rack_3:=[[0.00,-0.00,-600],[1,1.30482E-6,-1.85776E-8,-
4.88024E-6],[-2,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
CONST robtarget p_BlisterPosRef:=[[0.00,-0.00,-0.00],[1,1.30482E-6,-1.85776E-8,-4.88024E-
6],[-2,-1,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
VAR robtarget p_DejadaBlister:=[[0,0,0],[1,1.30482E-6,-1.85776E-8,-4.88024E-6],[-2,-1,-
1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];

!*********************************************************
!               UTIL DESCARGA
!*********************************************************
CONST robtarget p_DejarDisco_MEB:=[[20,380,12],[2.33354E-09,-0.98255,-
0.185998,8.30761E-09],[1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget p_DejarDisco_MQB:=[[0,0,0],[2.33354E-9,-0.98255,-0.185998,8.30761E-
9],[1,0,1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
CONST robtarget p_PounceDejada:=[[-372.03,-174.7,370.28],[-0.000000042,0.999390813,-
0.000001408,-0.034899893],[1,0,2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
!
CONST robtarget p_DejadaEmergencia:=[[1385.00,675.00,-27.00],[6.73446E-8,-1.37958E-
7,1,3.28892E-8],[1,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];

!*********************************************************
!               VENTOSA
!*********************************************************
CONST robtarget p_PounceVentosa:=[[880.63,-201.48,-418.66],[0.00498332,2.30671E-
7,1.34461E-6,0.999988],[-2,0,-1,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
CONST robtarget p_Ventosa:=[[0,0,0],[0.390731128,0,0,-0.920504854],[-2,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
!
```

```
    !***********************************************************
    !               OBJETOS DE TRABAJO
    !***********************************************************
    PERS wobjdata w_Ventosa:=[FALSE,TRUE,"",[[-482.152,1297.947,579.786741206],[0,-
0.390731128,0.920504854,0]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata w_Rack_1:=[FALSE,TRUE,"",[[1286,-922,-273.5],[0,1,0,0]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata w_Rack_2:=[FALSE,TRUE,"",[[-414,-922,-273.5],[0,1,0,0]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata w_RackDescarga:=[FALSE,TRUE,"",[[-1834,510,30],[0,-0.92193,-
0.38736,0]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata w_RackDescarga1:=[FALSE,TRUE,"",[[1707.063232324,-
80.598577911,889.786741206],[0,1,0,0]],[[0,0,0],[1,0,0,0]]];
    PERS wobjdata w_UtilDejada:=[FALSE,TRUE,"",[[1272.461097176,-
190.999999917,104.674017619],[0.999390831,0.00000003,0.034899397,0.000000235]],[[0,0,
0],[1,0,0,0]]];

    !***********************************************************
    !               DATOS DE HERRAMIENTAS
    !***********************************************************
    PERS tooldata t_Manipulador:=[TRUE,[[0,0,161],[1,0,0,0]],[25,[0,0,100],[1,0,0,0],0,0,0]];
    PERS tooldata t_Ventosa:=[TRUE,[[0,0,161],[1,0,0,0]],[25,[0,0,100],[1,0,0,0],0,0,0]];

    !***********************************************************
    !               DATOS DE VELOCIDAES
    !***********************************************************
    PERS speeddata vLento:=[250,100,5000,1000];
    PERS speeddata vBlister:=[2500,200,5000,1000];
    PERS speeddata vMovimiento:=[4000,500,5000,1000];

ENDMODULE
```

2.2 MÓDULO MAIN

```
MODULE MODULO_MAIN
    !***********************************************************
    ! Descripción: Módulo principal de la estación de carga de discos de freno
    ! para los modelos MQB/MEB en VW Alemania
    !***********************************************************

    CONST num Dist_X:=490;
    CONST num Dist_Y:=230;
    CONST num DBlister_Z:=70;
    !
```

```
    PERS num nOffset_X:=-980;
    PERS num nOffset_Y:=460;
    PERS num nOffset_Z:=0;
    !
    PERS bool bManipuladorOK:=TRUE;
    PERS bool bHome:=FALSE;
    VAR num HomePos:=0;
    PERS bool bNumeroPrograma:=TRUE;
    PERS bool bTipoDisco:=TRUE;
    PERS bool bEscanerHecho:=FALSE;
    VAR num nNumeroProg:=0;
    PERS num nTipoDisco:=1;
    VAR num nBlister_Rack_1:=0;
    VAR num nBlister_Rack_2:=0;
    VAR num nBlister_Descarga:=0;
    !
    VAR clock TCRack;
    VAR clock TCInit;
    VAR num TCiclo;
    VAR num TCicloInit;
    !
    PERS bool bDiscoMontado:=FALSE;
    PERS bool bVentosaMontada:=FALSE;
    PERS bool bBlisterMontado:=FALSE;
    !
    PERS bool bMatrizSensoresRack_1{6,8};
    PERS bool bMatrizSensoresRack_2{6,8};
    !
    CONST pos
pMatrizPosCogida{8}:=[[0,0,0],[Dist_X,0,0],[2*Dist_X,0,0],[246,Dist_Y,0],[746,Dist_Y,0],[0,2*Dis
t_Y,0],[Dist_X,2*Dist_Y,0],[2*Dist_X,2*Dist_Y,0]];

    !===============<MAIN>===============
    ! PROYECTO:    TFG_JCH
    ! ROBOT:       IRB4600_60_205_C_01
    !====================================
    PROC Main()
        ClkStart TCInit;
        !
        initVariables;
        initHome;
        ClkStop TCInit;
        !
        WHILE bHome=TRUE AND bManipuladorOK=TRUE DO
            !cycleStart;
```

```
        ClkStart TCRack;
        Elegir_Programa;
        !
        TEST nNumeroProg
        CASE 0:
          bHome:=TRUE;
          !
        CASE 1:
          !Programa 1 = Trabajo en Rack_1
          Rack_1;
        CASE 2:
          !Programa 2 = Trabajo en Rack_2
          Rack_2;
        DEFAULT:
          ExitCycle;
        ENDTEST
        !
        ClkStop TCRack;
        !
        TiempoCiclo;
        !cycleEnd;
      ENDWHILE
      !
    ENDPROC

    !***************************************************************************
    !* Procedimiento Elegir_Programa
    !*
    !* Selección de programa según condiciones establecidas
    !***************************************************************************
    PROC Elegir_Programa()
      nNumeroProg:=0;
      nTipoDisco:=0;
      bNumeroPrograma:=FALSE;
      bTipoDisco:=FALSE;
      !
      WaitUntil di_RackActivo_1=1 XOR di_RackActivo_2=1\Visualize\Header:="ESPERANDO
RACK ACTIVO"\Message:="di_RackActivo_1=1 XOR di_RackActivo_2=1"\Icon:=iconError;
      !
      Reset do_CheckRackLleno;
      Set do_CheckRackLleno;
      waittime 0.2;
      !
      WHILE bNumeroPrograma=FALSE AND bTipoDisco=FALSE DO
        IF di_RackActivo_1=1 AND di_MQB=1 AND di_Rack1Lleno=1 THEN
```

```
            nNumeroProg:=1;
            nTipoDisco:=1;
            bNumeroPrograma:=TRUE;
            bTipoDisco:=TRUE;
            TPWrite "Rack 1 - MQB Selecionado";
          ELSEIF di_RackActivo_1=1 AND di_MEB=1 AND di_Rack1Lleno=1 THEN
            nNumeroProg:=1;
            nTipoDisco:=2;
            bNumeroPrograma:=TRUE;
            bTipoDisco:=TRUE;
            TPWrite "Rack 1 - MEB Selecionado";
          ELSEIF di_RackActivo_2=1 AND di_MQB=1 AND di_Rack2Lleno=1 THEN
            nNumeroProg:=2;
            nTipoDisco:=1;
            bNumeroPrograma:=TRUE;
            bTipoDisco:=TRUE;
            TPWrite "Rack 2 - MQB Selecionado";
          ELSEIF di_RackActivo_2=1 AND di_MEB=1 AND di_Rack2Lleno=1 THEN
            nNumeroProg:=2;
            nTipoDisco:=2;
            bNumeroPrograma:=TRUE;
            bTipoDisco:=TRUE;
            TPWrite "Rack 2 - MEB Selecionado";
          ELSE
            TPErase;
            TPWrite "Esperando numero de programa o eleccion de modelo correctos";
            bNumeroPrograma:=FALSE;
            bTipoDisco:=FALSE;
            !
          ENDIF
        ENDWHILE
        !
      ENDPROC

      PROC ActualizarPosiciones(num nNumeroRack)
        nBlister_Rack_1:=0;
        nBlister_Rack_2:=0;
        TEST nNumeroRack
        CASE 1:
          !Resetear contador de blisters rack 1
          Set do_ResetCont_Blisters_1;
          WaitUntil nBlister_Rack_1=0\Visualize\Header:="ESPERANDO RESETEO CONTADOR DE
BLISTERS"\Message:="Espera nBlister_Rack_1=0"\Icon:=iconError;
          Reset do_ResetCont_Blisters_1;
          !Establecer contador de blisters rack 1
```

```
        Set do_SetCont_Blisters_1;
        !ResetearMatriz;
        WaitUntil nBlister_Rack_1>0\Visualize\Header:="ESPERANDO RESETEO CONTADOR DE
BLISTERS"\Message:="Espera nBlister_Rack_1=0"\Icon:=iconError;
        !
        TEST nBlister_Rack_1
        CASE 1:
          !BLISTER 1
          IF di_HayPiezas_Rack_1_1=1 THEN
            IF di_HayPiezas_Rack_1_1_1=1 bMatrizSensoresRack_1{1,1}:=TRUE;
            IF di_HayPiezas_Rack_1_1_2=1 bMatrizSensoresRack_1{1,2}:=TRUE;
            IF di_HayPiezas_Rack_1_1_3=1 bMatrizSensoresRack_1{1,3}:=TRUE;
            IF di_HayPiezas_Rack_1_1_4=1 bMatrizSensoresRack_1{1,4}:=TRUE;
            IF di_HayPiezas_Rack_1_1_5=1 bMatrizSensoresRack_1{1,5}:=TRUE;
            IF di_HayPiezas_Rack_1_1_6=1 bMatrizSensoresRack_1{1,6}:=TRUE;
            IF di_HayPiezas_Rack_1_1_7=1 bMatrizSensoresRack_1{1,7}:=TRUE;
            IF di_HayPiezas_Rack_1_1_8=1 bMatrizSensoresRack_1{1,8}:=TRUE;
            !
          ENDIF
        CASE 2:
          !BLISTER 2
          IF di_HayPiezas_Rack_1_2=1 THEN
            IF di_HayPiezas_Rack_1_2_1=1 bMatrizSensoresRack_1{2,1}:=TRUE;
            IF di_HayPiezas_Rack_1_2_2=1 bMatrizSensoresRack_1{2,2}:=TRUE;
            IF di_HayPiezas_Rack_1_2_3=1 bMatrizSensoresRack_1{2,3}:=TRUE;
            IF di_HayPiezas_Rack_1_2_4=1 bMatrizSensoresRack_1{2,4}:=TRUE;
            IF di_HayPiezas_Rack_1_2_5=1 bMatrizSensoresRack_1{2,5}:=TRUE;
            IF di_HayPiezas_Rack_1_2_6=1 bMatrizSensoresRack_1{2,6}:=TRUE;
            IF di_HayPiezas_Rack_1_2_7=1 bMatrizSensoresRack_1{2,7}:=TRUE;
            IF di_HayPiezas_Rack_1_2_8=1 bMatrizSensoresRack_1{2,8}:=TRUE;
            !
          ENDIF
        CASE 3:
          !BLISTER 3
          IF di_HayPiezas_Rack_1_3=1 THEN
            IF di_HayPiezas_Rack_1_3_1=1 bMatrizSensoresRack_1{3,1}:=TRUE;
            IF di_HayPiezas_Rack_1_3_2=1 bMatrizSensoresRack_1{3,2}:=TRUE;
            IF di_HayPiezas_Rack_1_3_3=1 bMatrizSensoresRack_1{3,3}:=TRUE;
            IF di_HayPiezas_Rack_1_3_4=1 bMatrizSensoresRack_1{3,4}:=TRUE;
            IF di_HayPiezas_Rack_1_3_5=1 bMatrizSensoresRack_1{3,5}:=TRUE;
            IF di_HayPiezas_Rack_1_3_6=1 bMatrizSensoresRack_1{3,6}:=TRUE;
            IF di_HayPiezas_Rack_1_3_7=1 bMatrizSensoresRack_1{3,7}:=TRUE;
            IF di_HayPiezas_Rack_1_3_8=1 bMatrizSensoresRack_1{3,8}:=TRUE;
            !
          ENDIF
```

```
    CASE 4:
      !BLISTER 4
      IF di_HayPiezas_Rack_1_4=1 THEN
        IF di_HayPiezas_Rack_1_4_1=1 bMatrizSensoresRack_1{4,1}:=TRUE;
        IF di_HayPiezas_Rack_1_4_2=1 bMatrizSensoresRack_1{4,2}:=TRUE;
        IF di_HayPiezas_Rack_1_4_3=1 bMatrizSensoresRack_1{4,3}:=TRUE;
        IF di_HayPiezas_Rack_1_4_4=1 bMatrizSensoresRack_1{4,4}:=TRUE;
        IF di_HayPiezas_Rack_1_4_5=1 bMatrizSensoresRack_1{4,5}:=TRUE;
        IF di_HayPiezas_Rack_1_4_6=1 bMatrizSensoresRack_1{4,6}:=TRUE;
        IF di_HayPiezas_Rack_1_4_7=1 bMatrizSensoresRack_1{4,7}:=TRUE;
        IF di_HayPiezas_Rack_1_4_8=1 bMatrizSensoresRack_1{4,8}:=TRUE;
        !

      ENDIF
    CASE 5:
      !BLISTER 5
      IF di_HayPiezas_Rack_1_5=1 THEN
        IF di_HayPiezas_Rack_1_5_1=1 bMatrizSensoresRack_1{5,1}:=TRUE;
        IF di_HayPiezas_Rack_1_5_2=1 bMatrizSensoresRack_1{5,2}:=TRUE;
        IF di_HayPiezas_Rack_1_5_3=1 bMatrizSensoresRack_1{5,3}:=TRUE;
        IF di_HayPiezas_Rack_1_5_4=1 bMatrizSensoresRack_1{5,4}:=TRUE;
        IF di_HayPiezas_Rack_1_5_5=1 bMatrizSensoresRack_1{5,5}:=TRUE;
        IF di_HayPiezas_Rack_1_5_6=1 bMatrizSensoresRack_1{5,6}:=TRUE;
        IF di_HayPiezas_Rack_1_5_7=1 bMatrizSensoresRack_1{5,7}:=TRUE;
        IF di_HayPiezas_Rack_1_5_8=1 bMatrizSensoresRack_1{5,8}:=TRUE;
        !
      ENDIF
    CASE 6:
      !BLISTER 6
      IF di_HayPiezas_Rack_1_6=1 THEN
        IF di_HayPiezas_Rack_1_6_1=1 bMatrizSensoresRack_1{6,1}:=TRUE;
        IF di_HayPiezas_Rack_1_6_2=1 bMatrizSensoresRack_1{6,2}:=TRUE;
        IF di_HayPiezas_Rack_1_6_3=1 bMatrizSensoresRack_1{6,3}:=TRUE;
        IF di_HayPiezas_Rack_1_6_4=1 bMatrizSensoresRack_1{6,4}:=TRUE;
        IF di_HayPiezas_Rack_1_6_5=1 bMatrizSensoresRack_1{6,5}:=TRUE;
        IF di_HayPiezas_Rack_1_6_6=1 bMatrizSensoresRack_1{6,6}:=TRUE;
        IF di_HayPiezas_Rack_1_6_7=1 bMatrizSensoresRack_1{6,7}:=TRUE;
        IF di_HayPiezas_Rack_1_6_8=1 bMatrizSensoresRack_1{6,8}:=TRUE;
        !
      ENDIF
    DEFAULT:
    ENDTEST
    !*****************************************************************!
  CASE 2:
    !Resetear contador de blisters rack 2
```

```
        Set do_ResetCont_Blisters_2;
        WaitUntil nBlister_Rack_2=0\Visualize\Header:="ESPERANDO RESETEO CONTADOR DE
BLISTERS"\Message:="Espera nBlister_Rack_2=0"\Icon:=iconError;
        Reset do_ResetCont_Blisters_2;
        !Establecer contador de blisters rack 2
        Set do_SetCont_Blisters_2;
        !ResetearMatriz;
        WaitUntil nBlister_Rack_2>0\Visualize\Header:="ESPERANDO RESETEO CONTADOR DE
BLISTERS"\Message:="Espera nBlister_Rack_2=0"\Icon:=iconError;
        !
        TEST nBlister_Rack_2
        CASE 1:
          !BLISTER 1
          IF di_HayPiezas_Rack_2_1=1 THEN
            IF di_HayPiezas_Rack_2_1_1=1 bMatrizSensoresRack_2{1,1}:=TRUE;
            IF di_HayPiezas_Rack_2_1_2=1 bMatrizSensoresRack_2{1,2}:=TRUE;
            IF di_HayPiezas_Rack_2_1_3=1 bMatrizSensoresRack_2{1,3}:=TRUE;
            IF di_HayPiezas_Rack_2_1_4=1 bMatrizSensoresRack_2{1,4}:=TRUE;
            IF di_HayPiezas_Rack_2_1_5=1 bMatrizSensoresRack_2{1,5}:=TRUE;
            IF di_HayPiezas_Rack_2_1_6=1 bMatrizSensoresRack_2{1,6}:=TRUE;
            IF di_HayPiezas_Rack_2_1_7=1 bMatrizSensoresRack_2{1,7}:=TRUE;
            IF di_HayPiezas_Rack_2_1_8=1 bMatrizSensoresRack_2{1,8}:=TRUE;
          ENDIF
        CASE 2:
          !BLISTER 2
          IF di_HayPiezas_Rack_2_2=1 THEN
            IF di_HayPiezas_Rack_2_2_1=1 bMatrizSensoresRack_2{2,1}:=TRUE;
            IF di_HayPiezas_Rack_2_2_2=1 bMatrizSensoresRack_2{2,2}:=TRUE;
            IF di_HayPiezas_Rack_2_2_3=1 bMatrizSensoresRack_2{2,3}:=TRUE;
            IF di_HayPiezas_Rack_2_2_4=1 bMatrizSensoresRack_2{2,4}:=TRUE;
            IF di_HayPiezas_Rack_2_2_5=1 bMatrizSensoresRack_2{2,5}:=TRUE;
            IF di_HayPiezas_Rack_2_2_6=1 bMatrizSensoresRack_2{2,6}:=TRUE;
            IF di_HayPiezas_Rack_2_2_7=1 bMatrizSensoresRack_2{2,7}:=TRUE;
            IF di_HayPiezas_Rack_2_2_8=1 bMatrizSensoresRack_2{2,8}:=TRUE;
          ENDIF
        CASE 3:
          !BLISTER 3
          IF di_HayPiezas_Rack_2_3=1 THEN
            IF di_HayPiezas_Rack_2_3_1=1 bMatrizSensoresRack_2{3,1}:=TRUE;
            IF di_HayPiezas_Rack_2_3_2=1 bMatrizSensoresRack_2{3,2}:=TRUE;
            IF di_HayPiezas_Rack_2_3_3=1 bMatrizSensoresRack_2{3,3}:=TRUE;
            IF di_HayPiezas_Rack_2_3_4=1 bMatrizSensoresRack_2{3,4}:=TRUE;
            IF di_HayPiezas_Rack_2_3_5=1 bMatrizSensoresRack_2{3,5}:=TRUE;
            IF di_HayPiezas_Rack_2_3_6=1 bMatrizSensoresRack_2{3,6}:=TRUE;
            IF di_HayPiezas_Rack_2_3_7=1 bMatrizSensoresRack_2{3,7}:=TRUE;
```

```
        IF di_HayPiezas_Rack_2_3_8=1 bMatrizSensoresRack_2{3,8}:=TRUE;
      ENDIF
    CASE 4:
      !BLISTER 4
      IF di_HayPiezas_Rack_2_4=1 THEN
        IF di_HayPiezas_Rack_2_4_1=1 bMatrizSensoresRack_2{4,1}:=TRUE;
        IF di_HayPiezas_Rack_2_4_2=1 bMatrizSensoresRack_2{4,2}:=TRUE;
        IF di_HayPiezas_Rack_2_4_3=1 bMatrizSensoresRack_2{4,3}:=TRUE;
        IF di_HayPiezas_Rack_2_4_4=1 bMatrizSensoresRack_2{4,4}:=TRUE;
        IF di_HayPiezas_Rack_2_4_5=1 bMatrizSensoresRack_2{4,5}:=TRUE;
        IF di_HayPiezas_Rack_2_4_6=1 bMatrizSensoresRack_2{4,6}:=TRUE;
        IF di_HayPiezas_Rack_2_4_7=1 bMatrizSensoresRack_2{4,7}:=TRUE;
        IF di_HayPiezas_Rack_2_4_8=1 bMatrizSensoresRack_2{4,8}:=TRUE;
      ENDIF
    CASE 5:
      !BLISTER 5
      IF di_HayPiezas_Rack_2_5=1 THEN
        IF di_HayPiezas_Rack_2_5_1=1 bMatrizSensoresRack_2{5,1}:=TRUE;
        IF di_HayPiezas_Rack_2_5_2=1 bMatrizSensoresRack_2{5,2}:=TRUE;
        IF di_HayPiezas_Rack_2_5_3=1 bMatrizSensoresRack_2{5,3}:=TRUE;
        IF di_HayPiezas_Rack_2_5_4=1 bMatrizSensoresRack_2{5,4}:=TRUE;
        IF di_HayPiezas_Rack_2_5_5=1 bMatrizSensoresRack_2{5,5}:=TRUE;
        IF di_HayPiezas_Rack_2_5_6=1 bMatrizSensoresRack_2{5,6}:=TRUE;
        IF di_HayPiezas_Rack_2_5_7=1 bMatrizSensoresRack_2{5,7}:=TRUE;
        IF di_HayPiezas_Rack_2_5_8=1 bMatrizSensoresRack_2{5,8}:=TRUE;
      ENDIF
    CASE 6:
      !BLISTER 6
      IF di_HayPiezas_Rack_2_6=1 THEN
        IF di_HayPiezas_Rack_2_6_1=1 bMatrizSensoresRack_2{6,1}:=TRUE;
        IF di_HayPiezas_Rack_2_6_2=1 bMatrizSensoresRack_2{6,2}:=TRUE;
        IF di_HayPiezas_Rack_2_6_3=1 bMatrizSensoresRack_2{6,3}:=TRUE;
        IF di_HayPiezas_Rack_2_6_4=1 bMatrizSensoresRack_2{6,4}:=TRUE;
        IF di_HayPiezas_Rack_2_6_5=1 bMatrizSensoresRack_2{6,5}:=TRUE;
        IF di_HayPiezas_Rack_2_6_6=1 bMatrizSensoresRack_2{6,6}:=TRUE;
        IF di_HayPiezas_Rack_2_6_7=1 bMatrizSensoresRack_2{6,7}:=TRUE;
        IF di_HayPiezas_Rack_2_6_8=1 bMatrizSensoresRack_2{6,8}:=TRUE;
      ENDIF
    DEFAULT:
    ENDTEST
    !
  DEFAULT:
  ENDTEST
  !
  Reset do_SetCont_Blisters_1;
```

```
    Reset do_SetCont_Blisters_2;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento InitHome
  !*
  !* Inicialización de estados para inicio
  !***********************************************************************
  PROC InitHome()
      !Comprobar correcto estado del manipulador para inicio
      IF (bVentosaMontada=FALSE AND bDiscoMontado=FALSE) THEN
      Home_Init;
        Abrir_Bridas;
        Cerrar_Bridas;
        bManipuladorOK:=TRUE;
      ELSEIF bVentosaMontada=TRUE THEN
        MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
        Dejar_Ventosa;
        Home_Init;
        !
      ELSEIF bDiscoMontado=TRUE THEN
        Dejada_Emergencia;
        Home_Init;
        !
      ELSE
        bManipuladorOK:=TRUE;
        TPErase;
        WaitUntil FALSE\Visualize\Header:="ESPERANDO MANIPULADOR
OK"\Message:="Comprobar correcto estado del manipulador para inicio!!!"\Icon:=iconError;
    ENDIF
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento PROGRAMA 2
  !*
  !* Cogida de Discos/Blisters en el Rack 2 & Dejada
  !***********************************************************************
  PROC InitVariables()
    bHome:=TRUE;
    bManipuladorOK:=FALSE;
    nNumeroProg:=0;
    nTipoDisco:=0;
    bNumeroPrograma:=FALSE;
```

```
    bTipoDisco:=FALSE;
    bEscanerHecho:=FALSE;
    Reset do_ResetCont_Blisters_1;
    Reset do_SetCont_Blisters_1;
    Reset do_ResetCont_Blisters_2;
    Reset do_SetCont_Blisters_2;
    Reset do_ResetCont_Blisters_3;
    Reset do_SetCont_Blisters_3;
    Reset do_Escaner_1;
    Reset do_Escaner_2;
    Set do_AreaLibreDejada;
    Set do_AreaLibreRackDejada;
    ResetearMatriz;
    !
ENDPROC

!*********************************************************************************
!* Procedimiento ResetearMatriz
!*
!* Reseteo de booleanas de la matriz
!*********************************************************************************
PROC ResetearMatriz()
    bMatrizSensoresRack_1{1,1}:=FALSE;
    bMatrizSensoresRack_1{1,2}:=FALSE;
    bMatrizSensoresRack_1{1,3}:=FALSE;
    bMatrizSensoresRack_1{1,4}:=FALSE;
    bMatrizSensoresRack_1{1,5}:=FALSE;
    bMatrizSensoresRack_1{1,6}:=FALSE;
    bMatrizSensoresRack_1{1,7}:=FALSE;
    bMatrizSensoresRack_1{1,8}:=FALSE;
    !
    bMatrizSensoresRack_1{2,1}:=FALSE;
    bMatrizSensoresRack_1{2,2}:=FALSE;
    bMatrizSensoresRack_1{2,3}:=FALSE;
    bMatrizSensoresRack_1{2,4}:=FALSE;
    bMatrizSensoresRack_1{2,5}:=FALSE;
    bMatrizSensoresRack_1{2,6}:=FALSE;
    bMatrizSensoresRack_1{2,7}:=FALSE;
    bMatrizSensoresRack_1{2,8}:=FALSE;
    !
    bMatrizSensoresRack_1{3,1}:=FALSE;
    bMatrizSensoresRack_1{3,2}:=FALSE;
    bMatrizSensoresRack_1{3,3}:=FALSE;
    bMatrizSensoresRack_1{3,4}:=FALSE;
    bMatrizSensoresRack_1{3,5}:=FALSE;
```

```
bMatrizSensoresRack_1{3,6}:=FALSE;
bMatrizSensoresRack_1{3,7}:=FALSE;
bMatrizSensoresRack_1{3,8}:=FALSE;
!
bMatrizSensoresRack_1{4,1}:=FALSE;
bMatrizSensoresRack_1{4,2}:=FALSE;
bMatrizSensoresRack_1{4,3}:=FALSE;
bMatrizSensoresRack_1{4,4}:=FALSE;
bMatrizSensoresRack_1{4,5}:=FALSE;
bMatrizSensoresRack_1{4,6}:=FALSE;
bMatrizSensoresRack_1{4,7}:=FALSE;
bMatrizSensoresRack_1{4,8}:=FALSE;
!
bMatrizSensoresRack_1{5,1}:=FALSE;
bMatrizSensoresRack_1{5,2}:=FALSE;
bMatrizSensoresRack_1{5,3}:=FALSE;
bMatrizSensoresRack_1{5,4}:=FALSE;
bMatrizSensoresRack_1{5,5}:=FALSE;
bMatrizSensoresRack_1{5,6}:=FALSE;
bMatrizSensoresRack_1{5,7}:=FALSE;
bMatrizSensoresRack_1{5,8}:=FALSE;
!
bMatrizSensoresRack_1{6,1}:=FALSE;
bMatrizSensoresRack_1{6,2}:=FALSE;
bMatrizSensoresRack_1{6,3}:=FALSE;
bMatrizSensoresRack_1{6,4}:=FALSE;
bMatrizSensoresRack_1{6,5}:=FALSE;
bMatrizSensoresRack_1{6,6}:=FALSE;
bMatrizSensoresRack_1{6,7}:=FALSE;
bMatrizSensoresRack_1{6,8}:=FALSE;
!
bMatrizSensoresRack_2{1,1}:=FALSE;
bMatrizSensoresRack_2{1,2}:=FALSE;
bMatrizSensoresRack_2{1,3}:=FALSE;
bMatrizSensoresRack_2{1,4}:=FALSE;
bMatrizSensoresRack_2{1,5}:=FALSE;
bMatrizSensoresRack_2{1,6}:=FALSE;
bMatrizSensoresRack_2{1,7}:=FALSE;
bMatrizSensoresRack_2{1,8}:=FALSE;
!
bMatrizSensoresRack_2{2,1}:=FALSE;
bMatrizSensoresRack_2{2,2}:=FALSE;
bMatrizSensoresRack_2{2,3}:=FALSE;
bMatrizSensoresRack_2{2,4}:=FALSE;
bMatrizSensoresRack_2{2,5}:=FALSE;
```

```
    bMatrizSensoresRack_2{2,6}:=FALSE;
    bMatrizSensoresRack_2{2,7}:=FALSE;
    bMatrizSensoresRack_2{2,8}:=FALSE;
    !
    bMatrizSensoresRack_2{3,1}:=FALSE;
    bMatrizSensoresRack_2{3,2}:=FALSE;
    bMatrizSensoresRack_2{3,3}:=FALSE;
    bMatrizSensoresRack_2{3,4}:=FALSE;
    bMatrizSensoresRack_2{3,5}:=FALSE;
    bMatrizSensoresRack_2{3,6}:=FALSE;
    bMatrizSensoresRack_2{3,7}:=FALSE;
    bMatrizSensoresRack_2{3,8}:=FALSE;
    !
    bMatrizSensoresRack_2{4,1}:=FALSE;
    bMatrizSensoresRack_2{4,2}:=FALSE;
    bMatrizSensoresRack_2{4,3}:=FALSE;
    bMatrizSensoresRack_2{4,4}:=FALSE;
    bMatrizSensoresRack_2{4,5}:=FALSE;
    bMatrizSensoresRack_2{4,6}:=FALSE;
    bMatrizSensoresRack_2{4,7}:=FALSE;
    bMatrizSensoresRack_2{4,8}:=FALSE;
    !
    bMatrizSensoresRack_2{5,1}:=FALSE;
    bMatrizSensoresRack_2{5,2}:=FALSE;
    bMatrizSensoresRack_2{5,3}:=FALSE;
    bMatrizSensoresRack_2{5,4}:=FALSE;
    bMatrizSensoresRack_2{5,5}:=FALSE;
    bMatrizSensoresRack_2{5,6}:=FALSE;
    bMatrizSensoresRack_2{5,7}:=FALSE;
    bMatrizSensoresRack_2{5,8}:=FALSE;
    !
    bMatrizSensoresRack_2{6,1}:=FALSE;
    bMatrizSensoresRack_2{6,2}:=FALSE;
    bMatrizSensoresRack_2{6,3}:=FALSE;
    bMatrizSensoresRack_2{6,4}:=FALSE;
    bMatrizSensoresRack_2{6,5}:=FALSE;
    bMatrizSensoresRack_2{6,6}:=FALSE;
    bMatrizSensoresRack_2{6,7}:=FALSE;
    bMatrizSensoresRack_2{6,8}:=FALSE;
    !
ENDPROC

!***********************************************************************
!* Procedimiento TiempoCiclo
!*
```

```
  !* Calcular el tiempo de ciclo y almacenarlo en archivo
  !*****************************************************************************
  PROC TiempoCiclo()
    TCiclo:=ClkRead(TCRack);
    TCicloInit:=ClkRead(TCInit);
    !
    EscribirTCEnArchivo;
    !
    ClkReset TCRack;
    ClkReset TCInit;
  ENDPROC

ENDMODULE
```

2.3 MÓDULO PROGRAMAS

```
MODULE MODULO_PROGRAMAS
  !*****************************************************************************
  ! Descripción:  Módulo de programas y rutinas de movimientos de la estación
  !*****************************************************************************

  !*****************************************************************************
  !* Procedimiento Rack_1
  !*
  !* Cogida de Discos/Blisters en el Rack 1 & Dejada
  !*****************************************************************************
  PROC Rack_1()
    VAR num nDisco;
    !
    TPErase;
    TPWrite "--------------------------------";
    TPWrite "         PROGRAMA 1          ";
    TPWrite "COGER DISCO/BLISTERS EN EL RACK 1";
    TPWrite "--------------------------------";
    !
    IF nTipoDisco=1 THEN
      WaitUntil di_MQBRack_1=1\Visualize\Header:="MENSAJE DE
ESPERA"\Message:="Modelo Introducido Incorrecto"\Icon:=iconError;
    ELSEIF nTipoDisco=2 THEN
      WaitUntil di_MEBRack_1=1\Visualize\Header:="MENSAJE DE
ESPERA"\Message:="Modelo Introducido Incorrecto"\Icon:=iconError;
    ENDIF
    !
    EscanearRack\NOCHECKEAR;
```

```
    Home_to_Pounce_1;
    !
    WaitUntil di_AreaLibreRack1=1\Visualize\Header:="MENSAJE DE
ESPERA"\Message:="Esperando Area Libre Rack 1"\Icon:=iconError;
    !
    EscanearRack;
    !Actualizar matriz de posiciones
    ActualizarPosiciones 1;
    WaitTime 1;
    !
    nDisco:=0;
    FOR nDisco FROM 1 TO 8 DO
      !
      IF di_RackActivo_1=1 THEN
        ManipuladorListo;
        !Recorrer matriz de posiciones
        !
        IF bMatrizSensoresRack_1{nBlister_Rack_1,nDisco}=TRUE THEN
          ObtenerOffsets(nDisco);
          Coger_Disco_1;
          Dejar_Disco;
          bMatrizSensoresRack_1{nBlister_Rack_1,nDisco}:=FALSE;
          PounceDejada_to_Pounce 1;
          !
        ENDIF
        !
      ELSE
        !Rack1 no seleccionado: ExitCycle.
        Home;
        ExitCycle;
        !
      ENDIF
    ENDFOR
    EscanearRack\NOCHECKEAR;
    !
    Coger_Ventosa;
    !
    EscanearRack;
    Coger_Blister_1;
    Dejar_Blister;
    Dejar_Ventosa;
    !
    Home;
    !
  ENDPROC
```

```
!***********************************************************************
!* Procedimiento Rack_2
!*
!* Cogida de Discos/Blisters en el Rack 2 & Dejada
!***********************************************************************
PROC Rack_2()
    VAR num nDisco;
    !
    TPErase;
    TPWrite "--------------------------------";
    TPWrite "         PROGRAMA 2          ";
    TPWrite "COGER DISCO/BLISTERS EN EL RACK 2";
    TPWrite "--------------------------------";
    !
    IF nTipoDisco=1 THEN
      WaitUntil di_MQBRack_2=1\Visualize\Header:="MENSAJE DE
ESPERA"\Message:="Modelo Introducido Incorrecto"\Icon:=iconError;
    ELSEIF nTipoDisco=2 THEN
      WaitUntil di_MEBRack_2=1\Visualize\Header:="MENSAJE DE
ESPERA"\Message:="Modelo Introducido Incorrecto"\Icon:=iconError;
    ENDIF
    !
    EscanearRack\NOCHECKEAR;
    Home_to_Pounce_2;
    !
    EscanearRack;
    !
    WaitUntil di_AreaLibreRack2=1\Visualize\Header:="MENSAJE DE
ESPERA"\Message:="Esperando Area Libre Rack 2"\Icon:=iconError;
    !
    !Actualizar matriz de posiciones
    ActualizarPosiciones 2;
    WaitTime 1;
    !
    nDisco:=0;
    FOR nDisco FROM 1 TO 8 DO
      !
      IF di_RackActivo_2=1 THEN
        ManipuladorListo;
        !Recorrer matriz de posiciones
        !
        IF bMatrizSensoresRack_2{nBlister_Rack_2,nDisco}=TRUE THEN
          ObtenerOffsets(nDisco);
          Coger_Disco_2;
```

```
            Dejar_Disco;
            bMatrizSensoresRack_2{nBlister_Rack_2,nDisco}:=FALSE;
            PounceDejada_to_Pounce 2;
            !
          ENDIF
          !
        ELSE
          !Rack2 no seleccionado: ExitCycle.
          Home;
          ExitCycle;
          !
        ENDIF
      ENDFOR
      EscanearRack\NOCHECKEAR;
      !
      Coger_Ventosa;
      !
      EscanearRack;
      Coger_Blister_2;
      Dejar_Blister;
      Dejar_Ventosa;
      !
      Home;
      !
    ENDPROC

    !***********************************************************************
    !* Procedimiento Home_to_Pounce_1()
    !***********************************************************************
    PROC Home_to_Pounce_1()
      MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
      bHome:=FALSE;
      MoveAbsJ jPounce_Rack_1,vmax,fine,t_Manipulador\WObj:=wobj0;
      !
    ENDPROC

    !***********************************************************************
    !* Procedimiento Home_to_Pounce_2()
    !***********************************************************************
    PROC Home_to_Pounce_2()
      MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
      bHome:=FALSE;
      MoveAbsJ jPounce_Rack_2,vmax,fine,t_Manipulador\WObj:=wobj0;
      !
    ENDPROC
```

```
!*****************************************************************************
!* Procedimiento Home()
!*****************************************************************************
PROC Home()
  MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
  bHome:=TRUE;
  !
ENDPROC

!*****************************************************************************
!* Procedimiento PounceDejada_to_PounceX()
!*****************************************************************************
PROC PounceDejada_to_Pounce(num nPounceRack)
  MoveJ p_PounceDejada,vmax,z100,t_Manipulador\WObj:=w_UtilDejada;
  TEST nPounceRack
  CASE 0:
    MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
    bHome:=TRUE;
  CASE 1:
    MoveAbsJ jPounce_Rack_1,vmax,z100,t_Manipulador\WObj:=wobj0;
  CASE 2:
    MoveAbsJ jPounce_Rack_2,vmax,z100,t_Manipulador\WObj:=wobj0;
  DEFAULT:
  ENDTEST
  !
ENDPROC

!*****************************************************************************
!* Procedimiento Home_Inicializacion()
!*****************************************************************************
PROC Home_Init()
  MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
  bHome:=FALSE;
  MoveAbsJ jPounce_Init,vmax,fine,t_Manipulador\WObj:=wobj0;
  !
  WaitTime 1;
  !
  MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
  bHome:=TRUE;
  !
ENDPROC

!*****************************************************************************
!* Procedimiento ObtenerOffsets(num nDisco)
```

```
!*
!* Obtiene los Offsets para la Posicion de Cogida Establecida
!*****************************************************************************
PROC ObtenerOffsets(num nDisco)
  !
  IF nDisco<1 OR nDisco>8 THEN
    TPErase;
    TPWrite "Error obteniendo los offsets";
    WaitUntil FALSE;
  ELSE
    nOffset_X:=-pMatrizPosCogida{nDisco}.x;
    nOffset_Y:=pMatrizPosCogida{nDisco}.y;
    !
    IF di_RackActivo_1=1 AND nTipoDisco=1 THEN
      nOffset_Z:=(nBlister_Rack_1*DBlister_Z)-DBlister_Z;
      !
    ELSEIF di_RackActivo_2=1 AND nTipoDisco=1 THEN
      nOffset_Z:=(nBlister_Rack_2*DBlister_Z)-DBlister_Z;
      !
    ELSEIF di_RackActivo_1=1 AND nTipoDisco=2 THEN
      nOffset_Z:=(nBlister_Rack_1*DBlister_Z)-DBlister_Z+10;
      !
    ELSEIF di_RackActivo_2=1 AND nTipoDisco=2 THEN
      nOffset_Z:=(nBlister_Rack_2*DBlister_Z)-DBlister_Z+10;
      !
    ELSE
      WaitUntil FALSE;
      !
    ENDIF
  ENDIF
  !
ENDPROC

!*****************************************************************************
!* Procedimiento Coger_Disco_1()
!*
!* Cogida de Discos en el Rack 1
!*****************************************************************************
PROC Coger_Disco_1()
  !
  MoveAbsJ jPounce_Rack_1,vmax,fine,t_Manipulador\WObj:=wobj0;
  !
  MoveJ pCentroRack_1,vmax,z200,t_Manipulador\WObj:=w_Rack_1;
  MoveJ Offs(p_CogerDisco,nOffset_X,nOffset_Y,-nOffset_Z-
50),vMovimiento,z50,t_Manipulador\WObj:=w_Rack_1;
```

```
    MoveL Offs(p_CogerDisco,nOffset_X,nOffset_Y,-
nOffset_Z),vLento,fine,t_Manipulador\WObj:=w_Rack_1;
    !Abrir_Bridas
    Abrir_Bridas;
    !Establecer carga del robot
    !SetPayLoad Pieza;
    !Establecer pieza en robot
    bDiscoMontado:=TRUE;
    bEscanerHecho:=FALSE;
    !
    MoveL Offs(p_CogerDisco,nOffset_X,nOffset_Y,-
nOffset_Z),vLento,fine,t_Manipulador\WObj:=w_Rack_1;
    MoveL Offs(p_CogerDisco,nOffset_X,nOffset_Y,-nOffset_Z-
50),vMovimiento,z50,t_Manipulador\WObj:=w_Rack_1;
    MoveJ pCentroRack_1,vmax,z200,t_Manipulador\WObj:=w_Rack_1;
    !
    MoveAbsJ jPounce_Rack_1,vmax,fine,t_Manipulador\WObj:=wobj0;
    !
    RETURN ;
    !Punto de Referencia de Cogida
    MoveL p_CogerDisco,vmax,fine,t_Manipulador\WObj:=w_Rack_1;
    !
  ENDPROC

  !***************************************************************************
  !* Procedimiento Coger_Disco_2()
  !*
  !* Cogida de Discos en el Rack 2
  !***************************************************************************
  PROC Coger_Disco_2()
    !
    MoveAbsJ jPounce_Rack_2,vmax,fine,t_Manipulador\WObj:=wobj0;
    !
    MoveJ pCentroRack_2,vmax,z200,t_Manipulador\WObj:=w_Rack_2;
    MoveJ Offs(p_CogerDisco,nOffset_X,nOffset_Y,-nOffset_Z-
50),vMovimiento,z50,t_Manipulador\WObj:=w_Rack_2;
    MoveL Offs(p_CogerDisco,nOffset_X,nOffset_Y,-
nOffset_Z),vLento,fine,t_Manipulador\WObj:=w_Rack_2;
    !Abrir_Bridas
    Abrir_Bridas;
    !Establecer carga del robot
    !SetPayLoad Pieza;
    !Establecer pieza en robot
    bDiscoMontado:=TRUE;
    bEscanerHecho:=FALSE;
```

```
    !
    MoveL Offs(p_CogerDisco,nOffset_X,nOffset_Y,-
nOffset_Z),vLento,fine,t_Manipulador\WObj:=w_Rack_2;
    MoveL Offs(p_CogerDisco,nOffset_X,nOffset_Y,-nOffset_Z-
50),vMovimiento,z50,t_Manipulador\WObj:=w_Rack_2;
    MoveJ pCentroRack_2,vmax,z200,t_Manipulador\WObj:=w_Rack_2;
    !
    MoveAbsJ jPounce_Rack_2,vmax,fine,t_Manipulador\WObj:=wobj0;
    !
    RETURN ;
    !Punto de Referencia de Cogida
    MoveL p_CogerDisco,vmax,fine,t_Manipulador\WObj:=w_Rack_2;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento  Dejar_Disco()
  !*
  !* CASE 1 => Dejar Disco en Util de Descarga Modelo MQB
  !* CASE 2 => Dejar Disco en Util de Descarga Modelo MEB
  !***********************************************************************
  PROC Dejar_Disco()
    !
    MoveJ p_PounceDejada,vmax,z50,t_Manipulador\WObj:=w_UtilDejada;
    Reset do_AreaLibreDejada;
    !
    TEST nTipoDisco
    CASE 1:
      !*************Dejada modelo MQB*************
      WaitUntil di_AreaLibreDejadaMQB=1\Visualize\Header:="ESPERANDO
SEÑAL"\Message:="Area Libre Dejada Disco MQB"\Icon:=iconError;
      !
      MoveJ
Offs(p_DejarDisco_MQB,0,0,50),vMovimiento,z20,t_Manipulador\WObj:=w_UtilDejada;
      MoveL p_DejarDisco_MQB,vLento,fine,t_Manipulador\WObj:=w_UtilDejada;
      !Cerrar bridas
      Cerrar_Bridas;
      !
      bDiscoMontado:=FALSE;
      !
      MoveL p_DejarDisco_MQB,vLento,fine,t_Manipulador\WObj:=w_UtilDejada;
      MoveL
Offs(p_DejarDisco_MQB,0,0,50),vMovimiento,z20,t_Manipulador\WObj:=w_UtilDejada;
      !
    CASE 2:
```

```
        !*************Dejada modelo MEB*************
        WaitUntil di_AreaLibreDejadaMEB=1\Visualize\Header:="ESPERANDO
SEÑAL"\Message:="Area Libre Dejada Disco MEB"\Icon:=iconError;
        !
        MoveL
Offs(p_DejarDisco_MEB,0,10,120),vMovimiento,z20,t_Manipulador\WObj:=w_UtilDejada;
        MoveL
Offs(p_DejarDisco_MEB,0,0,50),vMovimiento,z20,t_Manipulador\WObj:=w_UtilDejada;
        MoveL p_DejarDisco_MEB,vLento,fine,t_Manipulador\WObj:=w_UtilDejada;
        !Cerrar bridas
        Cerrar_Bridas;
        !
        bDiscoMontado:=FALSE;
        !
        MoveL p_DejarDisco_MEB,vLento,fine,t_Manipulador\WObj:=w_UtilDejada;
        MoveL
Offs(p_DejarDisco_MEB,0,0,50),vMovimiento,z20,t_Manipulador\WObj:=w_UtilDejada;
        !
    DEFAULT:
    ENDTEST
    !
    MoveJDO
p_PounceDejada,vmax,z50,t_Manipulador\WObj:=w_UtilDejada,do_AreaLibreDejada,1;
    !
    RETURN ;
    MoveL p_DejarDisco_MQB,vmax,fine,t_Manipulador\WObj:=w_UtilDejada;
    !
    MoveL p_DejarDisco_MEB,vmax,fine,t_Manipulador\WObj:=w_UtilDejada;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento Coger_Ventosa()
  !*
  !* Cogida de la Herramienta Ventosa del ToolStand
  !***********************************************************************
  PROC Coger_Ventosa()
    MoveJ p_PounceVentosa,vmax,z200,t_Manipulador\WObj:=w_Ventosa;
    !
    MoveJ Offs(p_Ventosa,0,0,-200),vmax,z100,t_Manipulador\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,0,0,-50),v200,z10,t_Manipulador\WObj:=w_Ventosa;
    MoveL p_Ventosa,vLento,fine,t_Manipulador\WObj:=w_Ventosa;
    !Abrir_Bridas
    Abrir_Bridas;
    bVentosaMontada:=TRUE;
```

```
    !
    MoveL p_Ventosa,vLento,fine,t_Ventosa\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,0,0,-50),v200,z10,t_Ventosa\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,0,0,-100),v200,z100,t_Ventosa\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,500,0,-200),v200,z100,t_Ventosa\WObj:=w_Ventosa;
    !
    MoveJ p_PounceVentosa,vmax,z200,t_Ventosa\WObj:=w_Ventosa;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento Dejar_Ventosa()
  !*
  !* Dejada de la Herramienta Ventosa sobre el ToolStand
  !***********************************************************************
  PROC Dejar_Ventosa()
    MoveJ p_PounceVentosa,vmax,z200,t_Ventosa\WObj:=w_Ventosa;
    !
    MoveJ Offs(p_Ventosa,500,0,-200),v1000,z100,t_Ventosa\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,0,0,-50),v200,z50,t_Ventosa\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,0,0,0),vLento,fine,t_Ventosa\WObj:=w_Ventosa;
    !
    Cerrar_Bridas;
    bVentosaMontada:=FALSE;
    !
    MoveL p_Ventosa,vLento,z10,t_Manipulador\WObj:=w_Ventosa;
    MoveL Offs(p_Ventosa,0,0,-50),v200,z50,t_Manipulador\WObj:=w_Ventosa;
    MoveJ Offs(p_Ventosa,0,0,-200),vmax,z200,t_Manipulador\WObj:=w_Ventosa;
    !
    MoveJ p_PounceVentosa,vmax,z200,t_Manipulador\WObj:=w_Ventosa;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento Coger_Blister_1()
  !*
  !* Cogida de Blister en el Rack 1 con Ventosa
  !***********************************************************************
  PROC Coger_Blister_1()
    MoveJ p_PounceVentosa_Rack_1,vmax,z200,t_Ventosa\WObj:=w_Rack_1;
    !
    p_CogerBlister_Rack_1.trans:=p_RefVentosa_Rack_1.trans+[0,0,-((nBlister_Rack_1-
1)*DBlister_Z)];
    !
    MoveJ p_CentroVentosa_Rack_1,vmax,z200,t_Ventosa\WObj:=w_Rack_1;
```

```
    MoveL Offs(p_CogerBlister_Rack_1,0,0,-
50),vMovimiento,z50,t_Ventosa\WObj:=w_Rack_1;
    MoveL Offs(p_CogerBlister_Rack_1,0,0,0),vLento,fine,t_Ventosa\WObj:=w_Rack_1;
    !
    Set do_Ventosa;
    WaitUntil di_VacioVentosa=1\Visualize\Header:="ESPERANDO VACIO ACTIVADO EN
VENTOSA"\Message:="Espera di_VacioVentosa=1"\Icon:=iconError;
    !
    bBlisterMontado:=TRUE;
    !
    MoveL Offs(p_CogerBlister_Rack_1,0,0,0),vLento,fine,t_Ventosa\WObj:=w_Rack_1;
    MoveL Offs(p_CogerBlister_Rack_1,0,0,-50),vBlister,z50,t_Ventosa\WObj:=w_Rack_1;
    MoveL p_CentroVentosa_Rack_1,vBlister,z100,t_Ventosa\WObj:=w_Rack_1;
    !
    MoveJ p_PounceVentosa_Rack_1,vBlister,z100,t_Ventosa\WObj:=w_Rack_1;
    !
    RETURN ;
    MoveJ p_RefVentosa_Rack_1,vmax,z200,t_Ventosa\WObj:=w_Rack_1;
    !
  ENDPROC

  !***************************************************************************
  !* Procedimiento Coger_Blister_2()
  !*
  !* Cogida de Blister en el Rack 2
  !***************************************************************************
  PROC Coger_Blister_2()
    MoveJ p_PounceVentosa_Rack_2,vmax,z200,t_Ventosa\WObj:=w_Rack_2;
    !
    p_CogerBlister_Rack_2.trans:=p_RefVentosa_Rack_2.trans+[0,0,-((nBlister_Rack_2-
1)*DBlister_Z)];
    !
    MoveJ p_CentroVentosa_Rack_2,vmax,z200,t_Ventosa\WObj:=w_Rack_2;
    MoveL Offs(p_CogerBlister_Rack_2,0,0,-
50),vMovimiento,z50,t_Ventosa\WObj:=w_Rack_2;
    MoveL Offs(p_CogerBlister_Rack_2,0,0,0),vLento,fine,t_Ventosa\WObj:=w_Rack_2;
    !
    Set do_Ventosa;
    WaitUntil di_VacioVentosa=1\Visualize\Header:="ESPERANDO VACIO ACTIVADO EN
VENTOSA"\Message:="Espera di_VacioVentosa=1"\Icon:=iconError;
    !
    bBlisterMontado:=TRUE;
    !
    MoveL Offs(p_CogerBlister_Rack_2,0,0,0),vLento,fine,t_Ventosa\WObj:=w_Rack_2;
    MoveL Offs(p_CogerBlister_Rack_2,0,0,-50),vBlister,z50,t_Ventosa\WObj:=w_Rack_2;
```

```
    MoveL p_CentroVentosa_Rack_2,vBlister,z100,t_Ventosa\WObj:=w_Rack_2;
    !
    MoveJ p_PounceVentosa_Rack_2,vBlister,z200,t_Ventosa\WObj:=w_Rack_2;
    !
    RETURN ;
    MoveJ p_RefVentosa_Rack_2,vmax,z200,t_Ventosa\WObj:=w_Rack_2;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento Dejar_Blister()
  !*
  !* Dejada de Blister en Rack Comun
  !***********************************************************************
  PROC Dejar_Blister()
    nBlister_Descarga:=0;
    !
    Reset do_AreaLibreRackDejada;
    !Resetear contador de blisters rack descarga
    waittime 0.1;
    Reset do_SetCont_Blisters_3;
    Set do_ResetCont_Blisters_3;
    WaitUntil nBlister_Descarga=0\Visualize\Header:="ESPERANDO RESETEO CONTADOR DE
BLISTERS DESCARGA"\Message:="Espera nBlister_Rack_3=0"\Icon:=iconError;
    Reset do_ResetCont_Blisters_3;
    !Establecer contador de blisters rack descarga
    Set do_SetCont_Blisters_3;
    Waittime 0.5;
    !
    p_DejadaBlister.trans:=p_BlisterPosRef.trans+[0,0,(-nBlister_Descarga*DBlister_Z)];
    !
    MoveAbsJ jPounce_Rack_Descarga,vBlister,z200,t_Ventosa\WObj:=w_RackDescarga;
    !
    WaitUntil di_AreaLibreRackDescarga=1\Visualize\Header:="ESPERANDO AREA LIBRE EN
RACK DE DESCARGA"\Message:="Espera di_AreaLibreRackDescarga=1"\Icon:=iconError;
    !
    MoveJ p_CentroVentosa_Rack_3,vBlister,z200,t_Ventosa\WObj:=w_RackDescarga;
    MoveL Offs(p_DejadaBlister,0,0,-100),vLento,z50,t_Ventosa\WObj:=w_RackDescarga;
    MoveL Offs(p_DejadaBlister,0,0,0),vLento,fine,t_Ventosa\WObj:=w_RackDescarga;
    !
    Reset do_Ventosa;
    WaitUntil di_VacioVentosa=0\Visualize\Header:="ESPERANDO VACIO DESACTIVADO EN
VENTOSA"\Message:="Espera di_VacioVentosa=0"\Icon:=iconError;
    !
    bBlisterMontado:=FALSE;
```

```
    !
    MoveL Offs(p_DejadaBlister,0,0,0),vLento,fine,t_Ventosa\WObj:=w_RackDescarga;
    MoveL Offs(p_DejadaBlister,0,0,-
100),vMovimiento,z50,t_Ventosa\WObj:=w_RackDescarga;
    MoveL p_CentroVentosa_Rack_3,vmax,z100,t_Ventosa\WObj:=w_RackDescarga;
    !
    MoveAbsJ jPounce_Rack_Descarga,vmax,fine,t_Ventosa\WObj:=w_RackDescarga;
    !
    Set do_AreaLibreRackDejada;
    !
    RETURN ;
    MoveJ p_BlisterPosRef,vmax,fine,t_Ventosa\WObj:=w_RackDescarga;
    !
  ENDPROC

  !***********************************************************************
  !* Procedimiento  Escaner()
  !*
  !* CASE 1 => Activacion del Escaner para Lectura de Discos en el Rack 1
  !* CASE 2 => Activacion del Escaner para Lectura de Discos en el Rack 2
  !***********************************************************************
  PROC EscanearRack(\switch NOCHECKEAR)
    TEST nNumeroProg
    CASE 1:
      !*************Escaneo sobre Rack 1*************
      IF bEscanerHecho=FALSE THEN
        !Iniciar Escaneo en Rack 1
        SET do_Escaner_1;
        WaitUntil di_EscanerFinalizado_1=0\Visualize\Header:="ESCANER
ACTIVO"\Message:="Espera di_EscanerFinalizado_1=0"\Icon:=iconError;
        !
        IF NOT PRESENT(NOCHECKEAR) THEN
          !Esperar Escaneo Finalizado Rack 1
          WaitUntil di_EscanerFinalizado_1=1\Visualize\Header:="ESPERANDO ESCANER
FINALIZADO RACK 1"\Message:="Espera di_EscanerFinalizado_1=1"\Icon:=iconError;
        ENDIF
        !
        bEscanerHecho:=TRUE;
        RESET do_Escaner_1;
      ELSE
        WaitUntil di_EscanerFinalizado_1=1\Visualize\Header:="ESCANER
ACTIVO"\Message:="Espera di_EscanerFinalizado_1=0"\Icon:=iconError;
        bEscanerHecho:=TRUE;
      ENDIF
      !*****************************************
```

```
      CASE 2:
        !*************Escaneo sobre Rack 2*************
        IF bEscanerHecho=FALSE THEN
          !Iniciar Escaneo en Rack 2
          SET do_Escaner_2;
          WaitUntil di_EscanerFinalizado_2=0\Visualize\Header:="ESCANER
ACTIVO"\Message:="Espera di_EscanerFinalizado_2=0"\Icon:=iconError;
          !
          IF NOT PRESENT(NOCHECKEAR) THEN
            !Esperar Escaneo Finalizado Rack 2
            WaitUntil di_EscanerFinalizado_2=1\Visualize\Header:="ESPERANDO ESCANER
FINALIZADO RACK 1"\Message:="Espera di_EscanerFinalizado_2=1"\Icon:=iconError;
          ENDIF
          !
          bEscanerHecho:=TRUE;
          RESET do_Escaner_2;
        ELSE
          WaitUntil di_EscanerFinalizado_2=1\Visualize\Header:="ESCANER
ACTIVO"\Message:="Espera di_EscanerFinalizado_2=0"\Icon:=iconError;
          bEscanerHecho:=TRUE;
        ENDIF
        !*********************************************
      DEFAULT:
      ENDTEST
      !
    ENDPROC

    !*****************************************************************************
    !* Procedimiento Abrir_Bridas()
    !*
    !* Abrir Bridas del Manipulador & Comprobacion
    !*****************************************************************************
    PROC Abrir_Bridas()
      !Abrir bridas
      Reset do_Cerrar_Bridas;
      Set do_Abrir_Bridas;
      waittime 0.5;
      !
      WaitUntil di_Bridas_Abiertas=1\Visualize\Header:="ESPERANDO BRIDAS
ABIERTAS"\Message:="Espera di_Bridas_Abiertas=1"\Icon:=iconError;
      bManipuladorOK:=FALSE;
      !
    ENDPROC

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

```
!* Procedimiento Cerrar_Bridas()
!*
!* Cerrar Bridas del Manipulador & Comprobacion
!*************************************************************************
PROC Cerrar_Bridas()
  !Cerrar bridas
  Reset do_Abrir_Bridas;
  Set do_Cerrar_Bridas;
  !
  WaitUntil di_Bridas_Cerradas=1\Visualize\Header:="ESPERANDO BRIDAS
CERRADAS"\Message:="Espera di_Bridas_Cerradas=1"\Icon:=iconError;
  bManipuladorOK:=TRUE;
  !
ENDPROC

!*************************************************************************
!* Procedimiento ManipuladorListo()
!*
!* Comprobación del correcto estado del manipulador
!*************************************************************************
PROC ManipuladorListo()
  !Comprobacion de correcto estado del manipulador
  Reset do_Cerrar_Bridas;
  Set do_Abrir_Bridas;
  WaitTime 0.1;
  Reset do_Abrir_Bridas;
  Set do_Cerrar_Bridas;
  waittime 0.1;
  !
ENDPROC

!*************************************************************************
!* Procedimiento Dejada_Emergencia()
!*
!* Dejada de disco en plataforma de emergencia
!*************************************************************************
PROC Dejada_Emergencia()
  MoveAbsJ jPounce_Rack_1,vmax,fine,t_Manipulador\WObj:=wobj0;
  MoveJ Offs(p_DejadaEmergencia,0,0,50), vMovimiento, z20,
t_Manipulador\WObj:=wobj0;
  MoveL p_DejadaEmergencia, vLento, fine, t_Manipulador\WObj:=wobj0;
  !Cerrar bridas
  Cerrar_Bridas;
  !
  bDiscoMontado:=FALSE;
```

```
    !
    MoveL p_DejadaEmergencia,vLento,fine,t_Manipulador\WObj:=wobj0;
    MoveL Offs(p_DejadaEmergencia,0,0,50),vMovimiento,z20,t_Manipulador\WObj:=wobj0;
    !
    MoveAbsJ jPounce_Rack_1,vmax,fine,t_Manipulador\WObj:=wobj0;
    MoveAbsJ jHome,vmax,fine,t_Manipulador\WObj:=wobj0;
    !
  ENDPROC
ENDMODULE
```

3. MÓDULO DE SISTEMA

```
MODULE EscribirTC(SYSMODULE)
  !
  !*****************************
  ! Escribir Tiempo Ciclo
  !*****************************
  !
  LOCAL VAR string strArchivo;
  LOCAL VAR iodev logData;
  LOCAL PERS string strRuta:="HOME:LOG_DATA";

  ! Init
  PROC ETCinit()
    !
  ENDPROC

  ! Escribir Datos En Archivo
  PROC EscribirTCEnArchivo()
    !
    VAR string strMain;
    VAR string strTiempoCiclo;
    VAR string strTiempoCicloInit;
    VAR string strNoProg;
    VAR string strNoBlister;
    VAR string strTipoProg;
    VAR string strTipoDisco;

    TEST nNumeroProg
    CASE 1:
      strTipoProg:="Rack 1";
      IF nTipoDisco=1 strTipoDisco:="MQB";
      IF nTipoDisco=2 strTipoDisco:="MEB";
      strNoBlister:=NumToStr(nBlister_Rack_1,0);
      !
    CASE 2:
      strTipoProg:="Rack 2";
      IF nTipoDisco=1 strTipoDisco:="MQB";
      IF nTipoDisco=2 strTipoDisco:="MEB";
```

```
      strNoBlister:=NumToStr(nBlister_Rack_2,0);
      !
    ENDTEST

    strNoProg:=NumToStr(nNumeroProg,0);
    strTiempoCiclo:=NumToStr(TCiclo,2);
    strTiempoCicloInit:=NumToStr(TCicloInit,2);
    strNoBlister:=NumToStr(n,2);

    ComprobarDir;

    strArchivo:=strRuta+"/"+"TFG_JOAN"+"_"+CDate()+".csv";

    IF IsFile(strArchivo)=FALSE THEN
      EscribirTitulos;
    ENDIF

    strMain:=strMain+strNoProg+";";
    strMain:=strMain+strTipoProg+";";
    strMain:=strMain+strTipoDisco+";";
    strMain:=strMain+strNoBlister+";";
    strMain:=strMain+strTiempoCiclo+";";
    strMain:=strMain+strTiempoCicloInit+";";

    Open strArchivo,logData\Append;
    Write logData,strMain;
    Close logData;

  ERROR
    EscribirTitulos;
    TRYNEXT;

  ENDPROC

  ! Escribir Titulos
  LOCAL PROC EscribirTitulos()

    Open strArchivo,logData\Write;
    Write logData,"NoPrograma;TipoPrograma;TipoDisco;NoBlister;"\NoNewLine;
    Write logData,"TiempoCiclo;TiempocicloInit";
    Close logData;
  ENDPROC

  ! Comprobar Directorio
  LOCAL PROC ComprobarDir()
```

```
    IF isFile(strRuta\Directory)=FALSE THEN
      MakeDir strRuta;
    ENDIF

  ERROR
    MakeDir strRuta;
    RETURN ;
  ENDPROC
ENDMODULE
```

ANEXO III: COMPONENTES INTELIGENTES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº7: ANEXO III: COMPONENTES INTELIGENTES

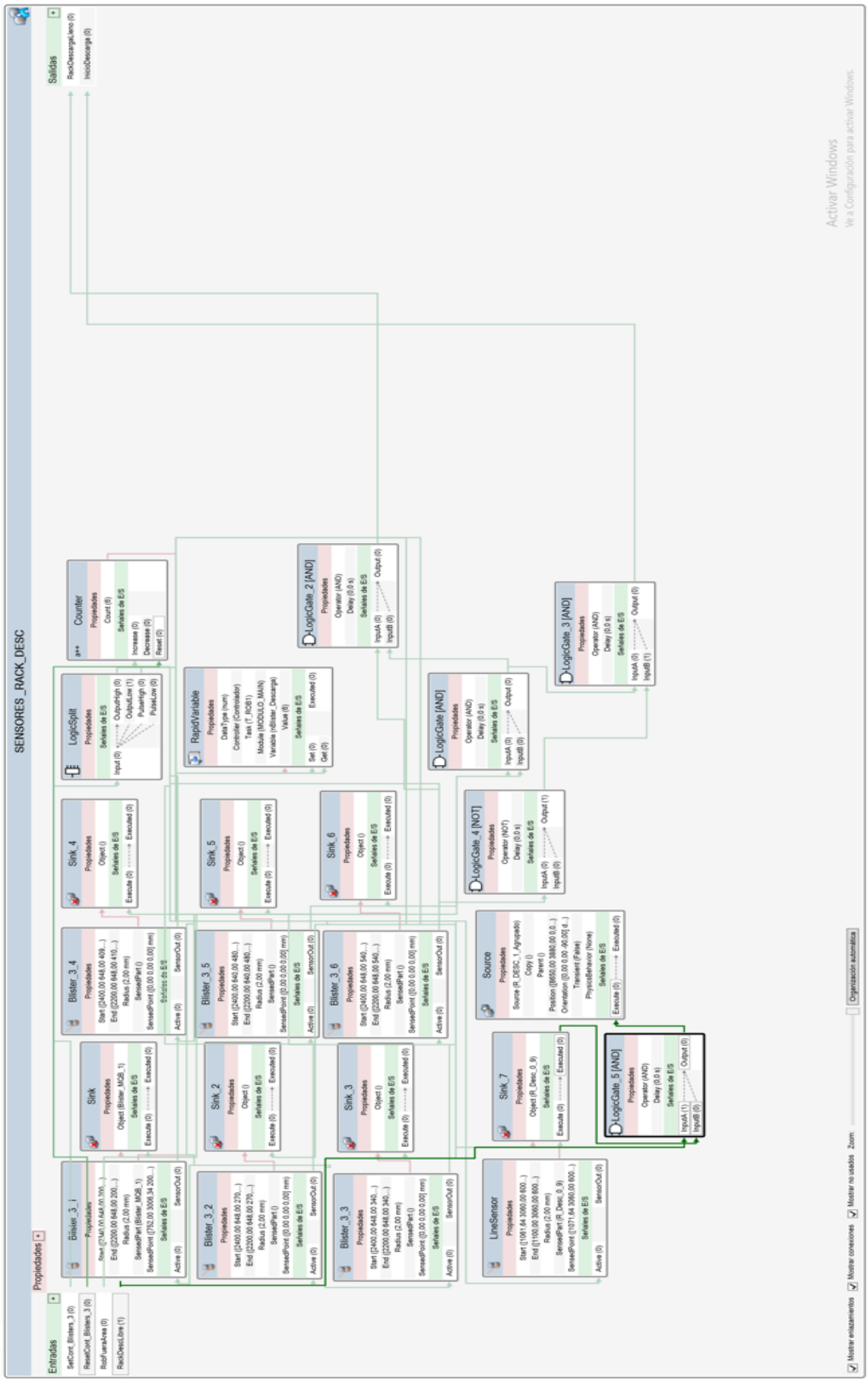


Figura 21.1 Diseño del componente Sensores_Rack_Desc en RobotStudio

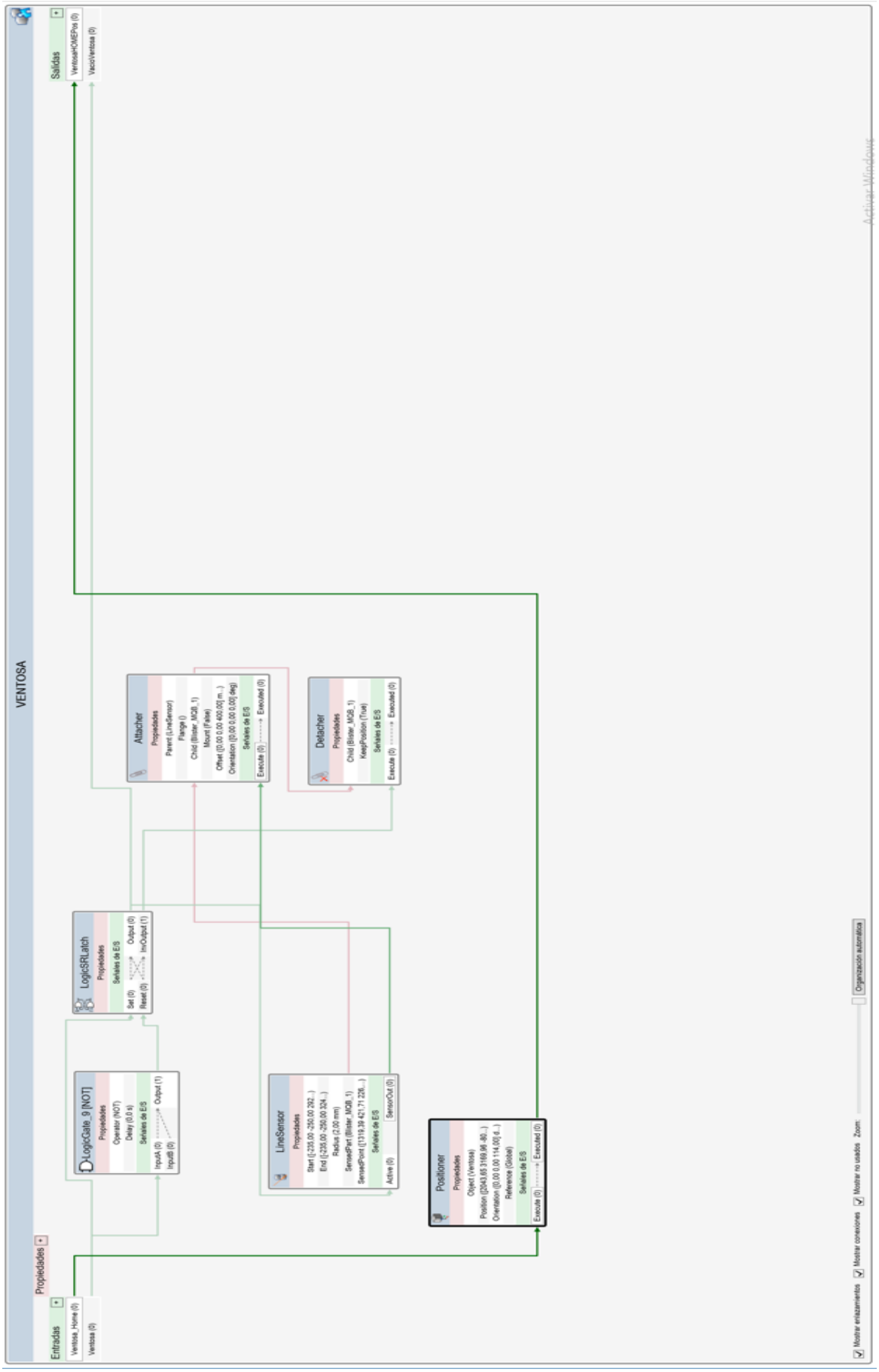


Figura 21.2 Diseño del componente Ventosa en RobotStudio

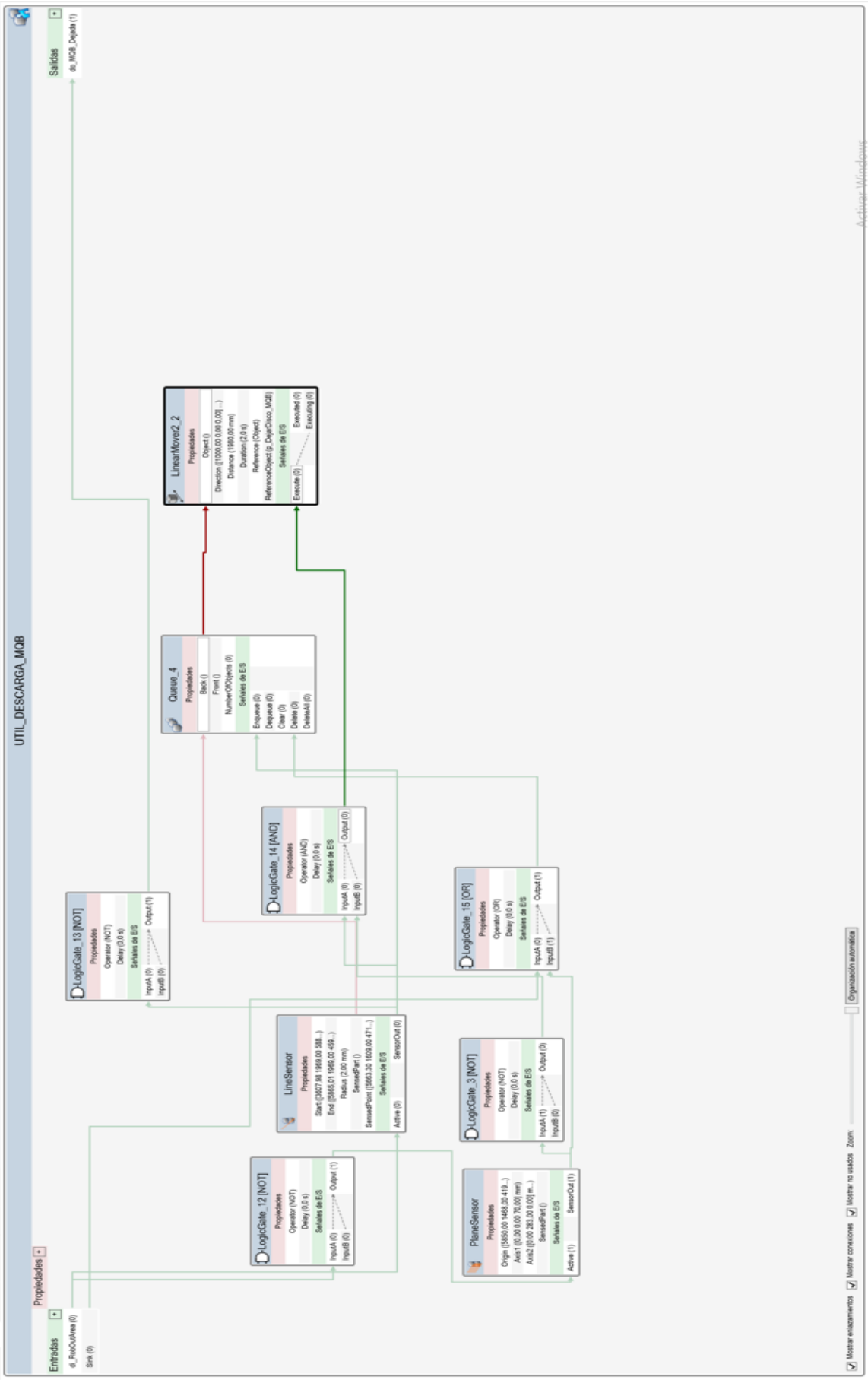


Figura 21.3 Diseño del componente Útil_Descarga MEB/MQB en RobotStudio

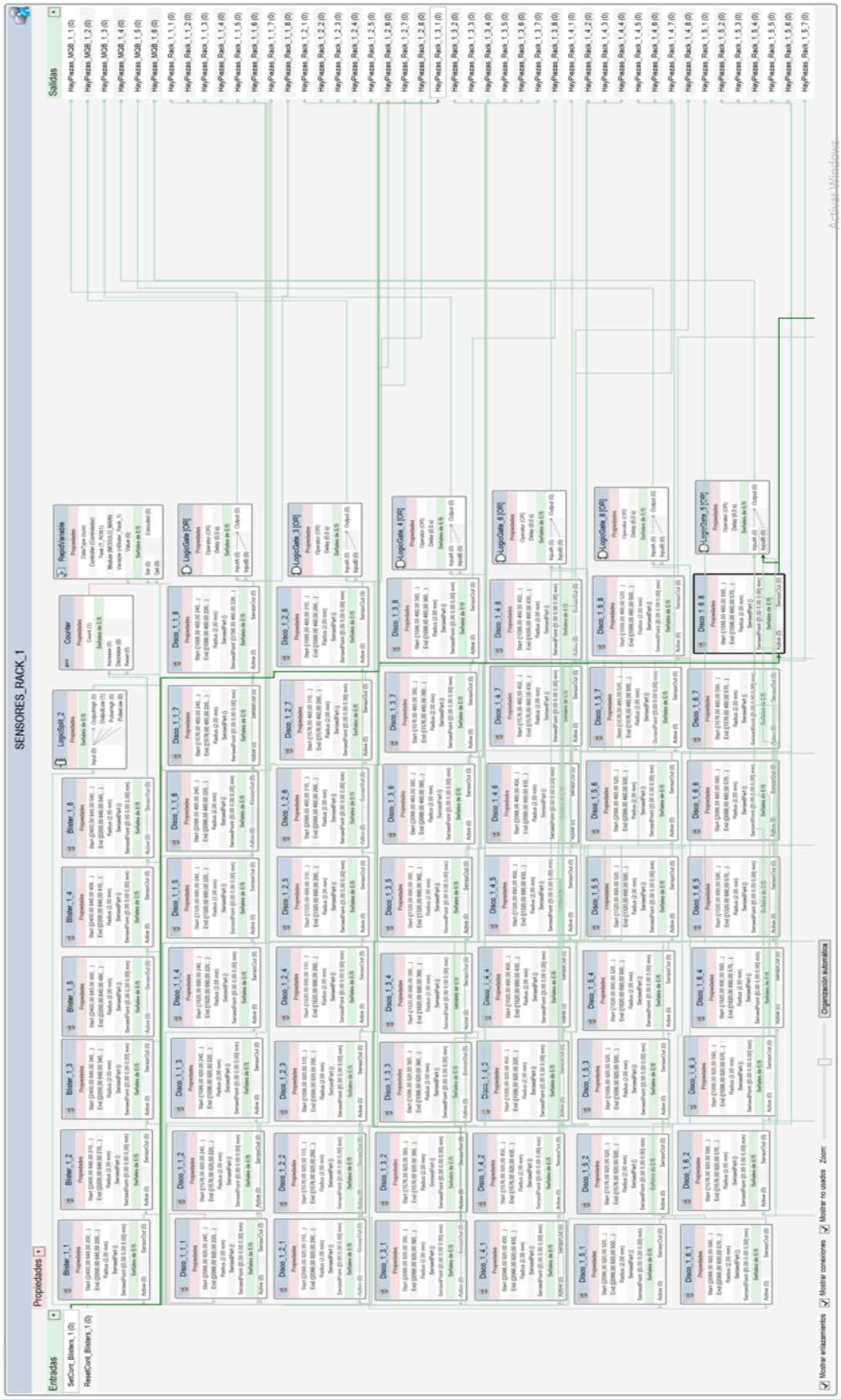


Figura 21.4 Diseño del componente Sensores_Rack_1 en RobotStudio

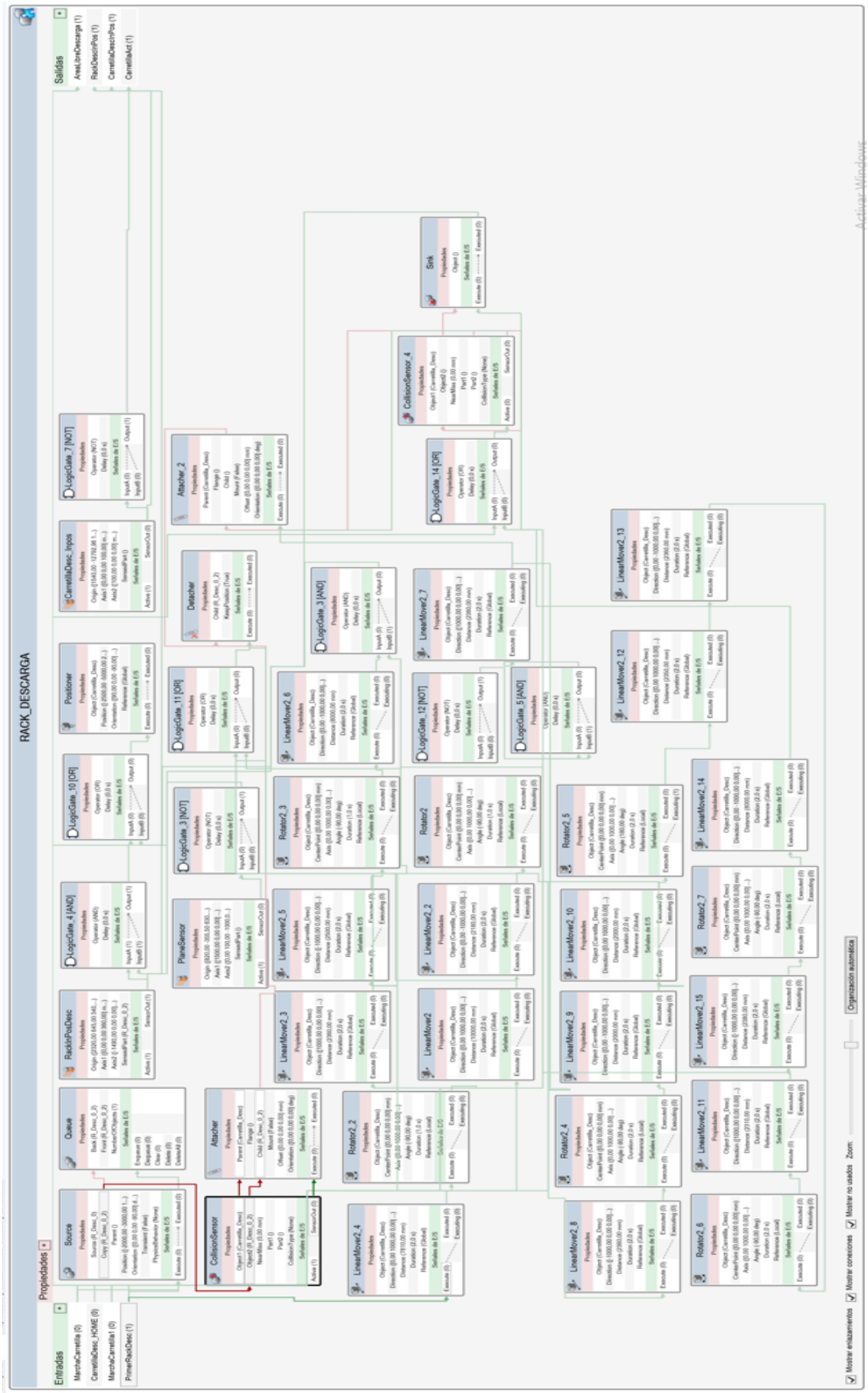


Figura 21.5 Diseño del componente Rack_Descarga en RobotStudio

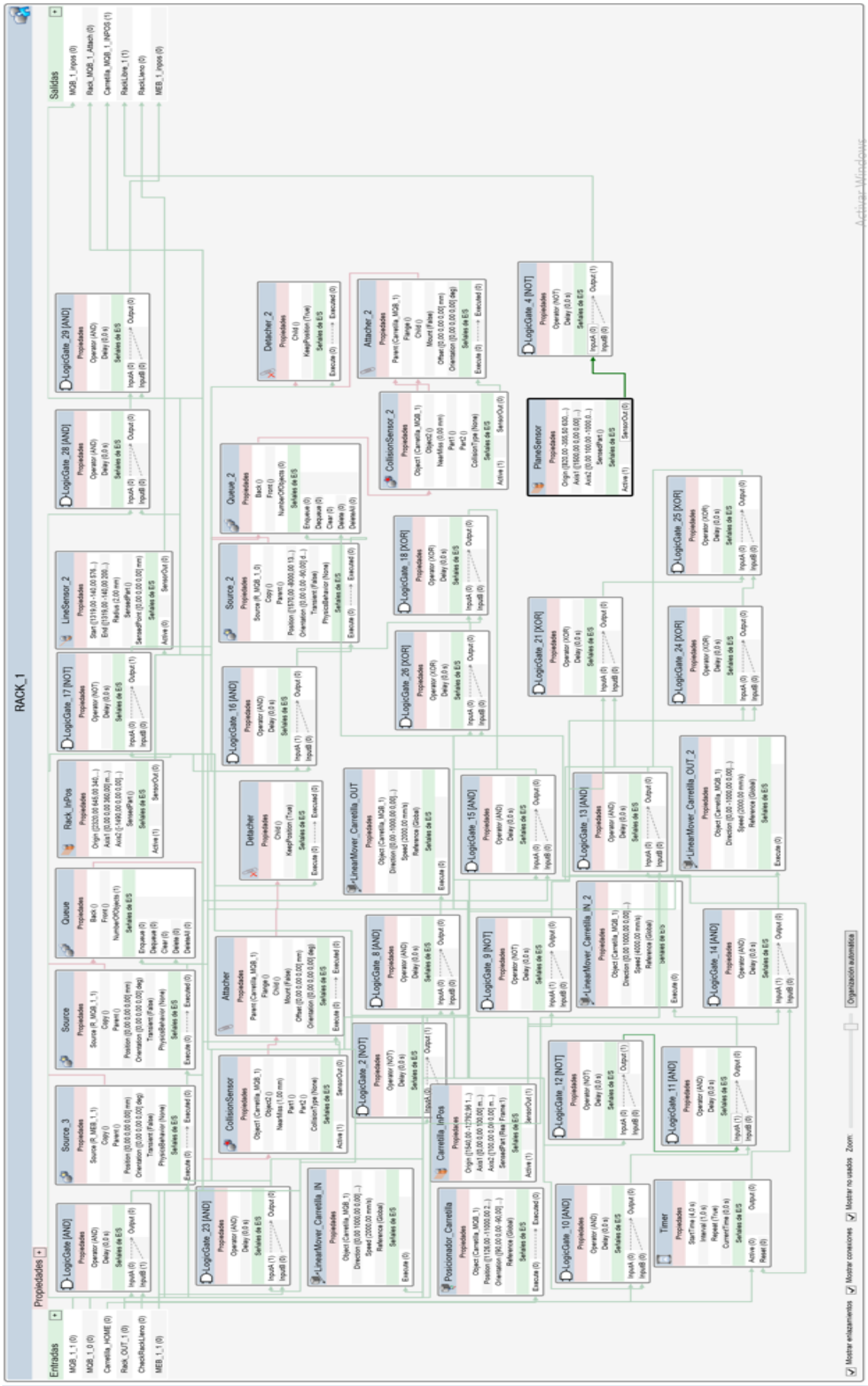


Figura 21.6 Diseño del componente Rack_1 en RobotStudio

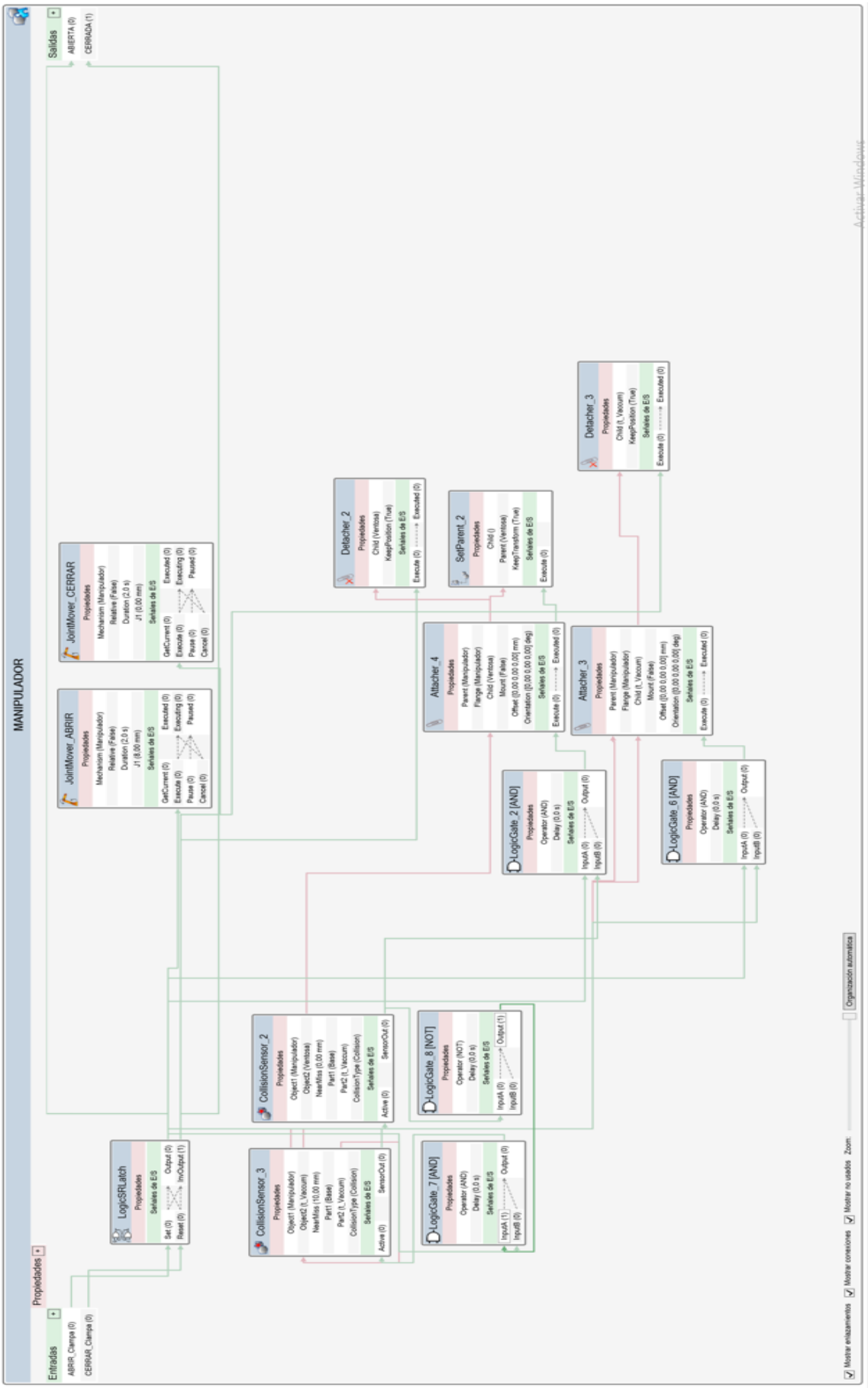


Figura 21.7 Diseño del componente Manipulador en RobotStudio

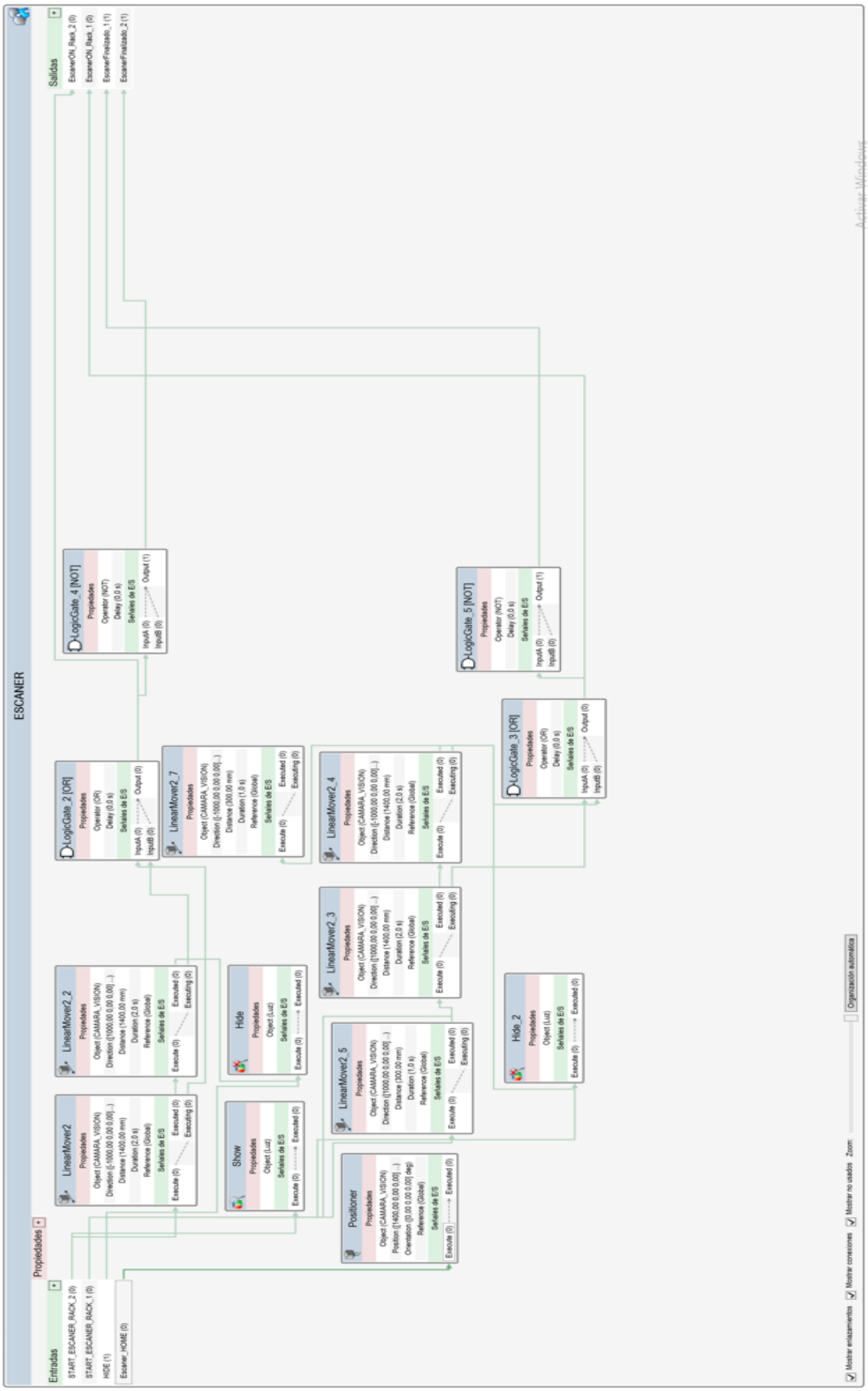


Figura 21.8 Diseño del componente Escáner en RobotStudio

ANEXO IV: MANUAL DEL USUARIO DE LOS APARTADOS EN LA SIMULACIÓN DE UNA ESTACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

PROYECTO DE AUTOMATIZACIÓN DE UNA LÍNEA DE PRODUCCIÓN DE DISCOS DE FRENO MEDIANTE LA PROGRAMACIÓN EN ROBOTSTUDIO DE UN ROBOT INDUSTRIAL ABB MODELO IRB4600_60_205

AUTOR: JOAN CALVO HERRERO

TUTOR: JOSÉ VICENTE SALCEDO ROMERO DE ÁVILA

Curso Académico: 2020-21

ÍNDICE DOCUMENTO Nº8: ANEXO IV: MANUAL DEL USUARIO DE LOS APARTADOS EN LA SIMULACIÓN DE UNA ESTACIÓN

1. ENSAMBLAJE MANIPULADOR

El ensamblaje del elemento Manipulador está formado por dos geometrías independientes a las cuales se les aplican una serie de uniones y restricciones formando el ensamblaje. Éstas dos geometrías son Base_Manipulador y Bridas, las cuales han sido diseñadas específicamente para poder coger todos los modelos de los discos, así como la herramienta Ventosa. Es por ello que las dimensiones de ambas han sido adaptadas explícitamente a dichas circunstancias que vienen proporcionadas por el cliente.

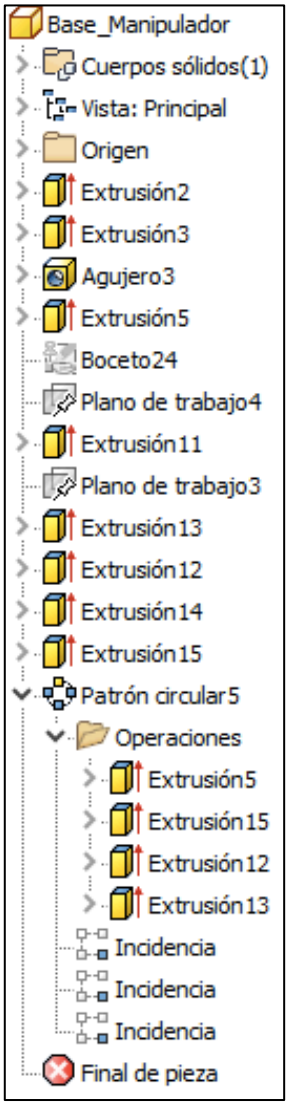


Figura 22.1 Modelado Base_Manipulador en Inventor

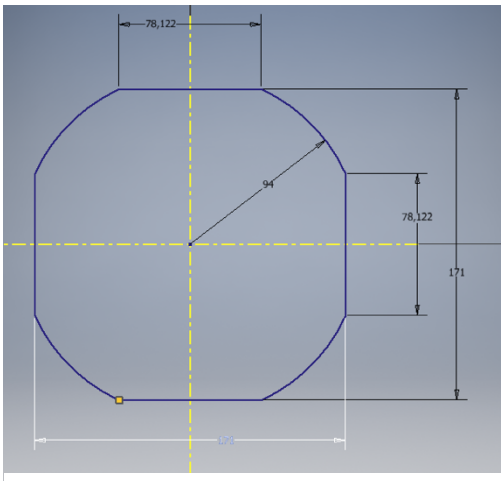


Figura 22.2 Boceto Base_Manipulador en Inventor

La geometría que forma la base del manipulador va acoplada al flanje del robot por la parte inferior de la misma. El boceto principal está formado mediante un solape de un círculo de 188mm de diámetro y un cuadrado con lados de 171mm.

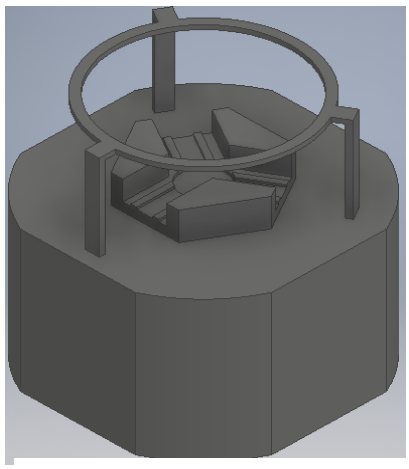


Figura 22.3 Geometría Base_Manipulador

Sobre este boceto se realiza una extrusión de 100mm en la dirección del eje del centro del boceto. Sobre un hexágono de 80mm y 20mm de altura se realizan 3 incisiones y las guías por las que se desplazan las bridas mediante un patrón circular de 120º. El círculo superior fija siempre la correcta posición de cogida y protege las bridas frente a colisiones. Para ello se realiza una extrusión de 3mm de la superficie entre dos circunferencias y se añaden unos soportes siguiendo el mismo patrón circular anterior.

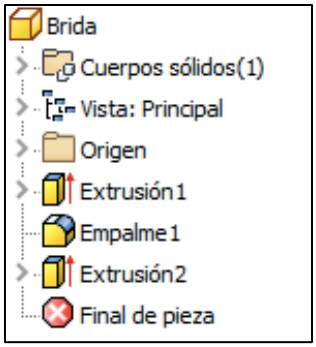


Figura 22.4 Modelado Brida en Inventor

La brida se realiza mediante una extrusión de 30mm sobre el boceto principal formado por el perfil de la geometría sobre su eje de simetría. Por ello la extrusión se realiza extrusionando la misma distancia por los dos lados del perfil. Por la parte inferior se realizan las incisiones con las que se acopla y deslizan las bridas a la base del manipulador. En la parte superior se realizan empalmes de 2mm en la parte por la que se cogen los discos facilitando así el ajuste de ambos. La altura de la brida y del hueco donde se encajan los discos se realiza con las medidas y a distancia exactas para realizar la cogida.

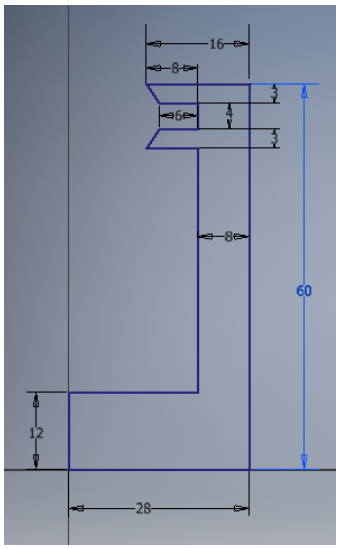


Figura 22.5 Boceto Brida en Inventor

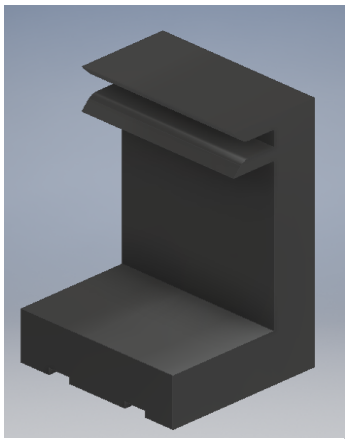


Figura 22.6 Geometría Brida

Una vez modeladas las geometrías se realiza un ensamblaje mediante la unión de las bridas en el hueco de la base del manipulador establecido para ello. Se utilizan ciertas restricciones de coincidencia de superficies y ejes, así como nivelación entre ambas. La brida cuenta con los grados de libertad necesarios para realizar el desplazamiento sobre las guías de 8mm de distancia simulando cuando están abiertas o cerradas. De todas formas, este procedimiento habrá que realizarlo de nuevo al crear el manipulador en RobotStudio. Para finalizar se realiza un patrón circular de la brida sobre el eje vertical del centro de la base del manipulador a 120º para crear las 3 bridas utilizadas.

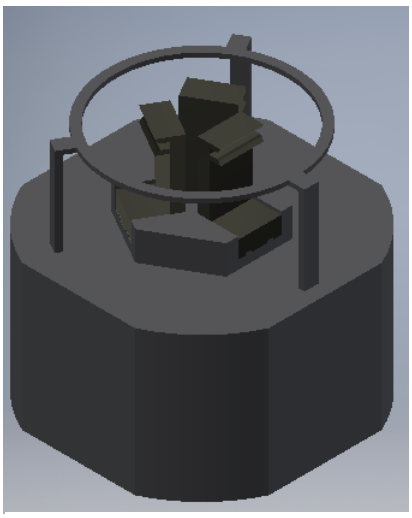


Figura 22.8 Ensamblaje Manipulador

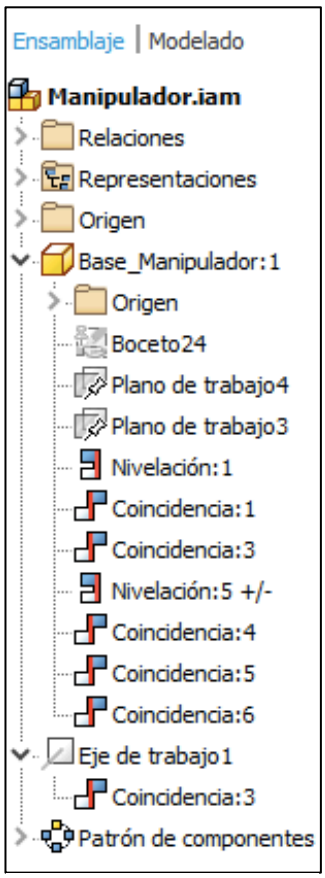


Figura 22.7 Ensamblaje Manipulador en Inventor

2. COMPONENTES INTELIGENTES

2.1 SEÑALES Y PROPIEDADES

En este apartado se explican con detalle las propiedades de los componentes inteligentes que hemos utilizado en la estación.

Señales y propiedades »

LogicGate

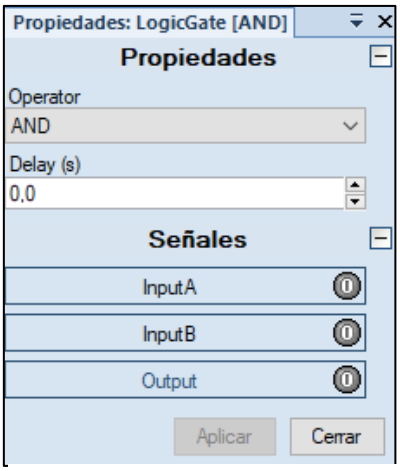


Figura 23.1 Propiedades LogicGate

Realiza una operación lógica con las señales digitales.

Operator: Operador lógico (AND,NOT,OR,XOR,NOP)

Delay : Tiempo de retardo antes de que la salida sea cambiada. Se mide en segundos.

InputA: Primera entrada
InputB: Segunda entrada

Output: Salida. Resultado de la operación.

Timer

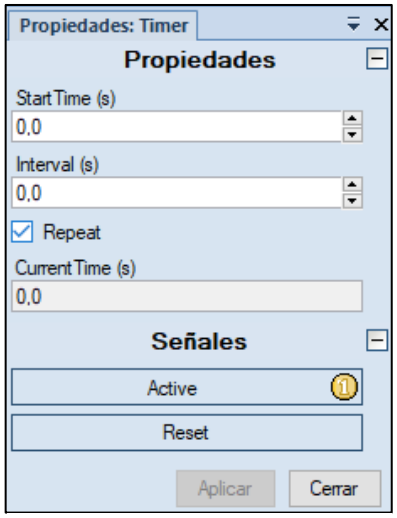


Figura 23.2 Propiedades Timer

Pulsa una señal digital con un intervalo de tiempo especificado durante la simulación.

Start Time: Tiempo que debe transcurrir antes del primer pulso
Interval: Intervalo entre impulsos. Se mide en segundos.
Repeat: Especifica si la señal debe pulsarse repetidamente o sólo una vez.
Current Time: Genera el tiempo de acción actual del temporizador en segundos.
Active: Activa el temporizador en alto (1)
Reset: Resetea el temporizador en alto (1)
Salida: Cambia a alto (1) y luego a bajo (0) con el intervalo especificado.

Sensores »

CollisionSensor

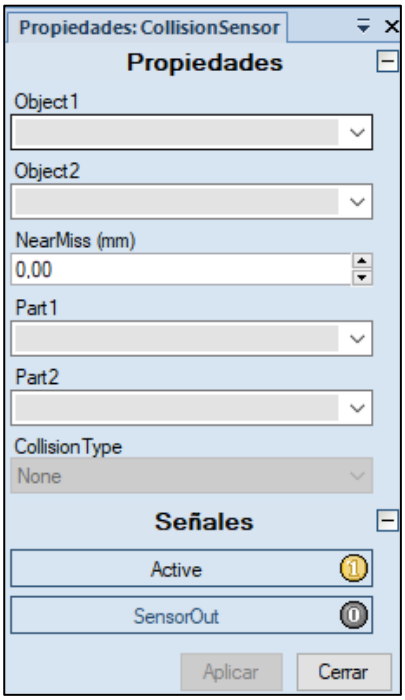


Figura 23.3 Propiedades CollisionSensor

Detecta la colisión entre objetos.

Object1: Primer objeto
Object2: Segundo objeto. Null para detectar con respecto a toda la estación.

NearMiss: Distancia de casi colisión o cero para detectar colisiones. Se mide en milímetros.

Part1: Primera pieza en colisión
Part2: Segunda pieza en colisión

CollisionType: Colisión (2), casi colisión (1) o ninguno (0)

Active: Activa el sensor en alto (1)
SensorOut: Cambia a alto (1) cuando se produce una colisión.

LineSensor

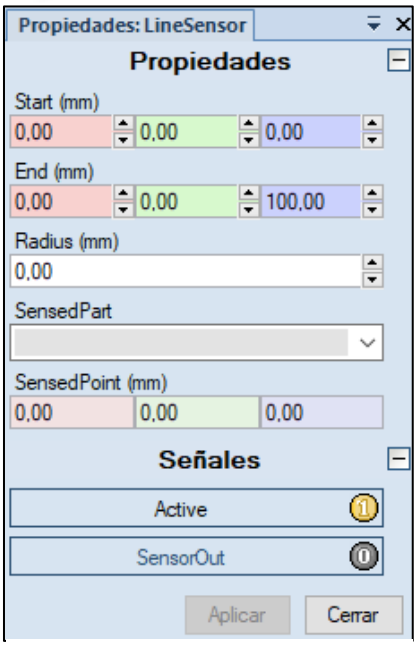


Figura 23.4 Propiedades LineSensor

Detecta si algún objeto corta una línea que une dos puntos.

Start: Vector que define el punto de inicio en milímetros.
End: Vector que define el punto de final en milímetros.
Radius: Radio del sensor en milímetros.

SensedPart: Define la pieza detectada por el sensor.
SensedPoint: Define el vector del punto en el que la línea corta la pieza más cercana en milímetros.

Active: Activa el sensor en alto (1)
SensorOut: Cambia a alto (1) cuando un objeto corta la línea.

PlaneSensor

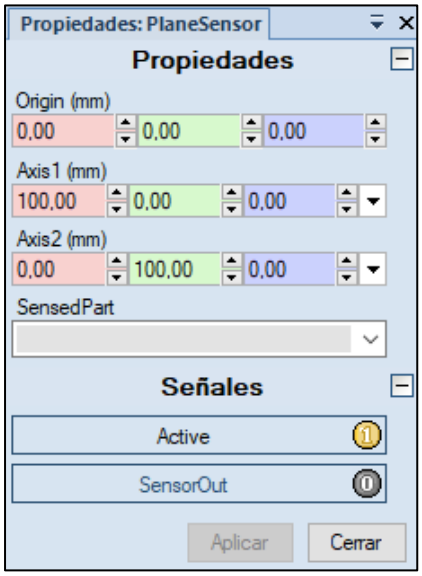


Figura 23.5 Propiedades PlaneSensor

Detecta si algún objeto corta un plano

Origin: Origen del plano en milímetros.

Axis1: Vector que define el punto del primer eje del plano.
Axis2: Vector que define el punto del segundo eje del plano.

SensedPart: Define la pieza detectada por el sensor.

Active: Activa el sensor en alto (1)
SensorOut: Cambia a alto (1) cuando un objeto corta el plano.

Acciones »

Attacher

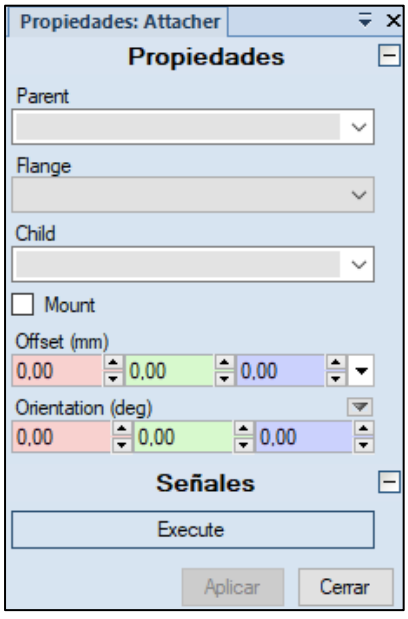


Figura 23.6 Propiedades Attacher

Conecta un objeto.

Parent: Objeto superior de conexión.
Flange: Brida de mecanismo al que conectar.
Child: Objeto a conectar.

Mount: Mueve el objeto hasta la brida superior de conexión.

Offset: Vector que define la posición relativa al objeto superior de conexión al utilizar Mount, en milímetros.
Orientation: Orientación relativa al objeto superior de conexión al utilizar Mount, en milímetros.

Execute: Entrada. En alto (1) para activa el attacher.
Executed: Salida. En alto (1) al completarse la operación.

Detacher

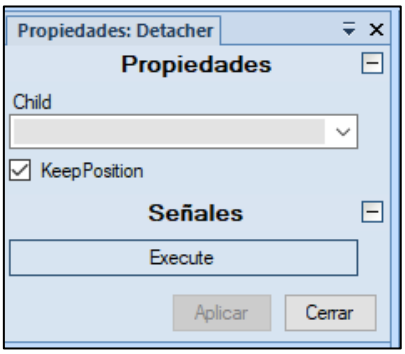


Figura 23.7 Propiedades Detacher

Desconecta un objeto conectado.

Child: Objeto conectado.
KeepPosition: Si no se activa, el objeto conectado vuelve a la posición original.

Execute: Entrada. En alto (1) para eliminar la conexión del objeto Child.
Executed: Salida. En alto (1) al completarse la operación.

Source

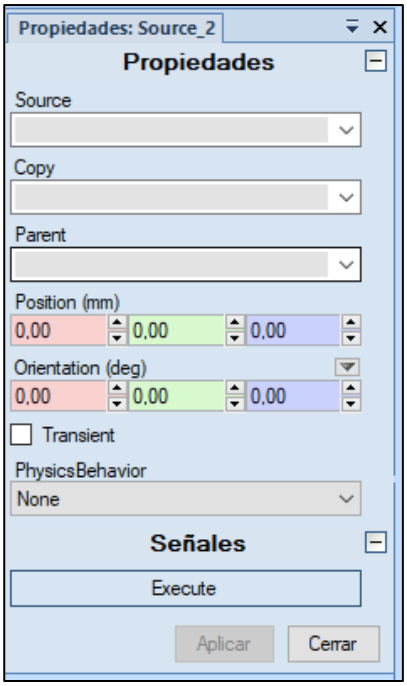


Figura 23.8 Propiedades Source

Crea una copia de un componente gráfico.

Source: Fuente que define el objeto a copiar.
Copy: Contiene el objeto copiado.
Parent: Donde se debe añadir la copia o vacío si debe tener el mismo objeto superior que el origen.

Position: Vector que define la posición de la copia respecto al objeto superior.
Orientation: Orientación de la copia respecto al objeto superior.
Transistent: Marca como temporales las copias creadas durante la simulación para evitar problemas de memoria.
PhysicsBehavour: Especifica el comportamiento físico de la copia.
Execute: Entrada. En alto (1) para crear la copia.

Executed: Salida. En alto (1) al completarse la operación.

Manipuladores »

LinearMover

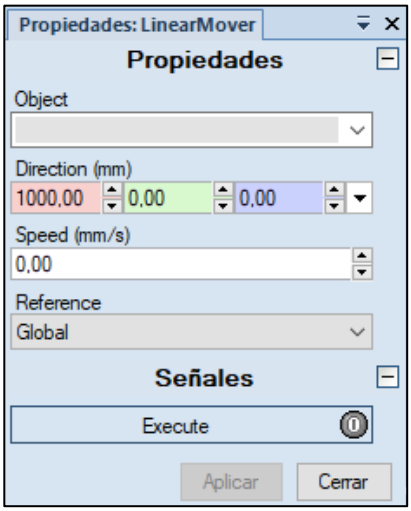


Figura 23.9 Propiedades LinearMover

Mueve un objeto en una trayectoria lineal.

Object: Objeto a mover.

Direction: Vector que define la dirección del objeto a mover en milímetros.
Speed: Velocidad en milímetros por segundo.
Reference: Sistema de coordenadas del objeto de referencia.

Execute: Entrada. En alto (1) para mover el objeto.
Executed: Salida. En alto (1) al completarse la operación.

Positioner

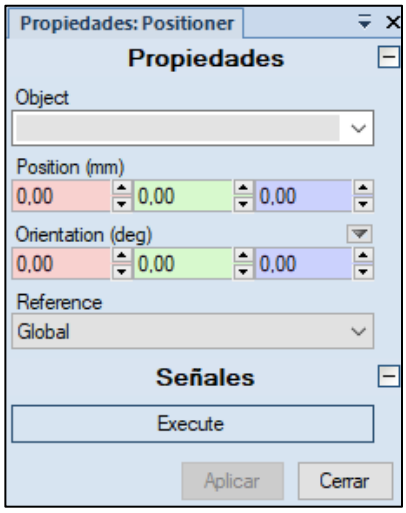


Figura 23.10 Propiedades Positioner

Define la posición y orientación de un objeto.

Object: Objeto a posicionar.

Position: Vector que define la posición del objeto.
Orientation: Orientación del objeto.
Reference: Sistema de coordenadas del objeto de referencia.

Execute: Entrada. En alto (1) para posicionar el objeto.
Executed: Salida. En alto (1) al completarse la operación.

JointMover

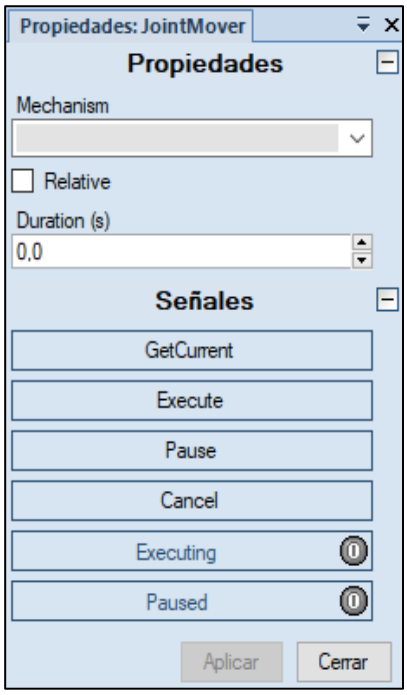


Figura 23.11 Propiedades JointMover

Mueve los ejes de un mecanismo

Mechanism: Mecanismo a mover.
Relative: Booleana que indica que los valores de los ejes son con respecto a la pose actual.
Duration: Tiempo de ejecución del movimiento.

GetCurrent: Entrada. Obtiene los valores actuales de los ejes cuando esta en alto (1).
Execute: Entrada. Inicia o reanuda el movimiento en alto (1).
Pause: Entrada. Pausa el movimiento en alto (1).
Cancel: Entrada. Cancela el movimiento en alto (1).

Executed: Salida. Cambia a alto(1) cuando se completa el movimiento.
Executing: Salida. Cambia a alto (1) durante el movimiento.
Paused: Salida. Cambia a alto (1) cuando el movimiento está pausado.

Otros »

Queue

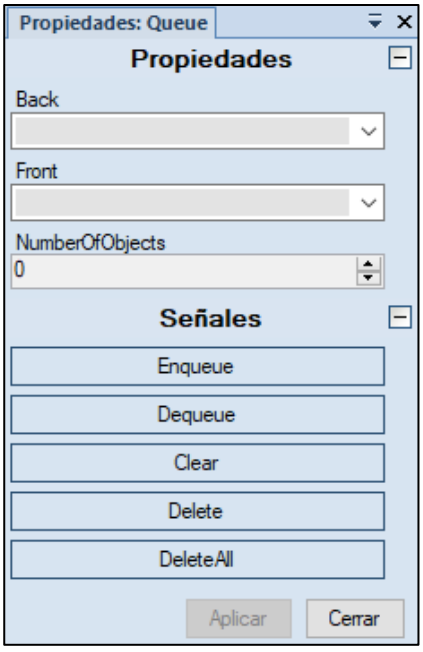


Figura 23.12 Propiedades Queue

Representa una cola de objetos que pueden manipularse como un grupo.

Back: Objeto a colocar en la cola.
Front: Primer objeto de la cola.

NumberOfObjects: Número de objetos en la cola.

Enqueue: Añade el objeto Back a la cola.
Dequeue: Elimina el objeto Front de la cola.
Clear: Vacía la cola.
Delete: Elimina el objeto Front de la cola y de la estación.
DeleteAll: Vacía la cola y elimina todos los objetos de la estación.

2.2 CREACIÓN DE COMPONENTES INTELIGENTES

Lo primero que debemos hacer antes de empezar a crear y diseñar los componentes inteligentes es posicionar los objetos a utilizar. En este caso deberemos situar en las posiciones establecidas tanto el rack lleno (componentes agrupados), el rack vacío y la carretilla elevadora tal y como se muestra en la figura siguiente.

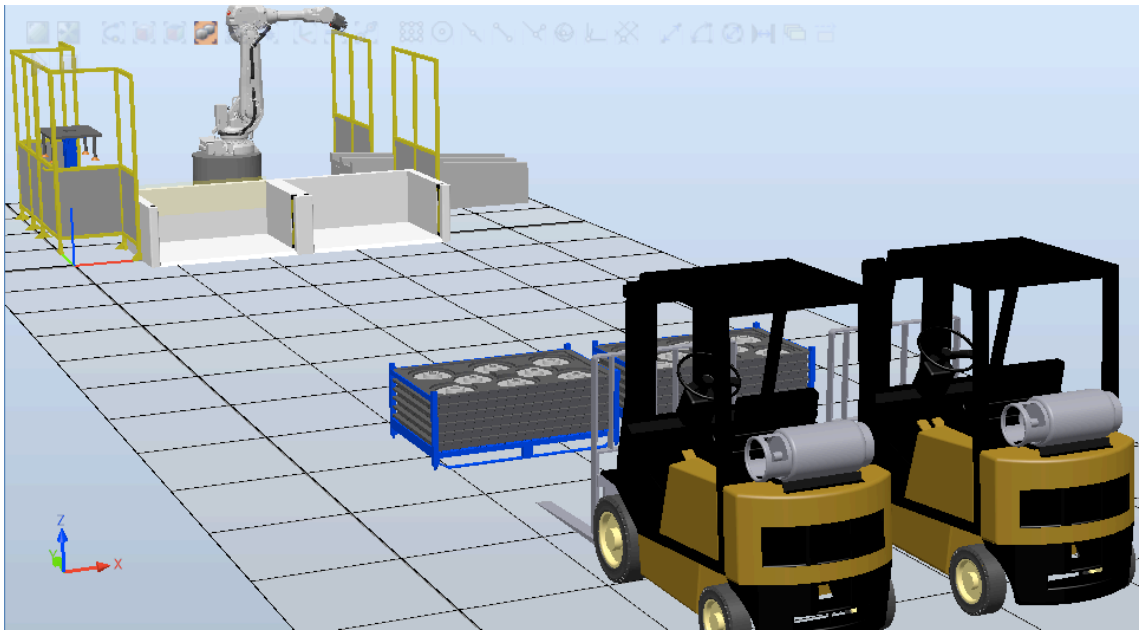


Figura 24.1 Creación de componentes inteligentes

Para crear el componente inteligente seleccionamos en la ventana de Modelado>Componente Inteligente e introducimos las geometrías a utilizar dentro del mismo arrastrándolos sobre el componente creados en la ventana de diseño de la parte izquierda.

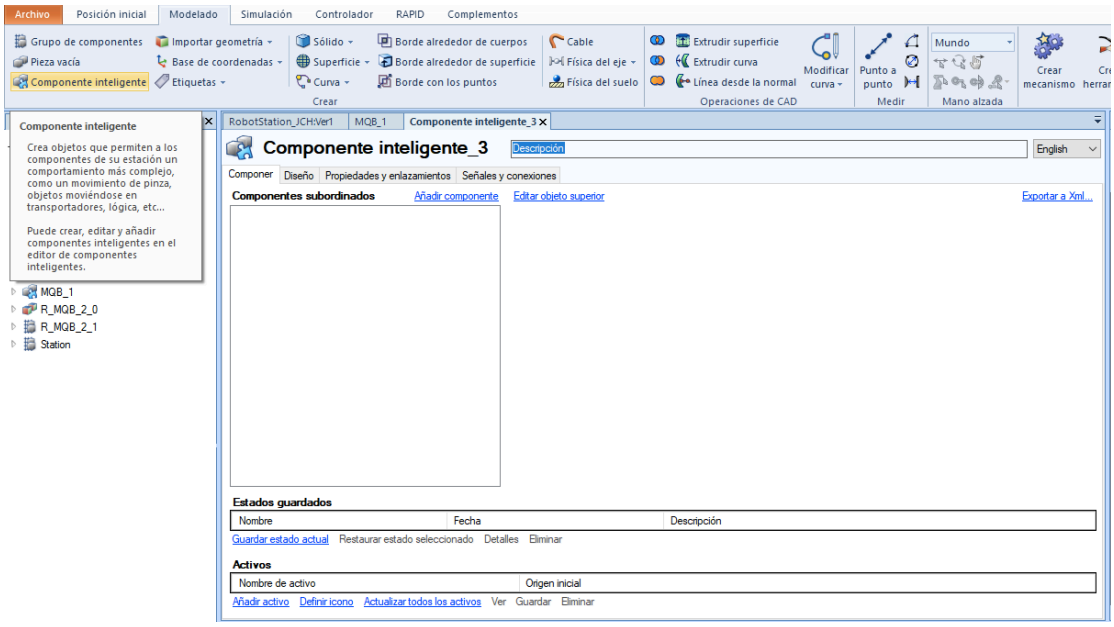


Figura 24.2 Pestaña de componente inteligente en RobotStudio

Ahora deberemos ir añadiendo los componentes que vamos necesitando durante el procedimiento. Podemos elegirlos haciendo clic en Añadir componente.

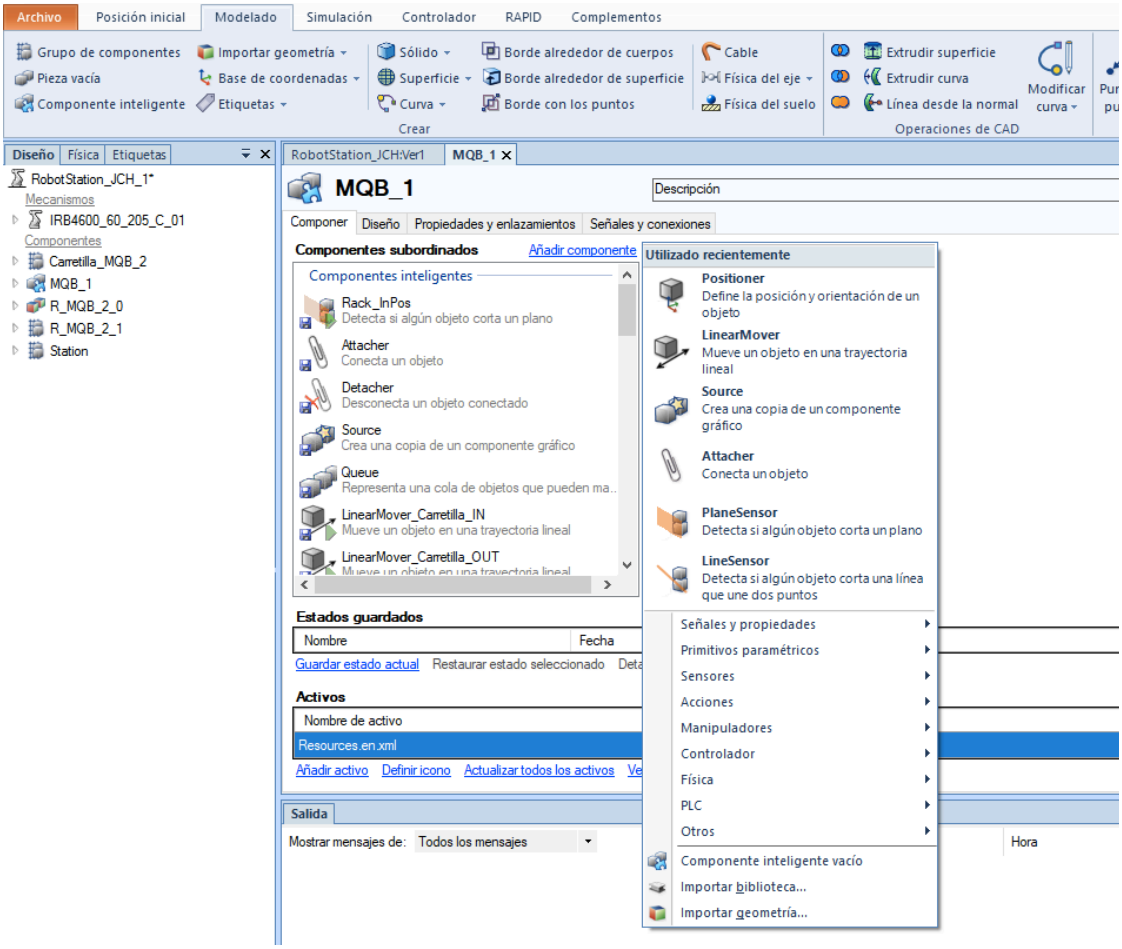


Figura 24.3 Añadir componentes inteligentes

Los componentes seleccionados irán apareciendo en la lista de componentes subordinados en la parte izquierda de la ventana. Cada componente subordinado tiene unas propiedades diferentes que se pueden modificar para adecuar dicho componente a las condiciones de funcionamiento necesitadas. En la parte derecha de la ventana podemos encontrar las propiedades de cada componente subordinado sobre el que hagamos clic.

Podemos crear señales de entrada y salida para gestionar los componentes subordinados utilizados. De esta manera podemos activar un componente mediante una entrada y recibir señales de salida como efecto de la ejecución del componente.

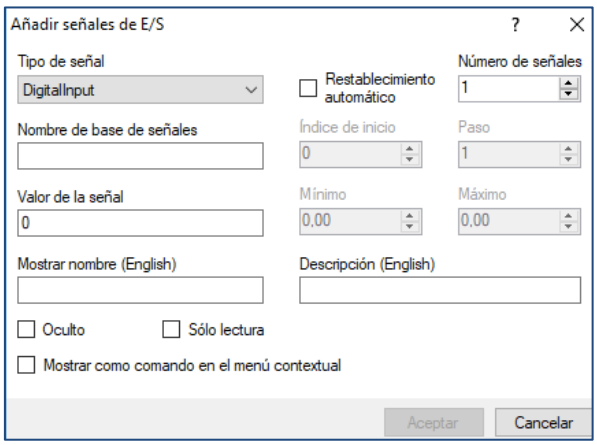


Figura 24.4 Añadir señales de E/S

Para visualizarlos de forma sencilla podemos acceder a la pestaña de Diseño. Posibilidad de enlazar mediante flechas de unión en esta pestaña o desde Señales y conexiones según nos resulte más cómodo.

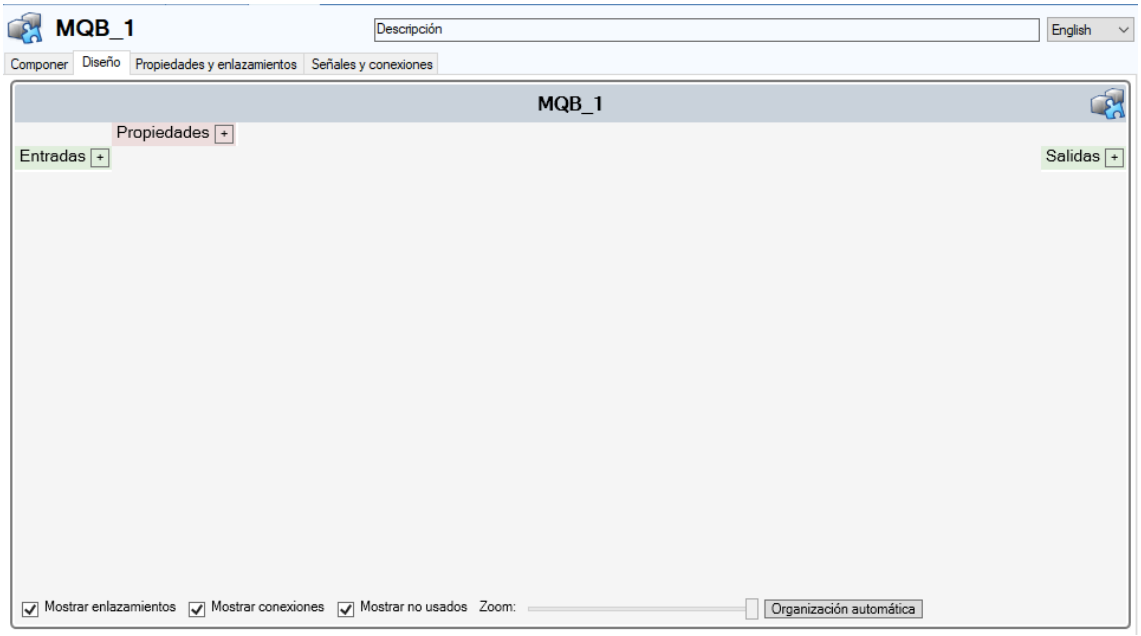


Figura 24.5 Pestaña de diseño para componentes inteligentes

Lo anteriormente explicado queda claro con el siguiente ejemplo, en el que al accionar la entrada A, el objeto Carretilla_MQB_2 se posicionara en el lugar establecido, activando como resultado la salida B cuando la operación se haya completado. Como se observa se pueden crear las conexiones rápidamente haciendo clic en la señal de entrada y arrastrando la flecha a la posición de ejecución del componente.

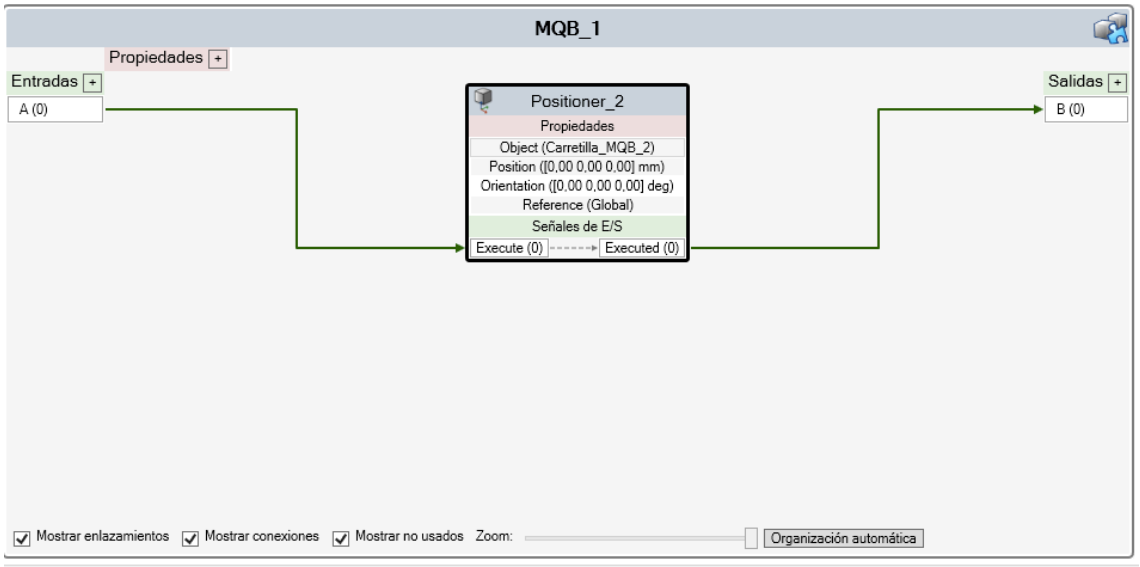


Figura 24.6 Diseño de componentes inteligentes

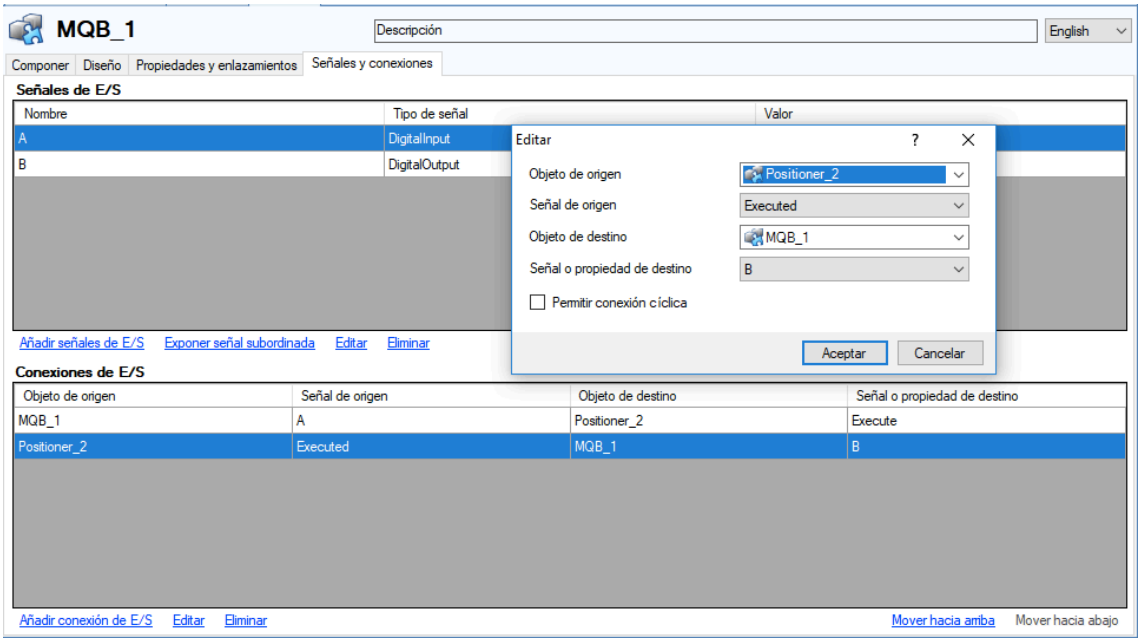


Figura 24.7 Pestaña de Señales y conexiones de SC

3. TRAYECTORIAS Y PUNTOS

Basándonos en un ejemplo vamos a explicar con detalle el procedimiento a seguir para crear una rutina de movimientos. En este ejemplo el robot, partiendo y terminando en Home, recorrerá las posiciones de todas las piezas de un blíster en orden.

Primero establecemos un punto llamado j_Home el cual marca la posición inicial del robot. Para ello seleccionamos Posición inicial>Programar posición, en la configuración de ejes por defecto del robot. También añadimos las posiciones para todos los discos del primer blíster desde Posición inicial>Posición>Crear punto. Se pueden establecer las posiciones utilizando los comandos proporcionados por el programa, en este caso el de círculo concéntrico.

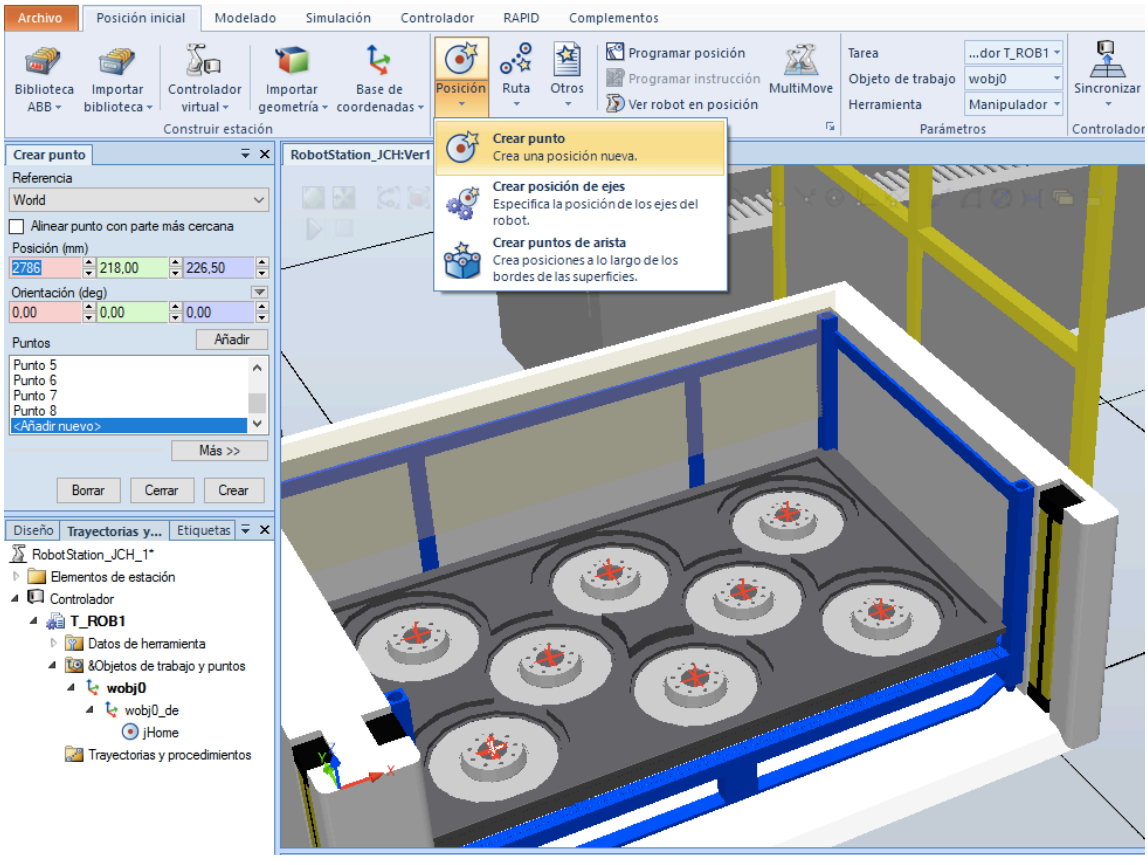


Figura 25.1 Crear un punto en RobotStudio

Haciendo clic con el botón derecho sobre los puntos y seleccionando la opción correspondiente, podemos colocar la herramienta o el robot en el punto para configurar orientaciones y comprobar accesibilidad.

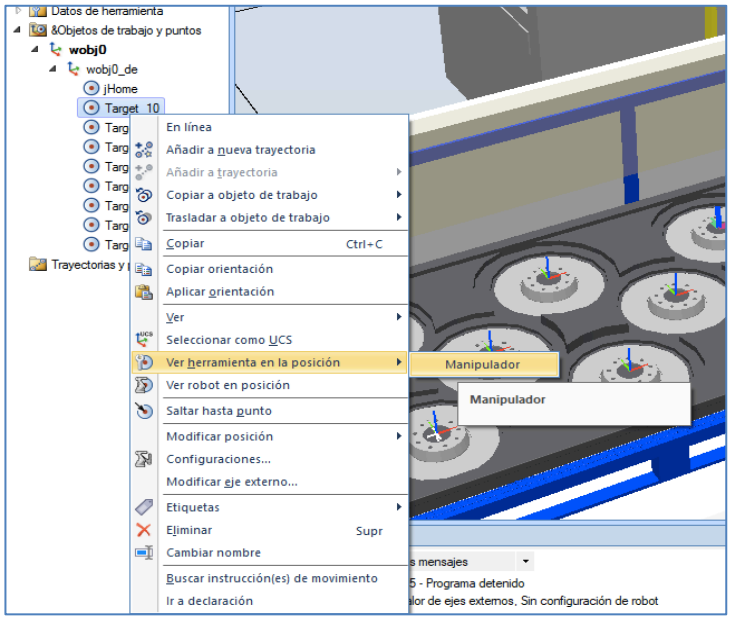


Figura 25.2 Ver herramienta en posición en RobotStudio

Figura 25.3 Puntos creados en RobotStudio

Podemos crear una trayectoria desde Posición inicial>Ruta>Trayectoria vacía y arrastrar los puntos anteriormente creados para formarla como se muestra en el ejemplo siguiente. *Path_10* es el nombre de la trayectoria que se crea por defecto.

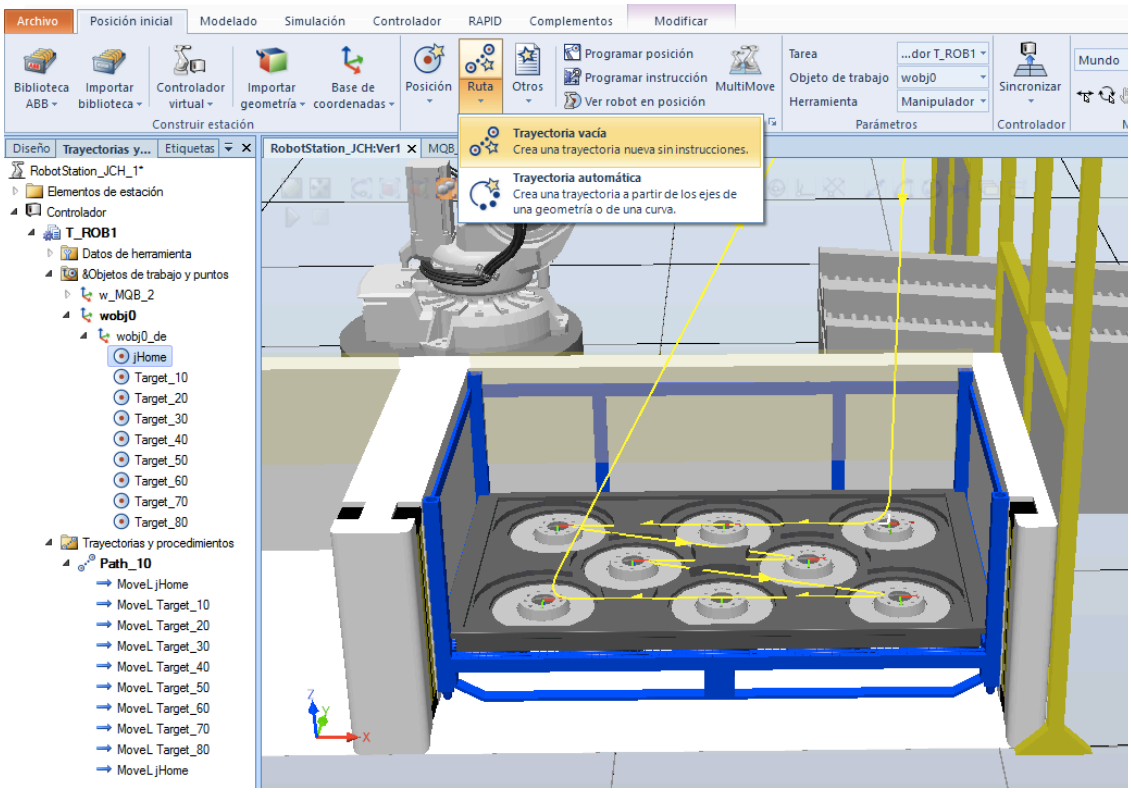


Figura 25.4 Crear trayectorias en RobotStudio

Una vez seleccionada la correcta orientación de la herramienta en el punto podemos copiar y pegar la orientación en el resto de puntos, evitando así que el robot pueda realizar giros extraños de los ejes debido a distintas orientaciones de los mismos en cada punto.

Para que las posiciones queden vinculadas al rack, vamos a establecer un nuevo *workobject* para dicho rack. El sistema de referencia del objeto de trabajo estará situado en la posición de cogida del disco 1 del primer blíster para cada caso. A partir de estos objetos de trabajo quedaran referenciadas todas las posiciones del resto de piezas.

Para ello debemos crear una nueva posición en el punto del primer disco como se ha hecho anteriormente. Una vez creada hacemos clic sobre la opción Convertir punto en objeto de trabajo, tal y como se muestra en la figura siguiente.

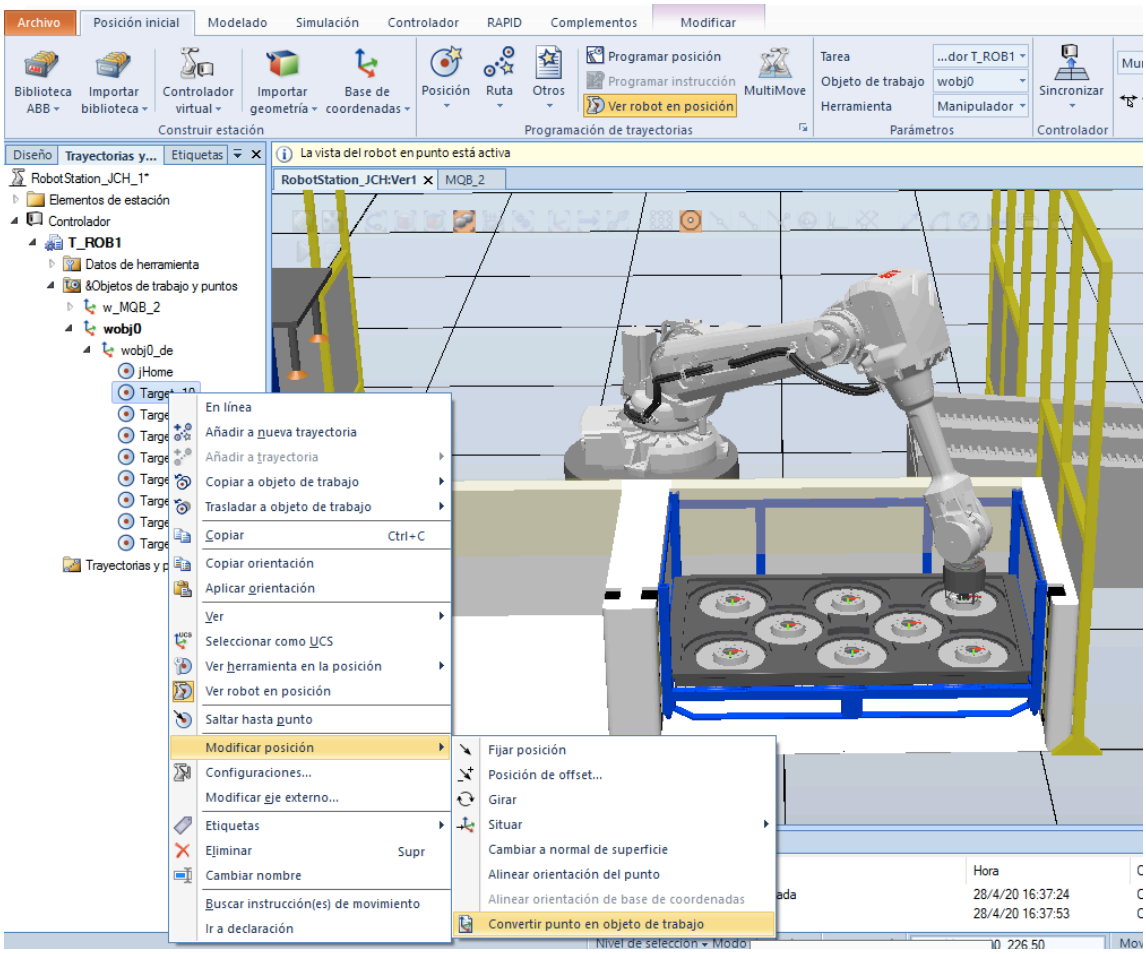


Figura 25.5 Convertir punto en objeto de trabajo en RobotStudio

Una vez creada la trayectoria se sincroniza con RAPID, tal y como se explica en la memoria, para obtener el código del programa. En el *Módulo1* predeterminado se crea la trayectoria Path_10 con las posiciones que previamente se han añadido.

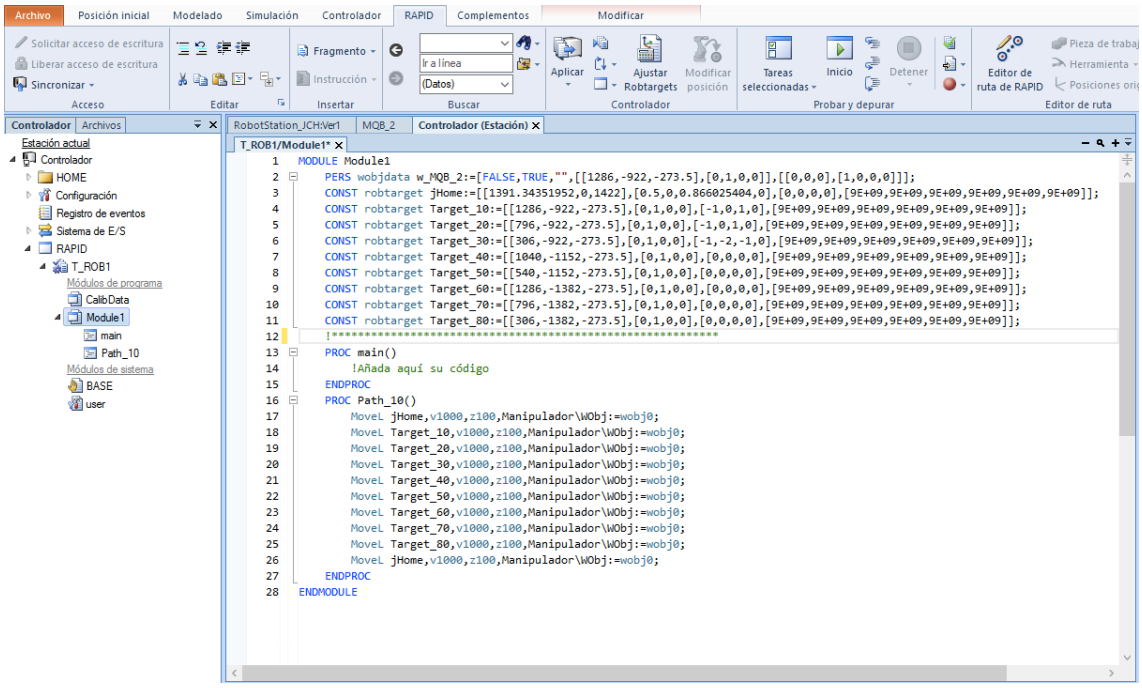


Figura 25.6 Sincronización con RAPID

Ejecutando el código se puede visualizar el robot desplazándose por todas las piezas. La simulación se puede configurar para que sea de ciclo continuo o que se detenga al finalizar el mismo. Esto se puede modificar en la configuración de simulación en la pestaña Simulación.

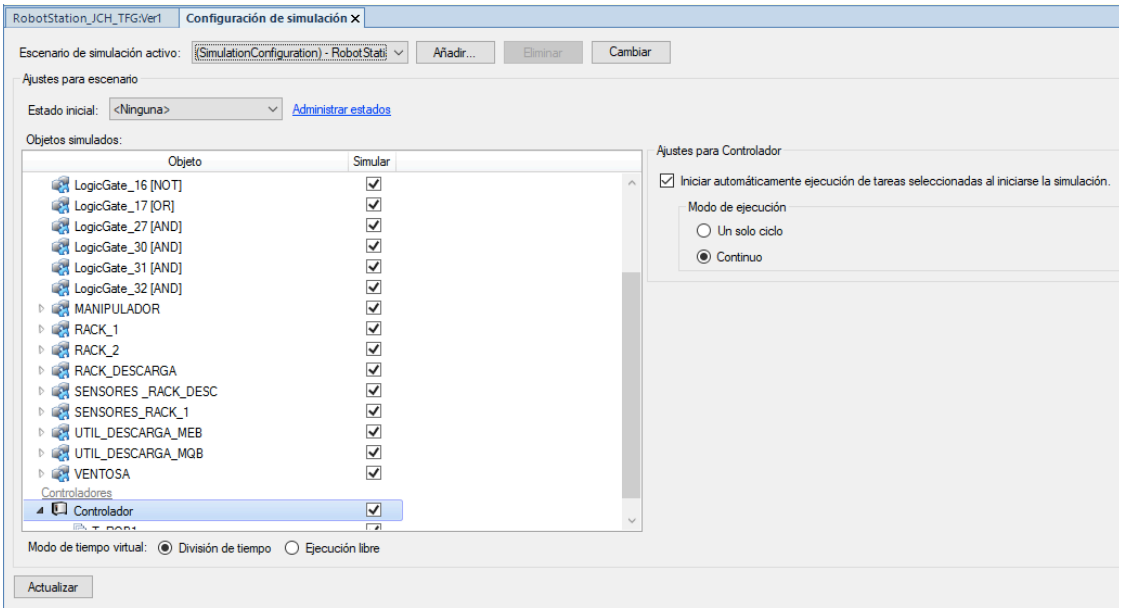


Figura 25.7 Configuración de la simulación

Es posible controlar la simulación para su funcionamiento en distintos modos, así como controlar las entradas y salidas de la simulación a través de los comandos específicos para ello y que se encuentran en las distintas pestañas de RobotStudio.

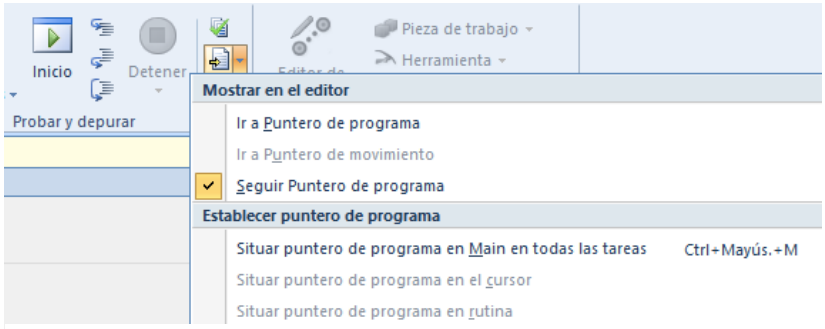


Figura 25.8 Comandos para opciones de puntero

Figura 25.9 Controles de simulación

Figura 25.10 Controlador de E/S de la simulación

Abriendo el Virtual Controller (Teach Pendant Virtual) se pueden realizar todas las funciones necesarias como mover el robot, teachear posiciones, recorrer el programa etc.

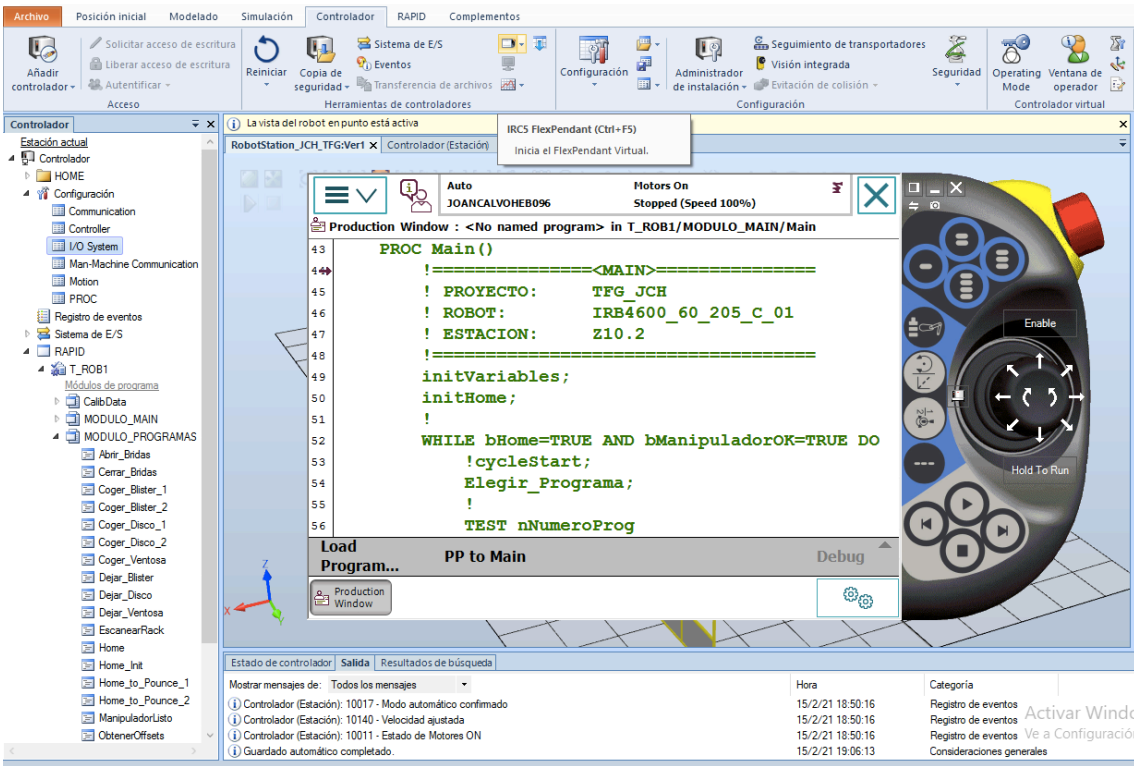


Figura 25.11 Virtual FlexPendant en RobotStudio