UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

ÁREA DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

PROYECTO FIN DE CARRERA

Ingeniería Industrial

SIMULACIÓN DE PROCESOS DE PRODUCCIÓN ROBOTIZADOS MEDIANTE EL PROGRAMA ROBOTSTUDIO

Autor del Proyecto: Víctor Martínez León

Tutor del proyecto: Santiago Martínez de la Casa Díaz

JULIO 2008

AGRADECIMIENTOS

La finalización de este proyecto fin de carrera supone el final de una etapa en la vida. En toda esta etapa, muchas son las personas que me han apoyado y ayudado a llevar el barco a buen puerto. Es por eso que hoy les doy las gracias a todas ellas por haberme hecho llegar hasta aquí, empezando por mi familia, mis padres, tío y primos, que han estado presentes en cada momento necesario; a los amigos que siempre han estado dispuestos a prestarme su ayuda, sus alegrías y sus penas, que han sido más que necesarias en todo el camino, y que siempre me provocan y estimulan a mejorar quién soy; y en el tramo final del camino, a mi tutor que siempre ha estado dispuesto a solucionarme toda la cantidad de dudas que le he podido plantear, y que tiene gran parte de culpa de que este libro exista. Gracias a todos por haberme permitido compartir con vosotros esta etapa.

A todos ellos también va dedicado este documento, a todas las personas que han intervenido y hecho posible este cambio de ciclo en mi vida, y en especial a las que siguen estando dispuestas a continuar en esta nueva etapa, me sigan ayudando a mejorarme a mí mismo, me sigan haciendo sentir tan afortunado, y quieran disfrutar de los nuevos tiempos que se avecinan. Nos apoyaremos mutuamente. No habrá dudas. Siempre estaré dispuesto. Juntos recorreremos el nuevo camino.

ÍNDICE

AGRADECIMIENTOS....1
ÍNDICE....2
I - INTRODUCCIÓN....5
I.1 - MOTIVACIÓN....6
I.2 - ESQUEMA DEL PROYECTO....7
II - INDUSTRIALIZACIÓN DE LA CONSTRUCCIÓN....8
II.1 - IDEAS GENERALES....9
II.2 - PROYECTO MANUBUILD....11
01. Resumen del proyecto....12
02. Objetivos del proyecto....13
II.3 - MOBILE FACTORY....16
II.4 - SERVICE CORE....20
01. Definición....20
02. Objetivo....21
03. Esquema básico y componentes....22
04. Demostración....27
II.5 - EL ROBOT INDUSTRIAL....28
01. Importancia del robot en la industria....28
02. Características de un robot industrial....32
03. Simulación de robots industriales....36
III - ESTADO DEL ARTE EN SIMULACIÓN DE ROBOTS INDUSTRIALES....38
01. Sistemas no propietarios....39
02. Sistemas propietarios....43
IV - IMPLEMENTACIÓN EN ROBOTSTUDIO DE PROCESOS BÁSICOS DE ENSAMBLADO....52
IV.1 - PROGRAMACIÓN DE UNA ESTACIÓN Y SIMULACIÓN....53
01. Introducción a RobotStudio....53
a) Ejecución del programa....53
b) Entorno de usuario....54
c) Trabajar con la ventana gráfica....57
02. Creación de la estación....59

a) Importar un Robot 61
b) Creación de sólidos 64
c) Situar un objeto 65
d) Creación y conversión de archivos CAD con SolidWorks 66
e) Crear un mecanismo. Kinematic Modeler 66
f) Modificar posiciones. 68
g) Generar una librería 69
h) Rotar un objeto 69
i) Fijar piezas 69
j) Usar cotas en SolidWorks 72
k) Copiar/Aplicar orientación 73
l) Crear una herramienta 75
m) Creación de puntos en RS. Puntos de agarre 76
n) Crear Trayectorias. Utilizar objetos de trabajo. 78
o) Arrancar el VC. Mover el robot y recorrer trayectorias. 80
p) Asignar señales E/S. Tabla de eventos. 83
q) Sincronizar: Transferir elementos al VC 85
r) Utilización de E/S. ProgramMaker 86
s) Utilización de macros VBA para mover mecanismos 88
t) Utilizar conjuntos de colisión 90
u) Creación del resto de puntos y procesos 92

IV.2 - APLICACIÓN DE LA METODOLOGÍA PARA LA IMPLEMENTACIÓN DE UN PROCESO REAL DE ENSAMBLADO ... 106
01. Creación de la estación 106
02. Simulación del ensamblado 113
03. Transferencia al robot real 120

V - CONCLUSIONES Y TRABAJOS FUTUROS 127

VI - BIBLIOGRAFÍA Y REFERENCIAS 130

VI.1 - REFERENCIAS 131
VI.2 - BIBLIOGRAFÍA 132

VII - ANEXOS 133

VII.1 - GLOSARIO 134
VII.2 - ÍNDICE DE FIGURAS 135
VII.3 - LISTA DE ACCESOS DIRECTOS EN ROBOTSTUDIO 138

VII.4 - PROGRAMAS RAPID 144
01. Programa Rapid del ensamblado del SC 144
02. Programa Rapid de la estación creada para transferir al robot real 155
03. Programa Rapid de la estación transferido al robot real 160
VII.5 - PROGRAMAS DE VBA 168
VII.6 - TABLAS DE EVENTOS 170
01. Tabla de eventos del ensamblado del SC (capítulo IV.1) 170
02. Tabla del proceso creado para demostrar (capítulo IV.2) 171
VII.7 - LISTAS DE SEÑALES 172
VII.8 - HOJAS DE CARACTERÍSTICAS 173

I - INTRODUCCIÓN

I.1 - MOTIVACIÓN

I.2 - ESQUEMA DEL PROYECTO

I.1 - MOTIVACIÓN

La motivación de este proyecto fin de carrera surge a partir de la idea principal de realizar una demostración lo más fiel y clara posible del proceso constructivo de un módulo de aguas. Dicho módulo de aguas, denominado **Service Core**, (que se explica más detalladamente en el capítulo II junto con el resto de fundamentos teóricos), es uno de los productos usados como ejemplo que puede producir una célula robotizada integrada en una Factoría Móvil llamada **Mobile Factory**. Ambos desarrollos están incluidos dentro del proyecto **ManuBuild**. Este proyecto, dentro pues del cual se encuadra el presente proyecto fin de carrera, se trata de uno de los proyectos de colaboración europeos del Sexto Programa Marco de la Comisión Europea (6FP). La idea principal y objetivo del proyecto ManuBuild consiste en realizar el salto del proceso constructivo tradicional (construcción en la obra, agentes del proceso mal integrados y coordinados, etc.) a un proceso que aproveche las ventajas de la industrialización, tomando ejemplos de lo que ocurre en otros campos, como el automovilístico o el aeronáutico.

Para realizar esta muestra explicativa de la filosofía general de dicha célula, se usará un programa de software que permita simular el proceso, y que permita además la transferencia de los programas robóticos entre el software y el robot **IRB2400 M94**, características que pueden resultar provechosas en futuros trabajos en esta línea. Esta demostración no mostrará el proceso completo y definitivo de fabricación del Service Core, pero sí servirá para mostrar las características básicas del mismo, y su metodología de trabajo. Además, ayudará a posteriores desarrollos de simulación del proceso a que lleguen a ser lo más semejantes posibles al proceso real, mostrando una introducción útil sobre el uso del software, así como el procedimiento de trabajo a seguir en los posteriores diseños, simulaciones, transferencias entre robot y software, etc.

Así pues, se pueden resumir los propósitos de este proyecto fin de carrera en:

- Adquirir los **conocimientos y habilidades** necesarias para aprovechar lo máximo posible las ventajas de un **software de simulación robótica.**
- Mostrar la **filosofía general** de un proceso constructivo que desarrollará la célula robótica de la Mobile Factory.
- Sentar las **bases necesarias** para futuros trabajos en estos temas.

I.2 - ESQUEMA DEL PROYECTO

Este documento se divide principalmente en tres bloques:

- **Bloque de introducción y bases teóricas:**

  En este bloque se presentan las líneas básicas del proyecto fin de carrera, su organización y esquema, así como el fondo teórico en que se sustenta el mismo. Los capítulos que comprende son:

  - *Introducción*: Motivación y esquema del proyecto.
  - *Industrialización de la construcción*: Aspectos generales del concepto de industrialización en la construcción, presentación del proyecto ManuBuild y descripción de la Factoría Móvil, el Service Core y los robots industriales.
  - *Estado del arte*: Estado del arte en softwares de simulación de robots.

- **Bloque principal de desarrollo del proyecto:**

  En este bloque se explica con detalle el desarrollo de la simulación con el software elegido y se realiza una verificación del proceso de transferencia de programas del software al robot físico. A continuación se comentan las conclusiones y los futuros pasos a realizar. Contiene los capítulos:

  - *Implementación en RobotStudio:* Introducción al programa, metodología de desarrollo, proceso de programación y simulación de una estación y desarrollo de un modelo para probar la transferencia software-robot físico.
  - *Conclusiones y trabajos futuros.*

- **Bloque de documentación y referencias:**

  En este bloque se muestran todos los anexos que complementan al proyecto, así como todas las referencias y bibliografía considerada en el desarrollo del mismo. Los capítulos que lo componen son:

  - *Bibliografía y referencias:* Conjunto de literatura consultada en el desarrollo del proyecto y referencias relevantes.
  - *Anexos:* Recopilación de documentos que sirven de apoyo al desarrollo del proyecto. Incluye un glosario de términos comunes en robótica y/o usados en el presente documento, índice de figuras, y programas de rapid.

II - INDUSTRIALIZACIÓN DE LA CONSTRUCCIÓN

II.1 - IDEAS GENERALES

II.2 - PROYECTO MANUBUILD

01. RESUMEN DEL PROYECTO
02. OBJETIVOS DEL PROYECTO

II.3 - MOBILE FACTORY

II.4 - SERVICE CORE

01. DEFINICIÓN
02. OBJETIVO
03. ESQUEMA BÁSICO Y COMPONENTES
04. DEMOSTRACIÓN

II.5 - EL ROBOT INDUSTRIAL

01. IMPORTANCIA DEL ROBOT EN LA INDUSTRIA
02. CARACTERÍSTICAS DE UN ROBOT INDUSTRIAL
03. SIMULACIÓN DE ROBOTS INDUSTRIALES

II.1 - IDEAS GENERALES

El método de construcción de edificios **tradicional**, muy extendido actualmente dada su probada eficacia, se basa en el transporte de los materiales al sitio de construcción, en el que se desarrollan los diferentes tajos, los cuales presentan una multitud de características y necesidades que hacen del proceso **difícil de coordinar**, lo que suele resultar en **ineficacias** tales como trabajos parados en espera de otros o aumento de plazos de construcción. Además, es un proceso altamente artesanal y frecuentemente destructivo, por ejemplo, a la hora de incluir las instalaciones que precisan de generación de canalizaciones, y así se generan importantes cantidades de escombro y frecuentemente surgen problemas de fisuras y transmisiones acústicas no deseadas. Igualmente el **mantenimiento se hace complicado** debido a la dificultad de registro y reparación de las instalaciones, y de efectuar posibles cambios de distribución.

El sector presenta, por tanto, una serie de ineficiencias. La **industrialización** del proceso mejoraría la situación, pero sin embargo, la aplicación de procesos industrializados se ha visto obstaculizada por esta visión tradicional del sector. Aunque la producción de viviendas se ha incrementado mucho en los últimos tiempos, la industria de la construcción se ha quedado por detrás de las demás en cuanto a desarrollo tecnológico e integración**. Otras industrias han adoptado estrategias** que les han llevado a conseguir reducir los costes de producción y mejorar la productividad y la calidad de los productos finales. Algunas de las técnicas y estrategias de más renombre que se han aplicado satisfactoriamente en otras industrias, especialmente en la automoción, la aeronáutica y la naval, son:

- ERP: Sistemas de planificación de recursos
- Sistemas de diseño CAD (asistidos por computador) orientados a objetos.
- JIT: Producción justo a tiempo.
- DFMA: Diseño para la fabricación y el ensamblado.
- Diseño de prototipos y análisis de herramientas.

Para conseguir unos beneficios similares a las industrias mencionadas, el sector de la construcción está actualmente estudiando y desarrollando aplicaciones relacionadas con éstos y otros conceptos y técnicas. El concepto de **prefabricación** está muy presente en estos desarrollos.

Unos de los principales enfoques a tener en cuenta a la hora de realizar este paso a la industrialización, basado en la prefabricación, es la

automatización y/o robotización de los procesos constructivos. Automatizar y/o robotizar la construcción llevaría consigo una serie de ventajas:

- Menor dependencia de mano de obra directa. Más calidad, evitar repetitividad a los operarios, cualificación de mano de obra.
- Incremento de productividad. Aumento de velocidad de producción
- Mejora de la seguridad. Sistemas más seguros para el hombre.
- Incremento de la calidad. Reducción de la variabilidad.
- Ventajas competitivas. Reducción de costes.
- Mejor control del proceso. Mejor gestión de problemas.
- Mayor control del resultado final.
- Mejora de la flexibilidad. Sistemas multitarea.

Es por eso que dentro de la investigación hacia el paso a la industrialización de la construcción, uno de los temas incluidos es el desarrollo de factorías robotizadas, las cuales contribuyen a la robotización y automatización de los procesos. Por tanto ayudan a mejorar el proceso global en línea con las características comentadas, y son completamente compatibles con el desarrollo de conceptos de prefabricación de los elementos constructivos.

II.2 - PROYECTO MANUBUILD

Como ya se ha comentado, el presente proyecto fin de carrera está encuadrado dentro del proyecto colaborativo de investigación europeo **ManuBuild,** adscrito al Sexto Programa Marco de la Comisión Europea, que tiene una duración de 4 años (2005-2009). El proyecto aborda el posible cambio en la filosofía de la construcción del enfoque actual a un proceso más integrado, abierto, eficiente y seguro teniendo como línea básica el paso a la industrialización de todo el proceso.

Entre los diferentes participantes europeos son de destacar algunos como la empresa metalúrgica inglesa CORUS (coordinadora del proyecto), la constructora y promotora sueca NCC, la promotora inglesa Taylor Woodrow o el Centro de Investigación Tecnológico VTT en Finlandia. Las empresas españolas involucradas en el proyecto son la constructora DRAGADOS, la empresa Municipal de la Vivienda y el Suelo de Madrid EMVS, la constructora FCC, el centro tecnológico Labein y la Universidad Carlos III de Madrid. A continuación se muestra la lista íntegra de participantes en el proyecto.

1	Corus, *coordinador*	COR	UK
2	Dragados	DRA	ES
3	EMVS	EMV	ES
4	FCC	FCC	ES
5	Fraunhofer IAO	FHG	DE
6	Mostostal Warszawa	MOS	PL
7	NCC	NCC	SE
8	Taylor Woodrow	TWC	UK
9	VTT, S&T manager	VTT	FI
10	YIT	YIT	FI
12	Carlos III de Madrid	CA3	ES
13	CIRIA	CIR	UK
14	Consolis Technology	CON	FI
15	Enterprixe Software Ltd	ESL	FI
16	Graphisoft	GST	HU
19	ITB	ITB	PL
20	IVF	IVF	SE
21	Labein	LAB	ES
22	Nuova QUASCO	QUA	IT
24	TNO	TNO	NL
25	TU Munich	TUM	DE
26	University of Salford	SAL	UK
27	University of Stuttgart, IAT	UST	DE

Figura II.2.1– Lista de participantes en el proyecto ManuBuild

01. Resumen del proyecto

El punto de vista de ManuBuild abarca la **producción abierta** en la construcción, **métodos de fabricación en un nuevo ambiente** (fabricación off-site, mobile factory, ensamblado on-site, etc.) y nuevos **procesos de negocio** que se centran en el aumento del valor, y apoyados por las Tecnologías de Información y Comunicación (**ICT**) de una manera adecuada para conseguir obtener edificios personalizados, flexibles, accesibles y sostenibles, mejorando la calidad de vida del usuario y aportándole mayor valor mediante servicios relevantes. El objetivo industrial es, por tanto, crear un sistema de fabricación de edificios abierto, estableciendo un nuevo paradigma en la producción de edificios mediante la combinación de la fabricación eficiente en las fábricas y en obra y estableciendo un sistema abierto de componentes y productos que aseguren la diversidad de proveedores en un mercado abierto.

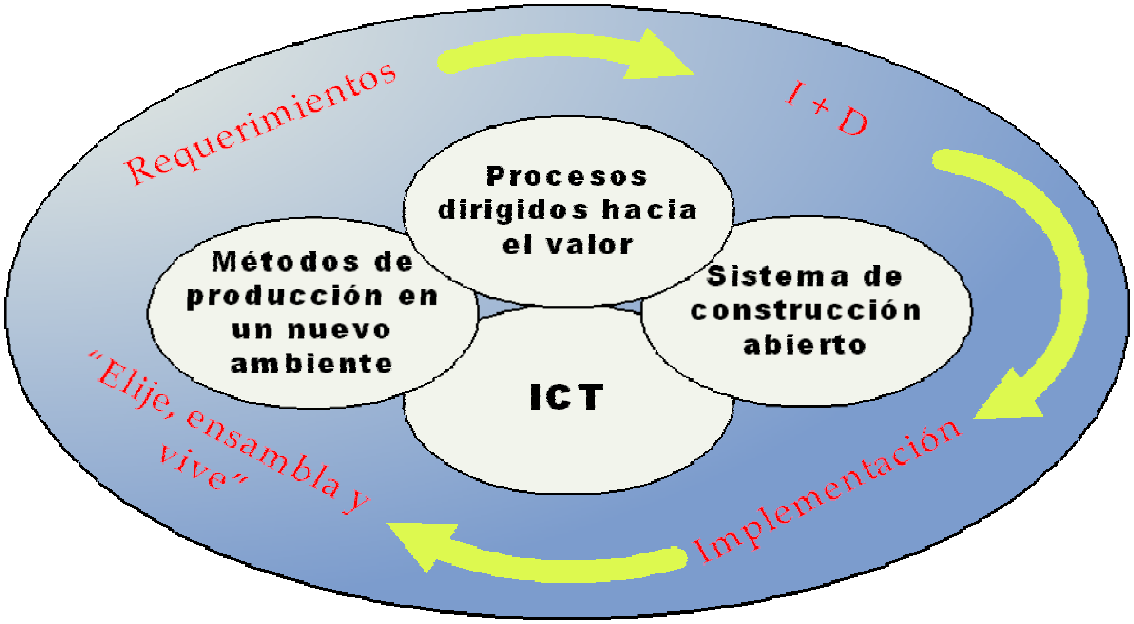


Figura II.2.2- Enfoque del proyecto ManuBuild

Los resultados se validarán y demostrarán en varios proyectos de edificación desarrollados por socios industriales en colaboración con varios ayuntamientos. Para el desarrollo del proyecto se estima un presupuesto de más de **18 millones** de euros.

Algunos de los impactos potenciales incluyen:

- Reducción de costes en más de un 50%.
- Reducción de tiempos en más del 70%.
- Reducción de accidentes en un 90%.
- Nueva oferta de empleo de calidad.

- Nuevas oportunidades de negocio.
- Seguridad en el trabajo.
- Reducción de residuos.
- Mejorar la satisfacción laboral.
- Mayor valor a los usuarios.

02. Objetivos del proyecto

Los elementos clave del sistema abierto de fabricación y construcción que propone ManuBuild incluyen los conceptos de construcción abierta, los procesos de negocio, las tecnologías de producción, y el apoyo de tecnologías de información y comunicación (ICT), como se ve en la **Figura II.2.3**.

El enfoque integrado de las actividades de I+D+i se basa en la búsqueda de cuatro objetivos técnicos y científicos, que buscan la creación del sistema de construcción abierto de la manera que vemos en la siguiente figura, y se explican a continuación.

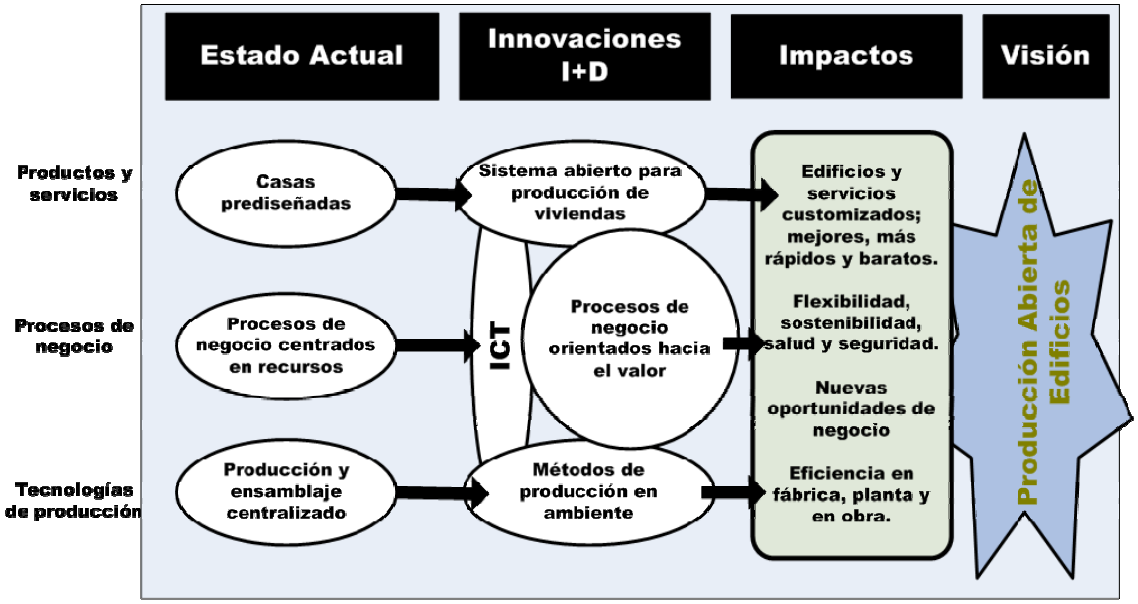


Figura II.2.3- Transformación del modelo de la construcción

Objetivo científico-técnico 1 (ST1): Creación de un **sistema abierto** para la fabricación de edificios ensamblados de manera eficiente en la obra:

- **Sistema de tipología flexible** que incluya al 30% de los tipos que existen hoy día. Así se establece una variada gama de posibilidades de personalización o **customización** del edificio.
- **Componentes inteligentes**, modificables a gusto del cliente, y con una capacidad de ensamblado rápida y sencilla, abarcando un 50% del valor del edificio. Se quiere integrar materiales

existentes y futuros además de las tecnologías actuales con las posibles innovaciones.

- **Conexiones inteligentes** que permitan ensamblar módulos "tipo enchufe" de manera fácil y rápida en la obra. Se pretenden usar estrategias que se complementen como conexiones estándar, nuevos materiales, conectores flexibles, etc., llegando éstas a ser hasta un 80% de las conexiones estructurales del edificio.

Objetivo científico-técnico 2 (ST2): **Procesos de negocio** orientados al **valor**: Nuevos enfoques y métodos que involucren al cliente y la comunidad en la planificación, diseño y personalización de edificios, re-ingeniería e integración del proceso de entregas. Algunos de ellos son:

- **Procesos de ejecución**: creación de métodos de asesoramiento e indicadores para el 95% de los componentes importantes de los edificios, y para los procesos de producción y aprovisionamiento.
- **Procesos de negocio:** se investigará en nuevos escenarios con posibilidad de potenciales mejoras en cuanto a procesos de creación de valor, como "Un mundo dependiente de la energía" o "Un mundo sensible al coste".
- Abarcar el **ciclo de vida completo:** creación modelos de servicios que estén presentes durante todo el ciclo de vida del edificio, incluyendo mantenimiento, rediseño, etc.

Objetivo científico-técnico 3 (ST3): Métodos de **fabricación en ambiente:** Procesos de fabricación y ensamblado en fábricas y obras orientados al incremento de valor, innovadores y optimizados. Combinación de las tecnologías existentes con nuevos desarrollos para obtener una construcción más rápida, flexible y segura que suponga reducciones de costes y tiempo. Se desarrollarán nuevos conceptos de fábrica, células de producción, procesos de fabricación específicos y logística.

- **Producción y pre-ensamblado** en fábrica: métodos y sistemas automatizados, flexibles y eficientes, aptos para la introducción de nuevos materiales y tecnologías de producción. Capacidad de entregar módulos bajo petición en menos de 5 días.
- **Factorías móviles**: creación de factorías móviles o portátiles que permitan realizar operaciones productivas y de ensamblaje en la obra y creando un entorno de trabajo limpio y seguro. Reducción del coste de transporte de componentes en un 80%. Ahorro en costes y tiempos y posibilidad de introducir la filosofía de fabricación "just in time".
- **Sistema logístico**: mejoras en los procesos logísticos que permitan mejorar el manejo y la entrega de módulos y componentes. La rotación del stock debe ser posible al menos 10 veces al año. Este sistema debe ayudarse de ICT para facilitar el plan de transporte, el manejo de material y las órdenes de producción.

- **Ensamblaje en obra:** creación de nuevos métodos y sistemas que reducirán el tiempo de productos producidos in situ a menos de 20 minutos. Se tendrá en cuenta la automatización con alta velocidad, alta precisión y tecnología de sensores integrados para el "ensamblaje exacto".
- **Calidad y seguridad:** optimización de la seguridad, el cumplimiento de requisitos medioambientales y de objetivos productivos para conseguir la entrega de edificios con cero defectos y cero accidentes.

Objetivo científico-técnico 4 (ST4): **Apoyo de ICT:** nueva infraestructura de ICT y herramientas clave:

- **Configuración del edificio interactiva** y herramientas de evaluación: se pretende conseguir un proceso orientado al cliente de planificación, diseño, configuración, personalización, etc. Esto reducirá la fase de diseño y planificación en un 60%.
- **Nueva infraestructura ICT**: abarca el uso de componentes industrializados, la distribución de la producción y la coordinación del ensamblado in situ.
- **Catálogos de componentes inteligentes**: estándares de lenguajes descriptivos de componentes que permitan el desarrollo de catálogos.
- **Gestión logística y plan de ensamblaje**: deben coordinar el suministro desde los distintos orígenes, el replanteamiento y simulación de alternativas para los procesos de ensamblado, la localización de lugares de producción, etc.

Este proyecto fin de carrera, que trata sobre la simulación de un posible proceso productivo desarrollado por una factoría móvil, está por lo tanto directamente encuadrado dentro del objetivo ST3 que estudia la línea de nuevos métodos y procesos de producción en ambiente.

II.3 - MOBILE FACTORY

Como ya se ha comentado, cada vez más se está estudiando y aplicando el cambio a un proceso más industrializado e integrado. Este enfoque pretende asemejar el proceso constructivo a aquéllos que se producen en otros sectores, como el de la automoción o el aeronáutico. En estos sectores, se ha conseguido una fuerte productividad de los procesos basada en los conceptos de estandarización, reducción de desperdicios, planificación y diseño integrado de procesos involucrando a todos los agentes del mismo, etc. Para avanzar en la **industrialización** de los procesos un concepto clave y presente en estos principios es la **prefabricación**.

Las aplicaciones de prefabricación en el sector de la construcción están típicamente poniendo de manifiesto que uno de los factores clave para desarrollar estas técnicas es el **coste de transporte** y el coste de implantación y desmontaje de las instalaciones necesarias en la obra. A la hora de evaluar los costes de transporte son factores importantes: el valor de la mercancía, el coeficiente de transporte (no transportar aire), y la cantidad. Muchas veces, el coste de carga, transporte y descarga de unidades prefabricadas es tan alto, que no resulta beneficioso frente al método tradicional. En esos casos se tiende a implantar una factoría temporal en la obra, a pesar de los costes de montaje y desmontaje de la misma.

Además de por este criterio, como ya se ha comentado, el desarrollar e implantar una factoría temporal conllevaría varias ventajas derivadas de la automatización y robotización de procesos, consiguiéndose un mayor grado de industrialización del proceso constructivo.

Un desarrollo intermedio entre la fabricación tradicional y el uso de talleres temporales en obra, es la **factoría móvil**, una factoría que se transporta, monta, usa y desmonta fácilmente y presenta flexibilidad en cuanto a los procesos que es capaz de realizar. Esta factoría aglutinaría por tanto las ventajas de la robotización y las posibles derivadas de las características logísticas del proyecto en cuestión. Se compone básicamente de una célula robótica transportada en un camión estándar, y diseñada para el propósito concreto.

Las características técnicas de esta factoría dependen de multitud de factores. Una primera clasificación a tener en cuenta en el desarrollo es la que se refiere a las tareas que se podrán realizar. Las **tareas** más típicas que puede desarrollar un sistema automático son:

- Ensamblado: tareas simples y complejas.
- Manipulación de materiales: de diferentes pesos y tipos.
- Distribución de materiales: con o sin tratamiento.
- Preparación: para otras tareas, como el tratamiento.

- Acabados: para dotar de ciertas características al producto.
- Inspección: para verificar propiedades.

Además, existen **otros factores** a tener en cuenta a la hora de generar un desarrollo de este tipo, como por ejemplo:

- La arquitectura del sistema: una máquina, complejo, etc.
- Flexibilidad del sistema: Multitareas o máquinas especializadas.
- Movilidad: Sistemas fijos o móviles.
- Autonomía: Sistemas autónomos, controlados, o mixtos.

Teniendo en mente todos estos factores, se introduce el concepto de la Mobile Factory, cuyo diseño se describe un poco más adelante, para que presente una buena flexibilidad, aporte un entorno seguro de trabajo, sea autónoma en energía, gestión y operación y tenga un flujo de proceso configurable.

La Mobile Factory, pues, debe ser diseñada, en lo que se refiere a diseños software y hardware, selección de herramientas, componentes y tareas, etc, para estar dotada de las siguientes características:

- Flexibilidad: debe ser capaz de realizar distintas operaciones, tareas y productos sin cambios significativos en el diseño mecánico.
- Movilidad: debe poder ser incluida en un contenedor estándar para ser transportada en un camión, cumpliendo las restricciones espaciales.
- Autonomía: debe necesitar la mínima intervención humana posible y ser autónoma energéticamente.
- Automática: las tareas deben estar automatizadas, sin necesitar operarios.
- Ensamblado: es la tarea principal de la factoría. Las piezas deben ser aptas para un rápido y sencillo ensamblado.
- Reconfigurable: se debe poder reconfigurar en cuanto a software, para cambiar la secuencia de tareas a realizar, y a hardware, para adaptar el diseño al proceso.

Con todo ello se realiza el diseño de la célula mostrado en la **Figura II.3.1**, en la que se observa el despliegue del contenedor de la factoría y sus dimensiones características. Como vemos, sus dimensiones principales son:

- Longitud: Contenedor cerrado→ 6000 mm. Abierto→ 8380 ó 10760 mm
- Anchura: Cerrado→ 2350 mm. Abierto → 4730 ó 7110 mm.
- Altura: 2380 mm desde la base del contenedor.
- Área de operación: desde 28.38 hasta 76.5 m^2.

Las variaciones en las dimensiones se refieren a la posibilidad de abrir dos, tres o las cuatro hojas.

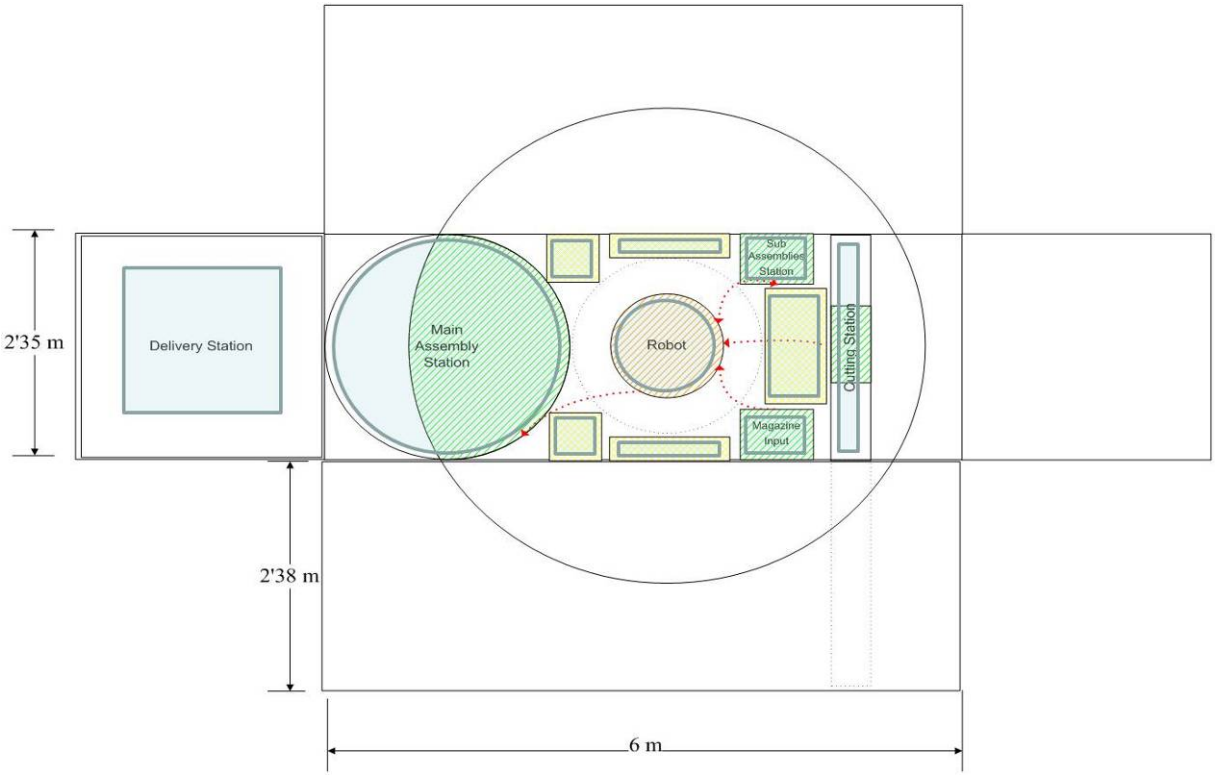


Figura II.3.1- Diseño de la célula de la factoría móvil

Las distintas áreas de la célula son:

- 4 estaciones de procesos:
  - Estación principal de ensamblado ("Main Assembly Station") formada por una mesa circular y un motor que la hace girar.
  - Estación de corte de piezas ("Cutting Station"), en la que se realizarían procesos de corte de hasta 2350 mm.
  - Estación de alimentación de kits de productos ("Magazine Input").

  - Estación de sub-ensamblado de pequeños componentes ("Subassembly station").
- 1 robot-manipulador industrial.
- 5 zonas de almacén de herramientas ("Tool warehouse areas").
- 2 zonas de salida de productos ("Delivery Station").

Este diseño general nos permite cumplir las restricciones de transporte, espacio, etc, y ofrecer las características y ventajas comentadas.

Ventajas

En resumen, el compendio de ventajas que se obtienen al utilizar la factoría móvil en el proceso constructivo se pueden clasificar en:

- Tiempo: Se optimiza el uso del tiempo, utilizando menos en realizar tareas y procesos. Se posibilita la opción de un trabajo continuado. Se reduce el volumen de los materiales que se transportan a la obra.
- Dinero: Se reducen los gastos de transporte, incluyendo amplias posibilidades de planificación logística. Se reduce el volumen de los materiales transportados. Se reducen los tiempos y costos de montaje y desmontaje de los equipos. Se reduce las ineficiencias y pérdidas de material.
- Calidad: Se aumenta el rendimiento del proceso y la precisión de las tareas. Una buena planificación ayudaría a reducir la cantidad de materiales y de mediciones necesarias.

Estas son las ideas que apoyan y presentan el concepto de la Mobile Factory. Uno de los productos que pretende producir, en línea con la filosofía del proyecto ManuBuild y las tendencias de industrialización en la construcción, es el **Service Core**, un módulo vertical de instalaciones que se explica más detalladamente en el siguiente apartado.

II.4 - SERVICE CORE

En la construcción tradicional, como se ha comentado anteriormente, la realización de las instalaciones se efectúa mediante rozas en los tabiques ya construidos, lo que produce escombros, fisuras y otras ineficiencias como la dificultad de registro y mantenimiento. La utilización de patinillos que concentran las instalaciones verticales mejora el proceso, evitando la generación de escombros y mejorando la accesibilidad y posibilidad de mantenimiento, pero comparte el problema de la difícil coordinación con otros procesos como la albañilería.

Es por eso que surge la idea de realizar instalaciones incluidas en módulos prefabricados, como el Service Core, que se beneficia de las características de la industrialización para mejorar el proceso de puesta en obra de las instalaciones. A continuación se introduce y se explica el concepto de Service Core, lo que nos ayudará a familiarizarnos con sus componentes y características.

01. Definición

Uno de los productos que se plantea que fabrique la Mobile Factory es el denominado como Service Core. Este producto es un módulo prefabricado de aguas, es decir, un **núcleo de instalaciones verticales** que transportan agua, como pueden ser el sistema de agua caliente sanitaria (ACS) y agua fría, desagües, calefacción, etc. Un módulo comprende el tramo de instalaciones que corresponde a una planta y/o vivienda. El conjunto de todos los módulos es lo que se denomina Service Core.

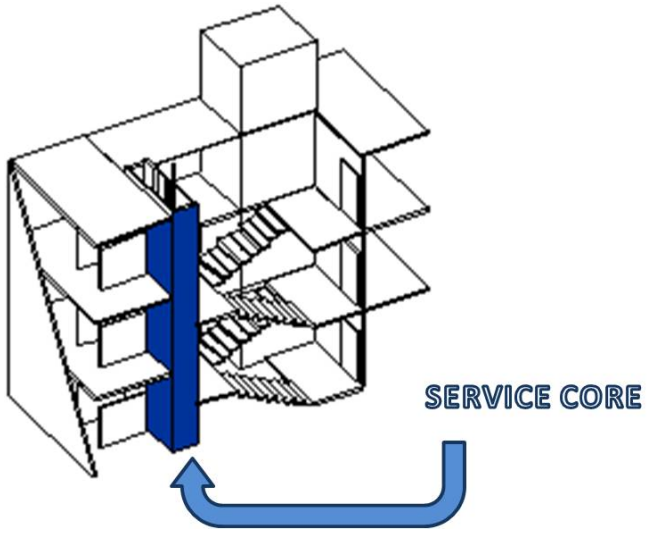


Figura II.4.1- Vista general de un Service Core completo

El Service Core se diseña para ser incluido en un módulo compartido por las cocinas y baños de las viviendas, lo que facilitaría la puesta en obra y el futuro mantenimiento.

Para aportar la debida flexibilidad el Service Core debe tener en cuenta diferentes materiales, diferentes configuraciones y diferentes tipos de instalaciones.

El módulo se puede adaptar a cualquier tipo de instalaciones de agua, y se trata pues de un método de fabricación válido de acuerdo con el diseño realizado, que será específico para cada edificio teniendo en cuenta sus requerimientos, gracias al uso de técnicas como CAD/CAE, DFMA, etc.

02. Objetivo

El objetivo principal del Service Core es hacer más **sencilla**, **rápida**, **segura** y de **más calidad** la puesta en obra de las instalaciones citadas, así como la calidad de los componentes usados en el proceso y la eficiencia y precisión del proceso mismo. Para ello este desarrollo debe situarse dentro del contexto de una construcción industrializada.

La consecución de este objetivo se basa en cuatro aspectos básicos:

- El módulo se debe **construir por tramos de planta**. De esta manera el módulo se fabricaría en un taller, factoría fija o móvil, y después sería transportado y/o colocado en su lugar correspondiente en la obra. Para su colocación sólo se requeriría conectarlo al módulo de la planta inferior de manera que toda la instalación vertical quede unida al final. Así se facilita el control de los trabajos y se mejora la calidad en la fabricación. También se sustituye la mano de obra en el tajo, que presenta peores condiciones de trabajo y seguridad, por mano de obra en taller o factoría, con la consiguiente mejora de las condiciones de trabajo.
- Cada módulo se compone de un bastidor metálico de soporte, de forma que se monta con **independencia de los tabiques** y fábricas de ladrillo. Por ello, la instalación de estos módulos no depende de los trabajos de albañilería, con lo que se mejora la coordinación de procesos y se flexibiliza la planificación de la obra.
- La composición de los módulos independiente de la tabiquería, hace que sean **muy fácilmente registrables** por sus frentes, lo que facilita enormemente el mantenimiento de las instalaciones incluidas.
- Un grado aún mayor de industrialización se obtiene al combinar la fabricación de los módulos en **talleres-factorías móviles**, con lo que se evitan los gastos de transporte de los módulos desde una factoría externa, y minimiza los costes de instalación y desmantelamiento del taller a pie de obra. Además facilita el uso

de técnicas JIT (producción justo a tiempo) lo que conlleva los consecuentes ahorros en tiempo y costes en la logística.

03. Esquema básico y componentes

El Service Core se compone de un bastidor o marco metálico al que se fijan todas las tuberías y desagües pertinentes en los cuartos húmedos. El diseño preliminar del módulo junto con el trazado del bastidor y un modelo real se muestran en la **Figura II.4.2**.

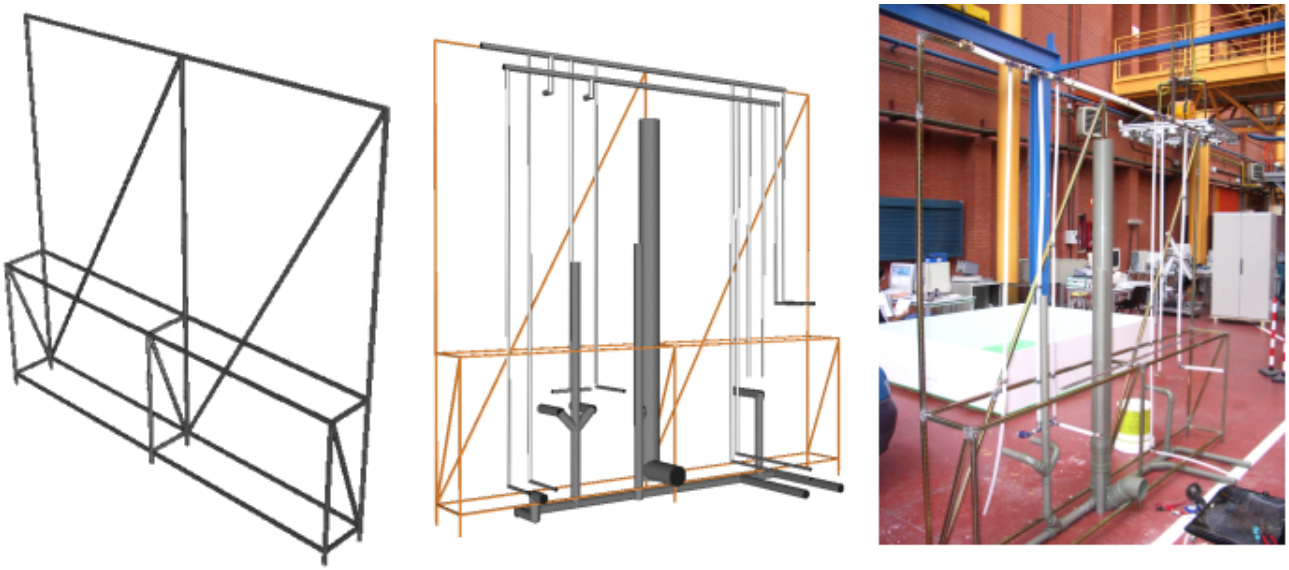


Figura II.4.2- Modelo del bastidor, diseño preliminar en 3D y modelo real del Service Core

En la figura siguiente se presenta una aproximación de las medidas relevantes al Service Core:

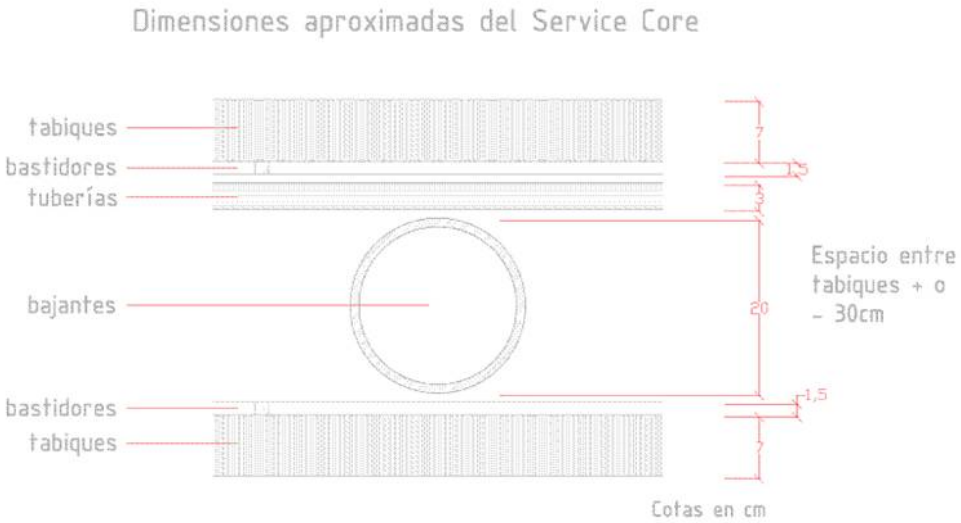


Figura II.4.3- Algunas dimensiones aproximadas del Service Core

A continuación se muestra una lista descriptiva inicial que sirve de preselección de los posibles componentes y herramientas que se integrarían en un Service Core:

Estructura: Una estructura metálica ligera usada para soportar todas las instalaciones pertinentes, y con fácil procesado. Los criterios utilizados a la hora de la elección de materiales deben ser el peso y la forma adecuada.

Un ejemplo de estructura podría ser el sistema "Quick Frame" de la compañía 80/20 Inc. (EEUU). Estaría compuesto básicamente por los siguientes elementos:

- Perfiles Quick Frame creados por extrusión de Aluminio 6105-T5 y con un acabado anodizado 204-R1.

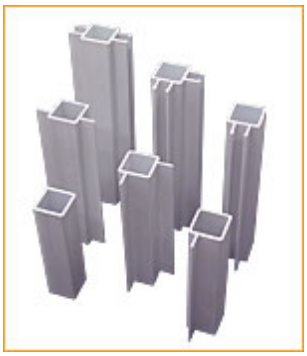


Figura II.4.4- Perfiles Quick Frame de 80/20 Inc.

- Esquinas Quick Frame de Nylon moldeado por inyección.

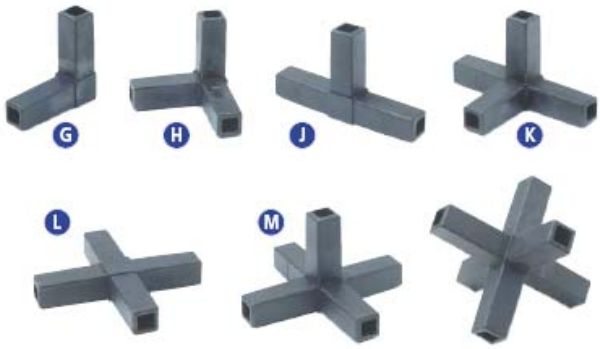


Figura II.4.5- Esquinas Quick Frame

El ensamblado de piezas no precisa de herramientas complejas (bastaría por ejemplo con un martillo de superficie suave) por lo que son fácilmente automatizables.

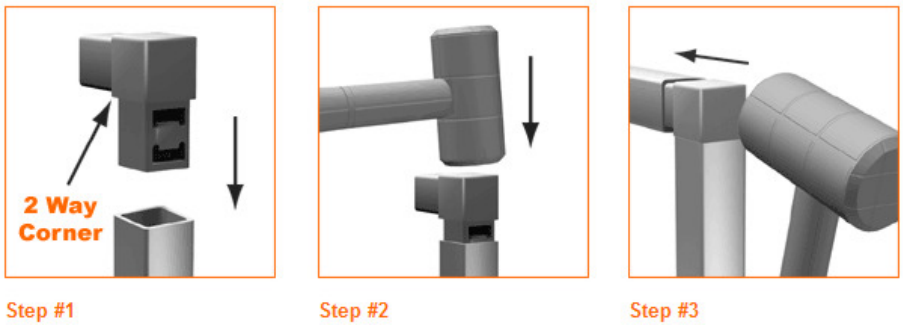


Figura II.4.6- Facilidad de conexión de piezas Quick Frame

Sistema de desagüe:

Los materiales principales a considerar para este sistema son:

- PVC (Cloruro de PoliVinilo): Las tuberías de PVC son los elementos más comúnmente usados, pues presentan unas características tales como:
  - Respetuoso con el medio.
  - Ofrecen vida de servicio duradera.
  - Fáciles de instalar y manejar.
  - Resistentes a la corrosión.
  - Eficientes en costes.
  - Aceptadas en la mayoría de códigos.

  Una tubería de PVC se fabrica por extrusión, presentando variedad de dimensiones y tamaños.
- PP (Polipropileno): Adecuado para instalaciones de desagüe interiores debido a su sensibilidad frente a la radiación ultravioleta. Sus características principales son:
  - Sólo tramos rectos.
  - Ligeros.
  - Flexibles.
  - Alta resistencia a impactos.
  - Excelente resistencia química.
  - Buen rango operativo de temperatura: desde menos de 0ºC hasta 60ºC (régimen continuo) o hasta 90ºC (puntualmente).
  - Vida de servicio duradera.
- ABS (Acrilonitrilo Butadieno Estireno): Las tuberías de ABS fueron originalmente desarrolladas a principios de los años 50,

para su uso en el campo del petróleo y la industria química. Sus propiedades clave son:

- Sólo tramos rectos.
- Acabado interior suave dotado de excelentes propiedades de flujo.
- Gran fuerza: puede aguantar cargas de terreno pesadas, bloques de cemento y cargas superficiales sin romperse.
- Resistente a daños mecánicos, incluso a baja temperatura.
- No es susceptible de putrefacción, oxidación o corrosión.
- No le afectan los químicos domésticos como los ácidos o alcalinos.
- Se pueden usar en construcciones resistentes al fuego.
- Absorbe el calor lentamente.
- Amplia gama operativa de temperatura: de -40ºC a 82ºC.

Todos los materiales comentados y otras posibilidades deben elegirse de acuerdo con los métodos de ensamblado requeridos para una fácil automatización. Por ejemplo, un método de ensamblaje muy sencillo que se puede usar es la unión por junta elástica o "elastic seal".

Otra propiedad deseable para el material es que tenga buenas propiedades de aislamiento acústico. Este requerimiento tiene que ver con la estructura de la tubería. Las tuberías fabricadas de paredes estructuradas presentan un comportamiento adecuado en este sentido.

Sistemas de agua fría y caliente sanitaria:

El Service Core incluye las instalaciones de distribución de agua fría y caliente. Los materiales que se pueden usar son:

- PB (Polibutileno): Muy usado en los 80 y 90, pero ha sido claramente sustituido por el PEX. Características principales:
  - Ligero.
  - Flexible, se puede enrollar.
  - Se puede doblar ante obstáculos sin requerir accesorios.
  - Más silencioso que una tubería rígida.
- PEX (Polietileno con enlaces cruzados): Es un material termoestable producido a partir de polietileno con enlaces cruzados de mediana o alta densidad. Las tuberías de PEX se han usado en distribución de agua caliente y fría y para calefacción radiante hidráulica en Europa durante muchos años. Características:

- Ligero.
- Flexible, se puede enrollar.
- Fácil de instalar, incluyendo doblamientos ante obstáculos sin precisar de accesorios.
- Gran resistencia al impacto.
- Gran resistencia química a sustancias domésticas.
- Más silencioso que las tuberías rígidas.
- Temperaturas de operación hasta 93ºC
- Larga vida de servicio.

Existen varias variaciones de este material con estructuras multicapa, por ejemplo, PEX-A1-PEX.

Normalmente estos materiales se unen con los respectivos accesorios mediante el uso de una herramienta mecánica. El método de unión más usado es la fijación ("clamping"). Este proceso implica la adaptación de la herramienta para los sistemas automatizados.

Fijaciones, cierres, etc:

Todas las estructuras se fijarán al bastidor metálico del Service Core. El sistema a usar para fijar las instalaciones al marco dependerá del peso del material, el diámetro de los tubos y la dificultad de ensamblaje.

Por tanto, las herramientas y métodos de fijación a usar dependerán de los materiales elegidos.

Figura II.4.7– Posible método de fijación

04. Demostración

Para su demostración, el Service Core se instalará en el edificio demostrador que se construirá en Madrid como un desarrollo dentro del proyecto ManuBuild.

El desarrollo de este módulo constituirá un sistema constructivo que persigue la industrialización de la fabricación y puesta en obra del núcleo vertical de instalaciones. Así se conseguirá un proceso con menores tiempos, mejores condiciones de trabajo y mejor calidad final del producto, así como se facilitará la coordinación con el resto de trabajos de la obra.

El concepto de Service Core no tiene nada que ver con el diseño tradicional de los servicios. Los módulos se construyen durante el proyecto, incorporando los servicios definidos. La conjunción de la fabricación del Service Core con la filosofía de la Mobile Factory lleva a un grado aún mayor de industrialización del proceso.

II.5 - EL ROBOT INDUSTRIAL

Dentro del desarrollo de la Mobile Factory se considera una herramienta robótica. El robot elegido a priori para el desarrollo del concepto de la Mobile Factory es de la serie IRB2400, de la que se dispone en el laboratorio de la Universidad Carlos III. En este subcapítulo se exponen algunas cifras y comentarios sobre el mercado relativo a la robótica que ponen de relieve la importancia de estos sistemas, así como las características principales de estas herramientas y las ventajas de su simulación.

01. Importancia del robot en la industria

El robot es una aplicación relativamente joven si se compara con otros desarrollos técnicos. La primera vez que se instaló un robot industrial ocurrió a la mitad del siglo XX. El primer robot con accionamiento eléctrico y controlado por computador vio la luz en 1974.

Hoy en día, muchas de las actividades productivas tienen en los robots industriales unos de sus más importantes componentes, los cuales permiten aprovechar las numerosas ventajas que ofrecen los procesos automatizados, como pueden ser una mayor rapidez, flexibilidad, etc. Las compañías de producción industrial están obligadas a presentar alta productividad y ahorro en costes para subsistir en el mercado. Las operaciones manuales no son rentables en los países que presentan la mano de obra más cara, por lo que estas operaciones tienden a trasladarse a los países con menores salarios. Por otro lado, cada vez se demanda mayor calidad de producto, que sólo puede conseguirse con la automatización. La inversión en robots facilita la posibilidad de mantener la producción en los países de renta alta y a la vez mejorar la calidad del producto. Además, ya que los robots se destinan a realizar a menudo tareas repetitivas y tediosas, frecuentemente peligrosas, la inversión en soluciones robóticas mejora las condiciones de seguridad de los trabajadores.

A continuación se muestran algunos datos concernientes a las cifras del mercado robótico mundial, que demuestran el constante crecimiento en el desarrollo de estas soluciones.

Algunas cifras históricas del mercado de la robótica

Jan Karlsson, perteneciente a la Comisión Económica de las Naciones Unidas para Europa (UN/ECE) manifestó, en un informe de febrero de 2001 [1], la evidencia de que nunca hasta entonces se habían producido tantas peticiones de robots industriales en la industria europea, apuntando a una aceleración en el cambio hacia la automatización. La UN/ECE estudia

regularmente el mercado de la robótica, conjuntamente con la IFR (Federación Internacional de Robótica). En el año 2000, las peticiones de robots industriales en Europa fueron un 25% más altas que en 1999 (ver en la **Figura II.5.1** a continuación). Fijándose en el último cuarto del año 2000, que mostró un incremento del 24%, parece que la inversión en sistemas robóticos no se desaceleró.

Las cifras europeas eran particularmente impresionantes, presentando un 31% de aumento en 1998 y un 12% en el año 1999. En la figura también se puede observar que los motores de la inversión en robótica en este periodo fueron las regiones europea y asiática. También se observa que después del destacado aumento del 60% en el año 1998 en Norteamérica, se produjo un bajón esperado en el año siguiente, mostrando la desaceleración de la economía americana.

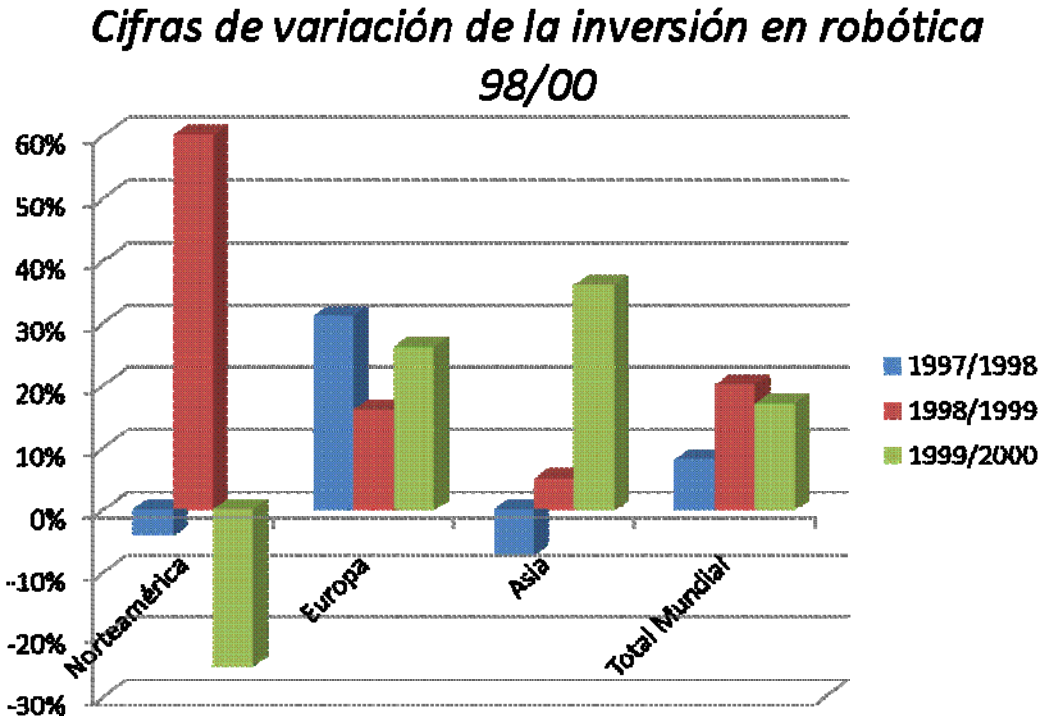


Figura II.5.1– Cifras de variación de la inversión en robótica 98/00

Lo normal era que el sector de la automoción liderara el mercado, como fue el caso en 1999. Sin embargo en el 2000 se produjo un importante cambio: industrias ajenas al automóvil a lo largo de todo el mundo incrementaron sus peticiones cerca de un 40% mientras el propio sector automovilístico sólo aumentó un 3% para operaciones de ensamblaje y un 9% en producción de componentes. En Asia y Europa, las industrias no automovilísticas incrementaron sus pedidos un 56% y un 41% respectivamente, mientras en Norteamérica se registró un escueto aumento del 5%, a diferencia de las caídas en el sector del automóvil en esa región.

El aumento en la inversión presentó varias causas. Una razón principal es la rápida reducción del precio de los robots en relación con los costes de mano de obra. Los precios de los robots en el año 2000 fueron de media un 44% inferiores que en 1990 (ver en la siguiente **Figura II.5.2**). Al mismo tiempo son robots con mucho mejor rendimiento que los del año 1990 en términos de versatilidad, velocidad, precisión y sobre todo potencia de cálculo. Si se tiene en cuenta el rendimiento para ajustar el precio (en la gráfica, la línea ajustada es la de rombos), se observa que incluso es mayor la bajada de precios. Se estima en menos de un tercio el coste de un robot del 2000 respecto a uno del 1990 con el mismo rendimiento.

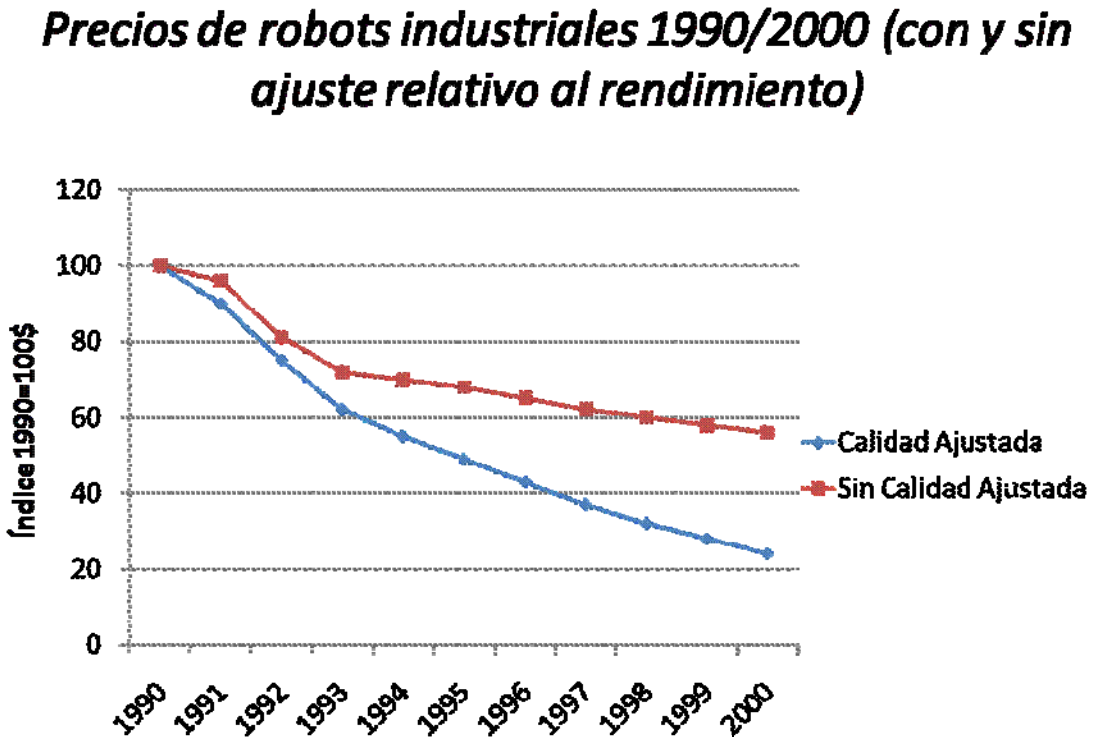


Figura II.5.2- Precios (en relación al año 1990) de robots industriales 1990/2000 (con y sin ajuste relativo al rendimiento)

Al mismo tiempo que el precio de los robots descendió y su calidad mejoraba, los costes de mano de obra subían con paso firme. Por ejemplo, en Estados Unidos el coste de la mano de obra subió un 43% en el periodo 1990-2000 (ver en la **Figura II.5.3**). En el mismo periodo los precios de los robots cayeron sobre un 60% sin tener en cuenta el ajuste por rendimiento, llegando al 80% si consideramos el ajuste, presentando un desarrollo precio/rendimiento similar al de los ordenadores personales. Cada año, los robots se hacían cada vez más eficientes en coste frente a los métodos manuales de producción.

Las cifras muestran claramente la aceleración de la inversión en robótica y la reducción de precios de estas soluciones. Según Karlsson, los

datos del año 2000 muestran la tendencia al cambio de un mercado protagonizado principalmente por la industria del automóvil, a un entorno en que otras industrias manufactureras, como la alimentación, así como otras muchas, tendrán un papel importante en el mercado.

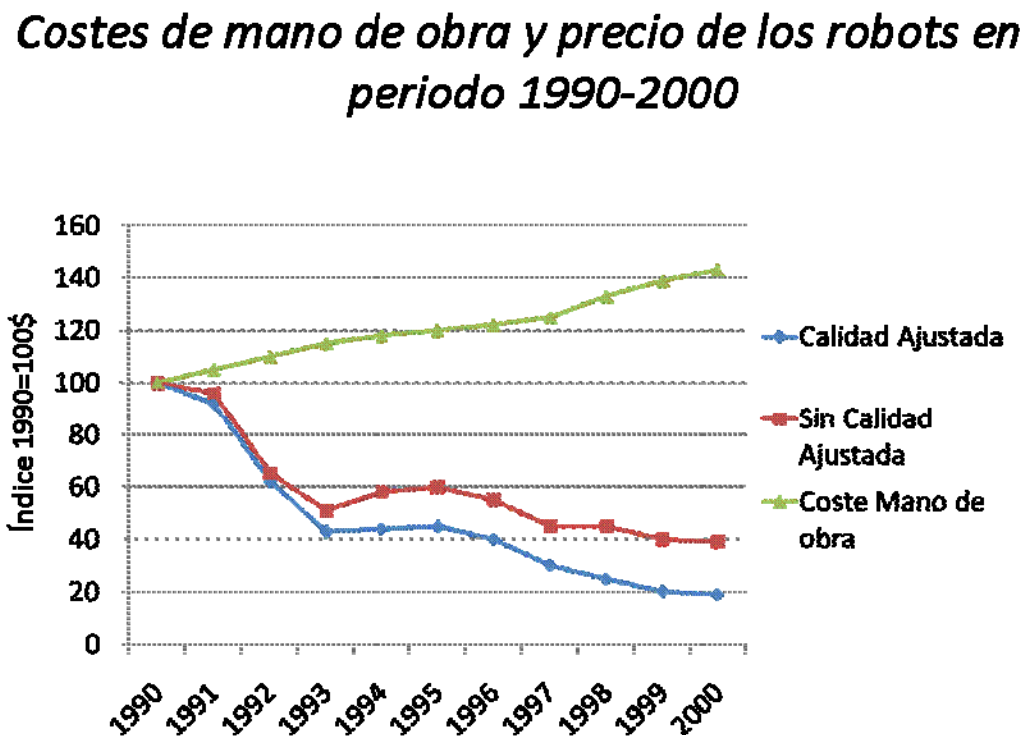


Figura II.5.3- Costes de mano de obra y precio de los robots en el periodo 1990-2000

Tendencias actuales del mercado

Un informe recientemente publicado por la Federación Internacional de Robótica (IFR), World Robotics 2007, establece que después de un repunte en 2005, el mercado mundial de nuevos robots se resintió en un 11% en 2006. Aún así, fue el segundo mayor resultado de la historia y las predicciones para el futuro son buenas. En relación con este informe, el integrante de ABB Ake Linqvist [2] señaló que los suministradores de robots están actualmente desarrollando nuevas aplicaciones, especializadas para pequeñas y medianas empresas, que permiten una interacción cooperativa entre hombre y máquina. También subrayó que las tecnologías en sensores, visión, control de fuerza, etc., se harán más avanzadas y se aplicarán más a menudo en los sistemas robóticos. Además, añadió que las aplicaciones para la industria fotovoltaica, la construcción, para operaciones logísticas y para el micro-ensamblado se están expandiendo, y que por todo ello el futuro de la robótica se presenta emocionante.

Este mismo informe concluye que en el 2007 la inversión en robots se habrá incrementado en todas las regiones debido a la inversión del sector de

la automoción y el aumento de la inversión de otros sectores como el eléctrico/electrónico, la industria química, la del metal y la de la alimentación. En particular en países como China, India y Rusia donde la inversión en la industria del automóvil ha sido hasta hoy de principal importancia, el suministro de robots a otras industrias va a empezar a crecer. El departamento de Estadística del IFR estima en un 10% el crecimiento mundial del mercado de robots industriales en 2007.

Mirando al futuro, Lindqvist se muestra positivo sobre el potencial que presentan los robots industriales. Estima en un 4% el incremento anual en 2008 y 2009 y piensa que las nuevas industrias diferentes de la automoción ganarán importancia en términos de robots instalados, puesto que ya las nuevas soluciones desarrollas por los suministradores de robots han resultado en una demanda creciente por parte de estas industrias.

Conclusiones

Por todo ello, se puede concluir que el mercado de la robótica está en constante crecimiento, por lo que resulta atractivo tener conocimiento sobre los desarrollos en este campo. Además, se constata que frente a la previa posición dominante y casi exclusiva del sector de la automoción como consumidor de estas soluciones, la tendencia muestra que muchas otras industrias, como la alimentaria, la construcción, la logística, etc., están cobrando gran importancia relativa en este mercado, aumentando su inversión en robótica y generando por tanto originales e innovadores desarrollos de aplicación en nuevos campos y tareas por parte de los fabricantes. Por tanto la robótica va a cobrar cada vez más importancia en todo tipo de áreas, y es interesante conocer los nuevos o potenciales desarrollos.

02. Características de un robot industrial

A continuación se presentan los componentes y características básicas de un robot industrial. El modelo concreto en que se detalla la explicación es la serie ABB IRBX400 con controlador S4, de la cual se dispone de varios ejemplos en el laboratorio de la Universidad Carlos III de Madrid. Estas nociones básicas son similares para la mayoría de diferentes modelos que ofrecen los distintos fabricantes, y sirven de conocimiento básico necesario y/o útil a la hora de encuadrar y desarrollar el presente proyecto fin de carrera. Algunas características más avanzadas del modelo se pueden consultar en la **hoja de características disponible** en los anexos (anexo VII.7).

Funcionamiento básico

La serie de robots ABB IRBX400 consta básicamente de tres elementos:

- Un controlador, o armario de control.

- Una unidad de programación, o *Teach Pendant*.
- Un manipulador o brazo.

En esta figura se observan los elementos:

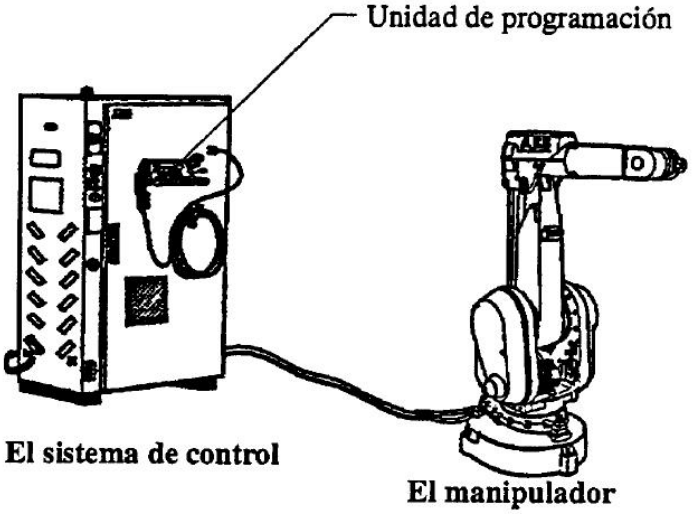


Figura II.5.4- Componentes de un robot industrial

El **manipulador** presenta una morfología antropomórfica con 6 grados de libertad. En la siguiente figura se detalla el manipulador del robot IRB2400.

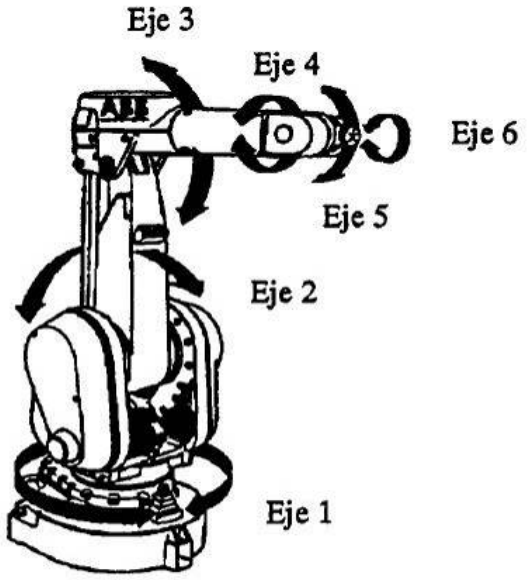


Figura II.5.5- Grados de libertad del IRB2400

El **controlador**, como el que se ve en la figura siguiente, contiene la electrónica de control del robot, en la que se procesan los programas, y todos los elementos de potencia, como transformador, drivers, etc…También está dotado de una unidad de disco extraíble de 3 ½, el interruptor y un panel de control con los mandos principales del robot.

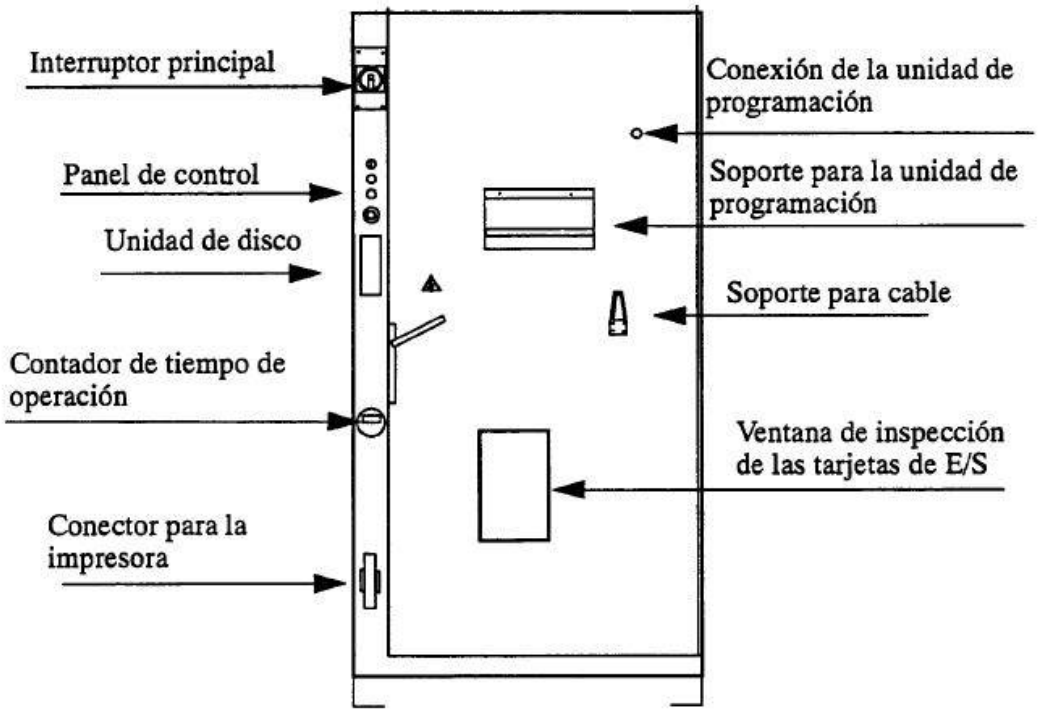


Figura II.5.6- Controlador del robot IRB2400

El **panel de control** (**Figura II.5.7** a continuación) presenta los modos de operación, los pulsadores de MOTORES ON (desactivar frenos y activar motores) y OFF y el pulsador de parada de emergencia. También indican si los motores tienen tensión o no, respectivamente. El selector de modo de operación permite elegir entre:

- Modo Auto: El modo normal en producción. Se ejecuta el programa sin la necesidad de un operario presente.
- Modo Manual: Modo de programación. El robot permite su programación a través de la unidad de programación. Se limita la velocidad a 250 mm/s.
- Modo Manual Velocidad Total: Modo de comprobación. El operario puede comprobar el funcionamiento de un programa sin que se limite la velocidad.

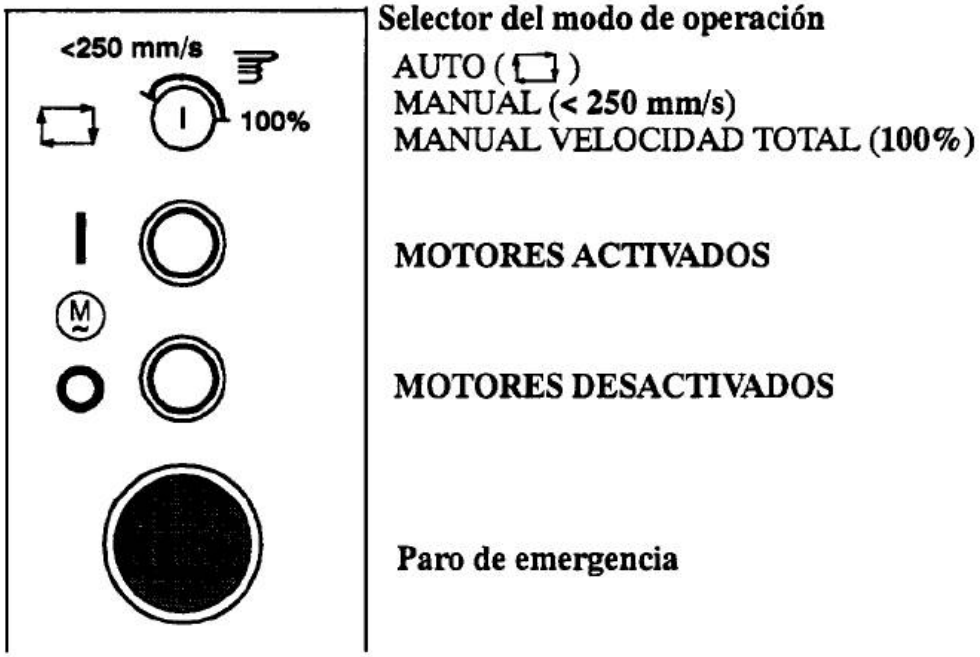


Figura II.5.7- Panel de control del controlador

La unidad de programación o *Teach Pendant* permite la interacción entre usuario y robot. Tiene un visualizador, una botonera, una parada de emergencia y una palanca de mando o *joystick.* Se muestra a continuación.

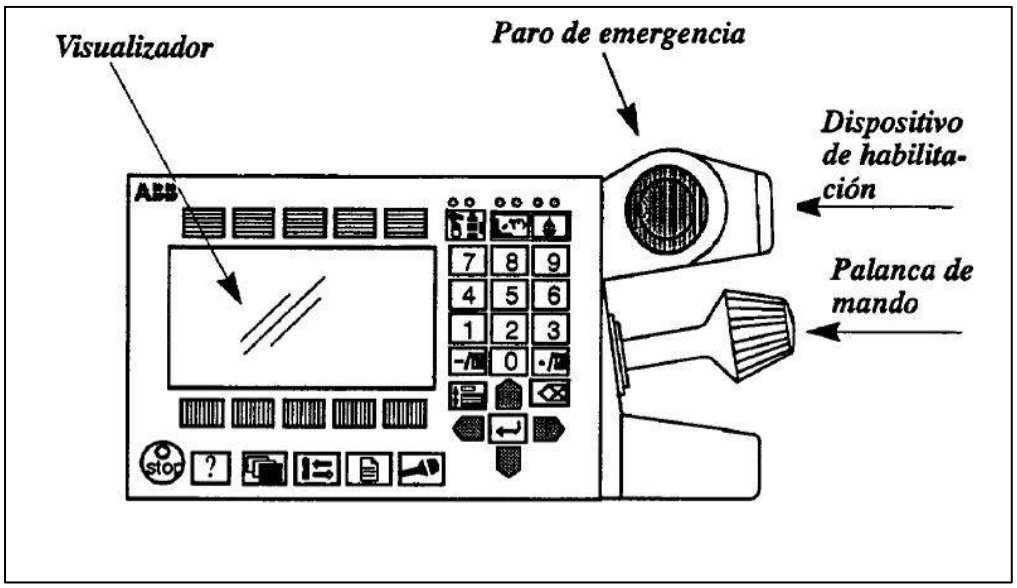


Figura II.5.8- Unidad de programación

El dispositivo de habilitación tiene tres posiciones. Sólo se activa en la intermedia, pasando a MOTORES ON. En el modo AUTO no se necesita activar este dispositivo.

Con la palanca de mando se permite el movimiento manual libre del brazo. Este aspecto es útil a la hora del mantenimiento, programación, etc.

En la siguiente figura se muestra el detalle de los diferentes botones disponibles en la unidad de programación.

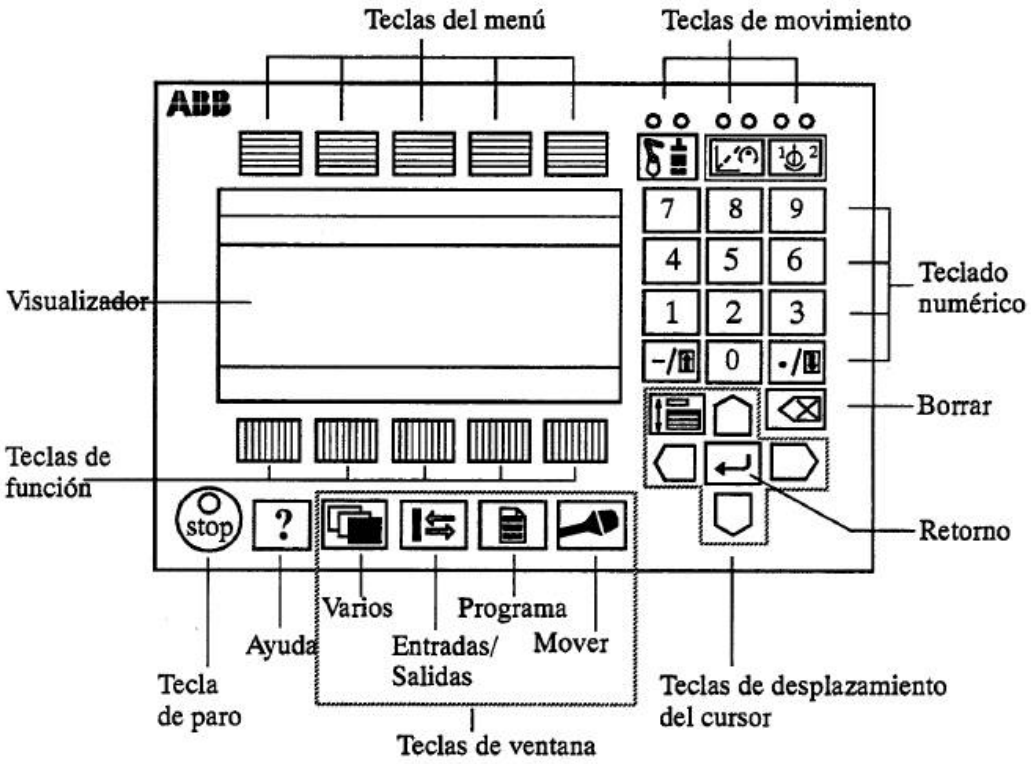


Figura II.5.9- Detalle de la unidad de programación

Seguridad

Una de las consideraciones más importantes a la hora de trabajar es la seguridad. Un robot industrial es una máquina muy potente y por tanto peligrosa. Es primordial respetar las normas de seguridad exigidas en el manejo de este tipo de herramientas. Entre otras medidas, se debe ser cuidadoso al entrar en el área del trabajo del robot (dependiendo del modo de funcionamiento), que tiene que estar debidamente acotada.

03. Simulación de robots industriales

La simulación es una herramienta que presenta varias ventajas aplicada a todo tipo de procesos y situaciones de la vida industrial y científica. Aplicada a la robótica, la posibilidad de simular estaciones en un ordenador nos ofrece una serie de ventajas muy provechosas, como pueden ser:

- **Programar** estaciones **off-line**, mientras la producción sigue su curso y sin perder, por tanto, el tiempo de producción que se necesitaría parar.
- Realizar **comprobaciones**, durante la programación off-line, de los problemas más comunes presentes en éste tipo de soluciones, como pueden ser colisiones, limites de alcance, tiempos de ciclo, etc., permitiendo así anticiparse a ellos antes de la implantación el robot real, y ahorrándonos posteriores revisiones.
- Probar posibilidad de **escenarios**, realizando comprobaciones ingenieriles de posibles alternativas.
- Desarrollar **presentaciones** útiles y claras con una simulación en 3D a la hora de presentar el producto o servicio a un cliente potencial. Se mejoraría así la percepción que se puede tener en un dibujo CAD en 2D, que a menudo son difíciles de ser percibidos adecuadamente cuando se trata de líneas de ensamblaje y robots.

Estas y otras características hacen de la simulación una técnica útil y provechosa para todo tipo de industria. Este conjunto de ventajas justifican y respaldan el desarrollo y los objetivos del presente proyecto fin de carrera.

III - ESTADO DEL ARTE EN SIMULACIÓN DE ROBOTS INDUSTRIALES

01. SISTEMAS NO PROPIETARIOS
02. SISTEMAS PROPIETARIOS

Hoy día existen en el mercado numerosas compañías dedicadas a la producción de robots industriales. Este hecho ratifica la importancia de la utilización de estas herramientas para conseguir un grado óptimo de automatización y productividad en las actividades industriales (como también manifiestan las cifras expuestas en el capítulo anterior).

Simular y programar estas soluciones de manera previa a su implementación permite obtener varias ventajas, como conseguir una ejecución posterior más eficiente y libre de errores que se pueden solucionar a priori, la posibilidad de comprobar desarrollos sin parar la producción, etc. Por ello, varias compañías productoras de robots industriales han desarrollado programas de modelado y simulación para sus propios sistemas, así como otras compañías especializadas en desarrollos de software relacionados con la robótica han creado programas que permiten la programación off-line y simulación de varias de las marcas de robots presentes en la industria. Atendiendo a estos criterios podemos dividir las diferentes soluciones que ofrece el mercado en **sistemas propietarios**, aquellos desarrollados y proporcionados exclusivamente para las herramientas de una compañía concreta, y **sistemas no propietarios**, todos aquellos de propósito general o que ofrecen la posibilidad de implementar robots de varias compañías diferentes.

A continuación se ofrece una muestra de algunas de estas soluciones, bajo la clasificación comentada:

01. Sistemas no propietarios

- **WORKSPACE 5**

Workspace 5™ (un producto de Flow Software Technologies) [3] ofrece la posibilidad de aprovechar las ventajas conocidas de la simulación off-line ofreciendo flexibilidad en cuanto al fabricante del robot que se desea usar; es decir, permite realizar una visualización y un análisis del proceso a efectuar antes de implantarlo en la fase de producción, sin importar el fabricante de la herramienta, pues implementa todas las principales opciones del mercado. Está desarrollado bajo las premisas de facilidad de uso y rápido aprendizaje.

Algunas de las características generales de este programa son:

- Creación, modificación y manejo de geometrías CAD 3D.
- Compatibilidad con archivos IGES y SAT (importar/exportar).
- Uso de VBA (Visual Basic para Aplicaciones).
- Creación de robots y mecanismos, incluyendo cinemáticas.
- Simulación de mecanismos aislados y múltiples.

- Simulación de programas creados en el programa, o importados de un robot real.
- Realización de pruebas de alcance muy precisas, análisis de tiempos de ciclo, y optimización de caminos.
- Detección y gestión de colisiones estáticas y dinámicas.
- Traducción y simulación bidireccional para varios lenguajes de robótica.
- Interpretación y traducción de señales I/O (entrada/salida).

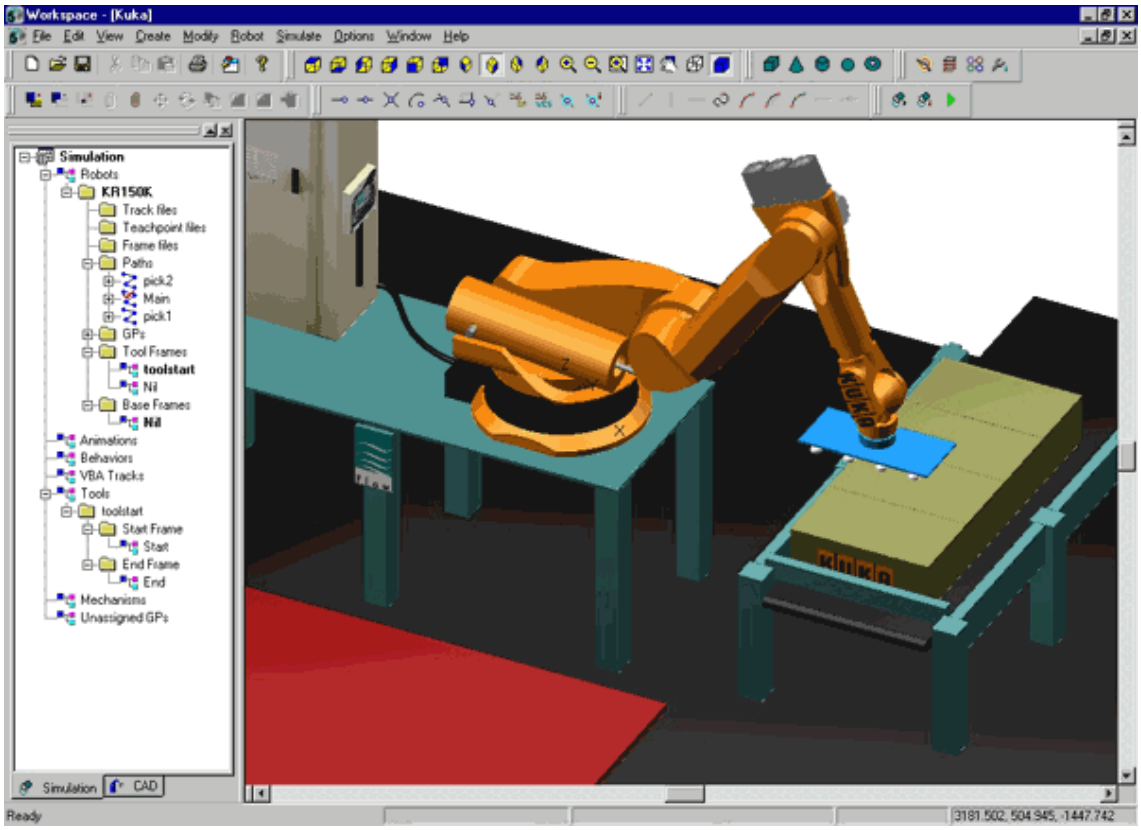


Figura III.1- Imagen del programa workspace5 con robot KUKA

Además, este programa aporta otras características avanzadas, como la posibilidad de crear trayectorias fácilmente a partir de superficies y mediante un asistente, o herramientas que muestran la capacidad de alcance de un robot al crearlo, para tener una idea primera válida de si su ubicación será útil o no.

La flexibilidad de este programa es quizá su punto más fuerte, como queda patente en la siguiente lista, que presenta todos los robots que el programa workspace5 es capaz de utilizar.

ABB	KUKA
ADEPT	MITSUBISHI
COMAU	MOTOMAN
DAIHEN	NACHI
FANUC	PANASONIC
FLOW	STAUBLI
KAWASAKI	

A su vez, ésta es la lista de lenguajes disponibles que el programa presenta:

- ABB Rapid
- Fanuc TP
- Motoman Inform II
- Kawasaki AS
- Kuka KRL
- Nachi Slim
- Panasonic Pres
- Siemens G-Code

Por tanto, es ésta la herramienta que presenta la mayor flexibilidad del mercado en cuanto a uso de fabricantes aportando muchas de las ventajas de la simulación off-line. Todas las características y ventajas mencionadas se pueden consultar en la página web del programa, así como información sobre los cursos que imparte la empresa y una selección de casos estudiados de uso del programa, que sirven para ilustrar mejor tanto sus características como sus ventajas llevadas a la práctica.

- **ROBOWORKS**

Roboworks [4] es un producto desarrollado por Newtonium. Es un software adecuado para:

- Modelado y animación 3D de sistemas robóticos y mecánicos.
- Uso como herramienta de docencia de cursos de grado e inferiores.
- Fácil adicción de gráficos 3D a lenguajes C/C++, C/C++ interpreter Ch, VB, VB.NET, LabView, etc sin necesidad de programación.
- Amplia colección de modelos de robots industriales.

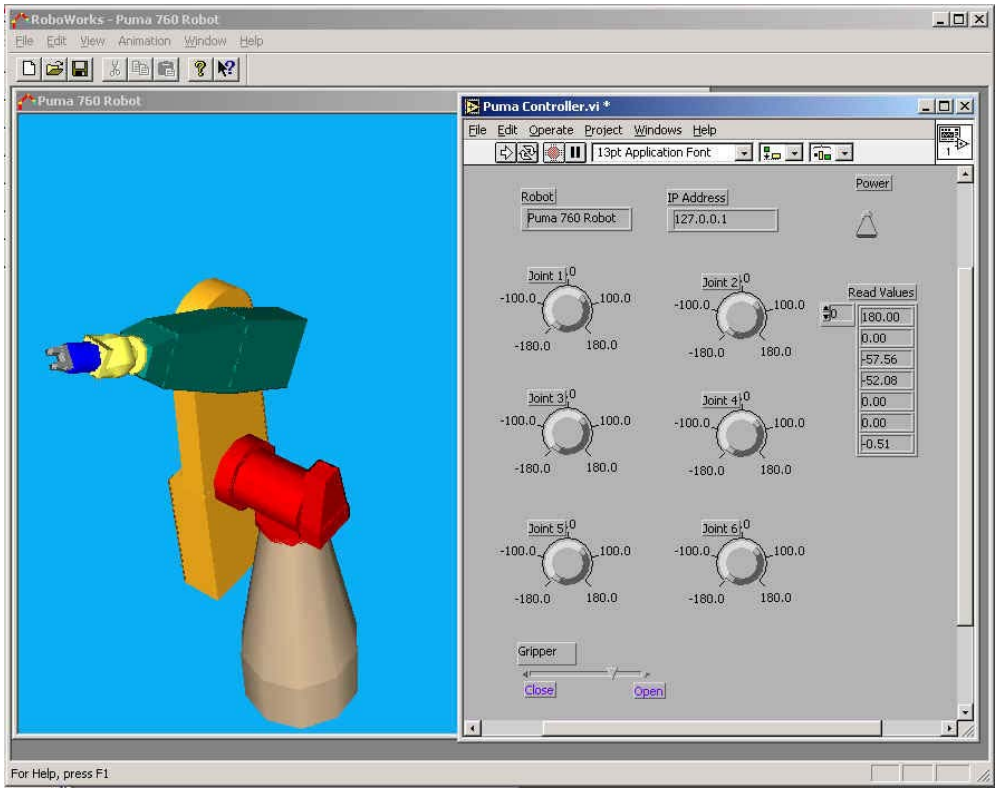


Figura III.2- Imagen del programa RoboWorks

Es un programa fácil de usar, válido para el modelado, simulación y animación de cualquier sistema físico. Algunas de sus ventajas son:

- Desarrollo de modelos intuitivo.
- Gráficos 3D interactivos y de gran calidad.
- Archivos de poco tamaño, poco agresivos con el espacio en disco.
- Animación completa incluso en la fase de creación del modelo.
- Interfaces con paquetes de software comunes.

En este último punto cabe destacar que muchos de los softwares de análisis y visualización tales como Matlab, MathCad y LabView, disponen únicamente de gráficos limitados a dibujos y tablas. RoboWorks es capaz de añadir modelos 3D animados a estos softwares. Es por tanto posible realizar el análisis en un cierto software a elección y ver y animar los resultados en RoboWorks.

El programa aporta una animación interactiva mediante el teclado, posible a la vez que se crea el modelo, a través de un archivo de datos o a través de RoboTalk™. RoboTalk es una interfaz de programación de aplicaciones de Código Abierto, que permite que cualquier programa que se ejecute en una plataforma que soporte TCP/IP pueda interactuar y controlar los modelos de RoboWorks en tiempo real.

En la siguiente lista se presenta un resumen de las características comentadas y algunas más:

- Múltiples vistas 3D con propiedades individualizadas.
- Modelado jerárquico, usando nodos de varios tipos.
- Plantillas 3D.
- Motor gráfico OpenGL.
- Animación a través del teclado.
- Animación a través de ficheros de datos ASCII.
- Código fuente RoboTalk como interfaz para programas externos.
- Interfaz LabView.
- Interfaz Matlab.
- Interfaz Visual Basic.

Así pues RoboWorks se presenta como una herramienta capaz de modelar y animar todo tipo de gráficos 3D, y con una alta capacidad de integración con algunas de las aplicaciones más comunes. Presenta un alto carácter generalista.

02. Sistemas propietarios

▪ INSER ROBÓTICA

Inser Robótica [5] es una empresa creada en 1986 y pionera en la instalación de sistemas robotizados en España.

En la actualidad, la compañía puede presumir de haber diseñado y montado más de 800 aplicaciones, que suponen la instalación de más de 1.000 robots, lo que avala la profesionalidad de la empresa y el grado de satisfacción de los clientes.

La empresa está especializada en el diseño, construcción, instalación y puesta a punto de sistemas robotizados de paletizado, soldadura, manipulación y carga y descarga, para diversos sectores.

La compañía distribuye en exclusiva para España y Portugal los robots industriales de las marcas **Kawasaki** y **Panasonic**.

Una de las principales líneas de la empresa es la distribución de sistemas integrados CAD-CAM-CAE de fabricación que conecten directamente el Departamento Técnico con el Departamento de Producción de una empresa. En este ámbito, para ofrecer una mayor flexibilidad y rentabilidad en los procesos productivos, así como facilidad en el manejo de los sistemas robotizados, Inser Robótica integra aplicaciones de simulación como complemento.

Así se puede ver en un PC el entorno en CAD idéntico a la instalación real, y se puede programar el sistema pudiendo ver los resultados simulados en tiempo real (CAM) y posteriormente volcarlos al robot. De la simulación se obtienen tiempos de ciclo y otra mucha información, con lo que es posible valorar el costo de la operación y mejorar la planificación de la producción (CAE).

Inser Robótica S.A. entrega el programa ya personalizado para el cliente y su sistema robotizado, con modelos 3D de la instalación real para trabajar directamente, sin retoques. Incluye licencia de uso y curso de formación.

Existen dos herramientas de software, cada una destinada a un fabricante de robots. Las principales características de estos programas son:

PC-ROSET

- Reducción de los costes.
- Programación off-line
- Optimización de los tiempos de ciclo del robot.
- Chequeo de las colisiones.
- Muestreo de la trayectoria y análisis del puesto robotizado.
- Posibilidad de trasladar programas entre distintas células de fabricación.
- Planificación del proceso, definiendo las fases necesarias para el mismo.
- Curva de aprendizaje del sistema muy corta, ya que el comportamiento del sistema y el interfaz es idéntico al del robot.

Figura III.3- Programa PC Roset para robots Kawasaki

DTPS II

- Reducción de los costes y plazos de entrega.
- Optimización de los ciclos de trabajo del robot.
- Eliminación de colisiones del robot en el proceso de diseño y simulación de las estaciones.
- Posibilidad de trasladar programas entre distintas células de fabricación.
- Prediseño de utillajes sin que interfieran con las trayectorias del robot.
- Planificación del proceso, definiendo las fases necesarias para el mismo.
- Posibilidad de generar programas parametrizados de piezas viendo los resultados simulados en tiempo real, con lo que es posible valorar el costo de la operación de soldadura.
- Sistema backup de programas que hace virtualmente infinito el número de programas del robot (con el límite de la capacidad del disco duro del PC).
- Curva de aprendizaje del sistema muy corta, ya que el comportamiento del sistema y el interface es similar al del robot.
- Posibilidad de convertir programas de distintos modelos de robots Panasonic y transferirlos entre distintas instalaciones (para la misma pieza).

Figura III.4– Programa DTPS II para robots Panasonic

Observamos cómo estas herramientas nos proporcionan las ventajas de la simulación off-line, así como un fácil aprendizaje de uso. Sin embargo, son herramientas específicas de dos fabricantes y normalmente se entregan listas para el uso, es decir, pensadas y preparadas para un propósito particular, perdiendo flexibilidad para nuestro propósito.

Toda la información relevante a esta empresa y sus herramientas se puede encontrar en su página web.

- **KUKA**

KUKA Robot Group [6], es un grupo que se encuentra en los lugares principales en el mercado de provisión de robótica industrial, y es además una de las empresas pioneras del mismo. KUKA ofrece una gama de robots industriales y sistemas de robots que cubre las clases de carga y los tipos de robots más habituales.

En 1996 KUKA se convirtió en la primera compañía en lanzar al mercado un robot industrial con unidad de control basada en PC, ofreciendo una trabajada interacción entre software, sistema de control y mecánica, una auténtica aplicación mecatrónica. Ésta es una de las principales características de sus soluciones hasta el día de hoy. Los estándares de KUKA empleados en esta tecnología han convencido a usuarios de robots de todo el mundo, por lo que la compañía es actualement el líder del mercado de robots industriales con unidades de control de base PC a nivel mundial.

Todos sus robots presentan una plataforma de control de base PC, llamada KR C, siendo posible por tanto aprovechar las ventajas de las tecnologías PC: diagnóstico a distancia, interfaz Windows, bus de campo, etc.

En cuanto a simulación de sus aplicaciones, la compañía dispone del programa KUKA.Sim, para ofrecer a sus clientes, antes de iniciar la puesta en servicio, la posibilidad de comprobar el proceso y, si procede, optimizarlo y validarlo.

El programa se compone de cuatro aplicaciones diferentes que se pueden utilizar coordinadamente para aprovechar todas las utilidades del programa, o por separado, si solo se quiere utilizar cierta funcionalidad. Estas aplicaciones se presentan a continuación.

KUKA.Sim Pro

Figura III.5- Imagen de KUKA.Sim Pro

Éste es el software para programar off-line robots de KUKA. Como una de sus características más destacables, se encuentra la posibilidad de conectarse en tiempo real con la aplicación KUKA.OfficeLite para realizar el control virtual, análisis de ciclos y generación de programas de una estación.

KUKA.Sim Layout

KUKA.Sim Layout es un programa que permite elaborar diseños en 3D de instalaciones equipadas con robots de KUKA. Con él se puede simular y examinar sin esfuerzo cualquier diseño y concepto.

Figura III.6- Imagen de KUKA.Sim Layout

KUKA.Sim Viewer

KUKA.Sim Viewer es un software que permite visualizar simulaciones elaboradas con KUKA.Sim Layout o KUKA.Sim Pro. Ya que permite observar simulaciones en 3D y leer notas, KUKA.Sim Viewer es la herramienta de comunicación ideal entre el proveedor y el cliente. Es gratuito.

Figura III.7- Imagen de KUKA.Sim Viewer

KUKA.OfficeLite

KUKA.OfficeLite es la unidad virtual de control de KUKA, y permite crear y optimizar programas para robots KUKA en cualquier PC.

KUKA.OfficeLite es casi idéntico al software estándar KR C. Gracias al uso de la interfaz de usuario original KUKA y de la sintaxis KRL, el manejo y la programación fuera de línea coinciden exactamente con los del robot. Los programas se pueden transmitir desde el sistema de programación KUKA.OfficeLite al robot por medio de una red o de un disquete.

Dispone de todo el repertorio de funciones de las respectivas ediciones del software de sistema.

Presenta comprobación de sintaxis mediante el compilador y el interpretador disponibles.

Presenta control completo de la ejecución de un programa de aplicación de robot. Ello permite optimizar la duración de los ciclos aunque éstos no se puedan determinar con exactitud debido a la ausencia de tiempo real preciso.

Las entradas originales se pueden simular.

KUKA.OfficeLite no se puede utilizar para controlar un robot.

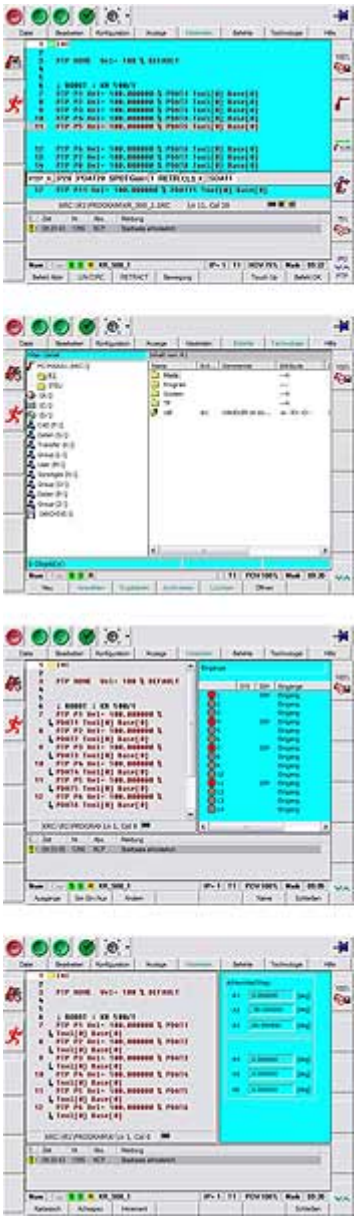


Figura III.8- Imágenes de KUKA.OfficeLite

Otras de las características generales de estas aplicaciones son:

Modelado

Uso de filtros incorporados para importar diseños CAD de otros sistemas o posibilidad de elaborar geometrías con las herramientas CAD de KUKA.Sim Pro.

Existen amplios catálogos electrónicos que se suministran con KUKA.Sim Layout, o modelos que se descargan de Internet. La mayoría de estos componentes del catálogo electrónico son configurables en dimensiones para ofrecer mayores posibilidades al usuario y ahorrar tiempo de diseño. Además, se incluyen de manera sencilla en la célula de robot, mediante un simple clic sobre los puntos "power-snap" de los componentes.

Simulación

Función de comprobación de los puntos incluidos en el programa del robot son alcanzables. Además, con el uso añadido del detector de colisiones se elabora un "buen" programa de robot y unos "buenos" diseños.

KUKA.Sim Pro permite elaborar y simular garras, pinzas de soldadura y otras estructuras cinemáticas.

Las líneas transportadoras del tipo "push-pull" (empuje-tracción) y las líneas transportadoras con velocidad constante pueden ser simuladas en KUKA.Sim Pro.

Realización de programas de robot con KUKA.OfficeLite directamente en KRL. Los programas de robot que han sido elaborados "sobre el terreno" pueden ser cargados uno por uno en KUKA.OfficeLite, de forma que estos programas pueden continuar operando sin problemas en KUKA.Sim Pro.

Comunicación

KUKA.Sim Pro está conectado en tiempo real con KUKA.OfficeLite, la unidad virtual de control de KUKA. Este programa corresponde en un 99% al software que opera en las unidades de control "reales" de KUKA. Eso permite pronosticar la duración de los ciclos con extraordinaria exactitud

El tamaño de los datos de las simulaciones elaboradas con KUKA.Sim Layout y KUKA.Sim Pro es extraordinariamente pequeño. Es por tanto sencillo y poco agresivo el enviar simulaciones por correo electrónico a clientes u otras personas interesadas en demostraciones de los procesos, que podrán visualizar con KUKA.Sim Viewer.

Toda la información relacionada con estas aplicaciones, así como la manera de adquirir licencias o las ofertas de cursos relacionados, se encuentran en la página web mencionada.

- **ROBOTSTUDIO**

RobotStudio [7] es el software para simulación y programación off-line que la empresa ABB desarrolla para sus robots industriales. Esta empresa

es líder en suministro de dichos robots, apuesta por proporcionar soluciones completas a sus clientes, y ya ha instalado más de 150.000 robots en todo el mundo.

RobotStudio forma parte de una familia de productos de software que la empresa ofrece a sus clientes para mejorar su productividad y reducir costes, y que sirven de apoyo a la gestión del ciclo de vida de sus soluciones robóticas.

Con esta herramienta, ABB ofrece las ventajas generales de la programación off-line, entre las que se pueden destacar el permitir programar los robots en PCs sin parar la producción o la posibilidad de preparar los programas con antelación, para mejorar la productividad. RobotStudio pues, incorpora herramientas que repercuten en la mejora de la rentabilidad de los sistemas, permitiendo llevar a cabo planes de formación, programación y optimización de sistemas sin parar los robots.

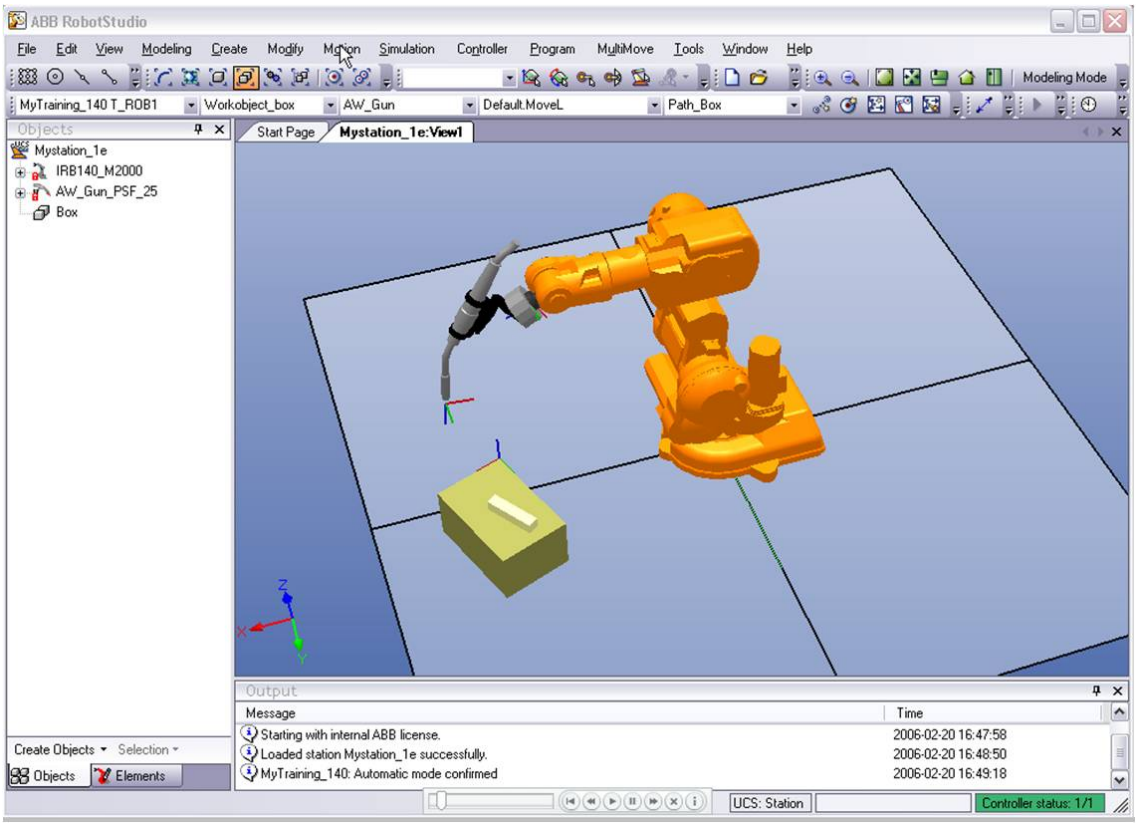


Figura III.9- Imagen de RobotStudio 5

Al instalar RobotStudio, se instala con él el **controlador virtual (VC)** que es una copia exacta del software real que usan los robots en la producción. Así se llevarán a cabo simulaciones muy realistas y se facilita el proceso de carga/descarga de programas del/al robot hacia/desde el PC.

Algunas características del programa se exponen a continuación:

- Importar **CAD:** Permite importar una amplia variedad de archivos CAD (IGES, STEP, VRML, VDAFS, ACIS, CATIA)
- Generación de caminos: Se pueden **generar caminos** de puntos a partir de una superficie o curva.
- Editor de Programas: Incorpora el software ProgramMaker, que permite realizar y modificar **off-line** programas RAPID de manera rápida y sencilla (en entorno Windows), repercutiendo en ahorros de tiempo de diseño.
- Permite comprobar si los puntos incluidos en el sistema son **alcanzables** o no con la configuración actual.
- Teach Pendant (unidad de programación) virtual: El programa dispone de la representación gráfica de un **Teach Pendant virtual** que permite realizar todas aquellas operaciones que se pueden realizar con el real.
- Tabla de eventos: Utilidad para gestionar eventos como colisiones o simulación de **señales I/O**. Las señales I/O se pueden visualizar.
- Detección de **colisiones**: El programa permite detectar colisiones de los diferentes juegos de objetos seleccionados.
- **VBA**, Visual Basic para Aplicaciones: RobotStudio permite usar VBA, que es una herramienta muy común y potente en aplicaciones informáticas.
- PowerPac's: Existen packs creados por ABB y sus socios, en lenguaje VBA, que extienden el uso del software, p. ej. permitiendo optimizar ciertas aplicaciones como las de soldado.
- Auténtica carga y descarga: Todo el programa RAPID puede ser **transferido del robot al software de simulación** y viceversa sin necesidad de ninguna interfaz o traductor. Es una cualidad única sólo suministrada por ABB gracias a su tecnología VirtualRobot technology.

Por tanto, este programa ofrece las ventajas comunes a la mayoría de alternativas. Toda la información comentada, además de más sobre casos estudiados, cursos, descargas, etc… se puede consultar en la página web.

Dado que el desarrollo del proyecto se basa en los robots ABB del laboratorio de la Universidad Carlos III de Madrid, estas características hacen que sea la opción más interesante, y por tanto la que se eligió para la realización de la simulación y demostración.

IV - IMPLEMENTACIÓN EN ROBOTSTUDIO DE PROCESOS BÁSICOS DE ENSAMBLADO

IV.1 - PROGRAMACIÓN DE UNA ESTACIÓN Y SIMULACIÓN

01. INTRODUCCION A ROBOTSTUDIO
02. CREACIÓN DE LA ESTACIÓN

IV.2 - APLICACIÓN DE LA METODOLOGÍA A LA IMPLANTACIÓN DE UN PROCESO REAL DE ENSAMBLADO

01. CREACIÓN DE LA ESTACIÓN
02. SIMULACIÓN DEL ENSAMBLADO
03. TRANSFERENCIA AL ROBOT REAL

IV.1 - PROGRAMACIÓN DE UNA ESTACIÓN Y SIMULACIÓN

01. Introducción a RobotStudio

En esta sección se realiza la presentación al usuario de la herramienta RobotStudio, de ABB. Se explican las generalidades del entorno y las funcionalidades del mismo. En las secciones posteriores se desarrollará una explicación más a fondo del programa, a la par que se explican los pasos dados para la programación y simulación de la estación. La terminología relativa al programa se puede consultar en la sección "Glosario".

Como ya se ha comentado con anterioridad, la herramienta elegida para desarrollar este estudio base del robotizado del Service Core, es el software de ABB RobotStudio. Esta herramienta se presenta en dos versiones: 3.0 y sucesivas, y 5.0 y sucesivas, entendiendo como sucesivas las actualizaciones, mejoras, nuevos lanzamientos, etc, (incluyendo las versiones 4.0 que se desarrollan a partir de la 3.2). Durante el presente año 2008, ABB lanzará la nueva versión del software, RobotStudio 2008.

La versión 5.0, la más moderna, trabaja con robots de controladores tipo IRC5 (igualmente, los robots ABB más modernos) mientras la 3.0 implementa controladores S4, que es el caso de nuestro robot, IRB2400_M94A (controlador S4 versión V21, la más antigua entre los S4). Por lo tanto, se va a realizar una visión general de la versión 3.0, que es la que se utiliza (aún así, muchas de las características del programa son compartidas por todas las versiones, lo que hace fácil el paso de la utilización de unas a otras).

Esta introducción muestra las características básicas del programa y su interfaz a tener en cuenta para poder realizar las acciones más comunes que se desarrollan a lo largo de la programación de una estación, como p. ej. cómo mover el entorno gráfico, el manejo de los menús, etc.

a) Ejecución del programa

Una vez instalado en el ordenador, el programa se ejecuta desde el menú Inicio→Programas→ABB Automation→RobotStudio y haciendo click en RobotStudio.

b) Entorno de usuario

Al abrir por primera vez el programa aparecerá una ventana similar a la de la figura siguiente:

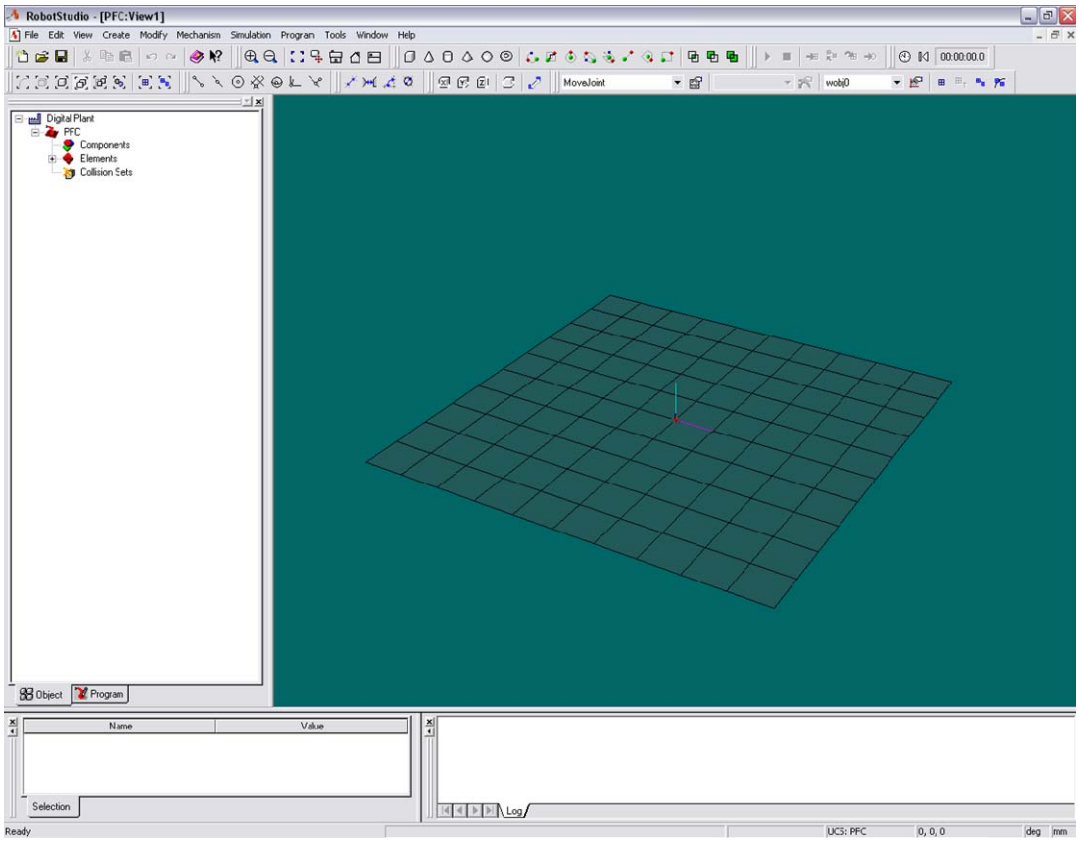


Figura IV.1.1- Entorno de usuario

Como se observa, presenta un **interfaz amigable** y similar a mucho de los programas informáticos más comunes en entorno Windows, como los del paquete office. Las distintas áreas de la pantalla se pueden ver en la **Figura IV.1.2** y son las siguientes:

- **Menús**: menús desplegables que permiten realizar todo tipo de acciones, desde abrir archivos hasta simular estaciones.
- **Barras de Herramientas**: Accesos rápidos a funcionalidades del programa.
- **Navegador**: Ventana en la que se muestran de manera jerárquica y se permiten seleccionar los objetos de la estación. Posee dos pestañas: Objetos y Programas
- **Ventana de salida**: Muestra mensajes de información del programa. La información se clasifica en diferentes pestañas (log, robotics, …)

- **Barra de estado**: Muestra algunos de los parámetros actuales de la estación, como el estado del controlador virtual, o la posición del UCS (Sistema de coordenadas del usuario). El UCS elegido es en el que se muestran las coordenadas.
- **Ventana gráfica**: En ella se pueden observar los objetos en 3D de la estación, seleccionarlos, etc. Es la extensión gráfica del navegador.
- **Ventana de propiedades** o de información: Muestra información diversa sobre los objetos seleccionados.

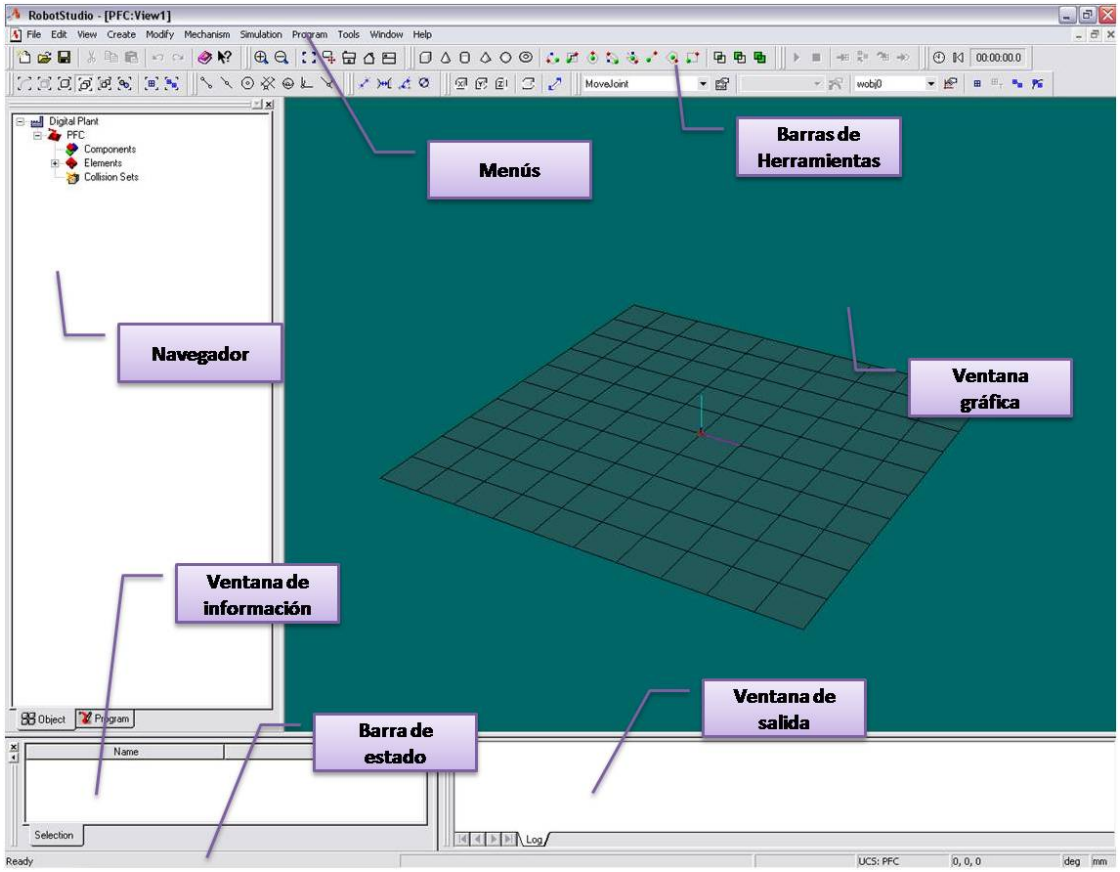


Figura IV.1.2- Áreas del entorno gráfico

El entorno es fácilmente **configurable y personalizable**. Por ejemplo, se pueden cambiar los tamaños de todas las ventanas, o los colores de la ventana gráfica. También se pueden ocultar o mostrar las diferentes ventanas. Incluso existe la posibilidad de abrir varias ventanas gráficas a la vez, que nos ofrezcan diferentes vistas de la estación. A fin de pequeño ejemplo, y para distinguir mejor en las capturas de pantalla el fondo de la estación de la rejilla del plano, se puede cambiar el color de fondo. Para ello, hay que hacer click derecho en el fondo y elegir un color deseado. El resultado se observa en la **Figura IV.1.3.**

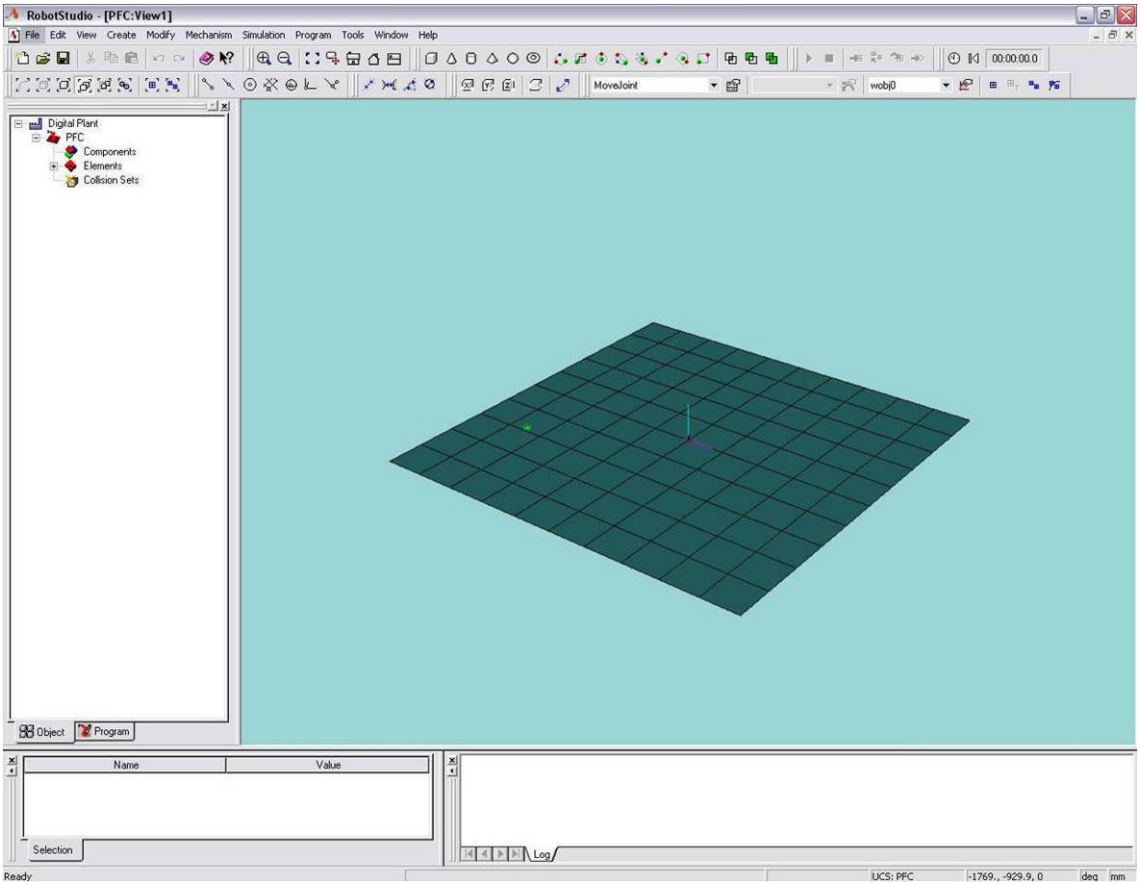


Figura IV.1.3– Fondo de la ventana gráfica cambiado

Los comandos de los menús cuentan con los más usuales en las aplicaciones informáticas comunes, como los menús:

- **File** (Archivo): usado para abrir/cerrar ficheros, guardar, etc.
- **Edit** (Editar): que contiene los comandos de copiar, pegar, deshacer, etc.
- **View** (Ver), el cual sirve para activar/desactivar las barras de herramientas y configurar las vistas.
- **Window** (Ventana), para crear vistas y configurar la alineación de las ventanas gráficas.
- **Help** (Ayuda) que contiene la ayuda del programa.

Más específicos del propio programa son el resto de menús, **Simulation**, **Program**, **Mechanism**, o **Create**. Muchos de los comandos de los menús necesarios para la realización del proyecto se comentarán a lo largo de este documento, al tiempo que se explica el proceso de creación de la estación, haciendo uso de gran parte de esos comandos de los que dispone el programa.

c) Trabajar con la ventana gráfica

Uno de los aspectos básicos más importantes para trabajar con el programa es tener un dominio de la ventana gráfica, en la que se desarrolla gran parte de la creación y gestión de la estación. Obtener una orientación óptima y saber cómo seleccionar objetos adecuadamente es primordial para realizar adecuadamente el trabajo. Algunas de las operaciones y consideraciones más importantes sobre la ventana gráfica son:

- **Selección**: Se selecciona un objeto haciendo click con el ratón sobre él en el área gráfica, o en el navegador, teniendo en cuenta el nivel de selección.
- **Nivel de Selección** (Selection Level): Establece el nivel de selección que se quiere ajustar al seleccionar los componentes. Hay varias posibilidades (pieza, mecanismo, curva…) que se eligen en el menú Ver o en la barra de herramientas.
- **Modo de Ajuste** (Span Mode): Establece el nivel de ajuste que se quiere tener al seleccionar, como por ejemplo el final de un objeto, el centro, una intersección, etc… que permite seleccionar justo el punto que se desea al pinchar en sus cercanías.
- **Rotar**: Al presionar a la vez los botones derecho y central (o bien la ruleta central) del ratón, o las teclas Ctrl y May junto con el botón derecho, se puede rotar el área gráfica sobre su punto central al mover el ratón.
- **Desplazar**: Si se presiona Ctrl y el botón derecho del ratón se puede desplazar el área gráfica "a mano alzada" desplazando el ratón. Con ello desplazaremos el punto central.
- **Zoom**: Se puede hacer zoom con el botón central o ruleta del ratón presionándola o pulsando Ctrl y el botón derecho, moviendo a la vez el ratón a izquierda o derecha.
- **Zoom de ventana**: Al mantener pulsado May y arrastrar el ratón con el botón derecho pulsado, se realiza un zoom sobre la selección.
- **Selección de área**: Manteniendo pulsado May y arrastrando el ratón con el botón izquierdo pulsado, se seleccionan todos los objetos incluidos en esa área, del nivel de selección ajustado.

Se debe notar que los comandos de movimiento del área gráfica, como rotar, dependen del modo de movimiento que tengamos seleccionado en View→Navigation→Navigation Type. Existen dos:

- TrackBall: El plano se desplaza y rota en torno al punto central.
- Walkthrough: El plano se desplaza y rota tomando como referencia el movimiento del puntero.

Otro de los comandos útiles a la hora de manejar la ventana gráfica es **Set Center View Point** (establecer como punto central de la vista) que centra

la vista sobre el punto actual señalado. Se puede ejecutar en la barra de herramientas de vistas o en View→Navigation→View Center (acceso rápido Alt+6).

Además, existen posibilidades de personalización en el menú **Tools→Customize** y en el **Tools→Options**, que permiten configurar diferentes aspectos gráficos, las unidades de las magnitudes, los botones de las barras, y muchas otras opciones.

Estas y otras acciones se pueden ejecutar desde las respectivas barras de herramientas, como la **barra Ver**, en la que además de hacer zoom se puede seleccionar el punto central, ajustar la vista para ver toda la estación, o elegir una vista del alzado, perfil o planta; y las **barras Nivel de Selección y Modo de Ajuste**.

Figura IV.1.4- Barras Ver, Nivel de Selección y Modo de Ajuste

También es útil tener en cuenta la ayuda propia del programa, disponible en el menú de ayuda (Help→Help) que puede indicar como solucionar algunos de los problemas que se presenten. Por ejemplo, es útil tener presente la lista de accesos directos que proporciona el programa, a la que se accede en la ayuda, en How to work with RobotStudio→How to work with shortcuts, y que se puede ver en los anexos a este documento (capítulo VII.3).

Estas indicaciones componen un conjunto de las nociones básicas para poder utilizar la interfaz del programa. Durante el desarrollo de la creación de la estación y posterior simulación, se comentarán muchos más de los aspectos y posibilidades que ofrece el programa y que permitirán al usuario conseguir un conocimiento amplio del mismo al tiempo que se desarrolla la estación que se pretende simular.

02. Creación de la estación

Una vez hecha la introducción básica al entorno del programa y a la navegación por el mismo, en esta sección se explica el desarrollo de la creación de la estación que servirá para simular un proceso de robotizado del Service Core. En este desarrollo se explicarán también muchas de las funcionalidades que ofrece el programa para conseguir realizar un modelado y simulación provechosos para los propósitos del usuario, que en nuestro caso son mostrar el procedimiento y filosofía básicos que se implementarán en la producción del Service Core. Además, con ello también se logrará, gracias a la explicación de muchas de sus funcionalidades, llegar a tener un conocimiento y dominio bastante amplio de la herramienta.

El procedimiento general que se sigue para realizar el diseño, creación y simulación de la estación se expone en la figura siguiente. Este flujograma presenta la visión general del proceso, admitiendo flexibilidad en su interpretación, ya que, por ejemplo, se pueden crear procesos y simularlos camino a camino, ó en conjuntos más amplios, como se explica a continuación.

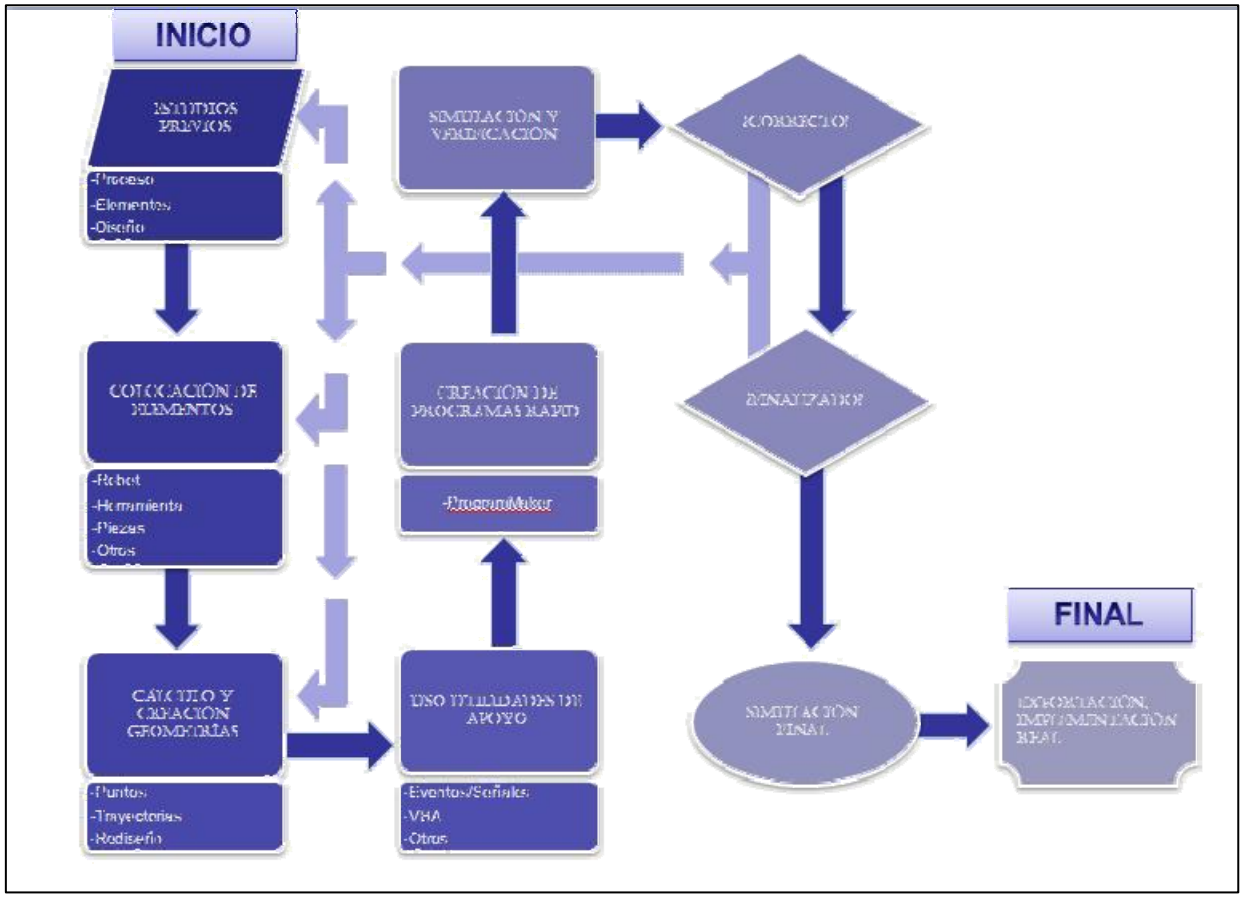


Figura IV.1.5- Flujograma del proceso de creación y simulación de una célula

Siguiendo este esquema, la realización de la simulación de una estación seguiría los siguientes pasos:

- Estudios previos, que detallarían el proceso, el diseño de la estación, la elección del robot y de la herramienta, de las piezas, etc, de la estación a simular.
- Colocación de los elementos elegidos, tales como el robot, las piezas, las herramientas, y otros objetos de la estación (mesas, posicionadores…).
- Cálculo y creación de las diferentes geometrías necesarias, es decir, de los puntos y las trayectorias que se pretende que siga el robot. Se pueden crear el número de geometrías que se considere oportuno:
    - Crear pequeños grupos (trayectoria por trayectoria, por ejemplo) y seguir con el proceso de simulación de cada uno, para comprobar la correcta elección de geometrías, puede resultar interesante (y volver a crear el siguiente grupo de geometrías hasta acabar con todas las necesarias).
    - Sin embargo también se pueden crear grupos más grandes, hasta incluso el total. Hay que tener en cuenta que a mayor número de puntos y trayectorias creados, más difíciles serán los posteriores rediseños de los mismos (en caso de que surjan fallos o se quieran realizar cambios).
- Uso de utilidades de apoyo, tales como la tabla de eventos, Visual Basic, y otras que ofrece el programa para completar la programación de nuestra estación (con la totalidad o con parte de las trayectorias).
- Creación de los programas Rapid necesarios para el correcto funcionamiento de la estación, ayudándose de la herramienta ProgramMaker.
- Simulación y verificación de que el resultado es el deseado. En caso de no haber realizado la programación y simulación del proceso completo, de existir fallos o de la voluntad de cambiar algún paso anterior, se vuelve al paso de los estudios previos, de la colocación de elementos, o del cálculo de geometrías, según lo que requiera el caso.
- Si todo es correcto y se ha realizado la simulación completa, el proceso ha terminado. Ya tenemos nuestra simulación lista para ser estudiada, presentada o transferida a un robot real.

Para saber aplicar la metodología mencionada, a continuación se muestran los pasos y comandos básicos del programa necesarios, aplicados a la creación y simulación del proceso de ensamblado del Service Core.

a) Importar un Robot

Una vez elegidos los elementos que se pretenden colocar, el primer paso para crear la estación es importar el robot que se quiere simular (en nuestro caso el IRB2400_M94A). Para ello simplemente se hace click en **File→Import→Library** (importar librería), o bien se pulsa Ctrl+M. En el subsiguiente menú desplegable se elige la ruta de librerías de robots (típicamente será C:\Archivos de programa\ABB Robotics\Library\Robots) y se selecciona la librería de nuestro robot: IRB2400_M94A.rlb, como ilustra la siguiente figura:

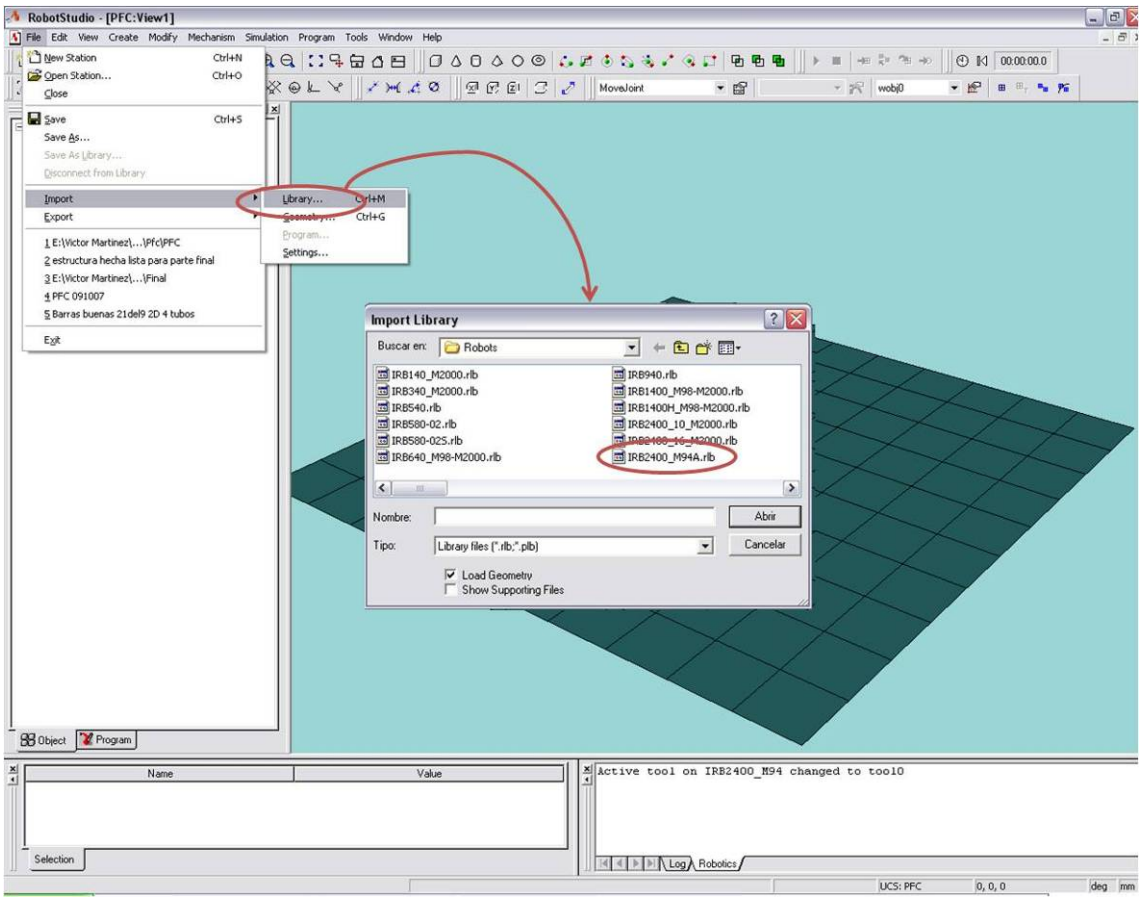


Figura IV.1.6- Importar el robot

Ya se puede ver el robot situado en el centro del plano (pto 0, 0, 0). El robot, como todos los elementos gráficos, también es configurable, pudiéndose cambiar su color total o parcialmente, y hacerlo invisible en el plano, en su totalidad (todo el mecanismo) o cada uno de sus componentes.

Se puede notar que el robot ha sido incluido en el navegador en la lista de componentes (**components**) del sistema, así como todos sus subcomponentes o piezas. La pestaña de objetos del navegador los clasifica en **componentes** de la estación (es decir, los gráficos creados o importados a la estación); **elementos** de la estación, que son todos aquellos puntos/objetivos y/o trayectorias/caminos presentes; y finalmente conjuntos de **colisión** (collision sets), que sirven para gestionar colisiones.

También se puede observar al pinchar en una pieza del robot, que ésta se selecciona, quedando resaltada en color rojo. Esto es debido a que se encuentra seleccionado el nivel de selección "Pieza". Se puede comprobar que si se elige el nivel "Mecanismo", al pinchar en cualquier punto del robot, se selecciona entero, puesto que el robot es un mecanismo. Otra posibilidad consiste en seleccionar los objetos pinchándolos en el navegador, y en este caso se selecciona sólo el objeto deseado, independientemente del nivel de selección actual, como en la figura siguiente, en la que se ha pinchado en pbase, la base del robot.

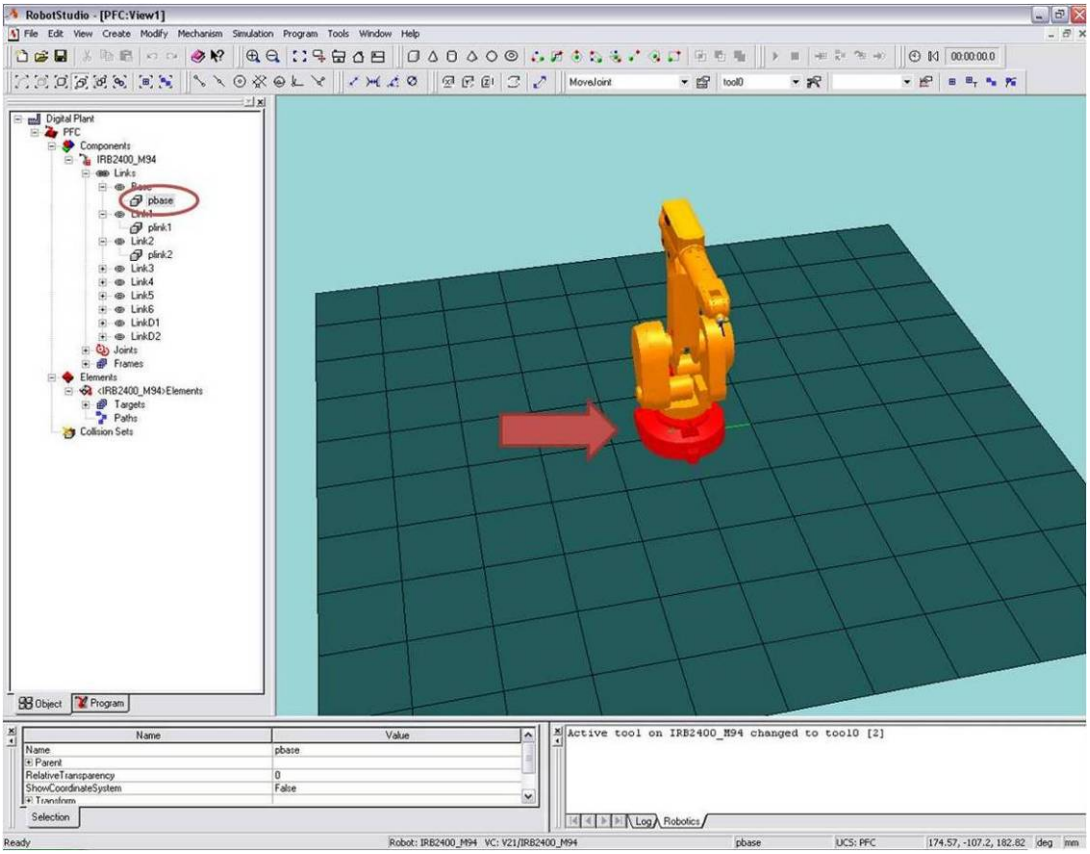


Figura IV.1.7- Base del robot seleccionada y vista en navegador

Se observa también como, tras importar la librería, aparece el mensaje siguiente en la ventana de salida, en la pestaña de robotics:

"Active tool on IRB2400_M94 changed to tool0"

Lo que quiere decir que el robot tiene activada por defecto la herramienta tool0, o lo que es lo mismo, su **TCP** (Tool Center Point), o punto central, está justo en la muñeca. El TCP de la herramienta activa es el punto que el robot posiciona, rota, etc, hasta coincidir con el punto de destino que se ha programado. Cualquiera que sea la herramienta activa, incluyendo tool0, su TCP se desplazará al punto programado (si el robot lo puede alcanzar).

Es importante fijarse también en la disposición espacial del robot, ya que es fundamental conocer perfectamente la posición y orientación espacial de cada componente. Si se observa la base, se vé el SDC (sistema de coordenadas) del mundo. Éste se compone de:

- Eje X positivo: Línea roja.
- Eje Y positivo: Línea verde.
- Eje Z positivo: Línea azul.

Por tanto, como se observa en la **Figura IV.1.7**, el robot se coloca con su brazo a lo largo del eje X positivo (plano XZ). Además, si se realiza un zoom sobre el extremo del brazo, es decir, sobre la muñeca, o se hace click derecho sobre ella (para esto es conveniente elegir el nivel de selección pieza) y se selecciona examinar ("**Examine**"), para conseguir tener una visión cercana de la muñeca, se observa que la orientación del eje de la muñeca tiene el sentido Z positivo hacia fuera del brazo. Además también se observa que el eje Y es el mismo que el del mundo, y el X está en el plano XZ del mundo. Lo vemos en la siguiente figura:

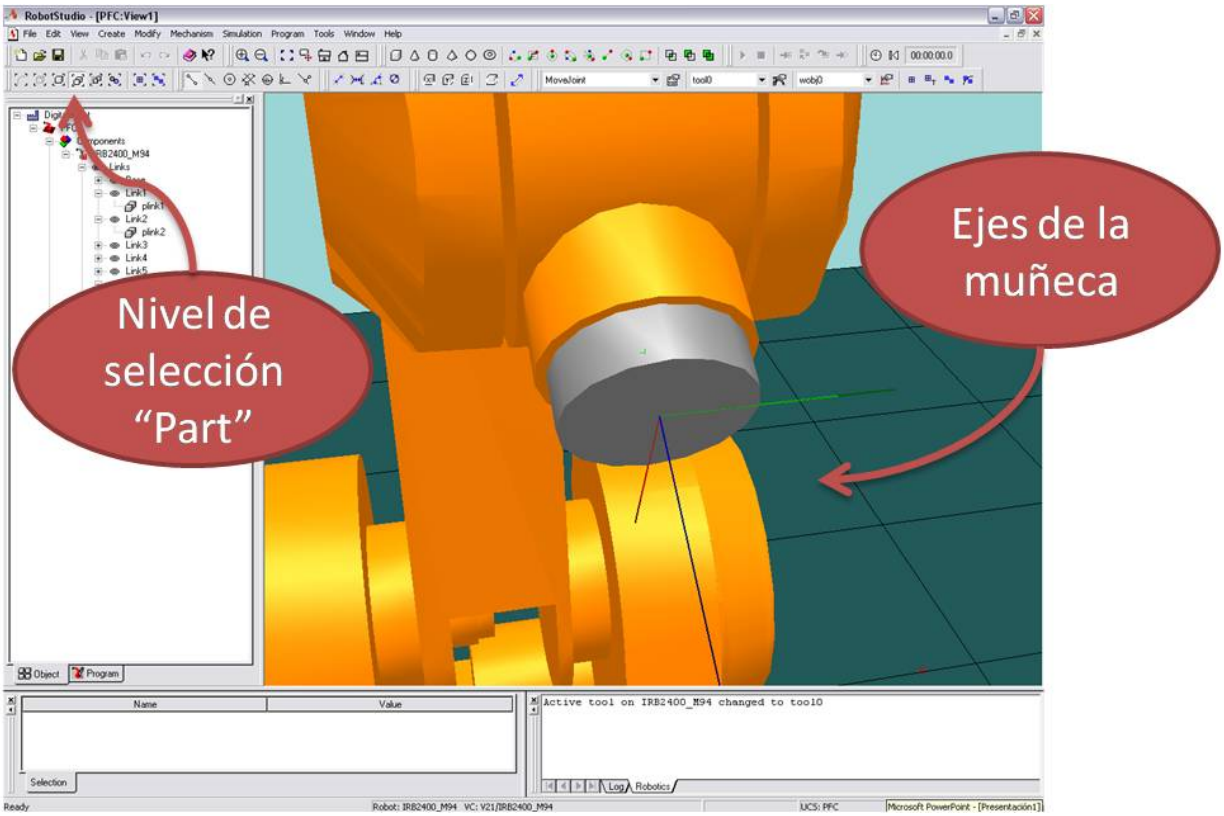


Figura IV.1.8- Ejes de la muñeca

b) Creación de sólidos

Para continuar con la generación de un entorno fiel al diseño de la Mobile Factory comentado en el apartado II.3 se utiliza la posibilidad que ofrece RobotStudio de crear sólidos. Esta operación se puede realizar desde el menú **Create** o desde la barra **Create Components,** como se ve en la **Figura IV.1.9**:

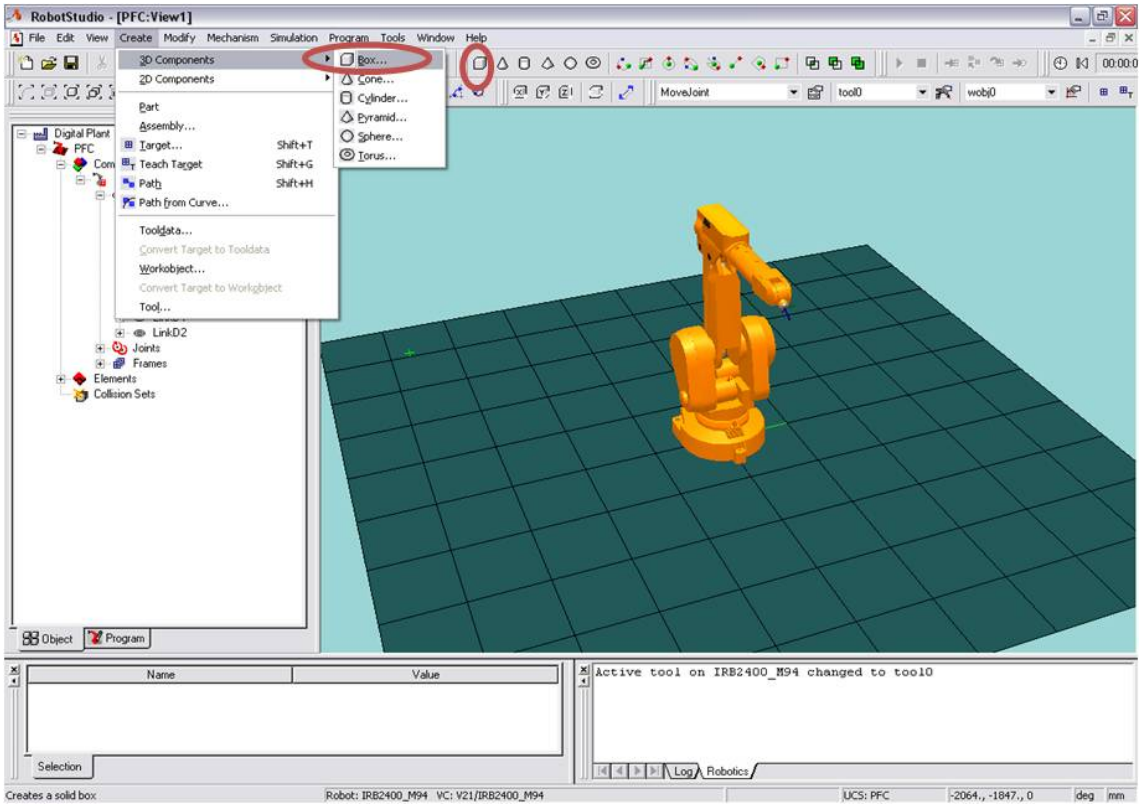


Figura IV.1.9- Las dos maneras de crear un objeto 3D

El layout se va a componer de dos elementos básicos:

1. Una mesa de alimentación de piezas, que hace las veces del transportador/alimentador de piezas o cinta transportadora.
2. La mesa de ensamblado, circular, y a la que se añade el motor **MTC500** (se puede consultar su hoja de características en los anexos, capítulo VII.8) para dotarla del movimiento rotatorio necesario para llegar a todos los puntos de ensamblado.

La mesa se puede generar así: Se crea un sólido 3D como se indica en la **Figura IV.1.9.** Aparece una ventana de diálogo con dos opciones: LWH (largo, ancho y alto) o 3 puntos. En la primera pestaña, se pide un punto de referencia y las dimensiones del sólido. Para elegir el punto de referencia, es útil usar el modo de ajuste de puntos de la cuadrícula ("**UCS Grid**"). Como se observa, **se puede aprovechar el efecto tanto del modo de ajuste como del nivel de selección al rellenar estas ventanas**, lo que resultará muy útil a lo largo de la creación de la estación. Se selecciona el modo de ajuste y se pincha en el punto de la cuadrícula del suelo que se encuentra en (1000, 500,

0) por ejemplo, observando cómo sus coordenadas **se incluyen** en la ventana de diálogo (para esto un campo del "punto de origen" se debe seleccionar con el ratón antes de pinchar en el punto de la ventana que queremos incluir en el diálogo). No es importante situar bien a la primera los objetos o puntos, ya que también se puede realizar su posterior reubicación como se verá más adelante. Una vez situado el origen de la mesa, se introduce directamente en el diálogo las dimensiones de la mesa de alimentación: 3000, 1500, 500. Con esto, ya se tiene dispuesta la mesa, que se sitúa delante del robot.

c) Situar un objeto

Pero en realidad se quiere que se sitúe a la derecha del robot (eje Y negativo) con lo cual se necesita resituarla. Situar objetos es sencillo con RobotStudio gracias al comando **Place.** Se selecciona la mesa con botón derecho y se pincha en **Modify→Place.** Este comando moverá el punto que se introduzca al nuevo punto que también se le proporcione, pertenezca o no al objeto, que se verá afectado de todas maneras, y por tanto **moviendo su SDC local** (esto se puede comprobar haciendo click derecho en el objeto, seleccionando "Show Coordinate System" y observando cómo este SDC, que en este caso coincide con el del mundo, varía al posicionar el objeto con Place). En el diálogo que aparece se selecciona el punto en el que se creó la mesa, (1000, 500, 0) como primer punto a mover ("From Point"), y como primer punto a dónde mover ("To Point") se elige en la cuadrícula el (-1500, 2000, 0). Existen varias pestañas en este diálogo (1-Point, 2-Points, 3-Points, By Frame) de las que en este caso se utiliza la primera de ellas. Una vez realizado esto, se puede elegir un color más oscuro (click derecho en el objeto, Set Color; si se elige "Advanced" podremos ver una previsualización) para la mesa, se renombra como Mesa_Alimentacion (click derecho sobre el objeto, Rename) y debe quedar un diseño semejante al de la **Figura IV.1.10**.

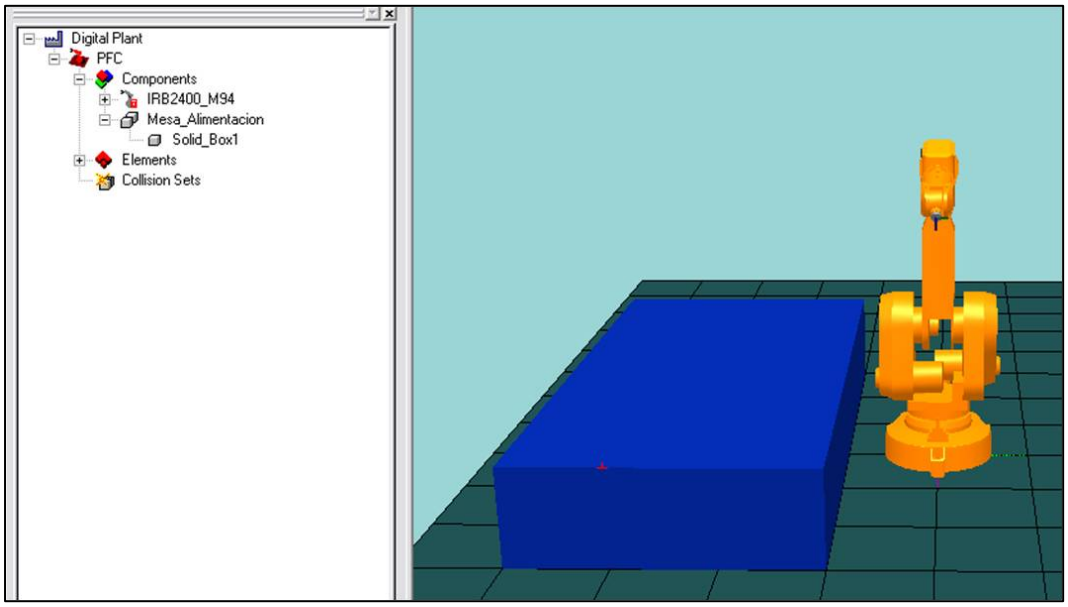


Figura IV.1.10- Mesa de alimentación reposicionada

Una vez situada la mesa de alimentación de piezas, se crea la mesa de salida o ensamblado. Estas dos mesas serán las únicas que creemos en esta estación, y no se tendrán en cuenta pues las mesas de preparación, etc, consideradas en el diseño de la Mobile Factory comentado en el capítulo II.3, pues es suficiente para mostrar la metodología de creación de las estaciones (las posibles futuras estaciones tendrán en cuenta muchos más aspectos) y de trabajo de la propia célula robótica.

d) Creación y conversión de archivos CAD con SolidWorks

Otra manera de incluir elementos en la estación es mediante importación de archivos de otros programas. Para simular la mesa de ensamblado, se crea una mesa circular de 3'5 metros de diámetro y 125 mm de espesor. Este sólido se generó en el programa **SolidWorks** que permite crear todo tipo de geometrías CAD, y además admite y es capaz de generar una cantidad considerable de tipos de archivo distintos (*.sldprt; *.igs; *.step; *.jpg; *.vda; *.cgr; *.eprt…) por lo que también sirve de improvisado conversor de formatos. De estos, los tipos admitidos por RobotStudio son step, igs, sat y vda. Se salva el archivo como Mesa.IGS. Por tanto ya se tiene el problema del soporte circular de la salida solucionado, y solo hay que importarla a RobotStudio con el menú File→Import→Geometry (Ctrl + G).

e) Crear un mecanismo. Kinematic Modeler

Además de esta mesa circular, se necesita el motor que la haga girar para permitir al robot llegar a la mayor parte de puntos de la mesa. Para ello, se descargan de la página web de ABB los modelos CAD del motor [8]. Estos modelos se pueden abrir en SolidWorks, y ya que están en un formato que no admite RobotStudio, **guardarlos y así convertirlos** a una extensión diferente, como step. Además, se necesita hacer un mecanismo con estas geometrías que incorporen el movimiento rotatorio al plato.

Para crear el mecanismo, se importan las dos geometrías (base y placa) a RobotStudio, una vez transformadas gracias a SolidWorks, a una nueva estación que se usa solamente para el fin de generar la librería del mecanismo. Una vez importadas, se observa cómo la placa se presenta centrada en el origen del mundo, mientras la base está desplazada. Se elige un color distinto para cada una de las dos partes para que se diferencien claramente.

Para crear un mecanismo, se usa la aplicación adicional que tiene RS llamada "**Kinematic Modeler**". Esta utilidad se lanza desde el menú Herramientas. Es necesario activarla previamente, si no lo está, en el comando "Add-ins" del mismo menú. Se despliega entonces una ventana de diálogo como la de la **Figura IV.1.11,** en la que se tienen que seguir los siguientes pasos:

1. Ponerle nombre al mecanismo: Motor_MTC500.
2. Elegir el tipo: "External Axis" o eje externo.

3. Añadir los elementos a ensamblar: botón derecho en "links", a continuación Add y seleccionar la base y la placa cada vez (hay que crear al menos dos "links" o partes del mecanismo para que exista movimiento relativo).

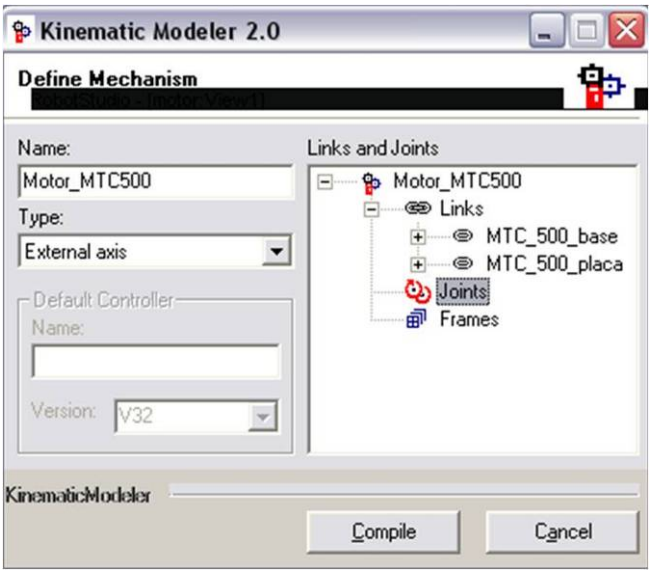


Figura IV.1.11 – Kinematic Modeler (con Links añadidos)

4. Añadir una unión o "joint", que constituye el eje de movimiento relativo. Click derecho en joint, luego add, y en las diferentes pestañas se introduce:
   - el nombre, p. ej. "eje",
   - el elemento padre y el hijo, siendo padre la base e hijo la placa, que es la que se mueve,
   - el eje de movimiento, p. ej. de (0, 0, -1000) a (0, 0, 1000), puesto que tiene que estar en el eje Z,
   - el tipo de movimiento, que será rotacional o "rotational",
   - y el tipo de límites, que se elegirá "constant" y en los límites se introducen los valores -3600 y 3600 grados.
5. Añadir una base al mecanismo, que constituirá su TCP. Se elige un nombre para el TCP y se selecciona el punto central del plato gracias al modo de ajuste "center", que deberá ser el (0, 0, 0). La orientación se elige igual a la del mundo.
6. Una vez añadido todo, pulsar compilar para crear el mecanismo.

Para comprobar el correcto funcionamiento del mecanismo, se pincha en el mecanismo creado en el navegador y se selecciona "**Mechanism Status**", o bien se selecciona el menú "**Mechanism→Show Mech Status Window**". En la

ventana que aparece se puede mover el mecanismo y verificar su correcto funcionamiento. Queda posicionar bien las partes.

f) Modificar posiciones.

Para unirlas, se combinan los comandos Modify→Place y **Modify→Position**. Este último es similar al anterior, pero a diferencia de situar un punto de un sólido, lo que se hace es **modificar directamente la posición** del sólido u objeto, caracterizada por la referencia del punto de origen de su SDC local. El punto de referencia de la placa es el (0, 0, 0). Se mueve con Position a (0, 0, 500) y se observa cómo se desplaza 500 mm hacia arriba todo el sólido. Es de particular interés fijarse en que la ventana de diálogo de este comando presenta cuatro opciones de elección de SDC, que son local (Local), padre (Parent), mundo (World) y UCS. El cambiar el SDC con estas opciones, implica que los puntos introducidos en la ventana pasan automáticamente al nuevo SDC, lo que resulta una utilidad cómoda a la hora de ejecutar ciertos comandos como éste.

Para juntar ambos sólidos, se elige el modo de ajuste centro, y se ejecuta Place sobre la base. El punto a mover se selecciona, gracias al modo de selección, al pinchar en el entorno del hueco circular que presenta la base. La operación resulta más fácil enfocando de cerca la pieza (botón derecho→**Examine**) para comprobar que se selecciona, con una cruz verde o roja, el punto deseado. Hecho esto se desenfoca la pieza (**Unexamine**) y se enfoca el plato, seleccionando gracias también al modo de selección que tenemos activo el punto central de la base circular de la placa. Se pincha con botón izquierdo sobre la base para asegurar que sea ella la que se posiciona (en la ventana de diálogo debe aparecer "Place" y el nombre del sólido a mover). Se pulsa OK y debe quedar una geometría como la de la **Figura IV.1.12**:

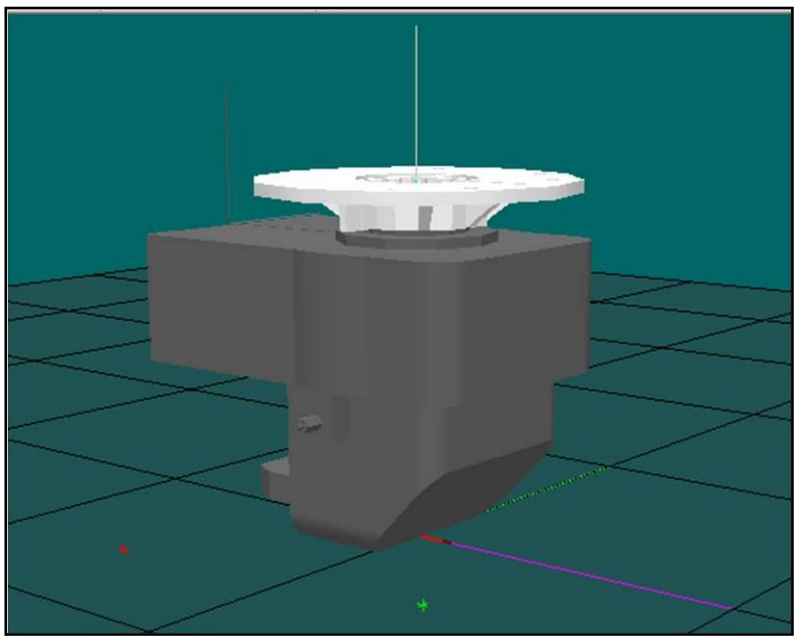


Figura IV.1.12- Piezas del motor unidas

g) Generar una librería

Hecho esto, sólo quedar pinchar en el mecanismo, seleccionar **File→Save As Library** (salvar como librería) y así se salva el modelo de motor completo como una librería que se puede utilizar. Se le denomina MTC500.rlb. Tener esta librería permitirá usarla en otros diseños.

Ya se tiene el motor y la mesa circular. Se importan ambos a la estación (recordando que el motor se importa como librería y la mesa como geometría, y esta última está en formato IGS) y se procede a su correcta ubicación. Deben estar al otro lado de la mesa de alimentación. Con lo cual, hay que colocar la mesa a 500 mm del robot, es decir, en **(0, 2250, 500),** puesto que tiene radio 1750 mm y debe estar elevada del suelo.

h) Rotar un objeto

La mesa está inclinada, hace falta girarla. Para ello se usa otro nuevo comando: **Modify→Rotate**, que permite rotar cualquier objeto sobre el eje que se desee y en el ángulo que se elija en torno a un punto. En este caso, hay que rotar 90º en el eje X local de la mesa y en torno al origen del SDC local de la mesa (para hacer estas rotaciones hay que fijarse en el eje local que se resalta en la ventana gráfica cuando se despliega la ventana de diálogo de rotar), quedando así la mesa por encima de la altura Z = 500 mm. Hay que fijarse en que la superficie superior, donde se efectuará el ensamblado de las piezas, está a una altura de **Z = 500 + 125 = 625 mm**. Se hace coincidir el motor con la base de la mesa, es decir, se sitúa el origen del plato en (0, 2250, 500) con la orientación adecuada.

Para que la estación tenga un mejor aspecto, se añade también un sólido que haga de base para el motor. Se crea un cilindro de, por ejemplo, 400 mm de altura y 3500 mm de diámetro, para tapar el motor y que parezca que éste se encuentra colocado en la base. Para ello hay que posicionar el cilindro debajo de la mesa circular. El espacio libre entre base y mesa circular permitirá observar el movimiento del motor.

i) Fijar piezas

Una vez hecho esto, hay que **unir o fijar** la mesa redonda de salida al motor, para que se mueva solidariamente con éste. Para ello simplemente se pincha en la mesa en el navegador y se arrastra hasta el motor. Aparecerá una ventana que pregunta si se quiere también **reposicionar** la pieza. Hay que contestar que **no**, pues eso cambiaría la posición de la placa, pasando su posición a cambiar su SDC de referencia del SDC del mundo al del motor. Se observa cómo aparece el nombre de la mesa bajo el árbol de componentes del motor, en concreto bajo su base ("frame"), con un símbolo de un clip.

El siguiente paso es **situar las piezas** que van a constituir el modelo de Service Core. Para ello previamente se ha estudiado el proceso constructivo (el paso "Estudios previos" del flujograma):

Para esta simulación sólo se creará una parte de la estructura metálica, la situada en el plano posterior, y sobre la que se colocan algunas de las

tuberías. Con esto, se consigue evitar cruzar los límites que impone el programa y que no permiten realizar el proceso completo, principalmente porque la versión de controladores V21 no es compatible con la utilización de posicionadores de piezas (alimentadores, cintas, etc) que evitarían la restricción espacial que se tiene al coger los objetos alineados en una mesa. Este punto es susceptible de mejora en futuros desarrollos de simulaciones de la célula, utilizando controladores que permitan usar posicionadores, otros softwares, etc… Sin embargo, el proceso que se va a realizar sí es válido para presentar la metodología de desarrollo de la estación y de funcionamiento de la célula robótica.

Así pues, hay que tener en cuenta la secuencia de ensamblado de piezas, a la hora de posicionarlas en la mesa de alimentación. Se debe empezar por un lado e ir girando la mesa hasta completar la estructura. Las barras que vamos a colocar son **6 barras de 1220 mm y 3 de 2300 mm**. En la **Figura IV.1.13** se observa el esquema y la secuencia elegida para las barras metálicas.

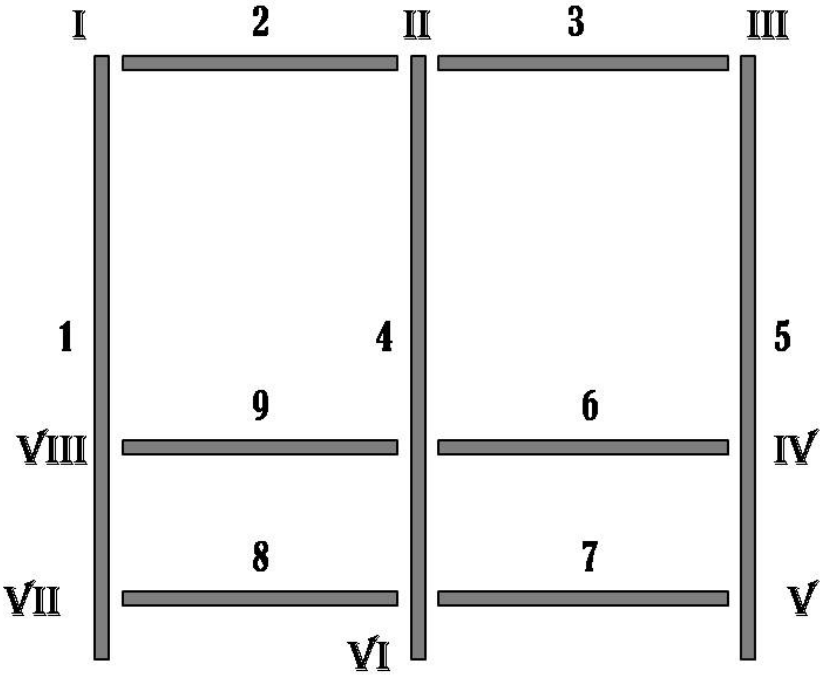


Figura IV.1.13- Esquema de la estructura metálica

Los números romanos indican las juntas entre barras que se simularán.

A continuación hay que colocar los tubos. En concreto, se van a colocar: un **tubo grande de 25x910 mm** que irá en la barra superior y funcionaría como alimentador del resto; y **tres tubos más, de 16x1620, 16x2100 y 16x582 mm** que se situarán de acuerdo al esquema de la **Figura IV.1.14:**

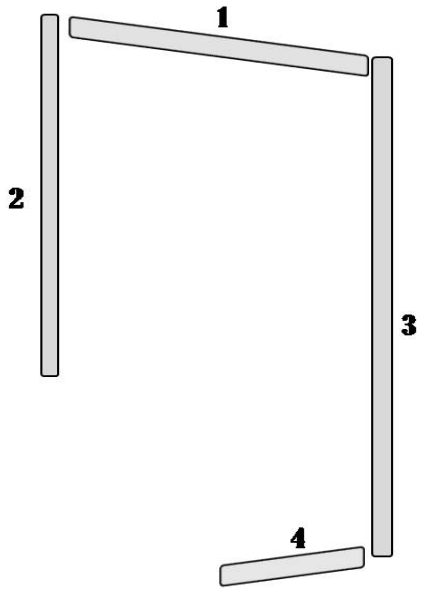


Figura IV.1.14- Esquema de las tuberías

Por tanto, la secuencia que debe seguir el robot a la hora de coger y ensamblar las piezas es:

1- Barra de 2300 mm. Se denominará Barra_2300_1

2- Barra de 1220 mm. Se denominará Barra_1220_2

3- Barra de 1220 mm. Se denominará Barra_1220_3

4- Barra de 2300 mm. Se denominará Barra_2300_4

5- Barra de 2300 mm. Se denominará Barra_2300_5

6- Barra de 1220 mm. Se denominará Barra_1220_6

7- Barra de 1220 mm. Se denominará Barra_1220_7

8- Barra de 1220 mm. Se denominará Barra_1220_8

9- Barra de 1220 mm. Se denominará Barra_1220_9

10- Tubo de 25x910 mm. Se denominará Tubo_25x910_1

11- Tubo de 16x1620 mm. Se denominará Tubo_16x1620_2

12- Tubo de 16x2100 mm. Se denominará Tubo_16x2100_3

13- Tubo de 16x583 mm. Se denominará Tubo_16x583_4

Así pues, en este orden se deben importar y posicionar en la mesa de alimentación. Todos los archivos han sido creados en SolidWorks y convertidos a formato STEP.

Se importa la primera barra de 2300 mm. Se hace invisible el robot y al examinarla, se observa que la barra se importa centrada en el origen del SDC del mundo y situada a lo largo del eje Z positivo. También que la ranura de la barra se encuentra hacia el lado Y negativo. Se observa en la siguiente figura:

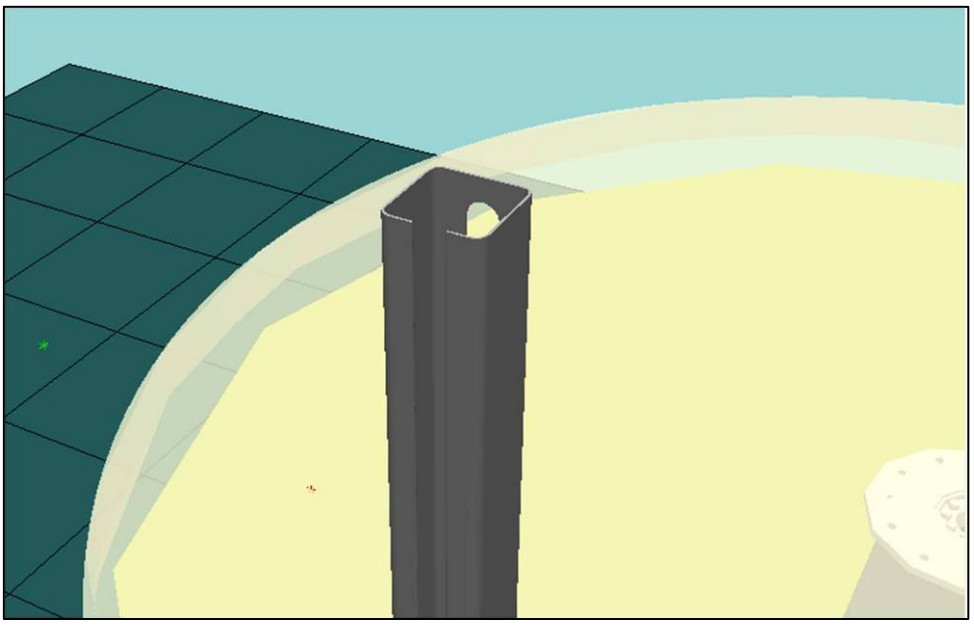


Figura IV.1.15- Orientación de la barra importada

Esta ranura se coloca orientada hacia abajo (para que quede hacia la parte posterior) al ensamblar el Service Core, por lo que es útil tener en cuenta su orientación al importar la barra. Para ayudarnos a colocar las barras se genera una línea en la mitad de la mesa de alimentación (X=0). Para ello se selecciona el modo de ajuste "punto medio" y se pincha cerca de la mitad del borde de la mesa cercano al robot. Mirando en la barra de estado se trata del punto (0, -500, 500). Se crea una línea desde ahí hasta el final de la mesa (0, -2000, 500) con la función **Create→2D Components→Line.**

j) Usar cotas en SolidWorks

Para situarla, primero se le da la orientación deseada, para lo que hay que rotarla primero -90º en Z y luego 90º en X respecto al origen, siendo todas estas operaciones en su **SDC local** que para la segunda rotación nótese que está rotado -90º en Z. Se sitúa el punto medio de la cara agujereada de la barra, que se debe ubicar a una cierta altura de la mesa. Para saber esa altura, se abre el modelo de la barra con SolidWorks y con la herramienta de Cota (Herramientas→Cota inteligente) vemos que la anchura de la barra es de **20 mm**. Se ilustra en la siguiente **Figura IV.1.16**:

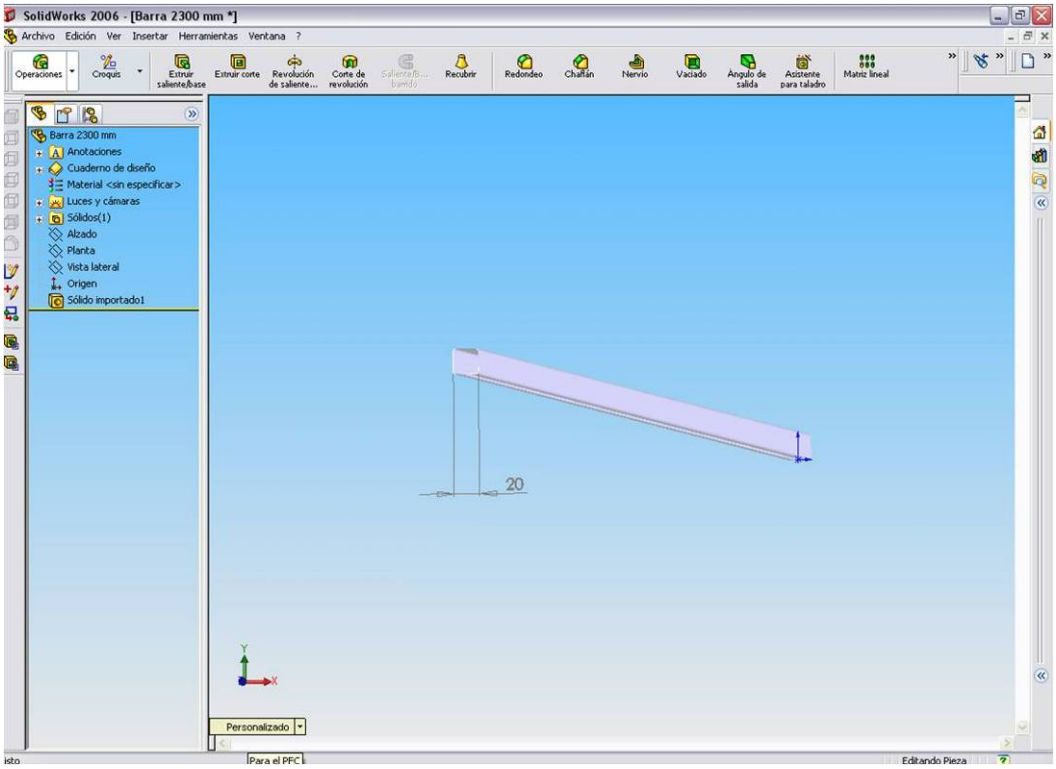


Figura IV.1.16– Cota de la anchura de la barra en SolidWorks

Por tanto, ahora se selecciona el punto central de la superficie agujereada de la barra (sin ranura) y se sitúa en (0, -600, 520) para lo que nos podemos ayudar de la línea (con el nivel de selección "curva" se puede pinchar en un punto de la línea, y así orientarse mejor) a la hora de ejecutar el comando Place. El resultado se puede ver en la **Figura IV.1.17.**

k) Copiar/Aplicar orientación

Para posicionar el resto de barras, el proceso será más sencillo. Hay que importarlas una a una a nuestra estación. Para tener más rango de actuación, lo primero que se puede hacer es mover la mesa de alimentación 100 mm hacia el robot (p. ej. con place, y también hay mover la barra 1 con lo que queda en (0, -500, 520) su punto central superior), de manera que se aprovecha todo lo posible su superficie. Tras importar cada barra, se hace uso del comando **Copy Orientation,** que permite aplicar la misma orientación de la barra 1 a la barra que se haya importado. Para ello se pincha en la ventana gráfica o en el navegador en la barra 1 con botón derecho, se selecciona Copy Orientation, a continuación en la barra importada, y se elige **Apply Orientation**. A continuación sólo resta situarla en la mesa de manera análoga a la barra 1 (50 mm más alejada del robot cada barra) y definirle el nombre deseado. También es posible realizar una **multiselección** (seleccionar varios objetos a la vez pulsando Ctrl) para facilitar el trabajo.

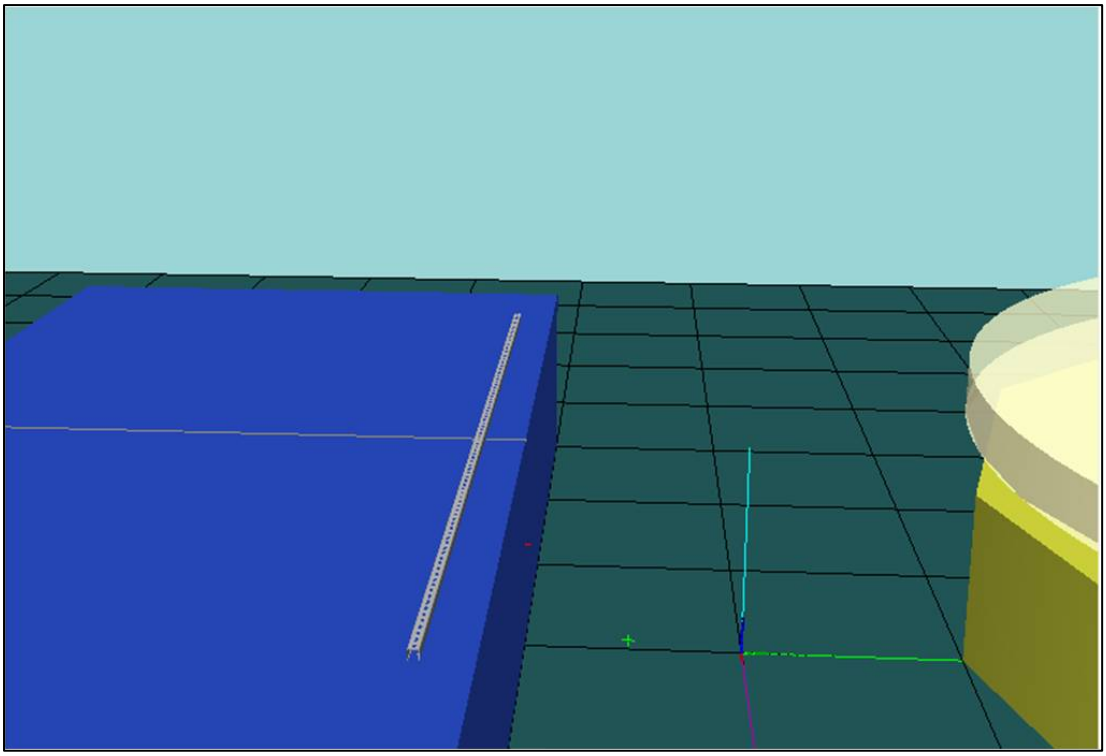


Figura IV.1.17- Barra 1 posicionada en la mesa

Para posicionar los tubos se sigue un procedimiento semejante. Se importa el primero de ellos, el de 25 mm de diámetro, y se observa que se importa con una orientación similar a con la que lo hacían las barras. Con el **modo de ajuste mitad ("mid"),** se observa que al seleccionarla se elige un punto a mitad de altura del tubo y en la posición en X de mayor magnitud, en concreto el **(12.5, 0, 455)**. Para facilitar su recolocación, se tumba a semejanza de lo que se hizo con las barras, es decir, rotando en torno a **(0, 0, 0) -90º en Y.** A continuación, queda situar ese mismo punto (superior centrado, ahora (-455, 0, 12.5) en la posición correspondiente en la mesa, 50 mm más alejada de la barra 9 y a una altura de **Z = 500 + 25 = 525 mm,** es decir **(0, -950, 525).** Se le puede dar además el nombre y un color blanco para que se distinga del resto de elementos. Se procede con las 3 tuberías restantes de manera semejante: importar, copiar orientación del tubo 1 y posicionar, teniendo en cuenta que estás irán a una altura de **Z = 500 + 16 = 516 mm.**

Al final de este proceso, la mesa debe quedar semejante a la de la **Figura IV.1.18**:

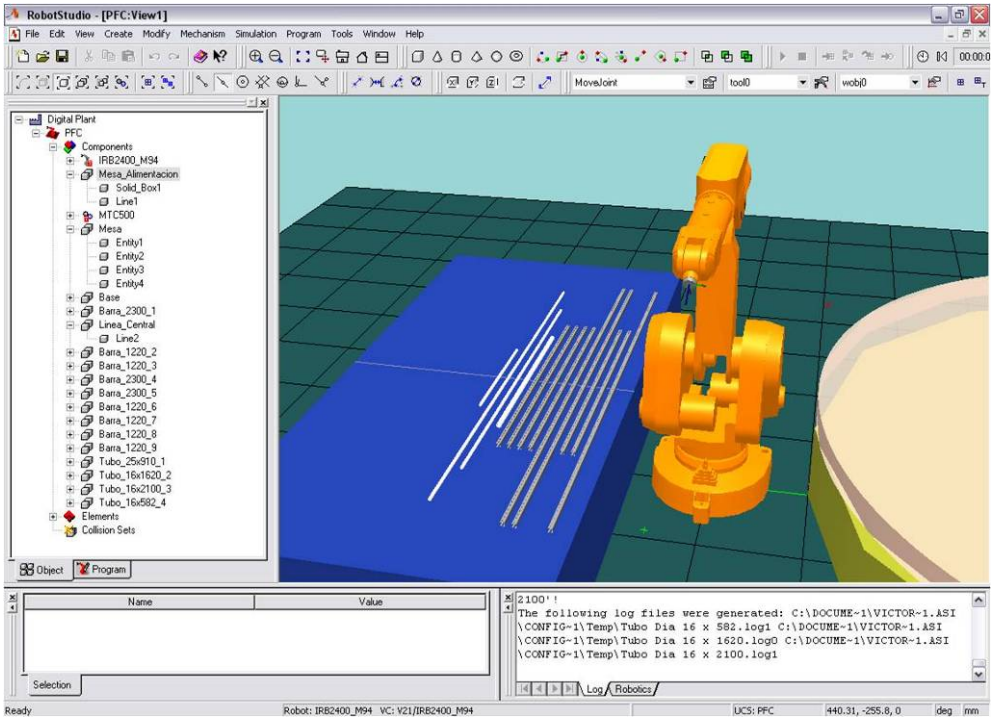


Figura IV.1.18– Barras y tubos posicionados en la mesa

l) Crear una herramienta

El siguiente paso a dar es la creación de la herramienta que usará el robot para ensamblar el Service Core. Se usa una herramienta común que es válida para operaciones de movimiento de piezas (agarre, suelta, ensamblado, etc). Hay que reseñar que el proceso de ensamblado que se va a simular no va a tratar de mostrar un procedimiento susceptible de realizarse en una célula real. El proceso consistirá en posicionar adecuadamente las piezas en su sitio, para mostrar la metodología a seguir en posteriores trabajos, y se añaden simulaciones de posibles tareas de soldadura. Para la simulación, nos servimos de una herramienta común y no estudiada específicamente.

Para crear la herramienta se ejecuta el comando **Tool** en el menú **Create.** En el asistente que se lanza, se presentan tres pasos o ventanas:

1. Elección del nombre, p. ej. MB_Tool, y del tipo de herramienta. Se elige **pieza existente** (“existing part”) y se selecciona la geometría de la herramienta que previamente hemos debido importar a la estación.
2. Elección de **nombre y posición del TCP**. Se le da un nombre al TCP. Para buscar una buena posición, con el modo de ajuste centro, se crea una línea entre las caras interiores de los dedos de la garra, de la manera que se muestra en la **Figura IV.1.19**. Luego con el modo de ajuste medio o “mid” se selecciona el punto

central de esa línea y se designa como TCP (en "Type in", con la misma orientación del SDC del mundo), que debe ser (-0.82, 55.1, 261.04) aproximadamente. A continuación se puede borrar la línea.

3. **Masa y centro de gravedad**. Estos parámetros no influyen decisivamente en nuestra simulación. Se eligen unos valores ficticios de 0.5 Kg y centro de gravedad (0, 25, 150).

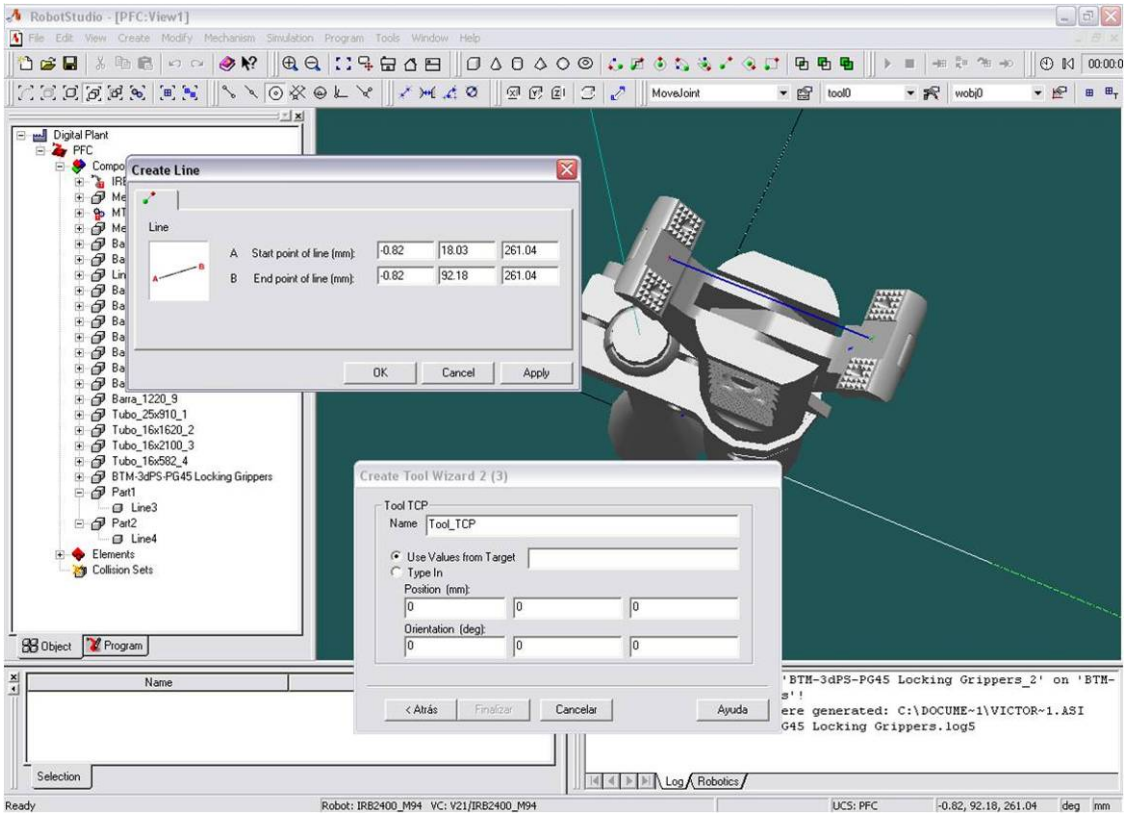


Figura IV.1.19- Crear herramienta y elegir TCP

Ya se dispone de la herramienta creada, como se comprueba en el navegador, en el que aparece en la lista de componentes con un símbolo de pistola de soldadura. Ahora hay que fijarla al robot como se ha comentado: pinchar y la arrastrar al robot en el navegador, eligiendo sí reposicionar para que se coloque en la muñeca del robot. Ya está listo el robot con su herramienta.

m) Creación de puntos en RS. Puntos de agarre.

Una vez el robot listo para crear los movimientos que desempeñará, se avanza al siguiente paso designado en la metodología de la **Figura IV.1.5**. Hay que crear los puntos de agarre de las piezas en la mesa de alimentación. Para crear un punto ("target") en RobotStudio basta con ir al menú "**Create**" y seleccionar "**Target**", o con el método rápido Máy+T. Se despliega una ventana de diálogo donde se puede elegir la posición y orientación del punto u objetivo, y también se puede seleccionar la referencia que deseemos utilizar. En nuestro caso se usa la referencia por defecto, que es la del

mundo, que además es la que está seleccionada como del usuario. La otra opción, Wobj u objeto de trabajo no aporta nada por ahora, pero se estudiará más adelante, ya que para situar puntos en la mesa circular es interesante crear el objeto de trabajo que defina esa mesa (en este caso, sin embargo, no hace falta definir un objeto de trabajo para la alimentación).

Así pues se crea el primer punto de agarre para la barra 1. Hay que recordar que el punto medio central superior de esta barra está situado en (0, -500, 520) y por tanto ahí es donde deberá desplazarse nuestro TCP. En cuanto a su orientación, hay que fijarse en que para que encaje bien la herramienta en la barra, y que la simulación sea más fiel a la realidad por tanto, el eje Y del TCP, que coincide con el de la muñeca, debe coincidir también con el del mundo. Y, además, el eje Z debe quedar en sentido contrario al del mundo. Por tanto, hay que girar 180º en Y, es decir, la orientación será (0, 180, 0). Se genera "Target1:1" en el árbol del navegador bajo la rama "Elements/Targets/wobj0/". Se cambia el nombre por **Pto_Barra_1**, y para comprobar que la herramienta se orienta correctamente al agarra la pieza, se hace click derecho en él y se selecciona "**View Tool at Target**" (ver la herramienta en el punto). Se observa en la **Figura IV.1.20**.

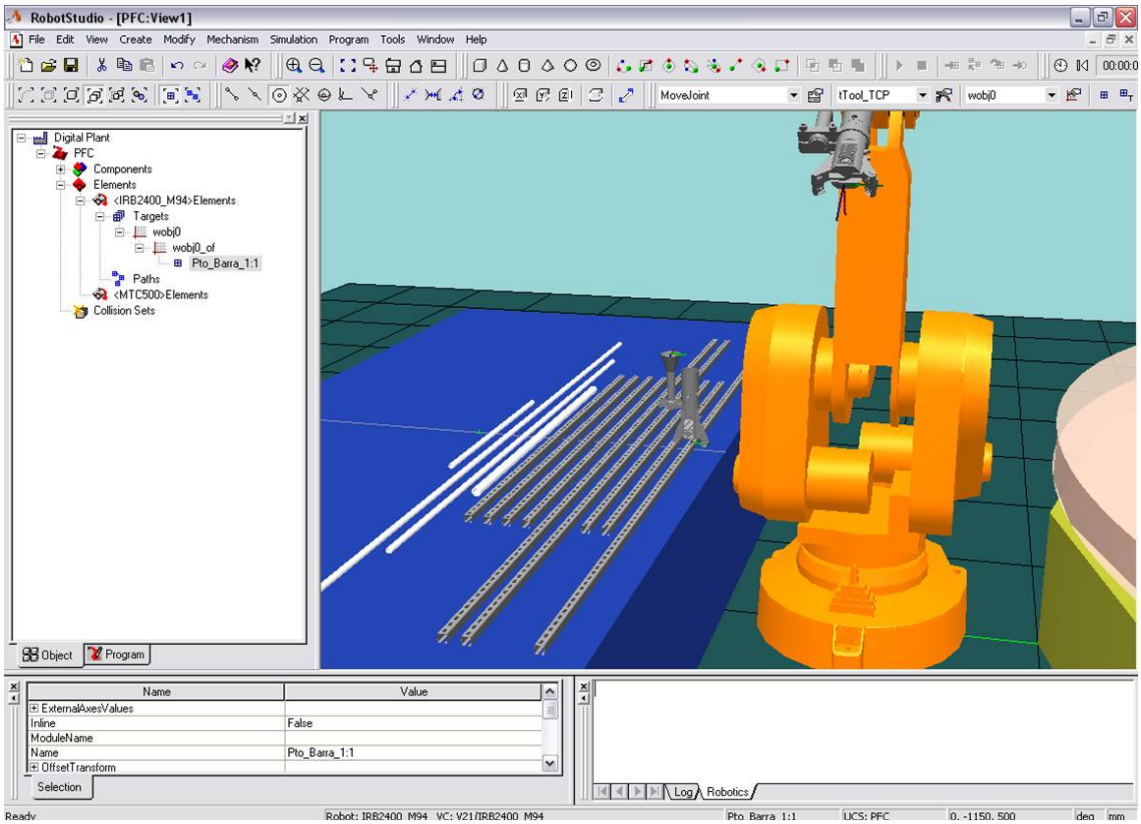


Figura IV.1.20- Vista de la herramienta en el punto Pto_Barra_1

Además, hay que crear un punto que sirva de aproximación al punto de cogida de la barra, que evitaría golpes bruscos o malas maniobras de aproximación en general en tareas reales. Se genera por ejemplo 50 mm por

encima del de agarre, es decir, en (0, -500, 570), y con la misma orientación. Se le llama **Ap_Barra_1**.

Asimismo, se crea un punto de aproximación general a la mesa de alimentación, que hará que el robot evite alturas bajas cuando manipule piezas y servirá de paso medio para las distintas trayectorias. Se genera por ejemplo en (0, -750, 1000) y con la orientación de los puntos de agarre. Se le da el nombre **Ap_Alim**. Con la misma idea, se crea un punto de paso medio entre la mesa de alimentación y la de ensamblado, en frente del robot. Se sitúa por ejemplo en (1000, 0, 1000) y con orientación del TCP (0, 180, 90) ya que se pretende que la pieza haya girado 90º positivos en Z del mundo al llegar a ese punto. Su nombre será **Paso_Medio**. Se crea también un punto que sirva de aproximación general a la mesa de ensamblado, por ejemplo en (0, 600, 1000) y con orientación (0, 180, 180) pues se quiere que la pieza haya girado 90º positivos en Z del mundo más respecto al paso medio. Se le da el nombre **Ap_Ens**.

n) Crear Trayectorias. Utilizar objetos de trabajo.

Hecho esto, ya casi están creados todos los puntos necesarios para crear la primera trayectoria ("Path"). Falta el punto en el que el robot debe dejar la barra 1, lo que se denominará como su punto de ensamblado. En la siguiente **Figura IV.1.21** se pueden observar las dimensiones características que serán útiles a la hora de realizar los cálculos de los puntos de ensamblado.

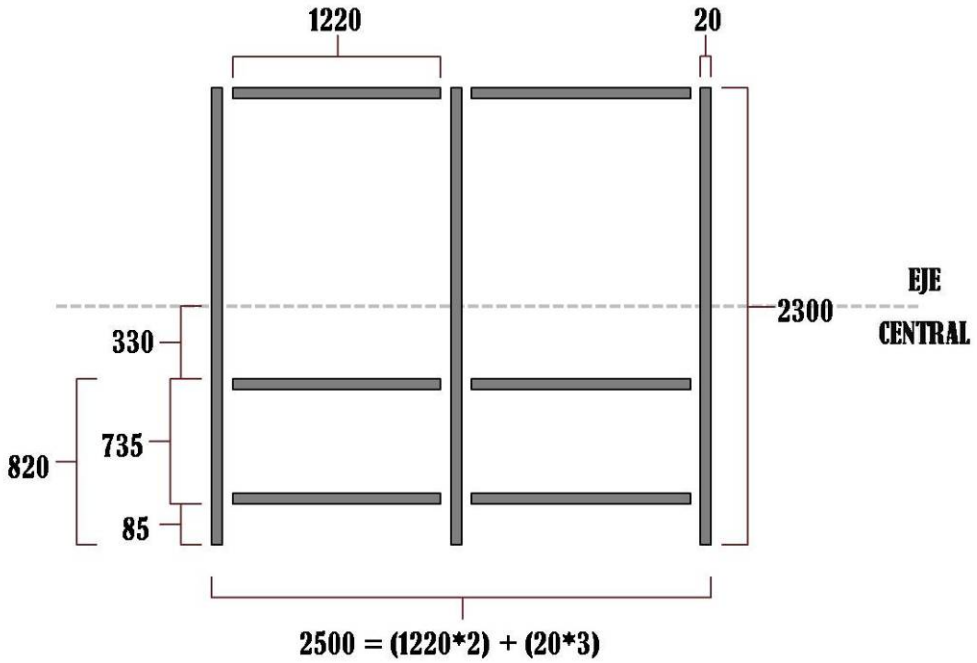


Figura IV.1.21- Dimensiones relevantes de la estructura metálica

Se va a **crear el objeto de trabajo Salida**, que permitirá referenciar los puntos a la mesa de salida y por tanto facilitará los cálculos de los puntos de ensamblado.

Para ello, en al menú "**Create**" se selecciona "**Workobject**". En la ventana que aparece, se introduce el nombre, Salida, y las coordenadas del objeto. Hay que introducir las coordenadas de su base ("uframe"), es decir, la posición del objeto en el mundo, y las de su SDC ("oframe"), es decir, el punto que se usará como origen del SDC de este objeto de trabajo, **tomando como referencia el uframe**. Por tanto, la posición del "uframe" será (0, 2250, 0) y la del "oframe" (0, 0, 625) para situar el SDC en el centro superior de la mesa de salida. Se dejan las orientaciones iguales que la del mundo. A partir de ahora, se puede usar este objeto como referencia a la hora de crear y situar puntos. En la barra de herramienta "Elements" existen varios menús desplegables en los que es posible elegir el objeto de trabajo activo, el TCP activo y el tipo de plantilla de instrucción activa (en la misma barra se dispone de los botones para crear puntos). Esto significa que si está el objeto de trabajo "Salida" activo, se puede referenciar la posición de un punto a este objeto, y además los puntos creados estarán incluidos bajo este objeto en el árbol del navegador. También se puede elegir desde el navegador el TCP y el Wobj activos, haciendo click derecho en los respectivos elementos.

Una vez creado el objeto "Salida", se crea el punto de ensamblado referenciado al mismo. Hay que asegurarse de que es el objeto activo y a continuación se crea el nuevo punto. En la ventana se elige como referencia "Wobj". Puesto que la estructura tiene un ancho de 2500 mm, y la barra tiene una anchura de 20 mm, el punto de ensamblado se debe encontrar a **2500/2 – 20/2= 1250 – 10 = 1240 mm** del centro de la mesa. Por tanto el punto de ensamblado será (0, -1240, 645) referido al "uframe" del objeto "Salida". La orientación es la misma que la del punto Ap_Ens. Si se quisiera ahora modificar su posición, se podría usar como SDC el "oframe" de "Salida", eligiendo en las ventanas de diálogo el sistema **padre** (Parent). Se denomina al punto de ensamblado **Ens_Barra_1**. Se crea también el punto de aproximación asociado a 50 mm de altura, con nombre **Ap_EnBarra_1**, y la misma orientación.

Ya se puede pues crear la primera trayectoria. Para ello se utiliza "**Create**" y a continuación "**Path**", o bien se usa el método rápido Máy+S, o bien en el navegador, se hace click derecho en "Paths" y se selecciona "Create Path". Se nombra a esta trayectoria **Path_Barra_1**. Las trayectorias, como cualquier otro objeto, se pueden hacer visibles/invisibles, cambiar su color, etc. Para añadir puntos a la trayectoria simplemente se arrastran en el navegador los puntos hasta la trayectoria deseada, con el orden adecuado. Esta trayectoria se compone de la siguiente secuencia:

Paso_Medio→Ap_Alim→Ap_Barra_1→Pto_Barra_1→Ap_Barra_1→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_1→Ens_Barra_1→Ap_EnBarra_1→Ap_Ens

Se realizan todos los arrastres y se comprueba en la pantalla gráfica como se va creando la trayectoria, que debe quedar similar a la de la **Figura IV.1.22**.

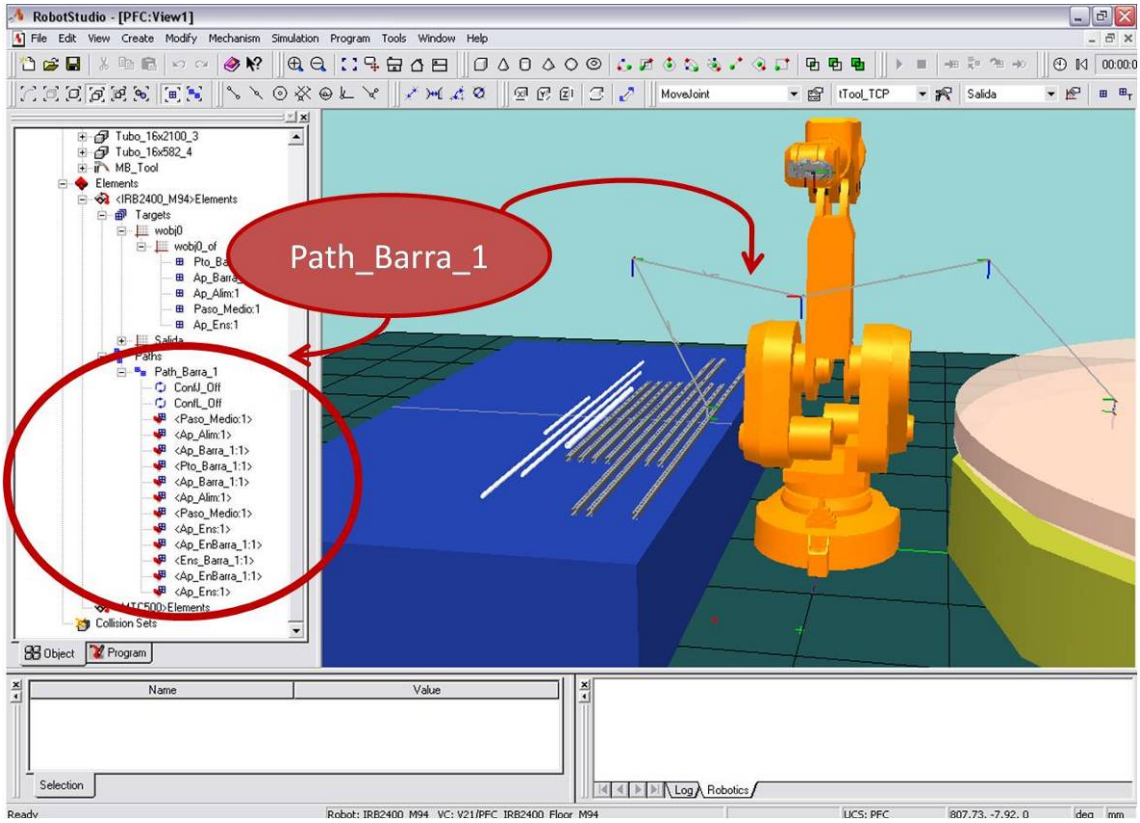


Figura IV.1.22- Path_Barra_1

Crear un trayectoria lleva ímplicito crear instrucciones de movimiento MoveJ, MoveL, MoveC, etc, entre los puntos de la misma. Estas instrucciones se crean en base a la plantilla activa en la barra de elementos o en Modify→Instruction Template (de manera similar a la herramienta activa o el SDC activo), que se pueden configurar como se desee o incluso crear nuevas plantillas de procesos, en las que ademas del tipo de movimiento se puede seleccionar la velocidad, precisión, etc.

o) Arrancar el VC. Mover el robot y recorrer trayectorias.

Una vez que se ha creado una trayectoria, incluso un simple punto, es conveniente recorrerla para comprobar que el robot realiza el movimiento que se pretendía y que no se sale de rango y/o intenta realizar movimientos extremos. Para ello se necesita asignarle un controlador virtual (VC) al robot que controle sus movimientos. Este controlador virtual es una copia exacta del controlador real del robot. Para seleccionarlo y arrancarlo, se hace click derecho en el robot en el navegador o el área grafica y se selecciona "**Setup Controller**", o bien, con el robot seleccionado se abre el menú "Mechanism" y se selecciona "**Controller→Setup**". Se despliega una ventana como la de la **Figura IV.1.23.**

Figura IV.1.23- Ventana de configuración del controlador

En esta ventana se puede arrancar el controlador que viene predeterminado o bien seleccionar otro. El controlador posee un software que se modifica automáticamente con cada operación que se realiza con el mecanismo, como incluir puntos o trayectorias en el robot, crear módulos de programas, mover el robot, etc, y por tanto si se efectúa cualquier operación que derive en un malfuncionamiento del controlador, es imposible (o muy difícil) volver a atrás y deshacerla. Por lo tanto es **muy importante** realizar una **copia del controlador** que será la que se use en cada proyecto. Se guarda la copia original como una copia de seguridad libre de cualquier cambio potencial que suponga un malfuncionamiento crónico del controlador. Para este propósito hay que pinchar en cambiar controlador ("**Change Controller**") desplegándose una ventana como la de la **Figura IV.1.24.** En esta ventana se puede navegar por los distintos tipos de VCs de que dispone el programa, copiarlos, borrarlos, crear nuevos, etc… Lo primero que se debe hacer es asegurarse de que la versión V21 es la seleccionada en "Current Directory" (Directorio Actual). Seguidamente se realiza una copia del robot IRB2400_Floor_M94 ("Create Copy") y se renombra ("Rename") por ejemplo PFC_IRB2400_Floor_M94. Ya solo queda seleccionarlo como VC elegido ("Set Current Robot") y cerrar. En la ventana anterior se selecciona arrancar ("Start Controller") y después de unos momentos se observa cómo en la barra de estado aparece resaltado en verde el nombre del robot y su VC, y el modo de operación, en este caso "Auto".

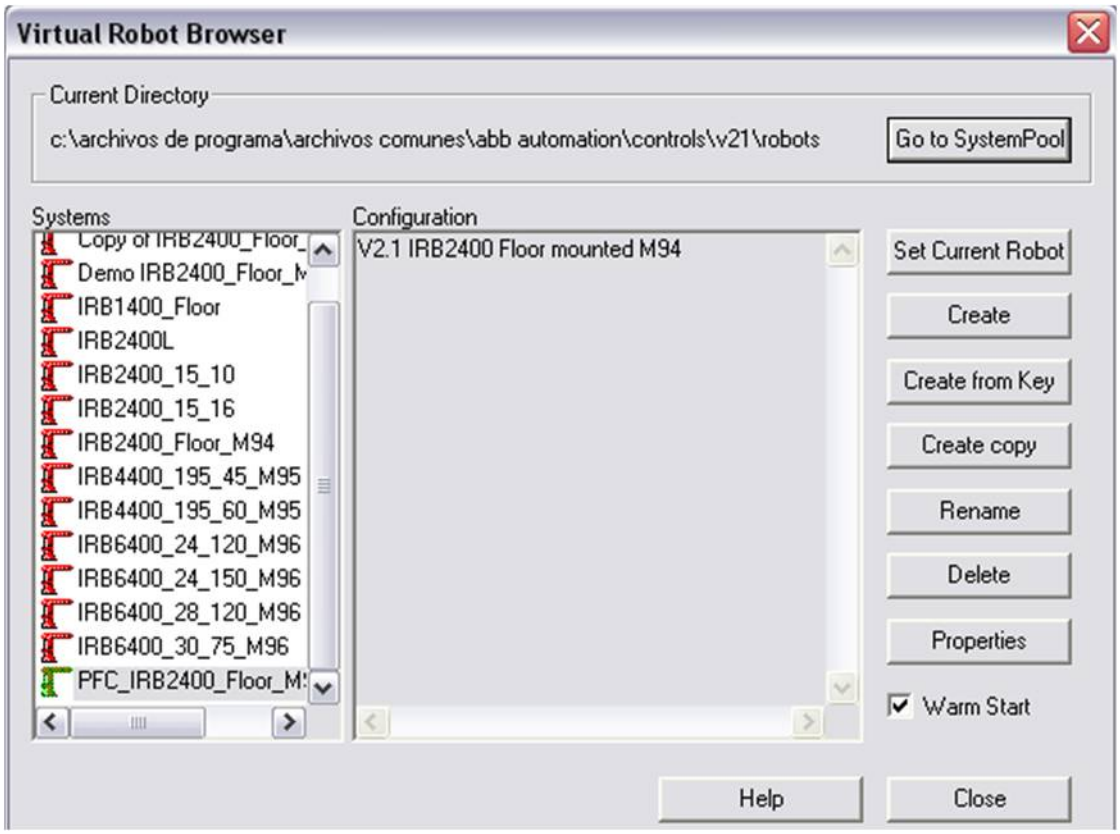


Figura IV.1.24- Navegador de Robots Virtuales

Ya se dispone del VC arrancado y se pueden recorrer trayectorias y mover el robot al punto que se desee. También es posible ver al robot en el punto objetivo, con una opción similar a la de ver la herramienta que en este caso se llama "**View Robot at Target**". Para recorrer una trayectoria basta con seleccionarla en el navegador con botón derecho y elegir la opción "**Move Along Path**", o con el menú "**Mechanism→Move→Move Along Path**" (Ctrl+A). De manera análoga se puede mover el robot a un punto. Se recorre la trayectoria Path_Barra_1 y se observa que el robot intentar realizar el movimiento de Paso_Medio a Ap_Alim de una manera incompatible para poder llegar al objetivo. Se soluciona moviendo más cerca el punto Ap_Alim, a (200, -750, 1200) de manera que llegue por un camino más corto y suave.

Al realizar un movimiento que provoca salida de rango, como el movimiento al Ap_Alim original que se ha intentado, el robot no deja ponerse en marcha de nuevo. Es necesario recurrir al "**Teach Pendant**" virtual para comprobar el error (como indica la ventana de salidas). Se ejecuta en "Mechanism→Teach Pendant" (F5). Hay que pasar a modo manual <250mm/s, pulsar OK (último botón de abajo del mando) para aceptar el error de fuera de rango, que aparece resaltado, se vuelve a pasar a Auto, de nuevo pulsar OK para verificar y poner los motores en ON. La **Figura IV.1.25** muestra el "Teach Pendant" virtual.

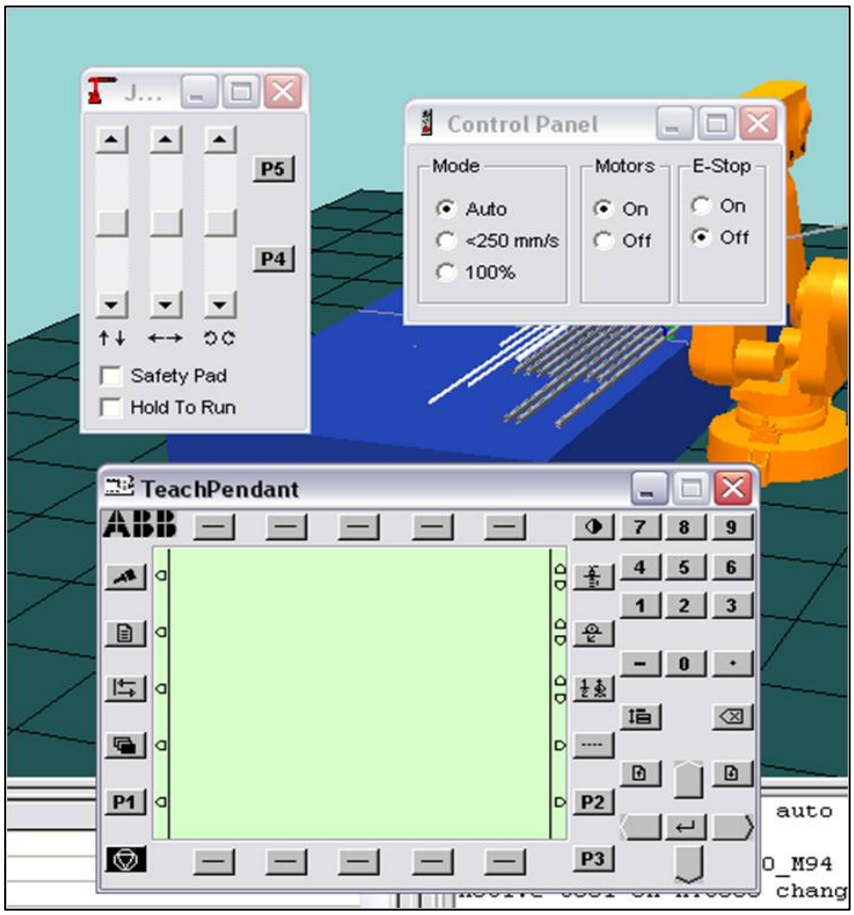


Figura IV.1.25- Captura del "Teach Pendant" Virtual

Ya se puede volver a recorrer el Path_Barra_1 de nuevo y comprobar que se realiza el movimiento correctamente. El siguiente paso será simular los procesos de sujeción y suelta/ensamblado de piezas.

p) Asignar señales E/S. Tabla de eventos.

Para hacer uso de señales de entrada/salida en RobotStudio es necesario manejar aplicaciones que el programa ofrece aparte de la creación del diseño de la estación, como son el ProgramMaker y la tabla de eventos.

La tabla de eventos es una tabla que permite gestionar todo tipo de eventos, como ligar señales a acciones y gestionar colisiones. Es una herramienta muy útil pero presenta algunos inconvenientes. El principal es que liga eventos a objetos uno a uno, con lo cual se necesitará una entrada de la tabla por cada acción que se quiera realizar sobre cada objeto, entendiendo por ejemplo fijar/soltar como una acción.

Lo primero hay que hacer para poder usar señales es configurarlas en el VC. Para ello, se abre el Teach Pendant virtual, se pulsa el botón de "Other Windows" (el cuarto de la columna izquierda) y se selecciona "**System Parameters**" con la tecla intro del Teach Pendant. Indica que hay que pasar a modo manual. Realizado esto, a continuación informa de que no existe programa para mostrar cargado. Se selecciona "System Parameters" otra vez, se pulsa el botón "File" y seleccionamos "**Add new parameters**". Se selecciona "**IOConfig**" y a continuación "**2DIG**". Los parámetros se han cargado. Hay que reiniciar con "File→Restart" y pulsando OK. El VC se reinicia con los nuevos parámetros configurados, con lo que ya es posible hacer uso de señales digitales. Para comprobarlo, se puede ejecutar el "**IOSimulator**" en el menú "Mechanism" (Ctrl+I) o bien haciendo click derecho en el robot, y se observa cómo las señales se pueden activar y desactivar en la tabla que se muestra.

Lo siguiente se debe hacer es asignarle los eventos y la señal a la barra 1 en la tabla de eventos. Se usará la señal digital do1 para esta barra. La lista de señales se puede consultar en los anexos (capítulo VII.7).

Se abre la tabla (**Simulation→Event Table**) y se añaden dos nuevas entradas ("**Add entry**"). Los campos se rellenan de la siguiente forma:

- Activation: ON. Evento activo en simulaciones y activaciones "a mano" de la señal.
- Period/Active: 1. Estas opciones sólo se usan al gestionar ciclos, que no es el caso.
- Event Type: IO. Tipo de evento E/S. Otros tipos son las colisiones, que se comentarán más adelante.
- Event Object: IRB2400_M94. El objeto que recibe o emite la señal.
- Event Params: do1=1. La señal que se gestiona. En este caso, el evento se produce cuando do1 pasa a 1 (flanco de subida).
- Action Type: Attach Stay. Fijar sin reposicionar. "Update" significa reposicionando.
- Action Object: MB_Tool. El objeto que realiza la acción.
- Action Params: Barra_2300_1. El objeto que recibe la acción.

La acción para do1=0 debe ser "Detach", es decir, soltar la pieza. De esta manera, la tabla debe quedar como ilustra la **Figura IV.1.26.**

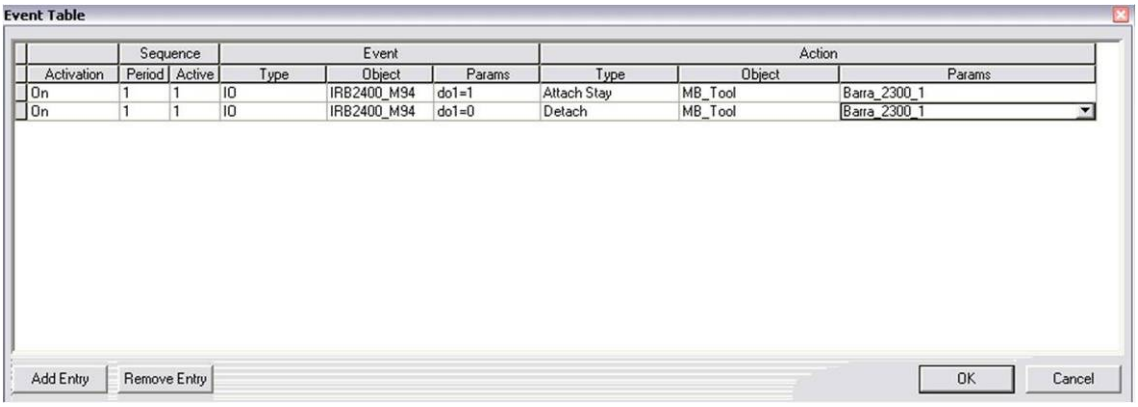


Figura IV.1.26- Tabla de eventos para la barra 1

q) Sincronizar: Transferir elementos al VC

Para poder simular los procesos, hay que cargarlos en el VC. La sincronización o transferencia de elementos de la estación al controlador se realiza **a nivel de trayectorias**. Se hace click derecho en Path_Barra_1 y se selecciona "**Sync to Virtual Controller**". Como es la primera vez que se sincroniza, es necesario crear un nuevo módulo, con nombre Module1 de manera predeterminada. Se pulsa OK para sincronizar el camino. Esto quiere decir que se ha incluido en un módulo de programa del VC, por lo que se puede consultar tanto a través del Teach Pendant virtual, como de la aplicación ProgramMaker, que se comenta un poco más adelante.

El efecto de la sincronización se puede percibir también en la pestaña "Programs" del navegador. Hay que abrirla y desplegar el árbol. También es necesario refrescarlo cada vez que se sincronice un nuevo camino, pinchando con botón derecho en el nombre del robot y seleccionando "Refresh". Se observa que aparece la trayectoria Path_Barra_1 bajo el Module1. Si se hace click derecho en ella y se selecciona "Execute Procedure" (ejecutar el proceso o rutina) el robot recorre el camino que se había creado, pero sin tener en cuenta señales puesto que aún no están incluidas en el programa.

Ésta es una de las maneras de simular la estación, de manera secuencial proceso a proceso. La manera más usual es crear un proceso "main" (botón derecho en el módulo y "**Insert main procedure**"), y arrastrar a él los procesos que se quieran incluir en el programa. De esta manera se incluyen las rutinas al proceso main (proceso principal de un módulo) de nuestro módulo. A continuación se simula con los comandos de RobotStudio (**Simulation→Play**, Ctrl+5). Esta acción ejecuta el programa que se encuentre en el VC. Si todas las rutinas están bajo el proceso main, como es el caso, el resultado es el mismo que si se ejecutara el proceso main de la manera que hizo antes con el Path_Barra_1. Se realiza una simulación y se comprueba que el resultado es el mismo que antes.

Una idea importante a recordar al simular es que **es conveniente grabar justo antes de cada simulación**, ya que la estación puede sufrir variaciones debido a eventos, señales, etc, durante la misma, que desubiquen los

elementos contenidos en ella, con lo que habría que volver a situar todo correctamente. Eso se evita grabando antes y cargando la versión previa de la estación después de cada simulación.

Creando un proceso main se crea una estructura de programa, que se puede consultar o modificar en el TeachPendant o en el ProgramMaker.

r) Utilización de E/S. ProgramMaker

El ProgramMaker es una aplicación aneja a RobotStudio que permite crear de forma simple, rápida y eficaz todo tipo de programas de RAPID, facilitando el trabajo con la inclusión de plantillas, accesos directos, etc. Además **se sincroniza directamente con el VC** que tengamos activo en nuestra estación de RobotStudio, lo que nos evita procesos de importación/exportación. Crear y modificar programas en el ProgramMaker es más simple y preciso que hacerlo directamente con el Teach Pendant o a través de editores estándar.

Se ejecuta el ProgramMaker (**Program→Edit Program**, Ctrl + E), y se puede observar como su interfaz es bastante sencilla y amigable. Se dispone de barras de menús y accesos rápidos en la parte superior; a la izquierda un árbol similar al de la pestaña de "program" del navegador de RobotStudio; en el centro la ventana de texto, dónde visualizar y editar programas; y a la derecha una serie de plantillas y utilidades para facilitar la creación y edición de programas. Se observa en la siguiente **Figura IV.1.27.**

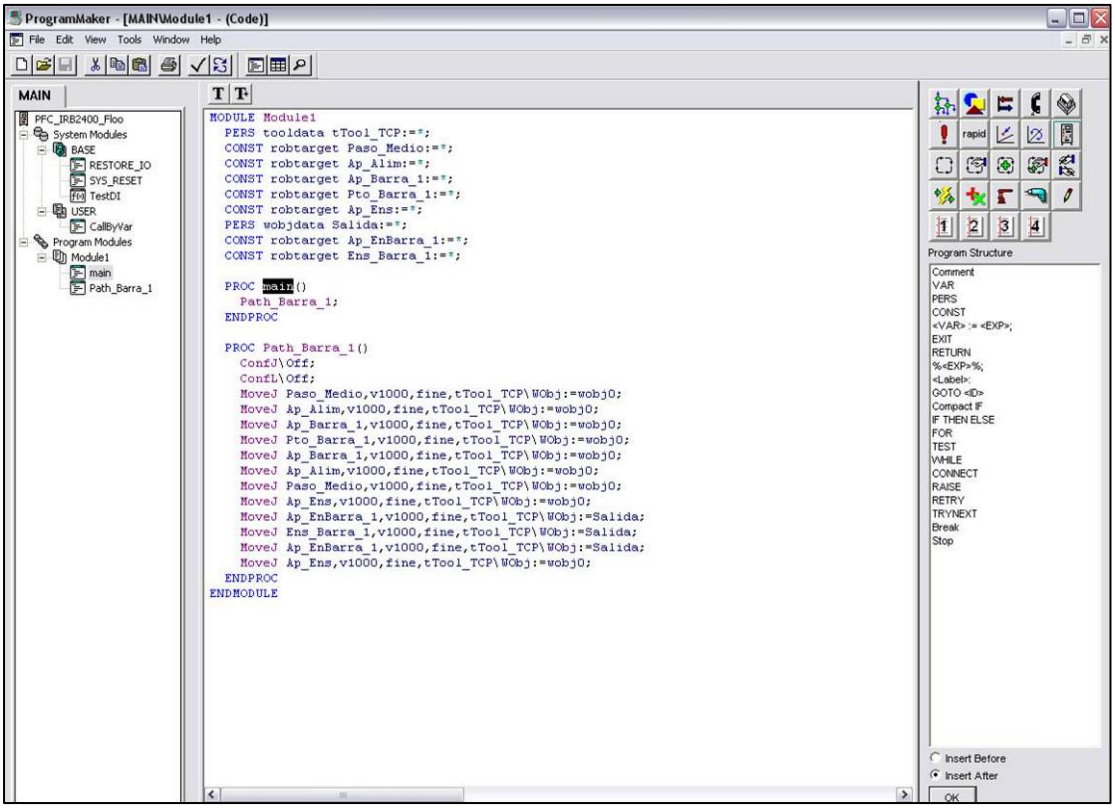


Figura IV.1.27- ProgramMaker con Path_Barra_1

Ahora hay que incluir las instrucciones relativas a E/S. Existen dos maneras:

- a mano, directamente escribiendo cualquier carácter en el texto del programa y editándolo.
- con las plantillas de la derecha, eligiendo el grupo en los botones de arriba a la derecha y la instrucción en la lista de abajo, con lo que simplemente hay que rellenar los parámetros a través de una ventana emergente.

Las instrucciones que se tienen que incluir son Set do1, para que la herramienta "coja" la barra (para que se fijen, como se definió en la tabla de eventos), que debe situarse por tanto después de llegar a Pto_Barra_1 y Reset do1, para "soltarla" cuando llegue a Ens_Barra_1. Es conveniente añadir justo antes y después de cada operación de activación/desactivación de señales, instrucciones de espera **WaitTime** para dar tiempo al robot a realizar las operaciones adecuadamente, evitando operaciones bruscas. Por ejemplo se pueden incluir esperas de 2 segundos. Al terminar se salva (File→Save Module o Save Program). Por tanto, el proceso final para la barra 1 quedaría como sigue:

```
MODULE Module1
  !Declaraciones
  PERS tooldata tTool_TCP:=[TRUE,[[-0.819937,55.1,261.04],
[1,0,0,0]],[0.5,[0,25,150],[1,0,0,0],0,0,0]];
  CONST robtarget Paso_Medio:=[[1000,0,1200],[0,-0.707107,0.707107,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Alim:=[[200,-750,1200],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_1:=[[0,-500,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_1:=[[0,-500,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Ens:=[[0,600,1200],
[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  PERS wobjdata
Salida:=[FALSE,TRUE,"",[[0,2250,0],[1,0,0,0]],[[0,0,625],[1,0,0,0]]];
  CONST robtarget Ap_EnBarra_1:=[[0,-1150,50],[0,1,0,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_1:=[[0,-1150,0],[0,1,0,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];

  PROC main()
    Path_Barra_1;
  ENDPROC

  PROC Path_Barra_1()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_1,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_1,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
    Set do1;
```

```
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Barra_1,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra_1,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Barra_1,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do1;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnBarra_1,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC
ENDMODULE
```

El texto final completo del programa está incluido en los anexos, capítulo VII.4.

Ahora se puede simular el proceso con las señales, de cualquiera de las maneras comentadas antes (Play o desde el árbol de programa) y ver cómo la barra se traslada de una a otra mesa. Hay que tener siempre presente que es importante **NO grabar** la estación después de simular, pues habría que recolocar la barra. Al terminar la simulación se vuelve a cargar la estación original sin salvar el progreso.

Se observa que el proceso se simula correctamente según lo esperado. Este mismo procedimiento se lleva a cabo para calcular el resto del proceso, como aplicación de la metodología de la **Figura IV.1.5**, realizándola por pasos o grupos pequeños de geometrías, es decir:

- Cálculo geométrico para generar los puntos de ensamblado.
- Generación de la trayectoria asociada.
- Sincronización al VC. Inclusión de señales e instrucciones RAPID.
- Simulación del proceso desarrollado hasta el momento para verificar.

Realizando simulaciones por pasos nos aseguramos de ir produciendo una simulación adecuada, y hacemos una mejor gestión de los errores, mucho más fáciles de afrontar si se trata de pequeños subprocesos que si se trata de un proceso completo mucho más amplio.

s) Utilización de macros VBA para mover mecanismos

Se sigue por tanto la misma metodología para obtener la simulación del ensamblado de todas las piezas. Otro de los complementos que se pueden usar a la hora de seguir la metodología de programación y simulación de estaciones es la posibilidad de usar programas VBA. Para poder realizar correctamente el proceso necesitamos implementar una manera de mover nuestra mesa de salida (de mover el motor al que está fijada). Típicamente la solución sería gestionar un eje externo con el VC del robot, pero la versión

de nuestro VC no nos permite esa posibilidad. Es por eso que se recurre a VBA.

Así pues, para mover la mesa de manera alternativa, se ha recurrido a usar un programa o macro de visual basic. RobotStudio incorpora la posibilidad de añadir este tipo de programas a las estaciones, y posee una amplia ayuda referente a los objetos que visual basic puede manejar en RobotStudio.

Se ejecuta el editor de visual basic en **Tools→Macro→Visual Basic Editor (Alt + F11).** A continuación se crea un nuevo módulo en el árbol de la izquierda, haciendo click derecho en Project (nombre de la estación)→insertar→módulo. A continuación se escribe el programa. Para ello hay que tener en cuenta:

- "Sub" es la instrucción de comienzo de una subrutina o programa.
- "Dim" es la instrucción requerida para declarar variables.
- Se pueden declarar variables fuera de las subrutinas (globales).
- "Set" es la instrucción que se usa para dar valores a variables.
- Existen varios tipos de variables, como mecanismos, números enteros, y números decimales.
- Cada objeto de VBA posee objetos y/o métodos asociados a los que se accede de forma jerárquica por medio de puntos. P. ej. mecanismo.ejes.nº de eje.valor, nos devuelve el valor de cierto eje de un mecanismo.
- El objeto "ActiveStation" se refiere a la estación abierta actual. El método "Refresh" asociado actualiza la estación.

Con todo esto en mente, el programa mostrado en la **Figura IV.1.28** sirve para mover la mesa 90º en sentido Z positivo:

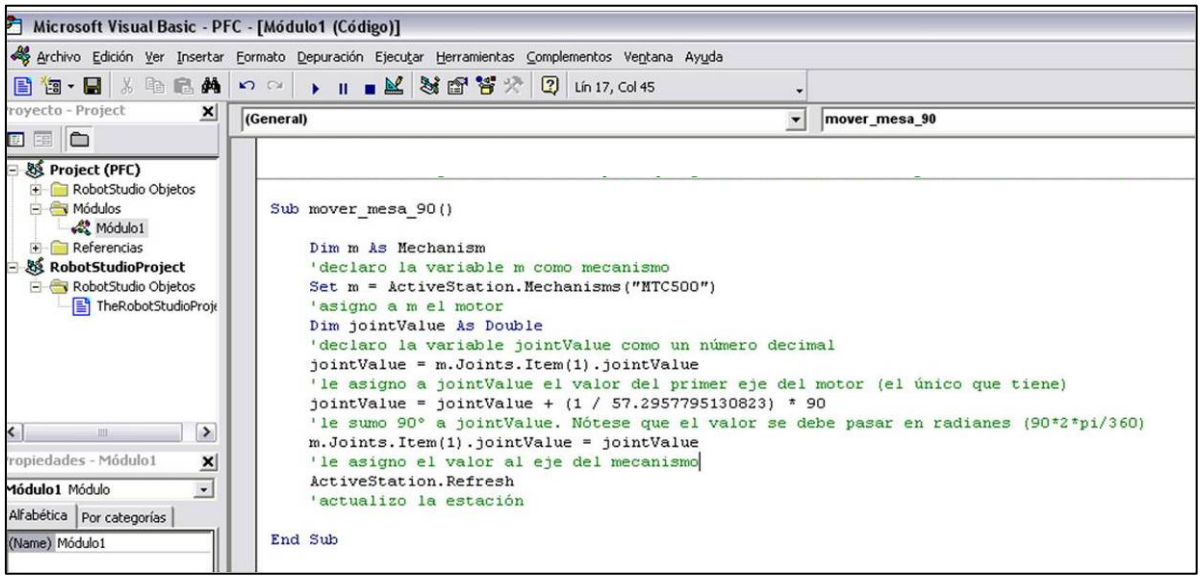


Figura IV.1.28- Macro para mover la mesa 90º

Se salva el programa y se cierra el editor de VBA. Hay que tener cuidado al salvar programas de VBA, pues **a la vez se salva la estación de RobotStudio abierta asociada,** con lo que puede suponer de pérdida de posicionamientos, etc, si por ejemplo se edita una macro justo después de una simulación y se salva la macro.

Una vez salvada, se puede comprobar su funcionamiento en el menú **Tools→Macro→Macros (Alt + F8).** Se selecciona el proyecto "Project(nombre de la estación) y se ejecuta la macro. El motor se mueve aunque es posible que no se perciba (se puede verificar que se ha movido viendo su estado "Mechanism Status"). Ahora hay que ser capaces de ejecutar la macro de otra manera, y no solamente "a mano".

Para ello se recurre de nuevo a la tabla de eventos, asignándole a un evento de E/S la acción de ejecutar la macro. Es decir, en "Event" seleccionar I/O y p.ej. **do26=1**, ya que no necesitaremos esta señal para otro propósito, y en "Action" el tipo "Macro", el objeto "Project" y en parámetros el nombre de la macro. Se necesita también incluir en el sitio adecuado del programa RAPID la instrucción de activación de la señal do26 (Set do26), en concreto se puede situar después del proceso Path_Barra_1 en el main. A continuación habrá que hacer un Reset de la señal para que quede disponible para sucesivos movimientos. También se incluye una espera entre el Set y el Reset de 1 segundo (WaitTime 1) como seguridad, para que dé tiempo a registrarse el cambio de estado de la señal.

Una vez realizado todo esto, se simula la estación para verificar que el proceso se simula correctamente. Al realizar la simulación en este punto, se observa que la barra no gira. En realidad, lo que ha ocurrido es que la mesa sí ha girado, pero no la barra. Eso es porque para que se muevan solidariamente es necesario fijarlas, cómo se ha visto anteriormente. Ahora bien, aunque ya se ha explicado cómo fijar "a mano" dos elementos, lo que se necesita es que se realice automáticamente en la simulación. Para ello puede ser de ayuda una funcionalidad más que nos ofrece el software RobotStudio: los conjuntos de colisión, combinados con macros, la tabla de eventos y el programa de RAPID de manera similar a cómo hemos simulado el agarre/suelte de las piezas.

t) Utilizar conjuntos de colisión

En el árbol del navegador existe una rama llamada "collision sets". Desde ahí se pueden crear y gestionar conjuntos de colisión. También desde el menú **Simulation→Collision**. Los conjuntos de colisión se pueden usar en la tabla de eventos, por lo que sirven para el propósito explicado. También es posible hacer que una colisión, o la proximidad de la misma, se resalte en la estación.

La idea es hacer que cuando una colisión, por ejemplo entre la barra 1 y la mesa de salida, esté activa, a través de la tabla de eventos se active una señal que active una macro que fije los elementos. Haciéndolo de la manera propuesta, sólo se necesitan dos conjuntos: uno para la dupla Mesa-barra 1 y

otro para la barra1 con todos los demás elementos. Esto permitirá que en el programa de VBA se pueda hacer referencia a todos los elementos fácilmente a través de un índice, ya que los conjuntos de colisión son también objetos de VBA. La macro se lanzará a través de la tabla de eventos cuando cada pieza se deposite en la mesa de salida (y su respectiva señal se haga 0).

Otra posibilidad que evitaría usar macros y conjuntos se efectuaría fijando a través de la tabla de eventos el elemento que se ha depositado en la mesa al elemento anterior, usando la condición de que su señal se hace 0 cuando se deposita. Sin embargo se opta por la primera opción para profundizar más en el uso de macros y conjuntos de colisión.

Un conjunto de colisión tiene dos grupos de elementos. Al chocar un elemento de un grupo con un elemento del otro se activa el evento de colisión, el cual se puede utilizar en la tabla de eventos. Como se ha mencionado antes, también es posible configurar una vista resaltada cuando se aproxima y/o cuando se realiza la colisión.

Por tanto se crea el primer conjunto de colisión, al que llamamos **Mesa_Barra1**. En el primer grupo se introduce la mesa de salida y en el otro la barra 1. Si se ejecuta ahora el programa se observa cómo al posarse la barra ambos elementos se resaltan en rojo, pero aún no gira la barra, puesto que no se ha añadido aún esta funcionalidad. Para evitar el resalte de la colisión, que no va a hacer falta, se desactiva la opción "**Enable Highlight Collisions**" en las opciones de las colisiones.

El siguiente paso es realizar la macro que fijará los elementos. Para ellos se usan los objetos de conjuntos de colisión ("**CollisionSets**") y piezas ("**Parts**"), y el método de fijar ("**Add**"). El programa quedaría como el de la **Figura IV.1.29**:

```
Sub attach_item_Mesa()
Dim c As CollisionSet
'declaro c como un conjunto de colision

Set c = ActiveStation.CollisionSets("Mesa_Barra1")
'le asigno el conjunto Mesa_Barra1

If c.Active = True Then
    Call ActiveStation.Parts("Mesa").Attachments.Add(c.ObjectsB.Item(1), True)
    'si la colisión está activa fijo las piezas

End If
End Sub
```

Figura IV.1.29- Macro fijar Barra 1 a la Mesa

Con esto se asegura que en el momento de fijar la barra a la mesa, está apoyada en ella, ya que se pone como condición al fijado que el conjunto de colisión esté activo.

Resta incluir en la tabla de eventos el lanzamiento de la macro que fijará las piezas. Cuando se deposite la barra 1 es cuando se debe ejecutar la macro. En ese momento su señal asociada, do1, se hace 0 para que la herramienta suelte la pieza. Aprovechándolo, se añade un evento que haga que cuando la salida do1 se haga 0 se lance la macro attach_item_Mesa().

Una vez hecho esto, esta primera parte del proceso debe quedar completada. Se simula la estación para comprobar que la pieza se traslada, se deposita y se mueve 90º positivos en Z.

Comprobada la correcta implementación del procesado de esta primera barra, y comentadas la práctica totalidad de las utilidades que vamos a usar, los pasos a seguir comprenden el cálculo teórico de las posiciones de ensamblado de las distintas piezas, la creación de puntos y trayectorias relativas a cada pieza, su sincronización al VC y la inclusión de señales y eventos, la simulación del proceso realizado hasta la última pieza realizada y la simulación final de todo el proceso, siguiendo la metodología presentada en este capítulo. Se procede, pues, con la comentada metodología.

u) Creación del resto de puntos y procesos

Barra 2

La barra 2 constituye junto a la barra 3, como se ve en la **Figura IV.1.13,** la parte superior de la estructura metálica. Mirando el esquema de la **Figura IV.1.21** se observa que el punto medio de la barra 2 se debe situar a **1220/2 + 20/2 = 620 mm** de la barra vertical central (X=620 en el SDC de la mesa) y a **2300/2 -20/2= 1140 mm** del eje central (Y=-1140 en la mesa). Así que ya se pueden crear los respectivos puntos **Ap_Barra_2** y **Pto_Barra_2** (los puntos en que se coloca la barra en la mesa de alimentación, 50 mm más alejados que los de la barra 1) y **Ap_EnBarra_2 = (620, -1140, 700) y Ens_Barra_2 = (620, -1140, 645).**

Es reseñable comentar que se pueden crear todos los puntos de agarre de las barras más fácilmente. Se seleccionan Ap_Barra_1 y Pto_Barra_1 a la vez, se pulsa Ctrl+C y a continuación seleccionando el objeto de trabajo 0 se pulsa Ctrl+V hasta tener el número de pares de puntos que se necesitan. Así será más fácil modificarlos en adelante, cuando se proceda a crear los respectivos caminos, pues probablemente sólo hará falta modificar una o dos dimensiones de su posición. Es importante elegir **NO** cuando el programa pida si se quiere actualizar la definición del punto ("**Update Target Definition**") para que no aplique los valores introducidos a los puntos madre que sirvieron para generar los nuevos.

A continuación se crea la trayectoria **Path_Barra_2** como sigue:

Paso_Medio→Ap_Alim→Ap_Barra_2→Pto_Barra_2→Ap_Barra_2→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_2→Ens_Barra_2→Ap_EnBarra_2→Ap_Ens

Se sincroniza al VC y se añade al main desde la pestaña de programas.

Ahora hay que crear las dos entradas en la tabla de eventos que asignen una señal a los procesos de agarre/suelte de esta barra, de manera análoga al caso anterior. Se elige la señal **do12**, por ejemplo. La lista completa de señales asignadas se puede consultar en los anexos (capítulo VII.7). Se añaden las instrucciones Set y Reset y las esperas en los lugares adecuados del programa (puntos Pto_Barra_2 y Ens_Barra_2).

El siguiente paso es hacer que la barra se fije para que se mueva solidariamente con la Barra 1. Para esto se crea, como se ha comentado anteriormente, un nuevo conjunto de colisión, que tendrá por un lado a la barra 1 y por el otro al resto de barras y tubos del programa. De esta manera serán fácilmente accesibles en el programa de VBA, por lo que es **importante ponerlos en el orden de ensamblado**. Se crea este conjunto y se le denomina "Ensamblado".

A continuación, pues, se crea la macro que hace que se vayan fijando las diferentes piezas y se muevan solidariamente con la barra 1. Esta macro es muy similar a la de fijar la barra 1 a la mesa, con la novedad del uso del índice para referenciar todos los elementos del grupo B del nuevo conjunto de colisión, y el uso de una variable global para gestionar ese índice. Quedaría como sigue en la **Figura IV.1.30**:

```
Dim numBarra As Integer
'La declaro como entero y global al módulo para que guarde el valor entre ejecuciones de las subrutinas

Sub attach_item_Ensamblado()

Dim c As CollisionSet
'declaro c como un conjunto de colision

Set c = ActiveStation.CollisionSets("Ensamblado")
'le asigno el conjunto "Ensamblado"

If numBarra < 1 Then numBarra = 1
'para que no dé fallo si no está inicializado numBarra

If c.Active = True Then
    Call ActiveStation.Parts("Barra_2300_1").Attachments.Add(c.ObjectsB.Item(numBarra), True)
End If
'si la colisión está activo, fijo la pieza número(numBarra) del grupo B a la barra 1

numBarra = numBarra + 1
'incremento el numero de barra a adjuntar
End Sub
```

Figura IV.1.30- Macro para fijar el resto de piezas

Se incluye por tanto el evento en la tabla asociando el paso a do12=0 a esta macro. En realidad, la condición de que esté activa la colisión sólo es útil en el primer caso (la primera barra con la mesa) puesto que a partir de ahí, estará activa para todo el resto de la simulación y no servirá para comprobar si las diferentes barras tocan a la mesa.

Ya está lista la segunda trayectoria. Se graba y se simula para comprobar.

Unión I

El siguiente proceso a simular será el de la unión de estas dos barras. Una de las posibilidades reales de unión de dos barras podría ser el sellado o soldadura de las mismas, usando una herramienta apropiada. No es el caso de nuestro modelo de herramienta, pero sí se puede simular una posible manera de realizar una unión de este tipo que sirva como ejemplo. Se llevará la herramienta a la unión de las barras con una orientación adecuada, y se hará esperar ahí unos segundos para simular un proceso de soldadura o sellado.

Para realizar este proceso, es necesario mover la mesa en sentido contrario al que se ha hecho hasta ahora, para que la unión sea alcanzable por el robot. Esto es muy sencillo de realizar con la base de la macro de mover 90º. Simplemente cambiando el valor, se genera una macro para que la mesa se mueva -45º, que servirá para este propósito. En la tabla de eventos se le asigna la señal **do27**, por ejemplo.

La manera más sencilla de realizar los cálculos del punto es partir del punto en la esquina de la estructura, la unión de las barras 1 y 2, antes de mover la mesa -45º, y después rotar ese punto en torno al centro de la mesa esos -45º, y seleccionar la correcta orientación.

El punto comentado estará por tanto en **X = 1220 + 10 +10 = 1240 mm** y en **Y = 2300/2 – 10 = 1140 mm** del centro de la mesa. Por tanto el punto que se busca es el **(1240, -1140, 645)** en el objeto Salida. Se crea también un punto de aproximación a altura **Z = 700 mm.** Se les nombra **Ap_Union_I y Pto_Union_I.** A continuación se rotan uno a uno respecto al centro de la mesa -45º en Z, para lo que se elige en el menú "Rotate" la opción "Parent" y el punto **(0,0,0).** Ya están correctamente situados. Se orientan para que la herramienta llegue recta a la unión, es decir, con la misma orientación que los puntos de ensamblado de las mismas barras 1 y 2. Se crea el **Path_Union_I** como sigue:

Ap_Union_I→Pto_Union_I→Ap_Union_I→Ap_Ens

Se sincroniza al VC y añade al main, y se añade una espera de 2 segundos en el programa después de llegar al punto de unión I, así como la instrucción para activar la señal do27 que gira -45º después del proceso de la barra 2 y después la desactivación. Se graba y simula para comprobar que se ejecuta correctamente.

Barra 3

Primero se crea una macro para mover la mesa **45º positivos**, a partir de cualquiera de las anteriores. Se le asigna en la tabla la entrada **do28** y se incluye en el programa después de la Unión_I, añadiendo también la correspondiente desactivación.

Los puntos de agarre de la barra 3 en la mesa de alimentación estarán situados 50 mm más alejados del robot que los de la barra 2, puesto que la

herramienta también incidirá en el centro de la misma. Creamos los puntos **Pto_Barra_3** y **Ap_Barra_3**.

El punto de colocación de esta barra debe ser **simétrico respecto al eje Y** del mundo del punto de la barra 2. Por tanto, se crean **Ap_EnBarra_3 = (-620, -1140, 700) y Ens_Barra_3 = (-620, -1140, 645)** en el objeto **Salida.** Mantenemos la misma orientación que los puntos de la barra 2.

A continuación, queda crear el Path_Barra_3 como sigue:

Paso_Medio→Ap_Alim→Ap_Barra_3→Pto_Barra_3→Ap_Barra_3→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_3→Ens_Barra_3→Ap_EnBarra_3→Ap_Ens

Se sincroniza al VC y se incluye en el main, se añaden a la tabla los eventos de **agarre y suelte** asociados a la entrada **do13** y la macro de fijar a la estructura al paso do13=0, y se añaden las respectivas instrucciones de Set, Reset y WaitTime al programa de manera análoga a las ocasiones anteriores. Se simula para comprobar el correcto funcionamiento y se sigue con la barra 4.

Barra 4

Los puntos de agarre de esta barra son singulares respecto a los de las anteriores. Si la herramienta incidiera en el centro de la barra, debería depositarla justo en eje central de la mesa de salida. Sin embargo, esto no es posible ya que el robot no alcanza hasta ese punto. Con lo cual, hay que pensar en otro punto de agarre de la barra. Por ejemplo, se agarra a 150 mm del extremo derecho de la barra, según la ve el robot. Es decir a **2300/2 – 150 = 1000 mm del eje Y** del mundo, y el punto de agarre sería **Pto_Barra_4 = (-1000, -650, 520).** Se crean también el **Ap_Barra_4** asociado **a Z=570 mm** y se les aplica la misma orientación que el resto de los puntos de agarre de las otras barras.

Para dejarla, el robot tendrá que situarse en el eje Y del mundo, y a 150 mm del borde superior de la estructura, o sea, a **2300/2 – 150 = 1000 mm** en el eje Y del centro de la mesa. Los puntos por tanto serán **Ap_EnBarra4 = (0, -1000, 700) y Ens_Barra4 = (0, -1000, 625)** en el objeto **Salida.** Hay que fijarse en que la orientación correcta tendrá que estar girada 90º negativos en el eje Z de los otros puntos de suelta (orientación que tienen los puntos creados al ser copiados de cualquier otro par de suelta), lo que se consigue con Modify→Position y eligiendo local para luego indicar -90 en Z.

Ya se puede crear el camino Path_Barra_4:

Paso_Medio→Ap_Alim→Ap_Barra_4→Pto_Barra_4→Ap_Barra_4→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_4→Ens_Barra_4→Ap_EnBarra_4→Ap_Ens

Se sincroniza al VC, se incluye en el main, se añanden a la tabla los eventos de agarre y suelte asociados a la entrada **do14** y la macro de fijar a la estructura al paso do14=0, y se añaden las respectivas instrucciones de Set, Reset y WaitTime al programa de manera análoga a las ocasiones

anteriores. Se simula para comprobar el correcto funcionamiento y se continúa con la unión II.

Unión II

Esta es la unión de las barras 2, 3, y 4. El punto de aplicación está por tanto **en el eje Y** del mundo, y estará a una distancia en **Y = 2300/2 – 20/2 = 1140 mm** del centro de la mesa de salida. Por tanto se crea el **Pto_ Union_II = (0, -1140, 645)** en el objeto Salida y el de aproximación **Ap_Union_II en Z=700**. Su orientación será girada 45º en Z respecto de los puntos de ensamblado de las barras 2 y 3, de manera que la herramienta incida de una manera apropiada, de manera semejante a lo que se haría en un proceso real. La orientación quedaría pues (180, 0, 45) respecto al mundo o a la mesa de salida.

Con esto se crea el Path_Union_II:

Ap_Union_II→Pto_Union_II→Ap_Union_II→Ap_Ens

Se sicroniza al VC, añade al main y se añaden en el programa una espera de 2 segundos cuando se llega al punto de la unión. Se graba y simula para comprobar y se sigue el proceso con la siguiente barra.

Barra 5

Esta barra se debe situar de manera análoga a la barra 1, exactamente en el mismo punto. Se crean primero los puntos de agarre de la barra 5 en el centro de la misma: **Pto_Barra_5 = (0, -700, 520) y Ap_Barra_5 = (0, -700, 570).** La orientación será la misma que el resto de puntos de agarre. Con esto ya se puede crear el Path_Barra_5 de la manera que sigue:

Paso_Medio→Ap_Alim→Ap_Barra_5→Pto_Barra_5→Ap_Barra_5→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_1→Ens_Barra_1→Ap_EnBarra_1→Ap_Ens

Se sincroniza la trayectoria al VC y se incluye en el main. Se añaden a la tabla de eventos la salida que gestionará el agarre/suelte de esta pieza, **do15**, y la asignación de la macro de fijar a la estructura al paso do15=0. Se añaden las esperas de 2 segundos y las instrucciones Set y Reset en los lugares adecuados del programa, así como la instrucción de girar la mesa 90º antes de realizar esta trayectoria (Set do26 entre el Path_Union_II y el Path_Barra_5 en el main, y a continuación el Reset) y se graba y simula para comprobar. Una vez comprobado, se pasa a la unión III.

Unión III

Este paso se calcula de manera similar a la unión I. Partiendo de la situación en que queda la estructura tras poner la barra 5 en la mesa, se calcula el punto de la unión III y luego se rota 45º negativos en Z en torno al centro de la mesa, para que coincida con la unión de las barras, que se girarán -45º desde la posición de ensamblado de la barra 5. Este punto será simetrico respecto al eje Y del punto de la unión I que se usó para luego girar. Es decir **(-1240, -1140, 645)** en el objeto Salida. Se crea junto a su

aproximación asociada y se rotan. Se les llama **Ap_Union_III y Pto_Union_III**. Tienen la misma orientación que el punto de unión I, para que la herramienta incida de una manera adecuada en la unión.

Se crea el **Path_Union_III** como sigue:

Ap_Union_III→Pto_Union_III→Ap_Union_III→Ap_Ens

Se sincronizan al VC y añaden al main, y se incluyen una espera de 2 segundos en el programa después de llegar al punto de unión III, así como la instrucción para activar la señal do27 que gira -45º después del proceso de la barra 5, y después la espera y la desactivación relativas. Se graba y simula para comprobar que se ejecuta correctamente.

Barra 6

Lo primero es incluir en el programa las instrucciones para mover la mesa 45º (do28) después del procesado de la unión III.

Los puntos de agarre tampoco estarán en la mitad de esta barra. Se debe coger, como se hizo con la barra 4, a 100 mm del extremo derecho según la ve el robot, es decir, a X = 1220/2 – 100 = 510 mm del centro de la barra (eje X). Por tanto, se crean los puntos **Pto_Barra_6 = (-510, -750, 520) y Ap_Barra_6 = (-510, -750, 570).** Misma orientación que los otros puntos de agarre.

El punto de ensamblado estará, mirando en la **Figura IV.1.21**, a 330 + 10 mm en X del centro de la mesa, y a 2500/2 – 100 (punto de cogida de la barra) – 20 (ancho de la barra 5)= 1130 en Y. Es decir, el punto **Ens_Barra_6 será (-340, -1130, 645)** en el objeto Salida. Se crea también el punto de aproximación en Z=700 mm y se les dá la orientación del punto de ensamblado de la barra 4, y se crea el Path_Barra_6:

Paso_Medio→Ap_Alim→Ap_Barra_6→Pto_Barra_6→Ap_Barra_6→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_6→Ens_Barra_6→Ap_EnBarra_6→Ap_Ens

Se sincronizan al VC y añaden al main. Se incluyen en la tabla de eventos los de agarre/suelte asociados a **do16**, y la macro de fijar a la estructura cuando do16 se hace 0. Se añaden en el programa las instrucciones referidas a los procesos de agarre/suelte. Se graba y se simula para comprobar que se ejecuta correctamente.

Unión IV

Esta es la unión de la barra 6 con la 5. El punto de unión estará por tanto a la misma distancia en X que la barra 6, o sea **X= -340** respecto al objeto Salida, y a 2500/2 – 10 = **1240 mm en Y.** Por tanto el punto será **Pto_unión_IV = (-340, -1240, 645)** en el objeto Salida. El punto de aproximación asociado se crea en Z = 700mm. La orientación de estos puntos se elige girada 45º en Z respecto a la de los puntos de ensamblado (es decir, se copia la orientación de estos puntos y luego se edita o giran los puntos). También sería adecuado girar -45º.

Se crea el **Path_Union_IV** como sigue:

Ap_Union_IV→Pto_Union_IV→Ap_Union_IV→Ap_Ens

Se sincroniza al VC y añade al main, y se incluye una espera de 2 segundos en el programa después de llegar al punto de unión IV. Se graba y simula para comprobar que se ejecuta correctamente.

Barra 7

Se añaden al programa las instrucciones para mover 90º la mesa (activar do26, esperar y desactivarla), que permitirá al robot llegar adecuadamente a realizar este ensamblado.

El punto de agarre se realiza en el centro de la barra. Por tanto **Pto_Barra_7= (0, -800, 520) y Ap_Barra_7 = (0, -800, 570).** Misma orientación que los otros puntos de agarre.

Mirando la **Figura IV.1.21**, se observa que se debe situar la barra a 330 + 735 – 10 (mitad de la barra) = 1055 mm en Y del centro de la mesa, y a 2500/2 – 20 – 1220/2 = 620 mm en X. Se crean pues los puntos **Ens_Barra_7 = (620, -1055, 645) y Ap_EnBarra_7 = (620, -1055, 700)**. Su orientación es la misma de la de los puntos de ensamblado de la barra 5. Ya se puede crear el **Path_Barra_7**

Paso_Medio→Ap_Alim→Ap_Barra_7→Pto_Barra_7→Ap_Barra_7→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_7→Ens_Barra_7→Ap_EnBarra_7→Ap_Ens

Se sincroniza al VC y añade al main. Se incluyen en la tabla de eventos los de agarre/suelte asociando la barra 7 a **do17**, y la macro de fijar a la estructura cuando do17 se haga 0. Se añaden en el programa las instrucciones referidas a los procesos de agarre/suelte (activación/desactivación do17 y esperas). Se graba y simula para comprobar que se ejecuta correctamente.

Unión V

Para llegar a esta unión hay que girar la mesa -45º. Se incluyen en el programa por tanto las instrucciones convenientes: activar do27, espera, desactivar.

Según está la mesa tras dejar la barra 7, el punto de unión es fácil de calcular. Está a la misma distancia en Y, y a 2500/2 – 10 (mitad de la barra 5) = 1240 mm en X. Es decir, los puntos serán **Pto_Unión_V = (1240, -1055, 645) y Ap_Unión_V = (1240, -1055, 700)** en el objeto Salida. Ahora se rotan en torno al centro de la mesa -45º en Z. Para finalizar, como orientación se usa por ejemplo la de los puntos de ensamblado de la barra 7.

Se crea el **Path_Union_V** como sigue:

Ap_Union_V→Pto_Union_V→Ap_Union_V→Ap_Ens

Se sincroniza al VC y añade al main, y se incluye una espera de 2 segundos en el programa después de llegar al punto de unión V. Se graba y se simula para comprobar que se ejecuta correctamente.

Barra 8

Hay que devolver la mesa a la posición en la que se ensambló la barra 7. Es decir, se añaden al programa las instrucciones para girar 45º la mesa (activar do28, esperar, desactivar).

La barra 8 sufre un proceso semejante a la barra 7. Se coge en su punto medio, por tanto se crean los puntos **Pto_Barra_8 = (0, -850, 520) y Ap_Barra_8 = (0, -850, 570)**. Misma orientación que los otros puntos de agarre.

Los puntos de ensamblado serán simétricos de los de la barra 7 respecto al eje Y del mundo. Es decir, **Ens_Barra_8 = (-620, -1055, 645) y Ap_EnBarra_8 = (-620, -1055, 700).** La orientación es la misma que la de los puntos de la barra 7.

Se crea el **Path_Barra_8:**

Paso_Medio→Ap_Alim→Ap_Barra_8→Pto_Barra_8→Ap_Barra_8→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_8→Ens_Barra_8→Ap_EnBarra_8→Ap_Ens

Se sincronizan al VC y añaden al main. Se incluyen en la tabla de eventos los de agarre/suelte asociando la barra 8 a **do18**, y la macro de fijar a la estructura cuando do18 se haga 0. Se añaden en el programa las instrucciones referidas a los procesos de agarre/suelte (activación/desactivación do18 y esperas). Se graba y simula para comprobar que se ejecuta correctamente. Se sigue con el proceso.

Unión VI

En esta unión confluyen las barras 4, 7 y 8. Para simular este proceso, se va a hacer que la herramienta incida dos veces con dos orientaciones diferentes, como si efectuara dos acciones de soldadura.

El punto está a Y = -1055 mm del centro de la mesa, al igual que las barras 7 y 8, y como está en el eje Y, X=0. Por tanto se crean los puntos **Ap_Union_VI = (0, -1055, 700) y Pto1_Union_VI = (0, -1055, 645)**. Éste último se duplica para crear el **Pto2_Union_VI**, con las mismas coordenadas espaciales pero que tendrá diferente orientación, como se ha comentado. Se hará que la orientación del punto de aproximación y del primer punto sean giradas 45º en Z respecto a los puntos de ensamblado de las barras 7 u 8, y la del Pto2 que sea la girada -45º respecto a estos mismos.

Se crea el **Path_Union_VI:**

Ap_Union_VI→Pto1_Union_VI→Ap_Union_VI→Pto2_Union_VI→Ap_Ens

Se sincroniza al VC y añade al main, y se incluyen esperas de 2 segundos en el programa después de llegar a cada uno de los puntos de unión VI. Se graba y se simula para comprobar que se ejecuta correctamente.

Unión VII

Para llegar a esta unión hay que girar la mesa 45º, de manera semejante a lo realizado con la unión V, empleando la señal do28 en este caso. Se añaden las instrucciones pertinentes en el main.

El punto de unión en la situación actual de la mesa es fácil de calcular. Es simétrico respecto al eje Y del punto de unión V. Por tanto los puntos serán **Pto_Unión_VII = (-1240, -1055, 645) y Ap_Unión_VII = (-1240, -1055, 700)** en el objeto Salida. Ahora se rotan en torno al centro de la mesa 45º en Z. Para finalizar, como orientación se usa por ejemplo la de los puntos de ensamblado de la barra 8.

Se crea el **Path_Union_VII** como sigue:

Ap_Union_VII→Pto_Union_VII→Ap_Union_VII→Ap_Ens

Se sincronizan al VC y añaden al main, y se incluye la espera de 2 segundos en el programa después de llegar al punto de unión VII. Se graba y simula para comprobar que se ejecuta correctamente.

Barra 9

Esta es la última barra a ensamblar. Para permitir que el robot llegue adecuadamente a posicionarla se necesita girar 45º la mesa respecto de la posición en la que se simula la unión VII. Para ello hay que incluir las instrucciones de activar la señal do28, esperar y desactivarla después de la instrucción de ejecutar la unión VII en el main.

Esta barra va a seguir un proceso similar al de la barra 6. Los puntos de agarre tampoco estarán en la mitad de la barra. Se agarra, como se hizo con las barra 4 y 6, a 100 mm del extremo derecho según la ve el robot, es decir, a X = 1220/2 – 100 = 510 mm del centro de la barra (eje X). Por tanto, se crean los puntos **Pto_Barra_6 = (-510, -900, 520) y Ap_Barra_6 = (-510, -900, 570).** Misma orientación que los otros puntos de agarre.

El punto de ensamblado estará, mirando en la **Figura IV.1.21**, en el punto simétrico respecto al eje Y del punto de la barra 6, es decir el punto **Ens_Barra_9 será (340, -1130, 645)** en el objeto Salida. Se crea también el punto de aproximación en Z=700 mm y se les dá la orientación del punto de ensamblado de la barra 6.

Se crea el **Path_Barra_9**:

Paso_Medio→Ap_Alim→Ap_Barra_9→Pto_Barra_9→Ap_Barra_9→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnBarra_9→Ens_Barra_9→Ap_EnBarra_9→Ap_Ens

Se sincroniza al VC y añade al main. Se incluyen en la tabla de eventos los de agarre/suelte asociados a **do19**, y la macro de fijar a la estructura

cuando do19 se hace 0. Se añaden en el programa las instrucciones referidas a los procesos de agarre/suelte. Se graba y simula para comprobar que se ejecuta correctamente.

Unión VIII

Unión de la barra 1 con la 9. Este punto será simétrico respecto al eje Y del de la unión IV. El punto de unión estará por tanto a la misma distancia en X que la barra, o sea **X= 340** respecto al objeto Salida, y a 2500/2 – 10 = **1240 mm en Y.** Por tanto el punto será **Pto_unión_VIII = (340, -1240, 645)** en el objeto Salida, y el punto de aproximación estará en Z = 700 mm. La orientación de estos puntos es la girada 45º respecto a la del punto de ensamblado 9.

Se crea el **Path_Union_VIII** como sigue:

Ap_Union_VIII→Pto_Union_VIII→Ap_Union_VIII→Ap_Ens

Se sincroniza al VC y añade al main, y se incluye una espera de 2 segundos en el programa después de llegar al punto de unión VIII. Se graba y simula para comprobar que se ejecuta correctamente.

Con esto ya se ha ensamblado el total de la estructura metálica que sirve como soporte a las instalaciones verticales. A continuación se pasa a simular el posicionado de una posible configuración básica de estas instalaciones, como se indica en la **Figura IV.1.14**. En este proceso no se va a simular el procesado de las uniones entre la tuberías, simplemente se posicionaran las tuberías para mostrar un esquema básico del Service Core. El procesado de las uniones entre las tuberías, e incluso entre los elementos estructurales, continúa en proceso de estudio dentro del proyecto Manubuild, por lo que aún queda por determinar los materiales y procesos de unión asociados. Se sigue pues con el simulado de esta parte.

Tubo 1

Hay que incluir un giro de 90º en Z desde la posición en que se realizó la unión VIII, añadiendo las instrucciones necesarias relativas a la señal do26 después del procesado de la unión.

El tubo 1 se situará centrado encima de las barras 2 y 3. Por tanto el robot lo cogerá en su punto medio. Tiene un diámetro de 25 mm. Así, los puntos que se deben crear son **Pto_Tubo_1 = (0, -950, 525) y Ap_Tubo_1 = (0, -950, 570)**. La orientación será la misma que la de cualquier punto de alimentación de las barras.

El punto de ensamblado estará en el eje Y. Estará a la misma distancia en Y del centro que la barra 2 o la 3, es decir, a 1140 mm. Debe quedar por encima de la barra 1, así pues Z = 625 + 20 + 25 = 670 mm. Por tanto los puntos de salida serán **Ens_Tubo_1 = (0, -1140, 670) y Ap_EnTubo_1 = (0, -1140, 750)** en el objeto Salida. La orientación de estos puntos será la misma que la de los puntos de ensamblado de la barra 1.

Se crea el **Path_Tubo_1**:

Paso_Medio→Ap_Alim→Ap_Tubo_1→Pto_Tubo_1→Ap_Tubo_1→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnTubo_1→Ens_Tubo_1→Ap_EnTubo_1→Ap_Ens

Se sincroniza al VC y añade al main. Se incluyen en la tabla de eventos los de agarre/suelte asociados a **do21**, y la macro de fijar a la estructura cuando do21 se hace 0. Se añaden en el programa las instrucciones referidas a los procesos de agarre/suelte. Se graba y se simula para comprobar que se ejecuta correctamente.

Tubo 2

Este tubo se situará en la parte derecha según mira el robot la mesa de salida. Es vertical a la estructura, con lo cual la orientación de los puntos de salida será como la de las barras 4, 6 y 9. Por tanto el robot cogerá la pieza de manera similar. En este caso se agarra el tubo a 200 mm del extremo derecho según lo ve el robot. Así el punto de agarre estará a 1620/2 – 200 = 610 mm en X. Este tubo tiene un diámetro de 16mm. Se crean pues los puntos **Pto_Tubo_2 = (-610, -1000, 516) y Ap_Tubo_2 = (-610, -1000, 570).** La orientación es la misma que la de agarre del tubo 1.

Hay que situarlo a 200 mm del tubo 1, es decir a 2300/2 – 200 – 10 = **940 mm en Y** del centro de la mesa. Debe situarse a partir del extremo del tubo 1, con lo cual a 910/2 + 16/2 = **463 mm en X** del centro de la mesa. Así pues, se crean los puntos **Ens_Tubo_2 = (463, -940, 661) y Ap_Tubo_2 = (463,-940, 750)** en el objeto Salida. Tienen la orientación del punto de ensamblado de la barra 4, 6 o 9.

Se genera el **Path_Tubo_2:**

Paso_Medio→Ap_Alim→Ap_Tubo_2→Pto_Tubo_2→Ap_Tubo_2→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnTubo_2→Ens_Tubo_2→Ap_EnTubo_2→Ap_Ens

Se sincroniza al VC y se agrega al main. Se añaden en la tabla de eventos los de agarre/suelte asociados a **do22**, y la macro de fijar a la estructura cuando do22 se hace 0. Se incluyen en el programa las instrucciones referidas a los procesos de agarre/suelte. Se graba y simula para comprobar que se ejecuta correctamente.

Tubo 3

Este tubo se situará en la parte izquierda según mira el robot la mesa de salida. Como vemos en la **Figura IV.1.14**, el proceso de este tubo es similar al del tubo 2. Así el punto de agarre estará a 2100/2 – 200 = 850 mm en X. Este tubo tiene también un diámetro de 16mm. Se crean pues los puntos **Pto_Tubo_3 = (-850, -1050, 516) y Ap_Tubo_2 = (-850, -1050, 570).** La orientación es la misma que la de agarre del tubo 1.

Hay que situarlo a 200 mm del tubo 1, es decir a 2500/2 – 200 – 10 = **1040 mm en Y** del centro de la mesa. Debe situarse a partir del extremo del tubo 1, con lo cual a 910/2 + 16/2 = **463 mm en X** del centro de la mesa. Así pues, se generan los puntos **Ens_Tubo_3 = (-463, -940, 661) y Ap_Tubo_3 =**

(463,-940, 750) en el objeto Salida. La orientación del punto se designa igual a la de los puntos de ensamblado de la barra 4, 6 o 9 o del tubo 2.

Se crea el **Path_Tubo_3:**

Paso_Medio→Ap_Alim→Ap_Tubo_3→Pto_Tubo_3→Ap_Tubo_3→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnTubo_3→Ens_Tubo_3→Ap_EnTubo_3→Ap_Ens

Se sincroniza al VC y añadimos al main. Añadimos en la tabla de eventos los de agarre/suelte asociados a **do23**, y la macro de fijar a la estructura cuando do23 se hace 0. Incluimos en el programa las instrucciones referidas a los procesos de agarre/suelte. Se graba y simula para comprobar que se ejecuta correctamente.

Tubo 4

Este es el último tubo y la última pieza del proceso simulado de ensamblaje del Service Core. Se necesita girar la mesa 180º para que el robot pueda situar el tubo. Por tanto, se introducen en el programa dos giros de 90º con las instrucciones pertinentes relativas a la señal do26.

El robot cogerá el tubo en su punto central. Este tubo tiene un diámetro de 16 mm. Por tanto, se generan los puntos de agarre **Pto_Tubo_4 = (0, -1100, 516) y Ap_Tubo_4 = (0, -1100, 570)**, con la misma orientación del resto de puntos de agarre.

Tendrá que ser situado a la misma distancia en X que el tubo 2 menos el radio del tubo 2 (ya que el tubo 2 se coloca desde arriba, y sin embargo éste es desde un lado; si se dejara a la misma distancia, se colocaría la parte externa del tubo 4 coincidente con el eje central del tubo 2), es decir, a **463 – 8 = 455 mm**. Hay que situarlo en la base del tubo 2, por tanto, a 2100 – 2300/2 + 8 (radio del tubo 4) = **958 mm en Y** del centro de la mesa. En Z, habrá que situarlo a 625 + 20 + 583/2 = **936,5 mm**. En cuanto a la orientación, será girada 90º en X local de los puntos, respecto a la del ensamblado del tubo 1, para que el tubo quede perpendicular a la mesa. Ya se pueden crear los puntos **Ens_Tubo_4 = (455, -958, 936.5) y Ap_Tubo_4 = (455, -958, 1000)** en el objeto Salida.

Queda crear el **Path_Tubo_4:**

Paso_Medio→Ap_Alim→Ap_Tubo_4→Pto_Tubo_4→Ap_Tubo_4→Ap_Alim→Paso_Medio→Ap_Ens→Ap_EnTubo_4→Ens_Tubo_4→Ap_EnTubo_4→Ap_Ens

Se sincroniza al VC y se añade al main. Se agregan en la tabla de eventos los de agarre/suelte asociados a **do24**, y la macro de fijar a la estructura cuando do24 se hace 0. Se incluyen en el programa las instrucciones referidas a los procesos de agarre/suelte. Se graba la estación y se simula para comprobar que se ejecuta correctamente.

Una vez hecho esto ya se dispone de la simulación completa que se pretendía del ensamblado del Service Core. Gracias al método de trabajo y

las herramientas aprendidas durante el proceso, se va a realizar en el apartado siguiente el modelado y simulación de un pequeño proceso constructivo, alternativo y representativo de un futuro módulo de instalaciones, que se llevará al robot real y se probará en el laboratorio, aplicando la metodología propuesta en este capítulo, y que servirá para verificar la utilidad del software y los conocimientos adquiridos.

La vista de la estación debe quedar semejante a como se observa en la **Figura IV.1.31**.

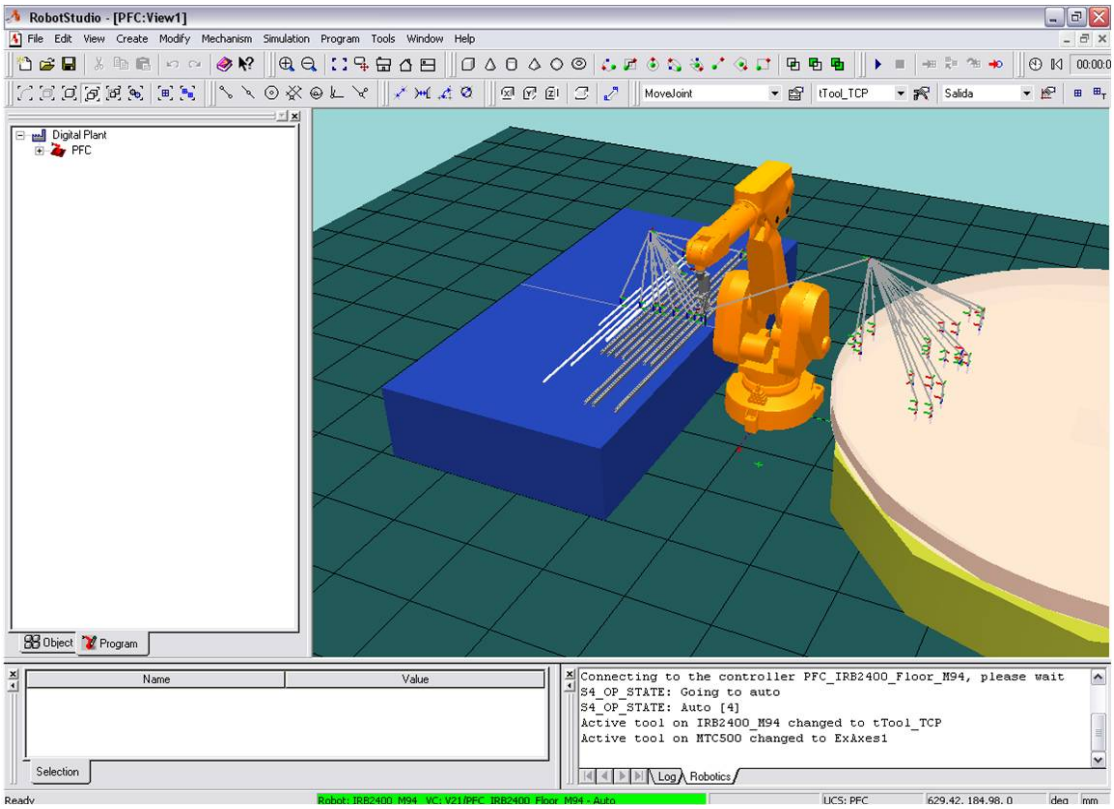
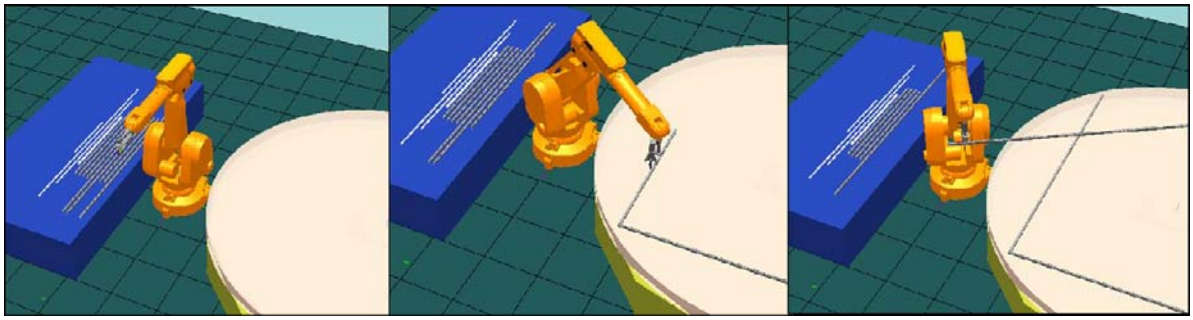


Figura IV.1.31- Visión final de la estación

Se realiza además una captura de video de la simulación que servirá como muestra visual de la filosofía de funcionamiento de la factoría. Para ello se hacen invisibles puntos y trayectorias para obtener una imagen más clara del proceso, y se utiliza una herramienta de carácter general de captura de video. A continuación se muestra una secuencia de este vídeo.



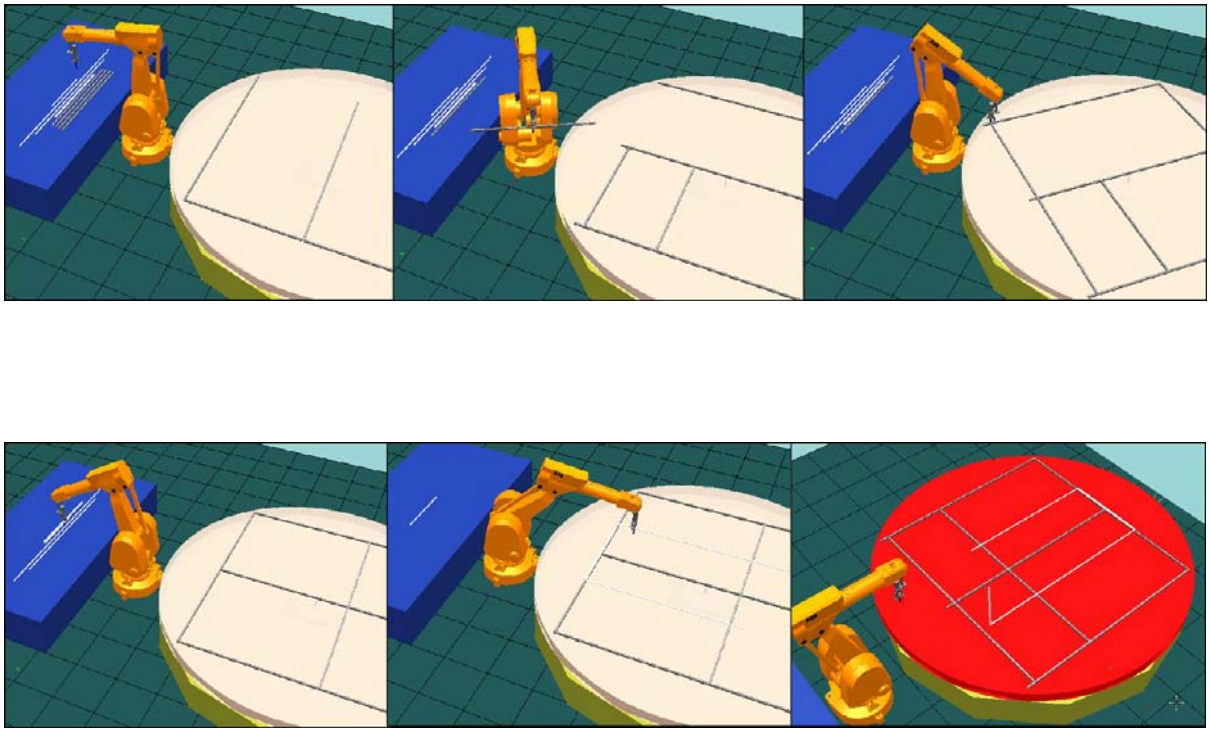


Figura IV.1.32 – Secuencia de la simulación del ensamblado

IV.2 - APLICACIÓN DE LA METODOLOGÍA PARA LA IMPLEMENTACIÓN DE UN PROCESO REAL DE ENSAMBLADO

Para complementar el proceso de creación y simulación estaciones en el programa RobotStudio que se ha realizado en el apartado anterior, en este apartado se realiza además la transición software-robot de un pequeño proceso análogo a la fabricación del Service Core, gracias a la disponibilidad de equipamiento del laboratorio de la universidad Carlos III de Madrid (robot ABB, herramientas, piezas para el diseño de la estación y el futuro Service Core a fabricar). Servirá como una pequeña demostración alternativa virtual y física. Se seguirá también la metodología mostrada en el flujograma de la **Figura IV.1.5**. Con ello, se conseguirá ampliar el conocimiento del programa a este proceso de transición, aplicando la metodología establecida a seguir y solucionando los posibles problemas que puedan surgir en una transición de este tipo.

01. Creación de la estación

Para ello primero se genera el layout correspondiente a esta pequeña demostración. Se va a realizar el ensamblado de una pequeña estructura cuadrada formada por elementos que se ensamblan de manera simple. Los elementos considerados presentan morfología paralelepípeda. La distribución elegida es parecida a la realizada en el modelo simulado en el capítulo IV.1. Habrá una mesa de alimentación de piezas y otra en la que se ensambla el producto. Ambas mesas se sitúan a ambos lados del robot, como se indica en la figura siguiente:

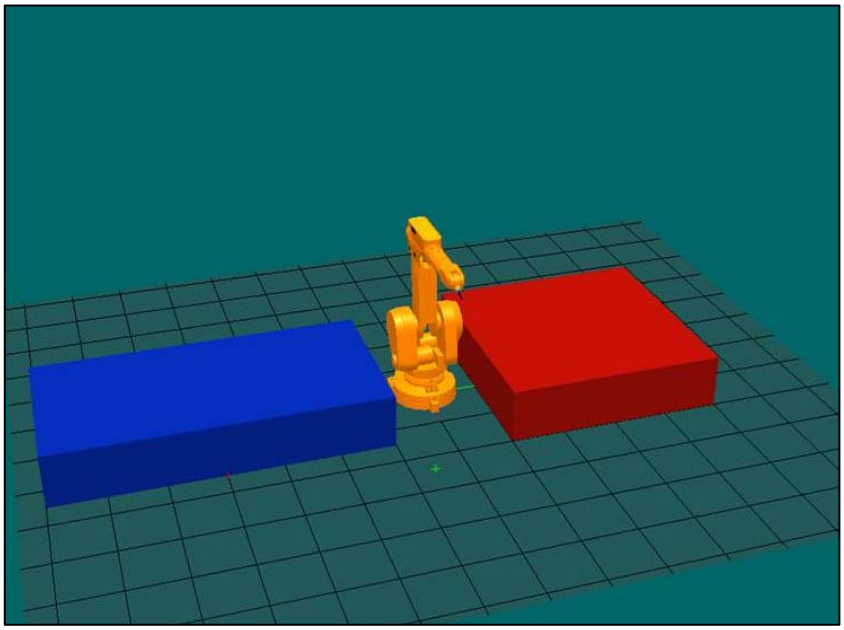


Figura IV.2.1- Layout de la demostración

Este layout es fácil de generar una vez realizada la simulación anterior. Simplemente se importa el robot IRB 2400_M94 y se crean dos mesas:

- Mesa Azul-Alimentación: Se genera un sólido de 1250x3000x500 mm y se sitúa a 500mm del origen (se puede corregir el emplazamiento inicial de la mesa con "Place"). Se le aplica un color al gusto, en este caso azul.
- Mesa Roja-Ensamblado: Se procede análogamente, con un sólido de 2000x2000x500 mm situado a partir de Y=500 mm. Se elige el color rojo.

Creación y situación de herramientas y piezas

Los siguientes pasos son crear una herramienta e importar las geometrías necesarias para la demostración.

En esta ocasión, en lugar de usar una geometría para crear la herramienta, se usará un gráfico estándar para imitarla, o "dummy". Para ello simplemente se selecciona utilizar pieza "dummy" (**"use dummy part"**) en lugar de usar una pieza existente en el primer diálogo del asistente para crear una herramienta, como se ve en la **Figura IV.2.2**:

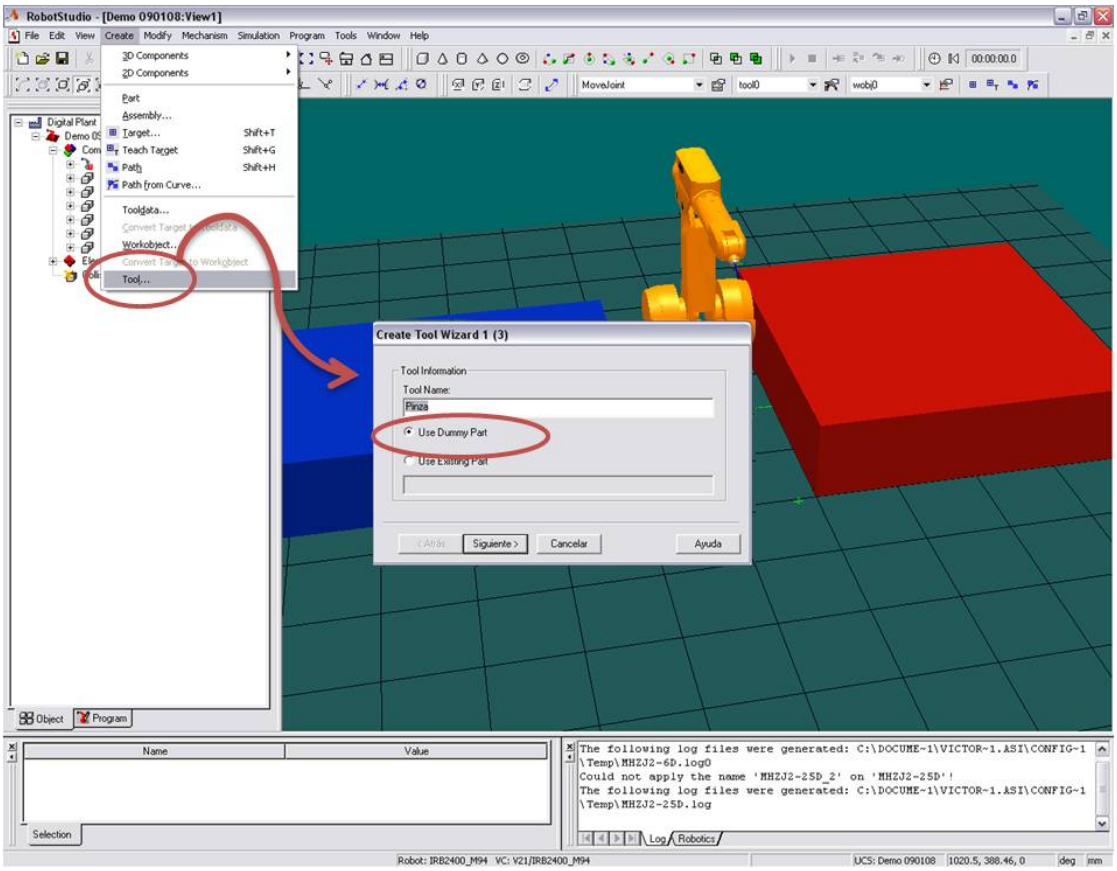


Figura IV.2.2- Crear herramienta con una "dummy part"

Además, se nombra la herramienta como "Pinza", a su TCP como **"Pinza_TCP"** y se posiciona en (0,0,100) con la misma orientación de la muñeca. Se selecciona 1 kg. de masa y se sitúa el centro de gravedad en

(0,0,35). Estas magnitudes son aproximadas ya que no se pretende que tenga relevancia en el proceso. Como ya se hizo con la simulación anterior, se agrega la herramienta al robot, pinchando en "Pinza" en el navegador y arrastrándolo hasta el robot. Se selecciona "Sí" cuando el programa pregunte si se quiere reposicionar la pinza. Ya está la herramienta lista, y se comprueba que es la herramienta activa en la barra situada sobre el área gráfica.

El siguiente paso es crear y posicionar las piezas. Los elementos se crean en solidworks a partir de croquis en 2D que se extrusionan para obtener la forma deseada, y utilizando los datos dimensionales que nos proporciona el fabricante. Los conectores de dos brazos (esquinas) presentan las dimensiones características que se observan en la **Figura IV.2.3**, y el resto de elementos son barras de sección cuadrada de 25 mm de lado, con huecos interiores para el ensamblado, de manera que ajusten perfectamente en los racores. Son necesarias **cuatro barras de 800 mm.**

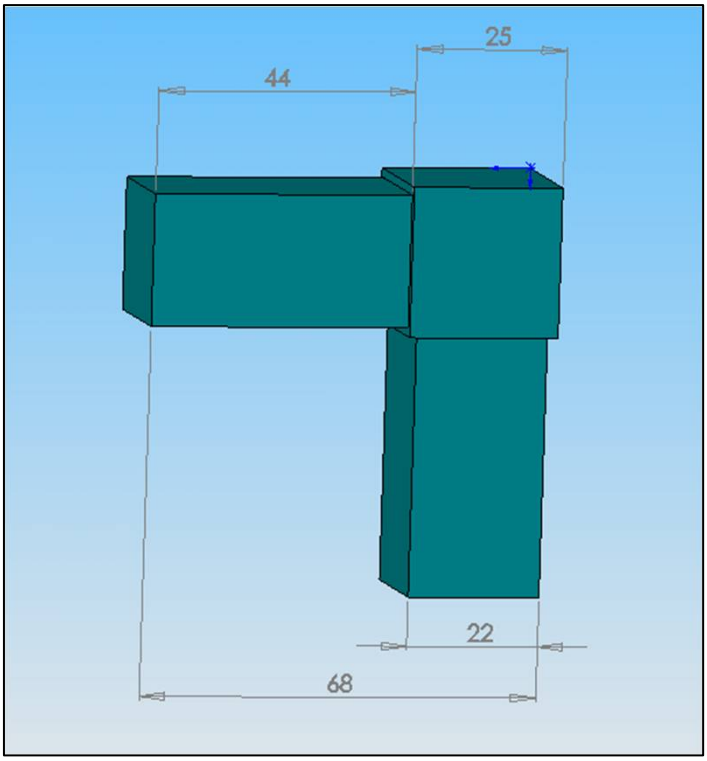


Figura IV.2.3- Cotas del conector de dos brazos en SW

En la siguiente figura se muestra una vista del elemento que se quiere producir ya ensamblado, con dimensiones generales a tener en cuenta en el cálculo de los puntos de la estación. La geometría que se pretende producir también tiene influencia en cómo se alimentarán las piezas, y por tanto en cómo se deben posicionar en la mesa azul.

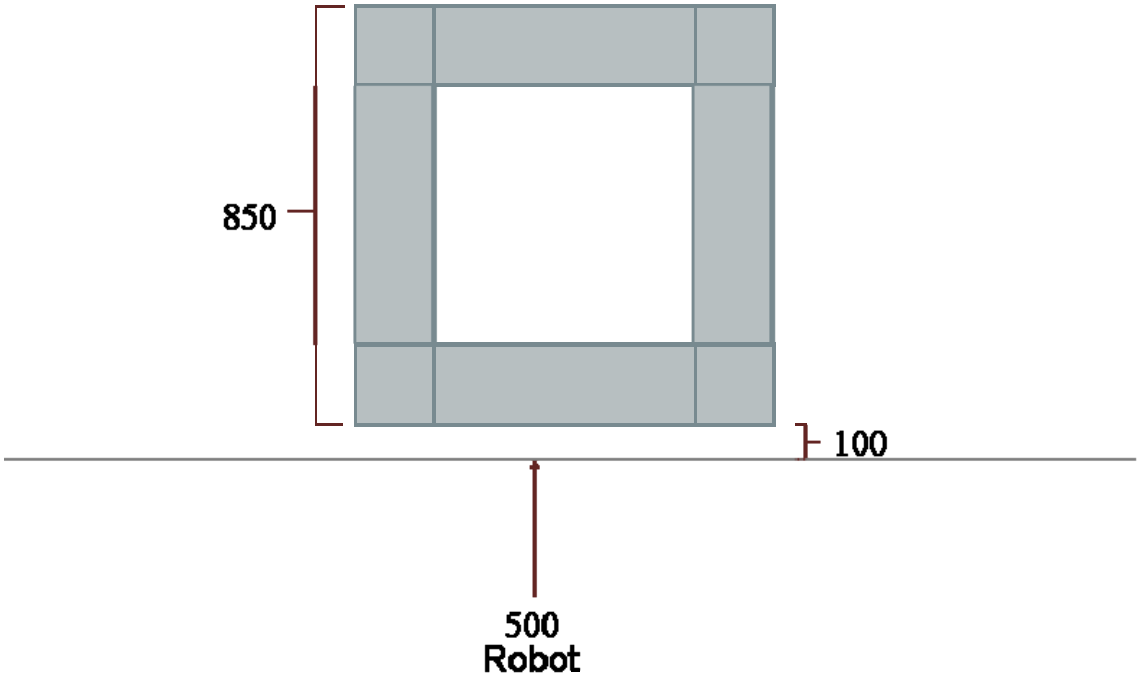


Figura IV.2.4- Dimensiones generales (mm) a tener en cuenta para el ensamblado

Se prosigue con la incorporación de elementos a la estación. Se puede crear una línea recta en la parte central de la mesa azul que ayudará a posicionar las piezas. Se crea una nueva curva entre los puntos (0,-500,500) y (0,-3500,500) para lo que puede ayudar del nivel de selección curva y el modo de ajuste punto medio. Se le da el nombre **"Linea_Central".** Se crea una línea más, a una altura de 25 mm de la mesa azul, que ayudará a posicionar los elementos, haciendo Ctrl+C en "Linea_Central" y Ctrl+V en "Components". Se sitúa con el comando "Place" y se le aplica un color diferente, p. ej. amarillo.

Ahora, se importan los cuatro conectores y las cuatro barras de 800 mm. Para posicionarlos, hay que fijarse en el proceso productivo que se muestra en la **Figura IV.2.5** pensado a partir del diseño del producto a ensamblar.

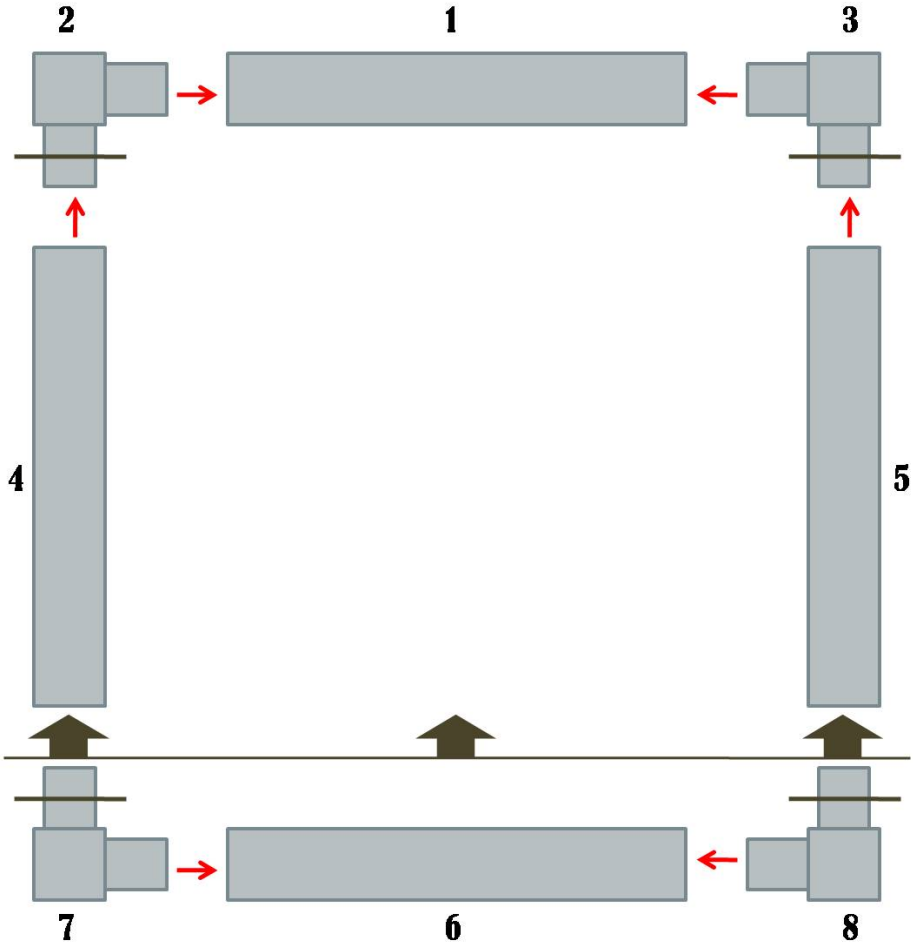


Figura IV.2.5 – Proceso productivo de la demostración física

Presenta dos peculiaridades importantes:

- Una: para realizar el proceso estudiado adecuadamente, la herramienta va a necesitar coger los racores por diferentes puntos, dónde indican las líneas marrones, lo que influye en la manera en que éstos se "alimentarán", o en cómo las cogerá el robot.
- Dos: El último proceso de ensamblado **se desglosa** en un subproceso de ensamblado de las piezas 6, 7 y 8 aparte, que luego se ensamblan al resto del sistema juntas en otro subproceso. Para este subproceso se necesitará el conveniente espacio en la mesa, que debe ser mayor de 69 mm (ancho total del racor, brazo + esquina). Estos procesos hay que tenerlos en cuenta a la hora del cálculo de los puntos y trayectorias que se realizarán más adelante.

Respecto a la primera peculiaridad comentada, se decide que la "alimentación" de piezas se realice con las piezas posicionadas con la misma orientación en la mesa. Ésta es una manera más cercana a cómo se esperaría en un proceso real, en el que la estandarización es una técnica básica. Por tanto, se van a colocar las piezas como queda en la siguiente figura.

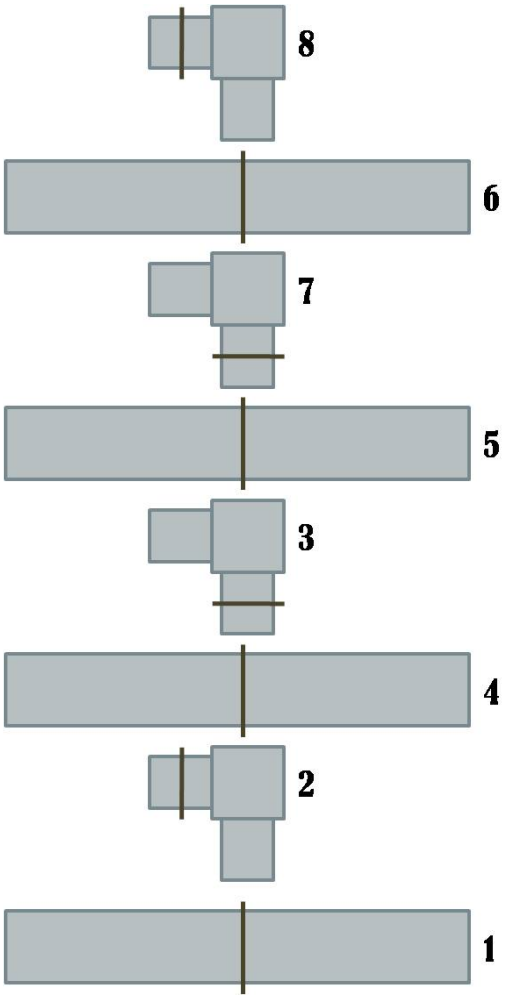


Figura IV.2.6 – Posicionamiento de las piezas en alimentación

Se observa cómo la herramienta incide en diferentes lugares de los racores para seguir el proceso productivo pretendido. La pinza actuará sobre el brazo del conector señalado con la línea marrón en la **Figura IV.2.6**.

Se posiciona la primera barra. Se coloca encima de la mesa, posicionando el punto de la mitad de la barra más alto en (0,-600,525), para lo que se puede hacer uso del comando "Place" y el modo de ajuste "Center" y/o de la línea amarilla que se creó antes. A continuación hace falta rotar el tubo 90º en el eje Z del mundo en torno al punto anterior (0, -600, 525) para dejarlo listo. Una vez hecho esto, se copia su orientación al resto de tubos y se posicionan en la mesa en los lugares indicados en la **Figura IV.2.6**, es decir, 150 mm más alejada del robot cada barra, para dejar espacio entre medias a los racores.

Para posicionar los racores se usa como referencia el punto central del elemento central de los mismos, o sea el punto (12.5, 12.5) de la esquina. Se observa que la orientación que presentan al importarlos es la correcta a la hora de colocarlos en la mesa. Ese punto se debe posicionar, en el caso del racor 2, en Y = -600 – 12.5 (media barra) – 44 (brazo racor) – 12.5 (punto medio esquina) = -654.5 mm, más un cierta holgura. Por ejemplo, en Y = -

700 mm. Se coloca por tanto el punto comentado en (0, -700, 525). El resto de racores se posicionan 150 mm más allá cada uno. Ya está lista la mesa de alimentación.

Al final de este proceso, la estación debe quedar similar a la de la siguiente **Figura IV.2.7**:

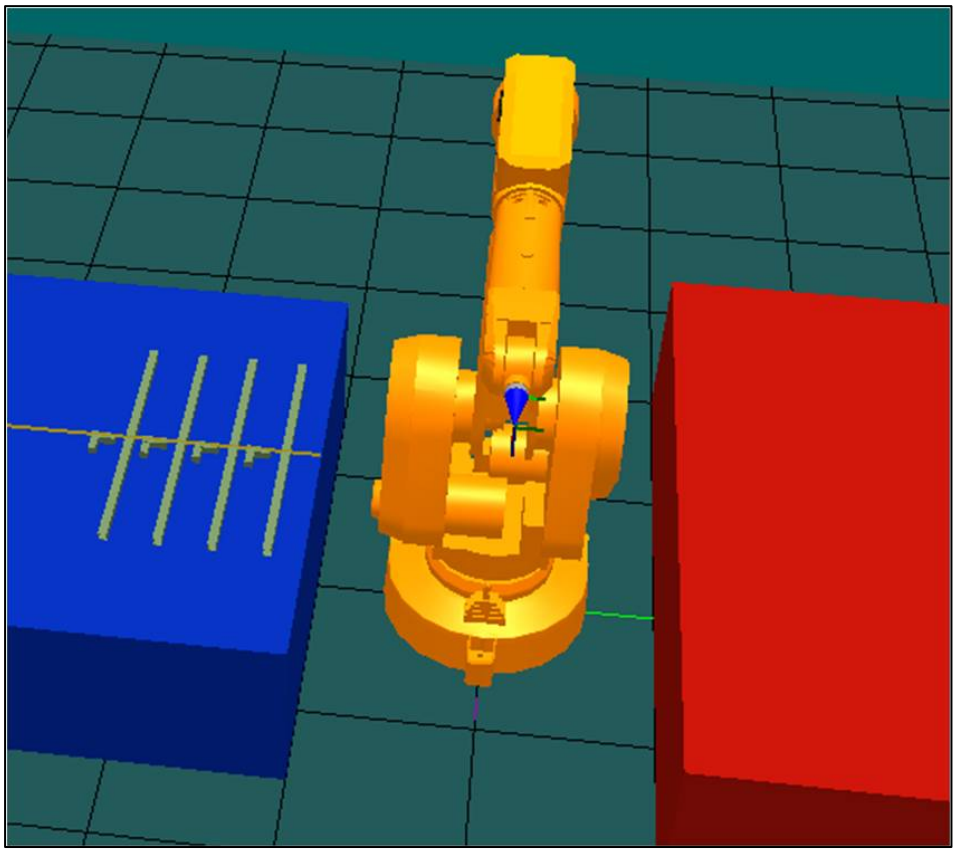


Figura IV.2.7- Conectores y tubos situados en la mesa

Una vez establecida la distribución de inicio de la estación, se prosigue con la activación y asignación de entradas/salidas. El proceso que se seguirá, siguiendo las pautas del flujograma de la **Figura IV.1.5** es crear primero todos los puntos y las trayectorias pieza por pieza, y a continuación introducir las entradas/salidas y simular el proceso.

02. Simulación del ensamblado

Activación y asignacion de entradas/salidas

Lo primero que hay que hacer es activar las señales de entrada y salida del robot que van a servir para comprobar en la simulación que el programa funciona correctamente, para lo que se realiza el mismo procedimiento que en el apartado IV.1:

1) Activar el controlador virtual (para lo que hay que recordar que primero es conveniente realizar una copia nueva, a la que se puede nombrar **Demo_IRB2400_Floor_M94**).
2) Cargar los parámetros de E/S: Abrir el **Teach Pendant** (o Unidad de programación virtual), poner en manual y seleccionar "system parameters" (parámetros del sistema). Elegir I/O Signals, y en el menú File seleccionar "add new parameters". Luego, en el menú "IOConfig" elegir "**1DIG**".
3) Reiniciar VC y comprobación: En File elegir reiniciar ("Restart") y OK. Pasar a modo auto y cerrar el Teach Pendant. Ya están las señales activadas. Para comprobarlo abrir el **IOSimulator** que se encuentra en el menú "Mechanism" y observar que hay señales disponibles.

Ahora se necesita asociar las señales a las piezas mediante la tabla de eventos. Se asocia **do1** a la pieza 1, **do2** a la pieza 2, **do3** a la 3, y así sucesivamente, de manera análoga a como se hizo en la simulación del apartado anterior. Se puede consultar la tabla de eventos de esta demostración en los anexos, capítulo VII.6.

Una vez asignadas, se pasa a generar las trayectorias. Como comprobación de la correcta generación de los puntos y trayectorias, es conveniente hacer que el robot las recorra ("Move Along Path") para ver que no hay problemas de alcance de los límites. Así, además de comprobar el correcto funcionamiento, se pueden depurar fallos de manera más inmediata y sencilla que si se dejara la comprobación sólo para el proceso final total. Se evita simular los subprocesos de cada pieza paso a paso, con lo que se ahorra el tiempo de sincronizar, modificar el programa cada vez etc. Es de esperar que ya se domine cómo afrontar estos pasos, que se realizarán todos conjuntamente al final y ya que no se espera que en las simulaciones de agarre/suelte se vaya a encontrar ningún problema.

Creación de trayectorias

Para empezar se genera un punto medio aproximadamente en la posición inicial del robot que servirá de paso intermedio entre las dos mesas. Se le llama **"Paso_medio"** y lo creamos en (1000, 0, 1500) con orientación (0, 120, 0).

-Trayectoria de la barra 1: Se crea ahora el primer punto para coger pieza: (0, -600, 525) y con orientación (0, 180, 0) en el mundo. Se le dá el nombre **Pto_Barra_1**. Es sencillo generar los demás puntos de alimentación de los tubos a partir de éste, pues solo cambian su coordenada Y a -750, -900 y -1050 respectivamente. También se genera el punto de aproximación al agarre, por ejemplo, en Z= 575 mm, con lo que **Ap_Barra_1 = (0, -600, 575)** y con la misma orientación de Pto_Barra_1. A partir de éste es sencillo crear las demás aproximaciones de manera semejante a cómo se ha procedido con los puntos de alimentación. También se define un punto que sirva de aproximación general para coger las piezas, por ejemplo en (0, -700, 900) y con la misma orientación que los puntos de alimentación de piezas. Se le nombra **Aprox_Azul**.

Se crea la primera parte de la nueva trayectoria (**Path_Barra_1**) con la siguiente secuencia:

Paso_medio→Aprox_Azul→Pto_Barra_1→Aprox_Azul→Paso_medio

Esta trayectoria implementará la secuencia de ensamblado del primer tubo.

Queda posicionar este tubo en la mesa roja. Se crea el punto **Aprox_Roja** en (0, 1000, 900) con orientación (0, 180, 180) en el mundo, para que ir de Aprox_Azul a Aprox_Roja pasando por Paso_medio resulte en un movimiento aproximadamente semicircular de la pieza centrado en el origen del mundo, y exigir así menos movimientos en los distintos ejes del robot. La primera barra va a constituir el travesaño superior de la estructura como se observa en la **Figura IV.2.5**. Como quiera que mide 800 mm. y los conectores tienen una cota de 44 mm por brazo (desde una base hasta el final del conector), permitiendo que la barra se introduzca en la totalidad de los 44 mm en cada conector tenemos que cada lado de la estructura mide: 800 + 25 + 25 = **850 mm de contorno**. Lo vemos en la **Figura IV.2.8.**

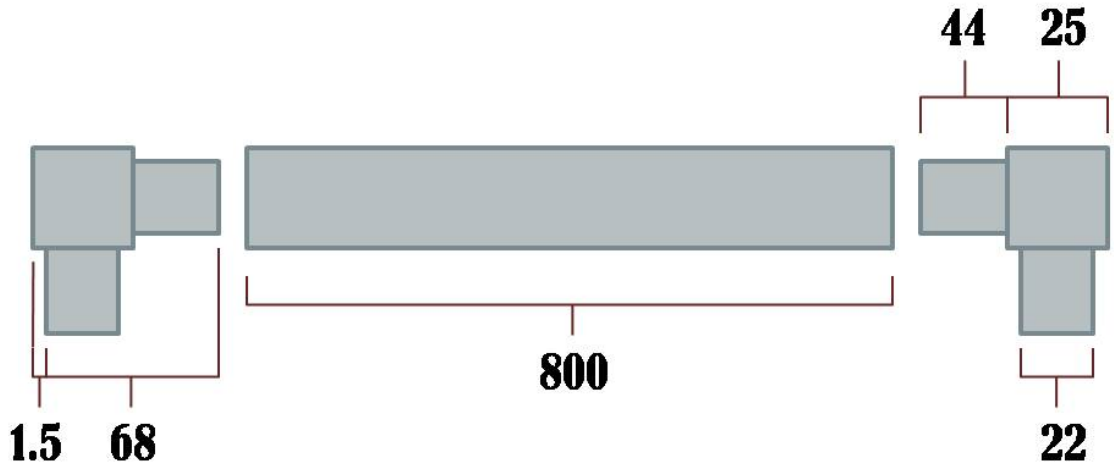


Figura IV.2.8 – Lado de la estructura

Como la mesa roja está situada a partir de Y=500 mm el punto de ensamblado de este primer tubo se crea por tanto por ejemplo en Y=1437'5

mm, llegando así la estructura hasta Y = 1450 mm (por eso la mitad de la barra se coloca en Y=1437'5), y así dejar el espacio libre en la zona de la mesa roja próxima al robot de unos 1450 – 850 – 500 = 100 mm que refleja la **Figura IV.2.4**. Este espacio será necesario para ensamblar el conjunto 6-7-8. Ya está listo el punto **Ens_Barra_1 = (0, 1437.5, 525)**. Se genera también el punto de aproximación **Ap_EnBarra_1 a Z = 600 mm.** La orientación será la girada 180º en Z de la de los puntos de agarre.

Se incluye la secuencia Aprox_Roja-Ap_EnBarra_1-Ens_Barra_1-Ap_EnBarra_1-Aprox_Roja al **Path_Barra_1** y se recorre con el robot. Este camino debe quedar parecido al que se muestra en la **Figura IV.2.9**.

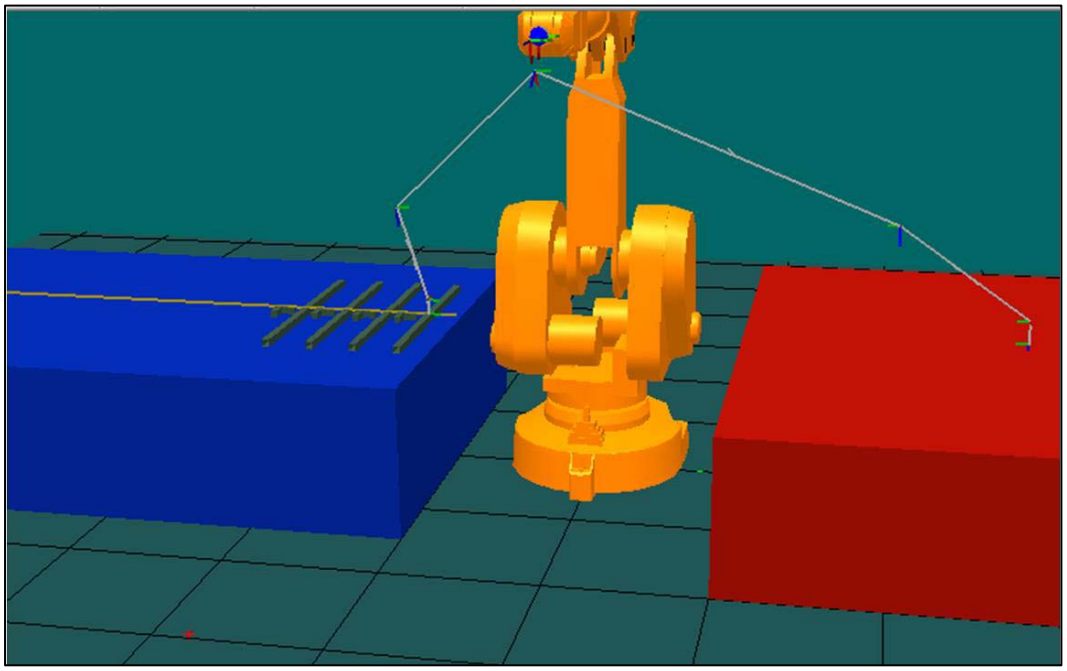


Figura IV.2.9- Path_Barra_1 de la demostración

-Trayectoria racor 2: Se crean los puntos de agarre para el racor 2. Hay que observar que la herramienta debe incidir sobre el brazo paralelo a las barras posicionadas en la mesa. Este punto de agarre se escoge a 20 mm del extremo del brazo, por tanto estará a **X = 69 – 20 – 12.5** (mitad de la esquina) **= 36.5 mm del eje Y**, y a la misma distancia en Y que el punto que se usa para situarlo (700 mm). Es decir, el punto a crear será **Pto_Racor_2 = (36.5, -700, 525)** y la aproximación **Ap_Racor_2 = (36.5, -700, 575).** A continuación se les aplica la misma orientación que la de los de agarre de la barra 1 (con Copy/Apply Orientation) y ya están estos puntos listos.

Una vez creados estos puntos se calculan el resto de puntos para crear la trayectoria para el racor 2. Puesto que la estructura tiene un lado, como se ha calculado antes, de 850 mm, la mitad de ese lado son 425 mm. Por tanto, ya que el robot coge al racor en la mitad del brazo, hay que situarlo a **425 –**

12.5 mm, o sea, en **X = -412.5 mm**. En Y, la distancia hasta la línea media de la barra 1 será de 69 (longitud del racor) – 20 (distancia de agarre hasta la base) – 12.5 (media barra 1) = 36.5 mm. Por tanto **Y = 1437.5 – 36.5 = 1401 mm.** Así pues el punto será **Ens_Racor_2 = (-412.5, 1401, 525).** Como la dirección de ensamblado es en el sentido del eje X positivo, se crea la aproximación en **X = -500 mm** (con esto se consigue que el racor no colisione con la barra, ya que existe una holgura de 500 – 412.5 = 87.5 mm, mayor que la longitud del racor, 69 mm). Es decir, **Ap_EnRacor_2 = (-500, 1401, 525).** La orientación de estos dos puntos será la **girada 90º en Z** del mundo respecto a la de agarre del racor.

Ya está lista la nueva trayectoria, **Path_Racor_2**, la cual se recorre con el robot, que implementa la secuencia:

Paso_medio→Aprox_Azul→Ap_Racor_2→Pto_Racor_2→Ap_Racor_2→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnRacor_2→Ens_Racor_2→Aprox_Roja.

-Trayectoria racor 3: Este racor se sitúa al lado contrario del dos. Sin embargo en este caso el robot agarra por el brazo posicionado perpendicularmente a las barras en la mesa de alimentación. Por tanto, en este caso el punto de agarre estará en X=0. De la misma manera que con el racor 2, se agarra la pieza a **20 mm de la base**, es decir, a **69** (ancho total del racor) **– 20 – 12.5** (distancia del punto de posicionamiento en la mesa al borde de la pieza) **= 36.5 mm** del punto de posicionamiento de la pieza en la mesa de alimentación. Por tanto, el punto de agarre estará en **Y= -850 + 36.5 = 813.5 mm**. Se crean los puntos **Ap_Racor_3 = (0, -813.5, 575) y Pto_Racor_3 = (0, -813.5, 525).** La orientación de estos puntos debe ser girada 90º en Z respecto a la de los puntos de agarre del racor 2, puesto que será necesario para que la herramienta incida de forma correcta en este punto (aunque en la simulación no se notará la diferencia debido a que se está usando una herramienta compuesta por un cono que imita el funcionamiento de una real).

El punto de ensamblado es el simétrico en X del del racor 2 como se ve en la **Figura IV.2.5.** Por tanto se generan sin más los puntos **Ens_Racor_3 = (412.5, 1401, 525) y Ap_EnRacor_3 = (500, 1401, 525).** La orientación será la girada 180º en Z del mundo respecto a la de los puntos de agarre de este racor. Se crea la nueva trayectoria, **Path_Racor_3**, la cual se recorre con el robot comprobando que funciona bien el proceso, que incluye la secuencia:

Paso_medio→Aprox_Azul→Ap_Racor_3→Pto_Racor_3→Ap_Racor_3→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnRacor_3→Ens_Racor_3→Aprox_Roja.

-Trayectoria barra 4: Esta barra se sitúa a la izquierda del ensamblado según lo ve el robot (**Figura IV.2.5.**). El robot cogerá la pieza en su punto

medio, de manera semejante a la barra 1. Por tanto, los puntos de agarre serán **Ap_Barra_4 = (0, -750, 575) y Pto_Barra_4 = (0, -750, 525).** La orientación de los puntos será la misma que la de los puntos de la barra 1.

El punto de ensamblado estará en el eje central de la barra. Se debe situar en **X= - 850/2 + 12.5** (mitad de la barra) = **-412.5 mm**. Se tiene que situar a la mitad de la estructura, por tanto en **Y = 1450 – 850/2 = 1025 mm**. Así pues se crean los puntos **Ens_Barra_4 = (-412.5, 1025, 525) y Ap_EnBarra_4 = (-412.5, 950, 525),** puesto que el movimiento de ensamblado es en el sentido positivo del eje Y**.** La orientación será la misma que la de los puntos de ensamblado del racor 2. Se genera la nueva trayectoria, **Path_Barra_4**, que se recorre con el robot para comprobarla, y que incluye la secuencia:

Paso_medio→Aprox_Azul→Ap_Barra_4→Pto_Barra_4→Ap_Barra_4→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnBarra_4→Ens_Barra_4→Ap_EnBarra_4→Aprox_Roja.

-Trayectoria barra 5: Esta barra se sitúa a la derecha del ensamblado según lo ve el robot (**Figura IV.2.5.**). El robot cogerá la pieza en su punto medio, de manera semejante a la barra 1. Por tanto, los puntos de agarre serán **Ap_Barra_5 = (0, -900, 575) y Pto_Barra_5 = (0, -900, 525).** La orientación de los puntos será la misma que la de los puntos de la barra 1.

El punto de ensamblado estará en el eje central de la barra. Se debe situar en **X= 850/2 – 12.5** (mitad de la barra) = **412.5 mm**. Se tiene que situar a la mitad de la estructura, por tanto en **Y = 1450 – 850/2 = 1025 mm**. Así pues se generan los puntos **Ens_Barra_5 = (412.5, 1025, 525) y Ap_EnBarra_5 = (412.5, 950, 525),** puesto que el movimiento de ensamblado es en el sentido positivo del eje Y**.** La orientación será la misma que la de los puntos de ensamblado de la barra 4. Se crea la nueva trayectoria, **Path_Barra_5**, y se recorre con el robot para comprobar, ejecutando la secuencia:

Paso_medio→Aprox_Azul→Ap_Barra_5→Pto_Barra_5→Ap_Barra_5→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnBarra_5→Ens_Barra_5→Ap_EnBarra_5→Aprox_Roja.

-Trayectoria barra 6: Esta barra es la primera del subproceso de ensamblado 6-7-8 (**Figura IV.2.5.**). El robot tiene que coger la pieza en su punto medio, de manera semejante a la barra 1. Por tanto, los puntos de agarre serán **Ap_Barra_6 = (0, -1050, 575) y Pto_Barra_6 = (0, -1050, 525).** La orientación de los puntos será la misma que la de los puntos de la barra 1.

Hay que crear el punto en el que se situará esta barra para ejecutar el ensamblado del conjunto 6-7-8 antes de incorporarlo al producto final. Las barras 4 y 5 llegan hasta una cota en Y de 1450 – 850 + 25 = 625 mm. Con lo cual para que las piezas 7 y 8 no colisionen bastará con situar la barra 6 en

Y = 540 mm. Así pues se crea el punto **Ens_Barra_6 = (0, 540, 525)** y el **Ap_EnBarra_6 = (0, 540, 600)**. La orientación será la misma de los puntos de ensamblado de la barra 1.

Se genera la trayectoria, **Path_Barra_6**:

Paso_medio→Aprox_Azul→Ap_Barra_6→Pto_Barra_6→Ap_Barra_6→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnBarra_6→Ens_Barra_6→Ap_EnBarra_6→Aprox_Roja.

-Trayectoria racor 7: Este racor unirá las barras 6 y 4 (**Figura IV.2.5.**). El robot tiene que coger la pieza en su brazo inferior, de manera semejante al racor 3 (a 36.5 del punto en que se posicionó la pieza, Y = 1000 mm) . Por tanto, los puntos de agarre serán **Ap_Racor_7 = (0, -963.5, 575) y Pto_Racor_7 = (0, -963.5, 525).** La orientación de los puntos será la misma que la de los puntos del racor 3.

El punto de ensamblado con la barra 6 se deberá situar a la misma distancia en X que la barra 4, o sea, X = -412.5 mm. En Y, deberá situarse a 69 – 20 – 12.5 = 36.5 mm del eje medio de la barra 4 (Y = 540 mm). Así pues se crea el punto **Ens_Racor_7 = (-412.5, 576.5, 600)** y el **Ap_EnRacor_7 = (-500, 576.5, 600),** para ejecutar el movimiento de ensamblado en el sentido positivo del eje X. La orientación será la misma de los puntos de agarre de esta barra.

Se genera la trayectoria, **Path_Racor_7** y la recorremos con el robot para comprobar:

Paso_medio→Aprox_Azul→Ap_Racor_7→Pto_Racor_7→Ap_Racor_7→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnRacor_7→Ens_Racor_7→Aprox_Roja.

Para que esta pieza se mueva solidaria a la barra 6, cosa que se necesita para simular el ensamblado del conjunto 6-7-8 en el resto del conjunto, hay que añadir los eventos necesarios en la tabla. Para conseguirlo, se añade el evento que hace que cuando se suelte la barra, es decir, do7 se haga 0, se fije a la barra 6.

-Trayectoria racor 8: Este racor unirá las barras 6 y 5 (**Figura IV.2.5.**). El robot tiene que coger la pieza en su brazo transversal, de manera semejante al racor 2 (a 36.5 del punto en que se posicionó la pieza, X = 0 mm, Y = 1150 mm). Por tanto, los puntos de agarre serán **Ap_Racor_8 = (36.5, -1150, 575) y Pto_Racor_8 = (36.5, -1150, 525).** La orientación de los puntos será la misma que la de los puntos del racor 2.

El punto de ensamblado con la barra 6 se deberá situar simétrico del del racor 7 respecto al eje X, es decir, **Ens_Racor_8 = (412.5, 576.5, 600)** y el **Ap_EnRacor_8 = (500, 576.5, 600),** para realizar el movimiento de

ensamblado en el sentido negativo del eje X. La orientación será la girada -90º en Z respecto de los puntos de agarre del racor.

Se crea la trayectoria, **Path_Racor_8** y se recorre con el robot para comprobar:

Paso_medio→Aprox_Azul→Ap_Racor_8→Pto_Racor_8→Ap_Racor_8→Aprox_Azul→Paso_medio→Aprox_Roja→Ap_EnRacor_8→Ens_Racor_8→Aprox_Roja.

Se añade también el evento asociado a do8=0→fijar a barra 6, de manera semejante a como se ha hecho con el racor 7.

-Trayectoria punto final 6F: Llevar el conjunto 6-7-8 a este punto completará este subproceso y el ensamblado completo del conjunto. Este punto se debe situar en **Y = 1450 – 850 +12.5 = 612.5.** Así moviendo la estructura del punto En_Barra_6 al que se crea, **En_Barra_6F = (0, 612.5, 600)** se consigue simular este proceso y terminar con la creación de puntos y trayectorias, creando la última trayectoria **Path_Barra_6F:**

Ap_EnBarra_6→Ens_Barra_6→Ens_Barra_6F→Ap_EnBarra_6→Aprox_Roja.

Añadir entradas/salidas y simular el proceso

Ya se dispone de todas las trayectorias generadas y las señales definidas en la tabla de eventos. Para simular el proceso resta:

- Sincronizar todas las trayectorias al VC.
- Añadir todas las trayectorias a un proceso main que creamos en el módulo añadido, en el orden correcto.
- Editar el programa de RAPID con el ProgramMaker, para añadir las señales y esperas donde sea preciso.

Todo esto se realiza de manera semejante a como se realizó la simulación del Service Core, por lo que se puede consultar el apartado anterior en caso de duda.

Para incluir las sentencias Set, Reset y WaitTime necesarias hay que recordar que se lanza la aplicación ProgramMaker en el menú Program→Edit Program de RobotStudio, y que es conveniente incluir sentencias de espera antes y después de coger/dejar una pieza para el correcto funcionamiento del programa RAPID (para que las acciones de agarre y suelta no se realicen precipitadamente). También se puede editar directamente el archivo de programa (*.prg) o de módulo (*.mod) con cualquier editor de texto convencional, pero de una manera, por tanto, menos eficiente. Los archivos prg y mod se pueden generar o salvar desde el ProgramMaker. El programa completo se puede consultar en los anexos, en el capítulo VII.4.

Una vez incluidas las sentencias de señales y espera al programa, se puede pasar a simular el ensamblado de la estructura. Realizado esto y comprobado el correcto funcionamiento de este proceso se puede realizar la transferencia del programa al robot real. La siguiente figura muestra una secuencia de la simulación.

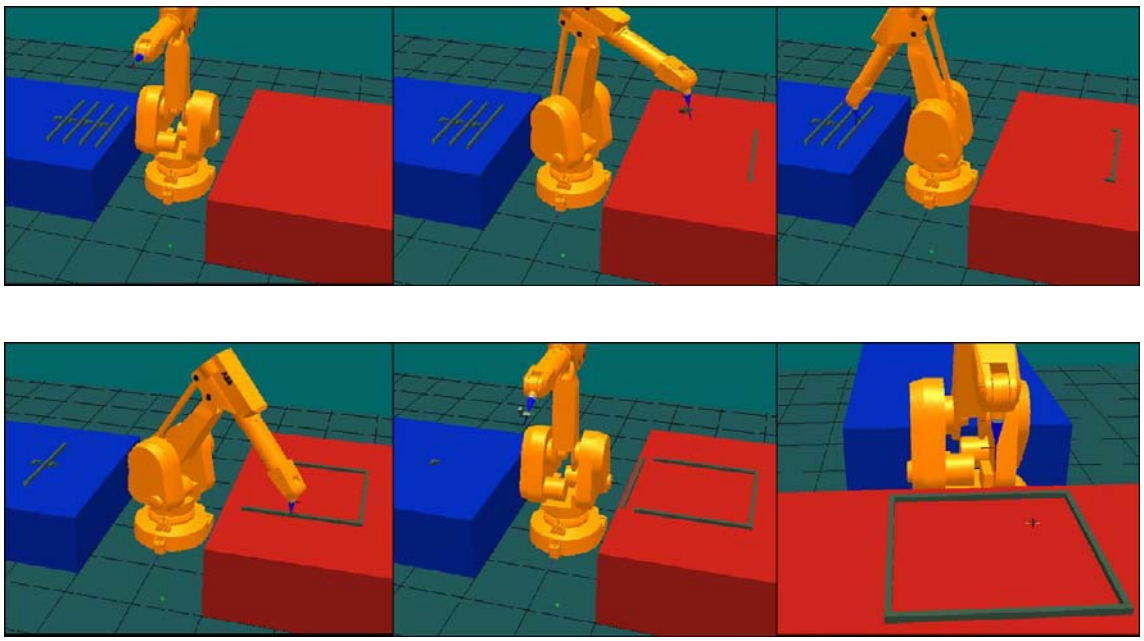


Figura IV.2.10 – Secuencia de la estación creada para transferir

03. Transferencia al robot real

Para realizar el paso del programa del software al robot real, hay que copiar el módulo o programa (archivo *.mod o *.prg) en un disco de 3 ½ para luego importarlo en el controlador real. Sin embargo, antes de importarlo se deberán hacer ciertas modificaciones, realizando un nuevo preestudio.

Se necesita cambiar el enfoque del proceso. El enfoque realizado en la simulación del apartado IV.2.02 no tenía en cuenta en profundidad el proceso real. En estos cabe esperar ineficiencias derivadas de las tolerancias de los materiales, de pequeños desajustes geométricos del robot (posiblemente debidos a una preinstalación de los elementos de la célula imprecisa, al desconocimiento de la misma, o al desgaste del robot a lo largo de los años de trabajo) o incluso de pequeñas derivaciones entre los procesos simulados y los reales, tales como un movimiento indeseado de una pieza una vez se suelta en la mesa o un desplazamiento de piezas durante un proceso real de ensamblado. Para ello se necesita la ayuda de ciertos elementos que sirvan de topes en el diseño real, para evitar algunas de estas irregularidades.

Los cambios que hay que introducir a la simulación anterior son básicamente dos: cambiar el orden de ensamblado 1-2 y el 6-7, para realizar el ensamblado del tubo en el racor y no al revés, y así poder restringir movimientos con el uso de los topes, que situaremos en el borde de los racores. Estos racores restringirán los grados de libertad necesarios para asegurarnos de que el proceso se ejecute con la suficiente precisión. El nuevo proceso a implementar se presenta en la **Figura IV.2.11.**

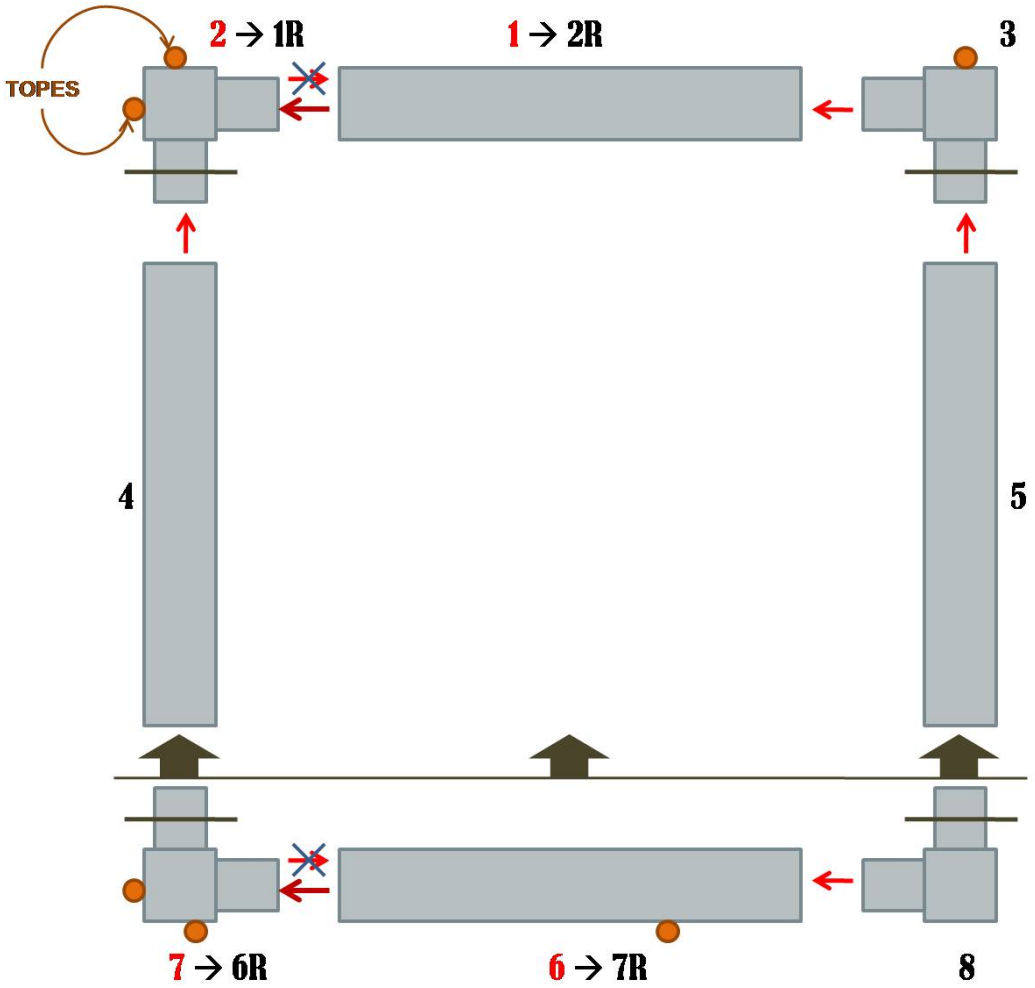


Figura IV.2.11 – Proceso productivo a implementar en el robot real

Como se observa, se ha denominado con R a los procesos que van a cambiar, y se necesitarán un total de 6 elementos para restringir los movimientos necesarios.

Para cambiar estos procesos, primero se procede a cambiar el nombre de los caminos asociados, es decir, se sustituye Path_Barra_1 por Path_Barra_2R, Path_Racor_2 por Path_Racor_1R, etc, para tener una visión general menos confusa. No parece necesario cambiar el nombre de los puntos, pero se puede hacer si se estima oportuno. Hecho esto, se procede a cambiar el orden en el proceso main: se sincronizan los cuatro nuevos

caminos al VC, añadiéndolos como nuevos, y en la pestaña de programa o bien en el ProgramMaker se cambia el main para que realice el nuevo proceso:

Path_Racor_1R;

Path_Barra_2R;

Path_Racor_3;

Path_Barra_4;

Path_Barra_5;

Path_Racor_6R;

Path_Barra_7R;

Path_Racor_8;

Path_Barra_6F.

Se procede ahora a estudiar los nuevos caminos. El racor 2 (camino 1R o racor 1R si se quiere) ahora no precisa de punto de aproximación horizontal, pero se puede sustituir por uno vertical (en Z). El punto de ensamblado sigue siendo **Ens_Racor_2 = Ens_Racor_1R = (-412.5, 1401, 600).** Es decir, se cambia **Ap_EnRacor_2 = (-500, 1401, 600)** por **Ap_EnRacor_1R = (-412.5, 1401, 700).** Se añade también después de soltar el racor. Es el único cambio necesario en este camino. Por tanto se tiene que Path_Racor_1R implementa la secuencia:

Paso_medio→Aprox_Azul→Ap_Racor_2→Pto_Racor_2→Ap_Racor_2→Aprox_Azul→Paso_medio→Aprox_Roja→**Ap_EnRacor_1R**→Ens_Racor_2→**Ap_EnRacor_1R** →Aprox_Roja.

En negrita se ha resaltado los únicos cambios necesarios. Se recorre con el robot para asegurar el correcto funcionamiento y se sincroniza al VC.

La barra o tubo 1, o si se quiere 2R, ahora necesita un punto de aproximación que hay que crear. El punto final de ensamblado de la barra es el mismo, **Ens_Barra_1 = (0, 1437.5, 600)**. Por tanto, para realizar el ensamblado en sentido negativo de X, cambiamos el anterior **Ap_EnBarra_1 = (0, 1437.5, 675)**, por **Ap_EnBarra_2R= (75, 1437.5, 600)**, siempre con la misma orientación. La trayectoria Path_Barra_2R queda entonces:

Paso_medio→Aprox_Azul→Ap_Barra_1→Pto_Barra_1→Ap_Barra_1→Aprox_Azul→Paso_medio→Aprox_Roja→**Ap_EnBarra_2R**→Ens_Barra_4→Aprox_Roja.

Se recorre con el robot mediante "Move Along Path" y se sincroniza al VC.

El siguiente camino a actualizar es el del racor 7, o si se quiere 6R. Ánalogamente a lo que se hizo con el racor 2/1R, hay que sustituir su punto de aproximación horizontal por uno vertical, ya que al ser ahora la primera pieza del conjunto 6-7-8 que se pone en la mesa, el movimiento no va a ser

horizontal. Además, el tope que se incluirá restringirá el movimiento horizontal. Como el punto de ensamblado es el mismo, por tanto se sustituye **Ap_EnRacor_7 = (-500, 576.5, 600),** por **Ap_EnRacor_6R = (-412.5, 576.5, 700).** Se añade también después de dejar la pieza. El camino **Path_Racor_6R** quedará como sigue:

Paso_medio→Aprox_Azul→Ap_Racor_7→Pto_Racor_7→Ap_Racor_7→Aprox_Azul→Paso_medio→Aprox_Roja→**Ap_EnRacor_6R**→Ens_Racor_7→**Ap_EnRacor_6R**→Aprox_Roja.

Se comprueba que el camino se puede recorrer y se sincroniza al VC.

Ya sólo falta actualizar el camino de la barra 6 ó 7R. El ensamblado de esta barra será en horizontal, y no se dejará en vertical como en el proceso anterior. Por tanto, ya que el punto final de ensamblado es el mismo, hay que cambiar **Ap_EnBarra_6 = (0, 540, 675)** por **Ap_EnBarra_7R = (75, 540, 600)**, para que el sentido del ensamblado sea el del eje X negativo. El camino **Path_Barra_7R** queda:

Paso_medio→Aprox_Azul→Ap_Barra_6→Pto_Barra_6→Ap_Barra_6→Aprox_Azul→Paso_medio→Aprox_Roja→**Ap_EnBarra_7R**→Ens_Barra_6→Aprox_Roja.

Se comprueba que se puede recorrer y se sincroniza al VC.

También hay que cambiar el Path_Barra_6F, puesto que incluía el punto Ap_EnBarra_6, y poner en su lugar el nuevo punto **Ap_EnBarra_6F = (0, 612.5, 675),** es decir, en la vertical del punto de ensamblado final del conjunto 6-7-8.

Por otro lado, se tiene que tener en cuenta que al empezar ahora el ensamblado del conjunto 6-7-8 por el racor 7/6R, hay que hacer que las otras dos piezas se fijen a él al ponerse en la mesa, en vez de a la barra 6/7R. Es decir, **cambiar la tabla de eventos** para que cuando do6 y do8 se hagan cero, se fijen los elementos al racor 7/6R (aunque en realidad el racor 8 se puede dejar como está, para fijarse al tubo 6/7R). Esto implica modificar el evento asociado a do7=0 que hace que se fije el racor 7/6R a la barra 6/7R, cambiando la señal y el orden de la acción. Además, como ahora es la barra 6/7R la que está unida al racor 7/6R y no al revés, el robot debe mover el conjunto 6-7-8 a través del racor 7/6R, es decir, hay que cambiar en la rutina Path_Barra_6F la señal do6 por do7.

Una vez realizado todo esto, se procede a realizar la simulación del proceso general para comprobar su correcto funcionamiento.

Para pasar el programa al robot real, se necesita también realizar las modificaciones concernientes a las salidas digitales. En la simulaciones se ha asociado un evento de agarre/suelte a una cierta entrada digital por pieza. En el caso del robot real, **la única señal** que se utiliza para fines de agarre/suelte **es la do1**. Esta señal controla la herramienta incluida en el robot, manejándola mediante un sistema de aire a presión que se abre o cierra según el estado de la señal. Por tanto, se tienen que cambiar todas las

señales presentes en el programa de RAPID por la señal do1. Hay que fijarse con atención en el sentido del circuito de aire, de lo que depende si el cerrar la herramienta corresponderá a activar o desactivar la señal.

Hecho esto, ya se dispone de un primer programa base que se puede grabar como módulo o programa en un disco de 3 ½ e importarlo al robot real, que se usará más adelante.

Implementación en el robot real

Este programa debe sufrir modificaciones derivadas de las especificaciones y características de los elementos y procesos de la estación real que se implementa en el laboratorio. Esta estación real cuenta con los siguientes elementos:

- Robot ABB IRB_2400_M94A con controlador S4.
- Sistema de barras de aluminio que sirven de bases (para las mesas).
- Planchas de madera sirviendo de superficie de ensamblado.
- Racores plásticos con capacidad de fácil ensamblado, similares a los Quick Frame reseñados en el capítulo II, y de las dimensiones simuladas en este capítulo IV.2.
- Barras metálicas con huecos interiores.
- Herramienta estándar para procesos de agarre/suelte.
- Escuadras metálicas que sirven de sujeción, tope o guía para el ensamblado de los elementos.

Hay que reajustar la definición de la herramienta, que difiere de los parámetros que se introdujeron en la simulación. Se mide la herramienta y se comprueba que su TCP está en (0, -24, 220), lo que se introduce en el programa.

El siguiente paso es comprobar que el robot recorre correctamente todo el programa. Para ello, se inserta el disco en el controlador, y se importa a través del mando de programación (Teach Pendant real) con el comando File→Open...→. Para ejecutar el proceso, se utiliza el menú Program/Test.

Modificaciones de los movimientos

También se introducen en el programa definitivo realizado por el robot real ciertas modificaciones de las sentencias de movimiento que no se tuvieron en cuenta en la creación de las simulaciones, pero que son comúnes en procesos reales. En resumen, se trata de:

- Cambiar la zona de aproximación a z50 para puntos intermedios (no de ensamblado ni de aproximación a ensamblado).
- Cambiar la velocidad a v200 para puntos de aproximación a ensamblado.

- Cambiar el movimiento a MoveL y la velocidad a v20 para ensamblados.

Aparte de estas modificaciones, se añaden algunos puntos adicionales al programa, que sirven de aproximación al ensamblado de una pieza y/o para la maniobra de vuelta al punto de aproximación general de la mesa, para conseguir que los movimientos del robot sean correctos y más precisos. Estos puntos son:

Ap1_EnRacor_3

Ap1_EnBarra_2R

Ap1_EnBarra_4

Ap1_EnBarra_5

Ap1_EnBarra_7R.

Problemas del proceso real

Durante esta fase, se observa que la célula presenta problemas a la hora de realizar los giros que se habían simulado correctamente. Esto parece deberse a que el robot hace uso de configuraciones de puntos (que designan en que cuadrante se debe situar el TCP) mientras que en el programa introducido no se hace uso de las mismas (se usan las instrucciones ConfL/OFF y ConfJ/OFF, y se usa la configuración [0, 0, 0, 0] para todos los puntos). También podría deberse a pequeñas derivaciones entre el controlador del software RobotStudio y el real. Para solucionar esta cuestión, se introducen cambios en las orientaciones de los puntos.

Otra cuestión a solucionar se refiere al posicionamiento del SDC del robot. El robot no está apoyado en el suelo, sino en una plancha metálica a la que se fija también el sistema óptico de seguridad. Esta base no se ha tenido en cuenta en la simulación, ya que esta se basa en los pasos dados en la simulación del apartado IV.1, que usa simplemente un robot como ejemplo de estación. Así pues, la altura 0 del robot no se encuentra a nivel del suelo, sino más arriba. Para resolver este problema se hace llegar al robot a la altura de las mesas para fijar la altura de los puntos en el programa, realizando las pertinentes modificaciones del mismo. Además, para tener referencias de los puntos de la mesa en el SDC del robot real, se añaden las rutinas PosRoja y PosAzul, que llevan la herramienta a distintos puntos de las mesas. Estos puntos sirven como referencia para la colocación de las mismas. Como añadidura, se realizan marcas en las patas de la mesa para coger su referencia respecto del suelo, y se fijan las tablas a la estructura de aluminio. También se usa una plantilla a escala real de las posiciones de las piezas en la alimentación, para ayudar a posicionar las mismas.

Además, del preestudio plasmado en la **Figura IV.2.11**, se decide finalmente prescindir del tope más cercano al robot (barra 7R), y además de añadir guías para ayudar a lo procesos de ensamblado de ciertas barras (2 y 4).

El programa definitivo transferido al robot real se puede consultar en los anexos, capítulo VII.4.

Imágenes del proceso real

Se muestra a continuación una secuencia del video del proceso real implementado en el laboratorio de la UC3M.

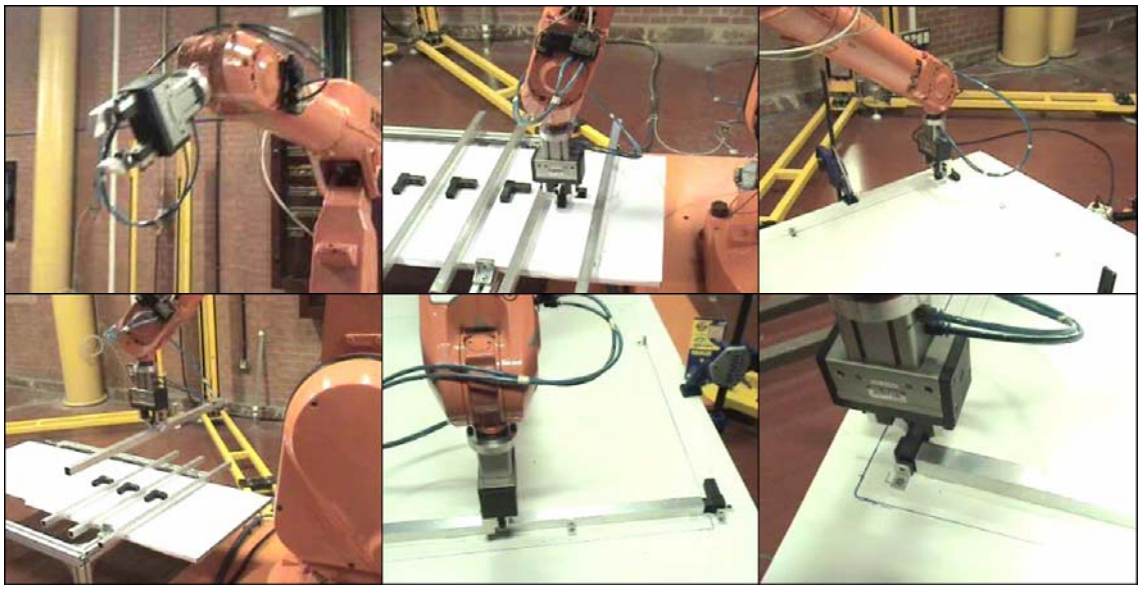
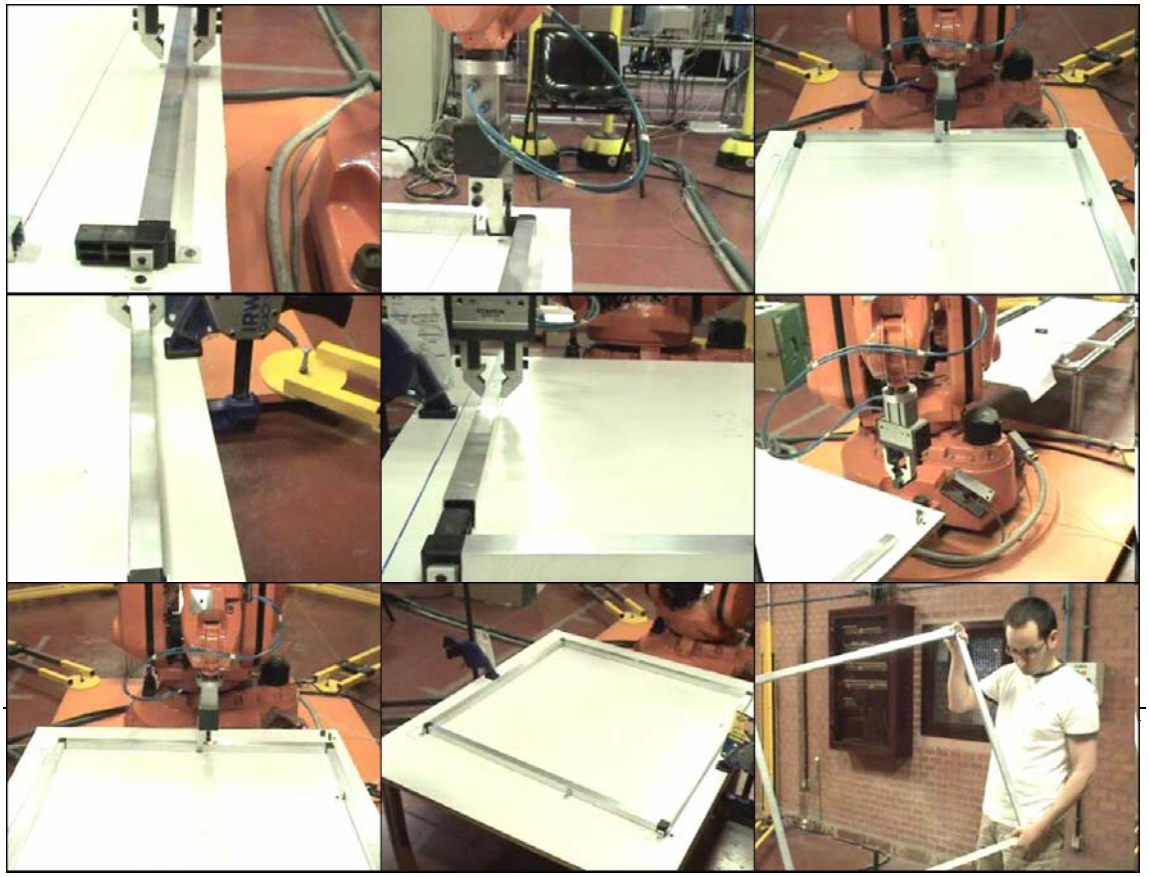


Figura IV.2.12 – Secuencia del proceso real implementado en el laboratorio



V - CONCLUSIONES Y TRABAJOS FUTUROS

Conclusiones

A lo largo del proyecto, a través de la creación de la estaciones a simular, se ha presentado, utilizado y explicado en extensión el software RobotStudio. Para la generación de estas simulaciones, se ha seguido una metodología presentada en el proyecto. Además, dichas simulaciones sirven para mostrar la filosofía de funcionamiento de la célula robótica de la Mobile Factory. Todo este conocimiento generado y los distintos estudios realizados sirven de apoyo para el futuro desarrollo dentro del proyecto, con lo cual se han conseguido cumplir los tres propósitos enunciados en la introducción de este documento, que eran: aprender a usar el software, mostrar el funcionamiento del proceso y sentar bases para futuros desarrollos.

Además, gracias al trabajo realizado en el proyecto, se han conseguido poner de manifiesto las ventajas de la simulación de procesos productivos robotizados, como la fácil inclusión y evaluación a priori de distintas opciones de diseño de la célula y el acorte de plazos de la generación de programas. A pesar de estas ventajas, se ha comprobado que la implementación en un robot real siempre presenta dificultades que precisan de un tiempo de resolución, y requiere un estudio detallado a priori que no se ha realizado tan minuciosamente en el proyecto (que está centrado en el aprendizaje de la herramienta e identificación de las futuras líneas de trabajo), para evitar los problemas derivados de las características, tolerancias, etc, de los elementos físicos reales. Incluso dentro de este estudio se podría incluir el diseño del propio sistema robótico, lo que llevaría a un conocimiento y diseño del robot más profundos, permitiendo evitar o suavizar futuros problemas previstos.

Trabajos futuros

En la línea de las conclusiones obtenidas en el proyecto, se plantean futuros desarrollos dentro del diseño de la Mobile Factory. A tener en cuenta especialmente en estos futuros desarrollos son las siguientes ideas:

- Usar la metodología presentada en este proyecto para generar los procesos productivos y el conocimiento del software adquirido.
- Especificar y diseñar con más detalle los elementos (herramientas, materiales, etc) a utilizar para desarrollar una demostración fiel de los procesos a realizar por la Mobile Factory. Realizando un diseño más preciso de los elementos se podrán evitar o aligerar futuros problemas en la implementación real.
- Existe la posibilidad de utilizar versiones del software más modernas, partiendo del conocimiento y los desarrollos de la actual versión, que aporten más funcionalidades. Por tanto se podría trabajar en la exportación de modelos de la versión actual (3.0) a otras más modernas, estudiando qué cambios serían necesarios y qué conceptos de los presentados en este proyecto

permanecerían o serían útiles. Incluso se podría complementar con la utilización de robots y herramientas más actuales, aportando también mejores prestaciones.

- Estudiar, utilizar y mejorar las soluciones presentadas a las dificultades a la hora de la implementación real del proceso (uso de sujeciones, topes, anclajes, etc).

VI - BIBLIOGRAFÍA Y REFERENCIAS

VI.1 - REFERENCIAS

VI.2 - BIBLIOGRAFÍA

VI.1 - REFERENCIAS

[1] United nations economic commission for europe un/ece news, Press Release ECE/STAT/01/01 Geneva, 13 February 2001 (http://www.unece.org/press/pr2001/01stat01e.htm)
[2] http://www.abb.com/cawp/seitp202/0069fbc7eae5a8d4852573920047e334.aspx, Accedido en Febrero 2008.
[3] http://www.workspace5.com, Accedido en Febrero 2008.
[4] http://www.newtonium.com, Accedido en Marzo 2008.
[5] http://www.inser-robotica.com/paletizadosoftware.php, Accedido en Febrero 2008.
[6] http://www.kuka.com/spain/es/products/software/kuka_sim/start.htm, Accedido en Febrero 2008.
[7] http://www.abb.es/product/seitp327/df90f6fe2c1ffc64c125725100252d4d.aspx, Accedido en Febrero 2008.
[8] http://www.abb.com/product/seitp327/5d8c9c5cfbf82509c12571770026a235.aspx?productLanguage=es&country=ES. Accedido en Enero 2008.

VI.2 - BIBLIOGRAFÍA

Proyecto ManuBuild, 6FP.

Proyecto Fin de Carrera "Simulación del modelo de negocio de la construcción en España desde el punto de vista del promotor privado", Carlos Bárcena, Marzo 2006,

Página web de la Asociación Europea de Tuberías y Accesorios Plásticos:

http://www.teppfa.org/Plastic-pipe-materials-a-fast-guide.htm

http://www.teppfa.com/Fast-Guide-to-Materials.asp

Prácticas de Automatización industrial – Universidad Carlos III.

Ayuda de RobotStudio 3.0.

Página web de abb: http://www.abb.es

Hoja de características del motor:

http://library.abb.com/GLOBAL/SCOT/SCOT241.nsf/VerityDisplay/1E1AAB65AACAC353C1256C8E0034741A/$File/Rotary%20unit.pdf

Hoja de características del robot:

http://library.abb.com/GLOBAL/SCOT/scot230.nsf/VerityDisplay/C768E5033C6206898525706200695A8C/$File/IRB2400_R3_US%2002_05.pdf

ABB Flexible Automation AB Rapid Referente on-line manual

VII - ANEXOS

VII.1 - GLOSARIO

VII.2 - ÍNDICE DE FIGURAS

VII.3 - LISTA DE ACCESOS DIRECTOS EN RS

VII.4 - PROGRAMAS RAPID

VII.5 - PROGRAMA DE VBA

VII.6 - TABLAS DE EVENTOS

VII.7 - LISTA DE SEÑALES

VII.8 - HOJAS DE CARACTERÍSTICAS

VII.1 - GLOSARIO

- ACS: Agua Caliente Sanitaria.
- CAD: Diseño asistido por ordenador (Computer Aided Design).
- CAE: Ingeniería asistida por ordenador (Computer Aided Engineering).
- CAM: Fabricación asistida por ordenador (Computer Aided Manufacturing).
- DFMA: Diseño para la fabricación y el montaje (Design for Manufacturing and Assembly).
- EC: Comisión Europea (European Comission).
- ERP: Sistemas de planificación de recursos (Enterprise Resource Planning).
- JIT: Producción justo a tiempo. (JIT)
- ICT: Tecnologías de Información y Comunicación.
- IFR: Federación Internacional de Robótica (International Federation of Robotics).
- MB: Proyecto ManuBuild.
- PEX: Polietileno con enlaces cruzados.
- PP: Polipropileno.
- PVC: PoliVinilo de Carbono.
- RAPID: Lenguaje de programación de robots de ABB.
- SDC: Sistema de coordenadas.
- ST: Objetivo científico-técnico (scientific-technical) del proyecto MB.
- TCP: Tool Center Point, punto de referencia de la herramienta.
- Teach Pendant/Flex Pendant: Unidad de programación del robot.
- UCS: Sistema de coordenadas del usuario (User Coordinate System).
- UN/ECE: Comisión Económica de las Naciones Unidas para Europa.
- VBA: Visual Basic para Aplicaciones.
- VC: Controlador virtual (Virtual Controller).

VII.2 - ÍNDICE DE FIGURAS

Figura II.2.1– Lista de participantes en el proyecto ManuBuild 11
Figura II.2.2– Enfoque del proyecto ManuBuild 12
Figura II.2.3– Transformación del modelo de la construcción 13
Figura II.3.1- Diseño de la célula de la factoría móvil 18
Figura II.4.1- Vista general de un Service Core completo 20
Figura II.4.2– Modelo del bastidor, diseño preliminar en 3D y modelo real del Service Core 22
Figura II.4.3– Algunas dimensiones aproximadas del Service Core 22
Figura II.4.4– Perfiles Quick Frame de 80/20 Inc 23
Figura II.4.5– Esquinas Quick Frame 23
Figura II.4.6– Facilidad de conexión de piezas Quick Frame 24
Figura II.4.7– Posible método de fijación 26
Figura II.5.1– Cifras de variación de la inversión en robótica 98/00 29
Figura II.5.2– Precios (en relación al año 1990) de robots industriales 1990/2000 (con y sin ajuste relativo al rendimiento) 30
Figura II.5.3– Costes de mano de obra y precio de los robots en el periodo 1990-2000 31
Figura II.5.4– Componentes de un robot industrial 33
Figura II.5.5– Grados de libertad del IRB2400 33
Figura II.5.6– Controlador del robot IRB2400 34
Figura II.5.7– Panel de control del controlador 35
Figura II.5.8– Unidad de programación 35
Figura II.5.9– Detalle de la unidad de programación 36
Figura III.1- Imagen del programa workspace5 con robot KUKA 40
Figura III.2– Imagen del programa RoboWorks 42
Figura III.3– Programa PC Roset para robots Kawasaki 44
Figura III.4– Programa DTPS II para robots Panasonic 45
Figura III.5– Imagen de KUKA.Sim Pro 46
Figura III.6– Imagen de KUKA.Sim Layout 47
Figura III.7– Imagen de KUKA.Sim Viewer 47
Figura III.8– Imágenes de KUKA.OfficeLite 48
Figura III.9– Imagen de RobotStudio 5 50
Figura IV.1.1– Entorno de usuario 54

Figura IV.1.2– Áreas del entorno gráfico 55
Figura IV.1.3– Fondo de la ventana gráfica cambiado 56
Figura IV.1.4– Barras Ver, Nivel de Selección y Modo de Ajuste 58
Figura IV.1.5- Flujograma del proceso de creación y simulación de una célula 59
Figura IV.1.7– Base del robot seleccionada y vista en navegador 62
Figura IV.1.8– Ejes de la muñeca 63
Figura IV.1.9– Las dos maneras de crear un objeto 3D 64
Figura IV.1.10– Mesa de alimentación reposicionada 65
Figura IV.1.11 – Kinematic Modeler (con Links añadidos) 67
Figura IV.1.12– Piezas del motor unidas 68
Figura IV.1.13- Esquema de la estructura metálica 70
Figura IV.1.14- Esquema de las tuberías 71
Figura IV.1.15– Orientación de la barra importada 72
Figura IV.1.16– Cota de la anchura de la barra en SolidWorks 73
Figura IV.1.17– Barra 1 posicionada en la mesa 74
Figura IV.1.18– Barras y tubos posicionados en la mesa 75
Figura IV.1.19- Crear herramienta y elegir TCP 76
Figura IV.1.20– Vista de la herramienta en el punto Pto_Barra_1 77
Figura IV.1.21- Dimensiones relevantes de la estructura metálica 78
Figura IV.1.22– Path_Barra_1 80
Figura IV.1.23- Ventana de configuración del controlador 81
Figura IV.1.24- Navegador de Robots Virtuales 82
Figura IV.1.25- Captura del "Teach Pendant" Virtual 83
Figura IV.1.26- Tabla de eventos para la barra 1 85
Figura IV.1.27- ProgramMaker con Path_Barra_1 86
Figura IV.1.28- Macro para mover la mesa 90º 89
Figura IV.1.29- Macro fijar Barra 1 a la Mesa 91
Figura IV.1.30- Macro para fijar el resto de piezas 93
Figura IV.1.31- Visión final de la estación 104
Figura IV.1.32 – Secuencia de la simulación del ensamblado 105
Figura IV.2.1- Layout de la demostración 106
Figura IV.2.2– Crear herramienta con una "dummy part" 107
Figura IV.2.3– Cotas del conector de dos brazos en SW 108
Figura IV.2.4- Dimensiones generales (mm) a tener en cuenta para el ensamblado 109

Figura IV.2.5 – Proceso productivo de la demostración física 110
Figura IV.2.6 – Posicionamiento de las piezas en alimentación 111
Figura IV.2.7– Conectores y tubos situados en la mesa 112
Figura IV.2.8 – Lado de la estructura 114
Figura IV.2.9– Path_Barra_1 de la demostración 115
Figura IV.2.10 – Secuencia de la estación creada para transferir 120
Figura IV.2.11 – Proceso productivo a implementar en el robot real 121
Figura IV.2.12 – Secuencia del proceso real implementado en el laboratorio 126

VII.3 - LISTA DE ACCESOS DIRECTOS EN ROBOTSTUDIO

Ayuda

Para	Pulse
Ver la ayuda de RobotStudio	F1
Ver la ayuda del API de RS	Alt + F1

Archivos

Para	Pulse
Crear una estación nueva	Ctrl + N
Abrir una estación existente	Ctrl + O
Cerrar la estación actual	Ctrl + F4
Salvar la estación actual	Ctrl + S
Importar una librería	Ctrl + M
Importar una geometría	Ctrl + G

Comandos de Edición

Para	Pulse
Deshacer el último comando	Ctrl + Z (Repetir para deshacer más acciones)
Repetir el último comando (Redo)	Ctrl + R (Repetir para rehacer más acciones)

Cortar la selección	Ctrl + X
Copiar la selección	Ctrl + C
Pegar desde el portapapeles	Ctrl + V
Borrar la selección	Supr

Configuraciones y Selección

Para	**Pulse**
Ver la ventana de diálogo del Modo de Ajuste	F2
Cambiar el Nivel de selección	F3 (pulse F3 de nuevo para cambiar al siguiente)
Establecer como UCS	Ctrl + U
Establecer el color del objeto seleccionado	Ctrl + Alt + C

Vistas y Navegación

Para	**Pulse**
Comandos de zoom:	
Zoom in (en la ventana gráfica)	Ctrl + Av Pag
Zoom out (en la ventana gráfica)	Ctrl + Re Pag

Comandos de vista:	
Ver todo (en la ventana gráfica)	Alt + 5
Ver el centro (en la ventana gráfica)	Alt + 6
Comandos de navegación:	
Seleccionar Tipo de Navegacion "TrackBall" (en la ventana gráfica)	Ctrl + B
Seleccionar Tipo de Navegacion "Walkthrough" (en la ventana gráfica)	Ctrl + W
Distribución del espacio de trabajo:	
Ver o ocultar el Navegador	Alt + 1 (repetir para rehacer)
Ver o ocultar la Ventana de salidas	Alt + 2 (repetir para rehacer)
Ver o ocultar la Ventana de propiedades	Alt + 3 (repetir para rehacer)
Ver o ocultar la Barra de estado	Alt + 4 (repetir para rehacer)
Resetear el espacio de trabajo a la vista predeterminada	Ctrl + Alt + R

Caminos y Puntos

Para	**Pulse**
Crear un punto nuevo (introduciendo sus coordenadas)	May + T
Crear un punto rápido o Teach Target (en la	May +

posición del TCP)	G
Crear un nuevo camino en el navegador	May + H

Modificar posiciones

Para	**Pulse**
Modificar el lugar del objeto seleccionado (respecto a otros objetos)	May + L
Modificar la posición del objeto seleccionado (tecleándola)	May + P
Rotar el objeto seleccionado	May + R
Copiar la orientación del objeto seleccionado	May + C
Aplicar orientación al objeto seleccionado	May + A
Alinear la orientación del punto(s) seleccionado(s)	Ctrl + Alt + O
Establecer la orientación del punto normal a la superficie	Ctrl + Alt + N
Editar el valor de los ejes externos del punto(s) seleccionado(s)	Ctrl + Alt + E

Instrucciones

Para	**Pulse**
Editar una instrucción	Ctrl + Alt + I
Insertar una instrucción de acción	Ctrl + Alt + A

Controlador Virtual

Para	Pulse
Abrir la ventana de configuración del VC	Ctrl + 4
Activar el controlador virtual	Ctrl + 5
Apagar el controlador virtual	Ctrl + 6

Movimientos del Robot y Simulación

Para	Pulse
Ver la ventana de estado del mecanismo seleccionado	F4
Ver la Unidad de Programación (Teach Pendant)	F5
Ver el simulador de E/S	Ctrl + I
Mover el robot al punto seleccionado	Ctrl + T
Mover el robot por el camino seleccionado	Ctrl + A
Llevar el robot directamente al punto seleccionado	Ctrl + J
Mover el robot a la posición de inicio	Inicio
Empezar una simulación (Play)	Ctrl + F5
Parar una simulación	May + F5

Programas

Para	Pulse
Editar Programa (inicia la aplicación ProgramMaker)	Ctrl + E
Sincronizar el programa al controlador virtual	Tab
Sincronizar el programa a la estación	May + Tab

Herramientas (medidas, macros y editores)

Para	Pulse
Activar la function "medir punto a punto"	Ctrl + Q (Repetir para desactivarla)
Crear una nueva macro	Alt + F8
Iniciar el Editor de Visual Basic	Alt + F11 (Repetir para cambiar de aplicación)

VII.4 - PROGRAMAS RAPID

01. Programa Rapid del ensamblado del SC

Éste es el programa de RAPID completo de la estación que simula un proceso de ensamblado del Service Core (capítulo IV.1)

```
MODULE Module1
  !Declaraciones
  PERS tooldata tTool_TCP:=[TRUE,[[-0.819937,55.1,261.04],
[1,0,0,0]],[0.5,[0,25,150],[1,0,0,0],0,0,0]];
  CONST robtarget Paso_Medio:=[[1000,0,1200],[0,-0.707107,0.707107,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Alim:=[[200,-750,1200],[0,0,1,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_1:=[[0,-500,570],[0,0,1,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_1:=[[0,-500,520],[0,0,1,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Ens:=[[0,600,1200],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  PERS wobjdata Salida:=[FALSE,TRUE,"",[[0,2250,0],
[1,0,0,0]],[[0,0,625],[1,0,0,0]]];
  CONST robtarget Ap_EnBarra_1:=[[0,-1240,75],[0,1,0,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_1:=[[0,-1240,20],[0,1,0,0],[0,0,0,0],
[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_2:=[[0,-550,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_2:=[[0,-550,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_2:=[[620,-1140,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_2:=[[620,-1140,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_I:=[[70.7109,-1682.91,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_I:=[[70.71,-1682.91,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_3:=[[0,-600,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_3:=[[0,-600,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra3:=[[-620,-1140,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra3:=[[-620,-1140,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra4:=[[-1000,-650,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra4:=[[-1000,-650,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_4:=[[0,-1000,74.9999],[0,-0.707107,
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_4:=[[0,-1000,20],[0,-0.707107,0.707107,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
```

```
  CONST robtarget Ap_Union_II:=[[0,-1140,75],[0,0.92388,0.382683,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_II:=[[0,-1140,20],[0,0.92388,0.382683,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_5:=[[-6.10292E-005,-700,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_5:=[[-6.10292E-005,-700,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_III:=[[-70.7109,-1682.91,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_III:=[[-70.7108,-1682.91,20.0001],
[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_6:=[[-510,-750,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_6:=[[-510,-750,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_6:=[[-340,-1130,75],[0,-0.707107,
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_6:=[[-340,-1130,20],[0,-0.707107,
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_IV:=[[-340,-1240,75],[0,0.92388,-0.382683,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_IV:=[[-340,-1240,20],[0,0.92388,-0.382683,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_7:=[[-6.01992E-005,-800,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_7:=[[3.01955E-005,-800,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_7:=[[620,-1055,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_7:=[[620,-1055,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_V:=[[130.815,-1622.81,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_V:=[[130.815,-1622.81,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_8:=[[0,-850,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_8:=[[0,-850,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_8:=[[-620,-1055,75.0001],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_8:=[[-620,-1055,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_VI:=[[0,-1055,75.0001],[0,0.92388,
0.382683,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto1_Union_VI:=[[0,-1055,20],[0,0.92388,0.382683,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto2_Union_VI:=[[0,-1055,20],[0,0.92388,-0.382683,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_VII:=[[-130.815,-1622.81,75],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_VII:=[[-130.815,-1622.81,20],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_9:=[[-510,-900,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Barra_9:=[[-510,-900,520],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_9:=[[340,-1130,75],[0,-0.707107,
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
```

```
  CONST robtarget Ens_Barra_9:=[[340,-1130,20],[0,-0.707107,
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Union_VIII:=[[340,-1240,75],[0,0.92388,-0.382683,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Union_VIII:=[[340,-1240,20],[0,0.92388,-
0.382683, 0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Tubo_1:=[[0,-950,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Tubo_1:=[[-4.36644E-005,-950,525.25],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnTubo_1:=[[0,-1140,125],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Tubo_1:=[[0,-1140,44.9999],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Tubo_2:=[[-610,-1000,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Tubo_2:=[[-610,-1000,516.25],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnTubo_2:=[[463,-940,125],[0,-0.707107,0.707107,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Tubo_2:=[[463,-940,36],[0,-0.707107,0.707107,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Tubo_3:=[[-850,-1050,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Tubo_3:=[[-850,-1050,516.25],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnTubo_3:=[[-463,-940,125],[0,0.766046,-0.642786,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Tubo_4:=[[0,-1100,570],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Tubo_4:=[[5.88315E-005,-1100,516.25],[0,0,1,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnTubo_4:=[[455,-958,375],[0,0.707107,
0,0.707107],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Tubo_4:=[[455,-958,311.5],[0,0.707107,
0,0.707107],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Tubo_3:=[[-463,-940,36],[0,-0.707107,0.707107,
0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnTubo_41:=[[200,-958,475],[0,1,0,0],
[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];

  PROC Path_Tubo_4()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Tubo_4,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Tubo_4,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
    Set do24;
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Tubo_4,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnTubo_4,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Tubo_4,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
```

```
    Reset do24;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnTubo_4,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Tubo_3()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Tubo_3,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Tubo_3,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
    Set do23;
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Tubo_3,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnTubo_3,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Auxtubo3,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Tubo_3,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do23;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnTubo_3,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Tubo_2()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Tubo_2,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Tubo_2,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
    Set do22;
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Tubo_2,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnTubo_2,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Tubo_2,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do22;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnTubo_2,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC
```

```
PROC Path_Tubo_1()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Tubo_1,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Tubo_1,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do21;
  !coger la pieza
  WaitTime 2;
  MoveJ Ap_Tubo_1,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_EnTubo_1,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ens_Tubo_1,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  Reset do21;
  !dejar la pieza
  WaitTime 2;
  MoveJ Ap_EnTubo_1,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Union_VIII()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_VIII,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto_Union_VIII,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_VIII,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_9()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_9,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_9,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do19;
  !coger la pieza
  WaitTime 2;
  MoveJ Ap_Barra_9,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_9,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ens_Barra_9,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  Reset do19;
  !dejar la pieza
  WaitTime 2;
  MoveJ Ap_EnBarra_9,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
```

```
ENDPROC

PROC Path_Union_VII()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_VII,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto_Union_VII,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_VII,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Union_VI()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_VI,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto1_Union_VI,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_VI,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto2_Union_VI,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_VI,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_8()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_8,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_8,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do18;
  !coger la pieza
  WaitTime 2;
  MoveJ Ap_Barra_8,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_8,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ens_Barra_8,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  Reset do18;
  !dejar la pieza
  WaitTime 2;
  MoveJ Ap_EnBarra_8,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Union_V()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_V,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto_Union_V,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_V,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC
```

```
PROC Path_Barra_7()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_7,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_7,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do17;
  !coger la pieza
  WaitTime 2;
  MoveJ Ap_Barra_7,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_7,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ens_Barra_7,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  Reset do17;
  !dejar la pieza
  WaitTime 2;
  MoveJ Ap_EnBarra_7,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Union_IV()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_IV,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto_Union_IV,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_IV,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_6()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_6,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_6,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do16;
  !coger la pieza
  WaitTime 2;
  MoveJ Ap_Barra_6,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_6,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ens_Barra_6,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  Reset do16;
  !dejar la pieza
  WaitTime 2;
  MoveJ Ap_EnBarra_6,v1000,fine,tTool_TCP\WObj:=Salida;
```

```
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Union_III()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_III,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto_Union_III,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_III,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_5()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_5,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_5,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do15;
  !coger la pieza
  WaitTime 2;
  MoveJ Ap_Barra_5,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_1,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ens_Barra_1,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  Reset do15;
  !dejar la pieza
  WaitTime 2;
  MoveJ Ap_EnBarra_1,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Union_II()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Ap_Union_II,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Pto_Union_II,v1000,fine,tTool_TCP\WObj:=Salida;
  WaitTime 2;
  MoveJ Ap_Union_II,v1000,fine,tTool_TCP\WObj:=Salida;
  MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_4()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Ap_Barra4,v1000,fine,tTool_TCP\WObj:=wobj0;
  MoveJ Pto_Barra4,v1000,fine,tTool_TCP\WObj:=wobj0;
  !llegada al punto
  WaitTime 2;
  Set do14;
  !coger la pieza
```

```
    WaitTime 2;
    MoveJ Ap_Barra4,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra_4,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Barra_4,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do14;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnBarra_4,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_3()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_3,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_3,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
    Set do13;
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Barra_3,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra3,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Barra3,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do13;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnBarra3,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Union_I()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Ap_Union_I,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Pto_Union_I,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    MoveJ Ap_Union_I,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_2()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_2,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_2,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
```

```
    Set do12;
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Barra_2,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra_2,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Barra_2,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do12;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnBarra_2,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_1()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_1,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_1,v1000,fine,tTool_TCP\WObj:=wobj0;
    !llegada al punto
    WaitTime 2;
    Set do1;
    !coger la pieza
    WaitTime 2;
    MoveJ Ap_Barra_1,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Alim,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Paso_Medio,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra_1,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ens_Barra_1,v1000,fine,tTool_TCP\WObj:=Salida;
    WaitTime 2;
    Reset do1;
    !dejar la pieza
    WaitTime 2;
    MoveJ Ap_EnBarra_1,v1000,fine,tTool_TCP\WObj:=Salida;
    MoveJ Ap_Ens,v1000,fine,tTool_TCP\WObj:=wobj0;
  ENDPROC


  PROC main()
    Path_Barra_1;
    Set do26;
    WaitTime 1;
    Reset do26;
    !girar la Mesa 90º
    Path_Barra_2;
    Set do27;
    WaitTime 1;
    Reset do27;
    !girar la Mesa -45º
    Path_Union_I;
    Set do28;
    WaitTime 1;
    Reset do28;
    !girar la Mesa 45º
```

```
Path_Barra_3;
Path_Barra_4;
Path_Union_II;
Set do26;
WaitTime 1;
Reset do26;
!girar la Mesa 90º
Path_Barra_5;
Set do27;
WaitTime 1;
Reset do27;
!girar la Mesa -45º
Path_Union_III;
Set do28;
WaitTime 1;
Reset do28;
!girar la Mesa 45º
Path_Barra_6;
Path_Union_IV;
Set do26;
WaitTime 1;
Reset do26;
!girar la Mesa 90º
Path_Barra_7;
Set do27;
WaitTime 1;
Reset do27;
!girar la Mesa -45º
Path_Union_V;
Set do28;
WaitTime 1;
Reset do28;
!girar la Mesa 45º
Path_Barra_8;
Path_Union_VI;
Set do28;
WaitTime 1;
Reset do28;
!girar la Mesa 45º
Path_Union_VII;
Set do28;
WaitTime 1;
Reset do28;
!girar la Mesa 45º
Path_Barra_9;
Path_Union_VIII;
Set do26;
WaitTime 1;
Reset do26;
!girar la Mesa 90º
Path_Tubo_1;
Path_Tubo_2;
Path_Tubo_3;
Set do26;
WaitTime 1;
Reset do26;
WaitTime 1;
Set do26;
WaitTime 1;
Reset do26;
```

```
    !girar la Mesa 180º
    Path_Tubo_4;
  ENDPROC
ENDMODULE
```

02. Programa Rapid de la estación creada para transferir al robot real

Éste es el programa de RAPID completo de la estación que se crea para realizar la transición al robot real (capítulo IV.2 sección 02, NO es el programa definitivo transferido al robot):

```
MODULE Module1
  PERS tooldata tPinza_TCP:=[TRUE,[[2.80493E-
005,0,100],[1,0,0,0]],[1,[0,0,35],[1,0,0,0],0,0,0]];
  CONST robtarget
Paso_medio:=[[1000,0,1500],[0.5,0,0.866025,0],[0,0,0,0],[9E+009,9E+009
,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Aprox_Azul:=[[0,-
700,900],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Ap_Barra_1:=[[0,-
600,575],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Pto_Barra_1:=[[0,-
600,525],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget
Aprox_Roja:=[[0,1000,975],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E
+009,9E+009,9E+009]];
  CONST robtarget
Ap_EnBarra_1:=[[0,1437.5,675],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+00
9,9E+009,9E+009,9E+009]];
  CONST robtarget
Ens_Barra_1:=[[0,1437.5,600],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009
,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Racor_2:=[[36.5,-
700,575],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Pto_Racor_2:=[[36.5,-
700,525],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Ap_EnRacor_2:=[[-
500,1401,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,
9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Racor_2:=[[-
412.5,1401,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+00
9,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Racor_3:=[[0,-813.5,575],[0,-
0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Pto_Racor_3:=[[0,-813.5,525],[0,-
0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
```

```
  CONST robtarget
Ap_EnRacor_3:=[[500,1401,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+00
9,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget
Ens_Racor_3:=[[412.5,1401,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+0
09,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_4:=[[0,-
750,575],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Pto_Barra_4:=[[0,-
750,525],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Ap_EnBarra_4:=[[-
412.5,950,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009
,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_4:=[[-
412.5,1025,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+00
9,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_5:=[[0,-
900,575],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget Pto_Barra_5:=[[0,-
900,525],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];
  CONST robtarget
Ap_EnBarra_5:=[[412.5,950,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+0
09,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget
Ens_Barra_5:=[[412.5,1025,600],[0,0.707107,0.707107,0],[0,0,0,0],[9E+0
09,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_6:=[[0,-
1050,575],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+0
09]];
  CONST robtarget Pto_Barra_6:=[[0,-
1050,525],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+0
09]];
  CONST robtarget
Ap_EnBarra_6:=[[0,540,675],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9
E+009,9E+009,9E+009]];
  CONST robtarget
Ens_Barra_6:=[[0,540,600],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E
+009,9E+009,9E+009]];
  CONST robtarget Ap_Racor_7:=[[0,-963.5,575],[ 0,-
0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Pto_Racor_7:=[[0,-963.5,525],[ 0,-
0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Ap_EnRacor_7:=[[-
500,576.5,600],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009
,9E+009]];
  CONST robtarget Ens_Racor_7:=[[-
412.5,576.5,600],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+0
09,9E+009]];
  CONST robtarget Ap_Racor_8:=[[36.5,-1150,575],
[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Pto_Racor_8:=[[36.5,-1150,525],
[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
```

```
  CONST robtarget
Ap_EnRacor_8:=[[500,576.5,600],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+0
09,9E+009,9E+009,9E+009]];
  CONST robtarget
Ens_Racor_8:=[[412.5,576.5,600],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+
009,9E+009,9E+009,9E+009]];
  CONST robtarget
Ens_Barra_6F:=[[0,612.5,600],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009
,9E+009,9E+009,9E+009]];

  PROC main()
    Path_Barra_1;
    Path_Racor_2;
    Path_Racor_3;
    Path_Barra_4;
    Path_Barra_5;
    Path_Barra_6;
    Path_Racor_7;
    Path_Racor_8;
    Path_Barra_6F;
  ENDPROC

  PROC Path_Barra_6F()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Ap_EnBarra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Barra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do6;
    WaitTime 2;
    MoveJ Ens_Barra_6F,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do6;
    WaitTime 2;
    MoveJ Ap_EnBarra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Racor_8()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_8,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Pto_Racor_8,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do8;
    WaitTime 2;
    MoveJ Ap_Racor_8,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnRacor_8,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Racor_8,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do8;
    WaitTime 2;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC
```

```
PROC Path_Racor_7()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ap_Racor_7,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Pto_Racor_7,v1000,fine,tPinza_TCP\WObj:=wobj0;
  WaitTime 2;
  Set do7;
  WaitTime 2;
  MoveJ Ap_Racor_7,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ap_EnRacor_7,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ens_Racor_7,v1000,fine,tPinza_TCP\WObj:=wobj0;
  WaitTime 2;
  Reset do7;
  WaitTime 2;
  MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_6()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
  WaitTime 2;
  Set do6;
  WaitTime 2;
  MoveJ Ap_Barra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ens_Barra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
  WaitTime 2;
  Reset do6;
  WaitTime 2;
  MoveJ Ap_EnBarra_6,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
ENDPROC

PROC Path_Barra_5()
  ConfJ\Off;
  ConfL\Off;
  MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ap_Barra_5,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Pto_Barra_5,v1000,fine,tPinza_TCP\WObj:=wobj0;
  WaitTime 2;
  Set do5;
  WaitTime 2;
  MoveJ Ap_Barra_5,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  MoveJ Ap_EnBarra_5,v1000,fine,tPinza_TCP\WObj:=wobj0;
```

```
    MoveJ Ens_Barra_5,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do5;
    WaitTime 2;
    MoveJ Ap_EnBarra_5,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_4()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_4,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_4,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do4;
    WaitTime 2;
    MoveJ Ap_Barra_4,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra_4,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Barra_4,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do4;
    WaitTime 2;
    MoveJ Ap_EnBarra_4,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Racor_3()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_3,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Pto_Racor_3,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do3;
    WaitTime 2;
    MoveJ Ap_Racor_3,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnRacor_3,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Racor_3,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do3;
    WaitTime 2;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Racor_2()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_2,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Pto_Racor_2,v1000,fine,tPinza_TCP\WObj:=wobj0;
```

```
    WaitTime 2;
    Set do2;
    WaitTime 2;
    MoveJ Ap_Racor_2,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnRacor_2,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Racor_2,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do2;
    WaitTime 2;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_1()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_1,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_1,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap_Barra_1,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnBarra_1,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Barra_1,v1000,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_EnBarra_1,v1000,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,fine,tPinza_TCP\WObj:=wobj0;
  ENDPROC
ENDMODULE
```

03. Programa Rapid de la estación transferido al robot real

Éste es el programa de rapid transferido al robot real, una vez introducidos los cambios comentados en el capítulo IV.2 sección 03:

```
MODULE Demo2RF
  PERS tooldata tPinza_TCP:=[TRUE,[[0,-24,220],
[1,0,0,0]],[1,[0,0,35],[1,0,0,0],0,0,0]];
  CONST robtarget
Paso_medio:=[[1000,0,1500],[0.5,0,0.866025,0],[0,0,0,0],[9E+009,9E+009
,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Aprox_Azul:=[[250,-
700,785],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Ap_Barra_1:=[[0,-
600,460],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
```

```
  CONST robtarget Pto_Barra_1:=[[0,-
600,300],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Aprox_Roja:=[[250,1000,785],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget
Aprox_Roja2:=[[250,1000,785],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009
,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_1:=[[2,1437,302],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Racor_2:=[[36.5,-
700,460],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Pto_Racor_2:=[[36.5,-
700,300],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Ens_Racor_2:=[[-
411,1403.5,302],[0.00029,0.99976,0.02206,-4E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Racor_3:=[[0,-
813.5,460],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Pto_Racor_3:=[[0,-
813.5,300],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Ap_EnRacor_3:=[[500,1396,301],[4E-005,0.02076,-
0.99978,5E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap1_EnRacor_3:=[[417.5,1396,400],[4E-005,0.02076,-
0.99978,5E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Racor_3:=[[417.5,1396,301],[4E-005,0.02076,-
0.99978,5E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_4:=[[0,-
750,460],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Pto_Barra_4:=[[0,-
750,300],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_4:=[[-411.5,950,306],[2E-
005,0.99999,0.00331,-9E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap1_EnBarra_4:=[[-411.5,950,455],[2E-
005,0.99999,0.00331,-9E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_4:=[[-411.5,1032,306],[2E-
005,0.99999,0.00331,-9E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_5:=[[0,-
900,460],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Pto_Barra_5:=[[0,-
900,300],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_5:=[[419.5,950,302],[0.0001,-0.00147,-
1,6E-005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap1_EnBarra_5:=[[419.5,950,455],[0.0001,-0.00147,-
1,6E-005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
```

```
  CONST robtarget Ens_Barra_5:=[[419.5,1026,302],[0.0001,-0.00147,-
1,6E-005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_Barra_6:=[[0,-
1050,460],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+0
09,9E+009,9E+009]];
  CONST robtarget Pto_Barra_6:=[[0,-
1050,300],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+0
09,9E+009,9E+009]];
  CONST robtarget Ens_Barra_6:=[[2,540,302],[0.00016,0.71107,-
0.70313,0.0001],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]]
;
  CONST robtarget Ap_Racor_7:=[[0,-
963.5,460],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Pto_Racor_7:=[[0,-
963.5,300],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+
009]];
  CONST robtarget Ens_Racor_7:=[[-
412.5,566.5,305],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+0
09,9E+009]];
  CONST robtarget Ap_Racor_8:=[[36.5,-
1150,460],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+0
09,9E+009,9E+009]];
  CONST robtarget Pto_Racor_8:=[[36.5,-
1150,300],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+0
09,9E+009,9E+009]];
  CONST robtarget Ap_EnRacor_8:=[[500,584,300],[0,0,-
1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Racor_8:=[[420,584,300],[0,0,-
1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ens_Barra_6F:=[[0,612.5,300],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_7R:=[[75,540,300],[0.00016,0.71107,-
0.70313,0.0001],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]]
;
  CONST robtarget Ap1_EnBarra_7R:=[[75,540,455],[0.00016,0.71107,-
0.70313,0.0001],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]]
;
  CONST robtarget Ap_EnBarra_6F:=[[0,612.5,485],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnBarra_2R:=[[75,1437,302],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap1_EnBarra_2R:=[[75,1437,455],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnRacor_1R:=[[-
411,1403.5,510],[0.00029,0.99976,0.02206,-4E-
005],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget Ap_EnRacor_6R:=[[-
412.5,566.5,510],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+0
09,9E+009]];
  CONST robtarget R1:=[[-40,1000,290],[0,0.707107,-
0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+009]];
  CONST robtarget
R2:=[[0,1060,290],[0,0,1,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+
009,9E+009]];
  CONST robtarget A1:=[[250,-
600,290],[0,0.707107,0.707107,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+00
9,9E+009,9E+009]];
```

```
  CONST robtarget A2:=[[250,-
700,290],[0,1,0,0],[0,0,0,0],[9E+009,9E+009,9E+009,9E+009,9E+009,9E+00
9]];

  PROC Path_Racor_6R()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_7,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Racor_7,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Racor_7,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnRacor_6R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Racor_7,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap_EnRacor_6R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Racor_1R()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    Set do1;
  MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_2,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Racor_2,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Racor_2,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnRacor_1R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Racor_2,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap_EnRacor_1R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_2R()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_1,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Pto_Barra_1,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
```

```
    WaitTime 2;
    MoveJ Ap_Barra_1,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap1_EnBarra_2R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ap_EnBarra_2R,v20,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Barra_1,v20,fine,tPinza_TCP\WObj:=wobj0;
WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap1_EnBarra_2R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_7R()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_6,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Barra_6,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Barra_6,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap1_EnBarra_7R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ap_EnBarra_7R,v20,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Barra_6,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap1_EnBarra_7R,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC main()
    Path_Racor_1R;
    Path_Barra_2R;
    Path_Racor_3;
Path_Barra_4;
    Path_Barra_5;
    Path_Racor_6R;
    Path_Barra_7R;
    Path_Racor_8;
    Path_Barra_6F;
  ENDPROC

  PROC Path_Barra_6F()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Ap_EnBarra_6F,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Barra_6,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveL Ens_Barra_6F,v20,fine,tPinza_TCP\WObj:=wobj0;
```

```
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap_EnBarra_6F,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Racor_8()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_8,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Racor_8,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Racor_8,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
MoveJ Ap_EnRacor_8,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Racor_8,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_5()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_5,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Barra_5,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Barra_5,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap1_EnBarra_5,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ap_EnBarra_5,v20,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Barra_5,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap1_EnBarra_5,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Barra_4()
ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Barra_4,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Barra_4,v100,fine,tPinza_TCP\WObj:=wobj0;
```

```
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Barra_4,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveL Ap1_EnBarra_4,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ap_EnBarra_4,v20,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ens_Barra_4,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap1_EnBarra_4,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_Racor_3()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_Racor_3,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Pto_Racor_3,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Reset do1;
    WaitTime 2;
    MoveJ Ap_Racor_3,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap1_EnRacor_3,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Ap_EnRacor_3,v200,fine,tPinza_TCP\WObj:=wobj0;
    MoveL Ens_Racor_3,v20,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    Set do1;
    WaitTime 2;
    MoveJ Ap1_EnRacor_3,v200,fine,tPinza_TCP\WObj:=wobj0;
MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_PosRoja()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    Reset do1;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ R1,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    MoveJ R2,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    MoveJ R1,v100,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Roja,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC

  PROC Path_PosAzul()
    ConfJ\Off;
    ConfL\Off;
    MoveJ Paso_medio,v1000,z50,tPinza_TCP\WObj:=wobj0;
    Reset do1;
```

```
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
    MoveJ A1,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    MoveJ A2,v100,fine,tPinza_TCP\WObj:=wobj0;
    WaitTime 2;
    MoveJ A1,v100,fine,tPinza_TCP\WObj:=wobj0;
    MoveJ Aprox_Azul,v1000,z50,tPinza_TCP\WObj:=wobj0;
  ENDPROC
ENDMODULE
```

VII.5 - PROGRAMAS DE VBA

Éste es el programa VBA completo de la estación que simula una parte del ensamblado del Service Core:

```
Dim numBarra As Integer
'La declaro como entero y global al módulo para que guarde el valor entre ejecuciones de las subrutinas

Sub attach_item_Ensamblado()

Dim c As CollisionSet
'declaro c como un conjunto de colision

Set c = ActiveStation.CollisionSets("Ensamblado")
'le asigno el conjunto "Ensamblado"

If numBarra < 1 Then numBarra = 1
'para que no dé fallo si no está inicializado numBarra

If c.Active = True Then
    Call ActiveStation.Parts("Barra_2300_1").Attachments.Add(c.ObjectsB.Item(numBarra), True)
End If
'si la colisión está activo, fijo la pieza número(numBarra) del grupo B a la barra 1

numBarra = numBarra + 1
'incremento el numero de barra a adjuntar
End Sub
```

```
Sub mover_mesa_90()

    Dim m As Mechanism
    'declaro la variable m como mecanismo
    Set m = ActiveStation.Mechanisms("MTC500")
    'asigno a m el motor
    Dim jointValue As Double
    'declaro la variable jointValue como un número decimal
    jointValue = m.Joints.Item(1).jointValue
    'le asigno a jointValue el valor del primer eje del motor (el único que tiene)
    jointValue = jointValue + (1 / 57.2957795130823) * 90
    'le sumo 90° a jointValue. Nótese que el valor se debe pasar en radianes (90*2*pi/360)
    m.Joints.Item(1).jointValue = jointValue
    'le asigno el valor al eje del mecanismo
    ActiveStation.Refresh
    'actualizo la estación

End Sub
```

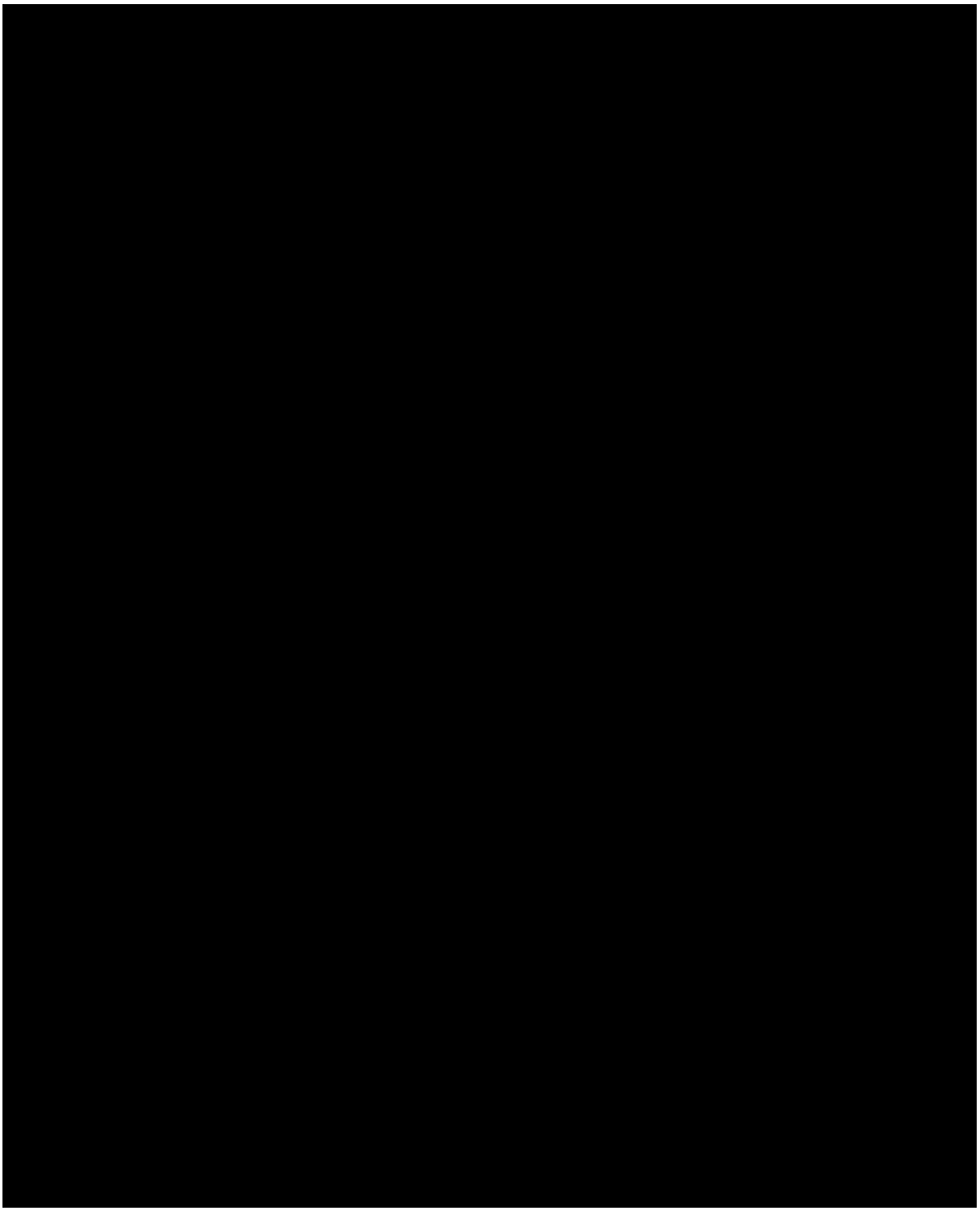
```
Sub mover_mesa_m45()

    Dim m As Mechanism
    'declaro la variable m como mecanismo
    Set m = ActiveStation.Mechanisms("MTC500")
    'asigno a m el motor
    Dim jointValue As Double
    'declaro la variable jointValue como un número decimal
    jointValue = m.Joints.Item(1).jointValue
    'le asigno a jointValue el valor del primer eje del motor (el único que tiene)
    jointValue = jointValue + (1 / 57.2957795130823) * -45
    'le sumo 90° a jointValue. Nótese que el valor se debe pasar en radianes (-45*2*pi/360)
    m.Joints.Item(1).jointValue = jointValue
    'le asigno el valor al eje del mecanismo
    ActiveStation.Refresh
    'actualizo la estación

End Sub
```

```
Sub mover_mesa_45()

    Dim m As Mechanism
    'declaro la variable m como mecanismo
    Set m = ActiveStation.Mechanisms("MTC500")
    'asigno a m el motor
    Dim jointValue As Double
    'declaro la variable jointValue como un número decimal
    jointValue = m.Joints.Item(1).jointValue
    'le asigno a jointValue el valor del primer eje del motor (el único que tiene)
    jointValue = jointValue + (1 / 57.2957795130823) * 45
    'le sumo 90° a jointValue. Nótese que el valor se debe pasar en radianes (45*2*pi/360)
    m.Joints.Item(1).jointValue = jointValue
    'le asigno el valor al eje del mecanismo
    ActiveStation.Refresh
    'actualizo la estación

End Sub
```

```
Sub attach_item_Mesa()
Dim c As CollisionSet
'declaro c como un conjunto de colision

Set c = ActiveStation.CollisionSets("Mesa_Barra1")
'le asigno el conjunto Mesa_Barra1

If c.Active = True Then
    Call ActiveStation.Parts("Mesa").Attachments.Add(c.ObjectsB.Item(1), True)
    'si la colisión está activa fijo las piezas

End If
End Sub
```

VII.6 - TABLAS DE EVENTOS

01. Tabla de eventos del ensamblado del SC (capítulo IV.1)



02. Tabla del proceso creado para demostrar (capítulo IV.2)

Esta es la tabla de eventos de la simulación del proceso de ensamblado realizada en el capítulo IV.2 sección 02, antes de ser revisada para transferir el programa al robot físico. Esta revisión sólo afectaría a las dos últimas entradas de la tabla, como se comenta en la sección IV.2.03.

VII.7 - LISTAS DE SEÑALES

Ésta es la lista de señales utilizadas en la simulación de la estación del capítulo IV.2.01

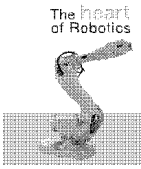
SEÑAL	ASIGNACIÓN	
do1	Coger/Dejar	Barra_2300_1
do12	Coger/Dejar	Barra_1220_2
do13	Coger/Dejar	Barra_1220_3
do14	Coger/Dejar	Barra_2300_4
do15	Coger/Dejar	Barra_2300_5
do16	Coger/Dejar	Barra_1220_6
do17	Coger/Dejar	Barra_1220_7
do18	Coger/Dejar	Barra_1220_8
do19	Coger/Dejar	Barra_1220_9
do21	Coger/Dejar	Tubo_25x910_1
do22	Coger/Dejar	Tubo_16x1620_2
do23	Coger/Dejar	Tubo_16x2100_3
do24	Coger/Dejar	Tubo_16x582_4
do26	Ejecutar	mover_mesa_90
do27	Ejecutar	mover_mesa_m45
do28	Ejecutar	mover_mesa_45

Ésta es la lista de señales utilizadas en la simulación de la estación del capítulo IV.2.02 (antes de la revisión para transferir al robot real).

SEÑAL	ASIGNACIÓN	
do1	Coger/Dejar	Tubo_Cuadrado_25x25_1
do2	Coger/Dejar	Racor_2vias_2
do3	Coger/Dejar	Racor_2vias_3
do4	Coger/Dejar	Tubo_Cuadrado_25x25_4
do5	Coger/Dejar	Tubo_Cuadrado_25x25_5
do6	Coger/Dejar	Tubo_Cuadrado_25x25_6
do7	Coger/Dejar/Fijar a 6	Racor_2vias_7
do8	Coger/Dejar/Fijar a 6	Racor_2vias_8

VII.8 - HOJAS DE CARACTERÍSTICAS

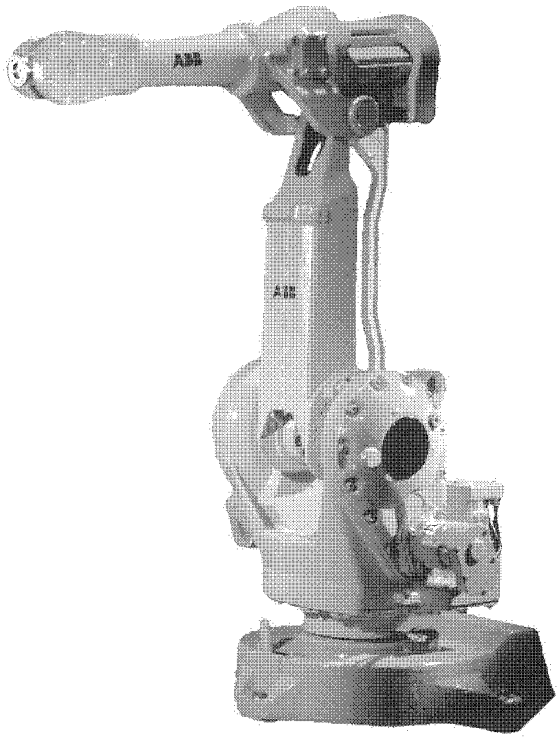
Robot IRB 2400



IRB 2400

Industrial Robot

MAIN APPLICATIONS
Arc welding
Cutting/Deburring
Glueing/Sealing
Grinding/Polishing
Machine tending
Material handling



Most popular industrial robot

IRB 2400 is the world's most popular industrial robot in its class. It comprises a complete family of application optimized robots that maximize the efficiency of your arc welding, process and tending applications.

The IRB 2400 is a real hard worker. It can take additionally 35 kg load on axis 1 and 15 kg additional load on the upper arm - still keeping 100 % duty cycle.

The IRB 2400L model has 1.8 meters reach, 7 kg load capacity, large working range and slim arm and wrist. Other models offer handling capacity of up to 20 kg, excellent motion control, large load offset and unlimited motion in axis 6. This means there's an IRB 2400 robot to give you excellent performance in your material handling, tending and process applications. All models offer you inverted mounting capability.

The compact design of the IRB 2400 ensures ease of installation. The robust construction and use of minimum parts contribute to high reliability and long intervals between maintenance.

The Foundry Plus version is washable with high pressure steam and it's supplied with increased environment protection meeting IP 67 standard.

ABB

IRB 2400

Industrial Robot

TECHNICAL DATA, IRB 2400 INDUSTRIAL ROBOT

SPECIFICATIONS

Robot version	Handling capacity	Reach	Protection
IRB 2400L	7 kg	1.8 m	Foundry, Clean room
IRB 2400-10	12 kg	1.5 m	FoundryPlus, Clean room
IRB 2400-16	20 kg	1.5 m	FoundryPlus, Clean room

Mounting	Floor and inverted all versions. Wall IRB 2400-10

Supplementary load	IRB 2400L	IRB 2400-10	IRB 2400-16
Upper arm,			
-wrist end	1 kg	2 kg	2 kg
-rear end	10 kg	10 kg	10 kg
Base unit	35 kg	35 kg	35 kg

Number of axes	
Robot manipulator	6
External devices	6
Integrated signal supply	23 poles, 50 V DC 10 poles, 250 V AC
Integrated air supply	Max. 8 bar

PERFORMANCE

Positional repeatability	0.06 mm (average result from ISO test)

Axis movements	IRB 2400L	IRB 2400-10	IRB 2400-16
Working range			
Positioning			
Axis 1, Rotation	360°	360°	360°
Axis 2, Arm	200°	200°	200°
Axis 3, Arm	125°	125°	125°
Re-orientation			
Axis 4, Wrist	370°	400°	400°
Axis 4, Option	–	Unlimited	Unlimited
Axis 5, Bend	240°	240°	240°
Axis 6, Rotation	800°	800°	800°
Axis 6, Option	Unlimited	Unlimited	Unlimited

Axis movements	IRB 2400L	IRB 2400-10	IRB 2400-16
Max. speed			
Positioning			
Axis 1, Rotation	150°/s	150°/s 90° *	150°/s
Axis 2, Arm	150°/s	150°/s 90° *	150°/s
Axis 3, Arm	150°/s	150°/s 90° *	150°/s
Re-orientation			
Axis 4, Wrist	360°/s	360°/s	360°/s
Axis 5, Bend	360°/s	360°/s	360°/s
Axis 6, Rotation	450°/s	450°/s	450°/s

*) For wall mounted version

ELECTRICAL CONNECTIONS

Supply voltage	200–600 V, 50/60 Hz
Rated power, supply transformer	4 kVA/7.8 kVA with external axes

PHYSICAL

	IRB 2400L	IRB 2400-10	IRB 2400-16
Dimensions			
Total height	1,731 mm	1,564 mm	1,564 mm
Manipulator bases	723x600 mm	723x600 mm	723x600 mm
Weight			
Robot	380 kg	380 kg	380 kg

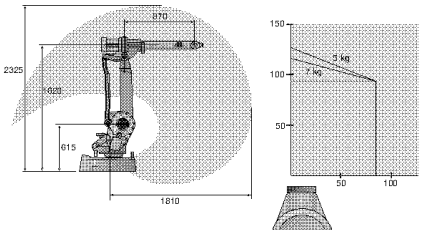
ENVIRONMENT

Ambient temperature			
Basic manipulator in operation	5°C - 45°C		
Relative humidity	Max. 95%		
Degree of protection	IRB 2400L	IRB 2400-10	IRB 2400-16
Standard and Clean Room versions	IP54	IP54	IP54
Foundry or Foundry Plus versions	IP55/67	IP67	IP67
Clean Room	US Federal Standard 209,class 100		
Noise level	Max. 70 dB (A)		
Emmission	EMC/EMI-shielded		

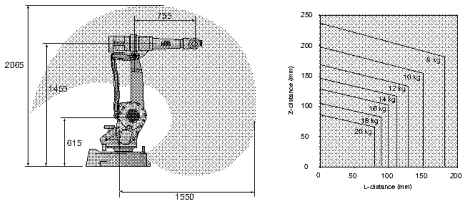
DATA AND DIMENSIONS MAY BE CHANGED WITHOUT NOTICE.

WORKING RANGE AND LOAD DIAGRAM

IRB 2400L



IRB 2400-10, IRB 2400-16

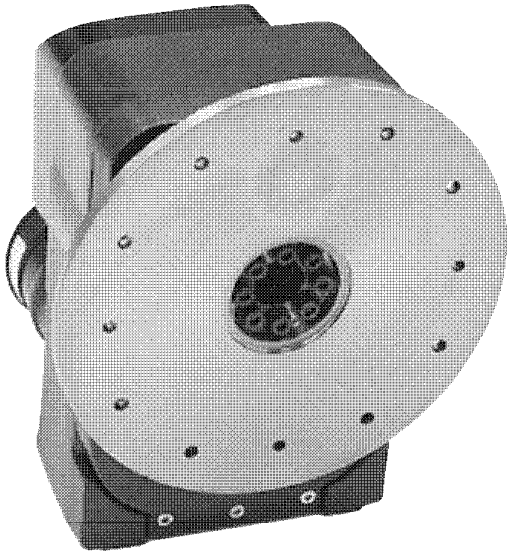


www.abb.com/robotics

Unidad rotatoria MTC 500

Rotary unit

Motion Range



Rotary unit

The rotary units are designed to be used in different situations when you need rotating movements on payloads from 250 kg up to 5000 kg.

The units are originally designed for arc welding, the motors are insulated and they are equipped with current collectors, but they can properly be used in a lot of different applications such as thermal and laser cutting, grinding material handling etc. For media transfer the rotary units can be equipped with a 24 channel electrical slip ring and/or a one or two channel swivel for air and water.

We can supply the units with cabling for fixed assembly or flexible cables if the rotary units will be placed on some kind of indexing table.

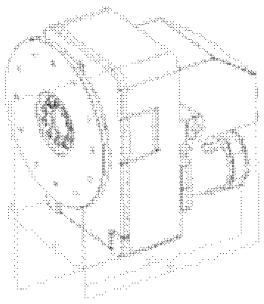
Our range of rotary units fits ABB robots IRB 140, 1400, 2400, 4400 and 6400.

Main features

- Short delivery time
- High flexibility
- Short installation time
- Coordinated motion for easy programming
- Easy to integrate

Rotary unit

Motion Range



	MTC			
	A	B	C	D
250	250	378	337	364
500	354	540	479	496
750	354	540	479	496
2000	415	612	567	606
5000	580	810	771	788

	Unit	MTC 250	MTC 500	MTC 750	MTC 2000	MTC 5000
Handling capacity	kg	250	500	750	2000	5000
Cont. torque	Nm	350	650	900	3800	9000
Max inertia	kgm^2	40	170	300	1200	3500
Max bending torque	Nm	650	3300	5000	25000	60000
Max speed	rpm	30	25	25	15	6.5
Max acc	rad/s^2	3,8	2.5	2.0	0.7	0.6
Repeatability r=500	mm	0.1	0.1	0.1	0.1	0.1
0 – 45 deg reorient	sec	1.1	1.4	1.5	2.3	2.5
0 – 90 deg reorient	sec	1.4	1.8	2.1	3.2	3.7
0 – 180 deg reorient	sec	2.0	2.5	2.8	4.4	6.0
0 – 360 deg reorient	sec	3.0	3.6	4.1	6.4	10.9
Emergency stop time	sec	< 0,5	< 0.5	< 0.5	< 0.6	< 0.9
Weight	kg	70	169	172	340	772

Dimensional drawings are found on www.abb.com/robotics were you also will find more detailed technical information.

PR10152EN_R1 April 2003
Cert Or. Printed in Sweden by Westerås Media Produktion, Västerås 2003.

www.abb.com/robotics
ABB reserves the right to change specifications without notice.

SE-695 82 Laxå Sweden, Tel +46 584 820 00, Fax +46 584 131 80