Universidad
Rey Juan Carlos

Práctica 2

Programación en lenguaje RAPID.
Uso de objetos de trabajo (WorkObjects).
Manejo de módulos y procedimientos

Julio S. Lora Millán

Juan Alejandro Castaño Peña

Diego Martín Martín

Í N D I C E

1. INTRODUCCIÓN Y OBJETIVOS

En la práctica 2 estudiaremos todo lo relativo a la programación del robot mediante el lenguaje RAPID utilizando RobotStudio. RAPID es el acrónimo de *Robotics Application Programming Interactive Dialogue*, y es un lenguaje de programación de alto nivel creado por ABB en 1994, como evolución de un lenguaje más antiguo denominado ARLA.

Otros fabricantes de robots industriales utilizan lenguajes muy similares. Destacamos los siguientes:

- KRL (Kuka)
- KAREL (Fanuc)
- V+ (Adept, Staübli)
- AS (Kawasaki)
- Val II (Unimation)

Antes de empezar es importante destacar que los lenguajes de programación de robots se suelen dividir en dos familias:

- **Lenguajes de programación por guiado o aprendizaje:** consiste en llevar manualmente el robot a los puntos necesarios para realizar una tarea, al tiempo que se registran dichos puntos, la trayectoria entre ellos y la configuración de la herramienta utilizada (Figura 1).
    - **Guiado pasivo:** si se desconectan los actuadores y el operador mueve manualmente el robot, se denomina *guiado pasivo directo*. Si se utiliza un "modelo ligero" del robot, se denomina *guiado pasivo por maniquí*.
    - **Guiado activo:** se realiza el movimiento manual del robot empleando los actuadores del mismo, controlados desde una botonera o joystick.
- **Lenguajes de programación textual:** consiste en realizar un programa basado en una secuencia de instrucciones en modo texto pertenecientes a un lenguaje determinado.

Por suerte, RAPID combina la programación textual con la programación por guiado activo.

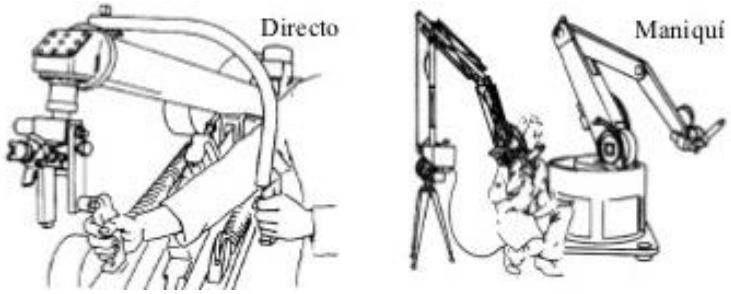

Figura 1. Lenguajes de programación por guiado (directo y por maniquí)

2. Uso de objetos de trabajo para definir posiciones y trayectorias. Configuraciones del robot.

3.1. ¿Qué es un objeto de trabajo o WorkObject?

Las instrucciones de movimiento de RAPID utilizan por defecto posiciones de trabajo relativas al sistema de coordenadas de la base del robot, como hemos visto en la práctica anterior Sin embargo, suele ser más interesante realizar la definición de posiciones con respecto a un sistema de coordenadas que se encuentre en la mesa de trabajo (**coordenadas de usuario**) o sobre el mismo objeto de trabajo (**coordenadas del objeto**), ambas representadas en la Figura 1.

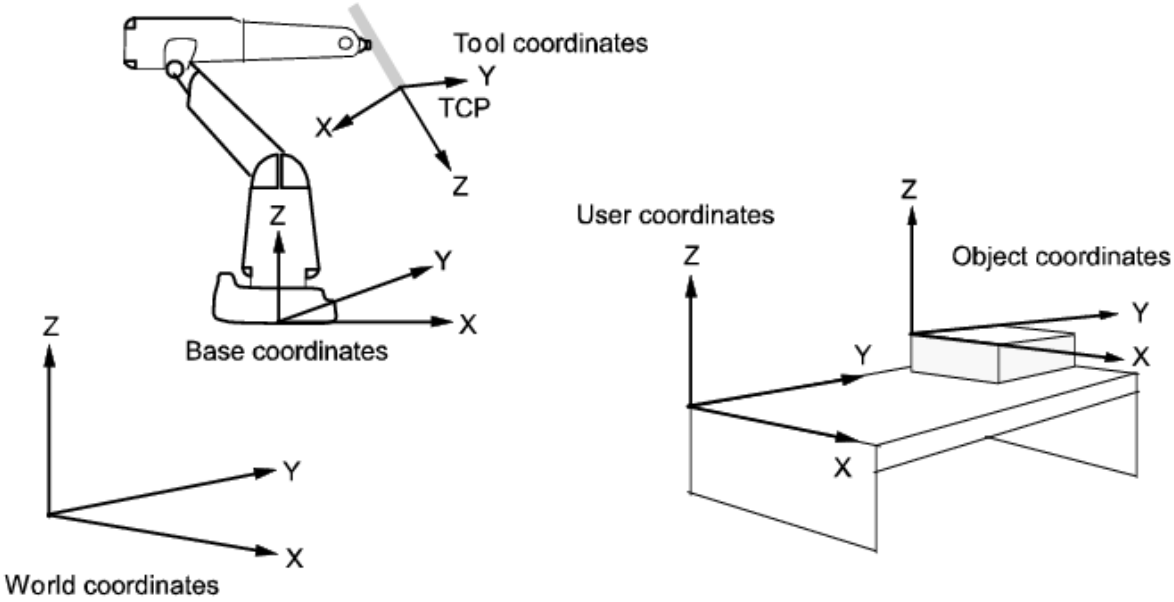

Figura 1. Sistemas de coordenadas existentes en una aplicación robótica

Programar trayectorias que utilicen posiciones referenciadas a los sistemas de referencia de usuario u objeto presenta ciertas ventajas:

- Al reposicionar la mesa o el objeto de trabajo en la estación, sólo es necesario **modificar la posición del sistema de coordenadas de usuario o de objeto**, y toda la trayectoria se reubicará automáticamente.
- Permite el trabajo con objetos movidos por ejes externos

Para especificar nuevos sistemas de referencia, RobotStudio utiliza el tipo de datos "objeto de trabajo" o WorkObject, que se puede insertar como argumento opcional en las instrucciones de movimiento, como veremos más adelante.

El sistema de coordenadas de la base del robot (sistema de referencia 0) es el que viene configurado por defecto, y se denomina "**WorkObject 0**" (wobj0). Normalmente se usa este

sistema de referencia para especificar las posiciones "**Home**" del robot (que se usa para el inicio y fin de las trayectorias de movimiento).

3.1. Creación, definición y uso de los objetos de trabajo.

Para definir un objeto de trabajo en RobotStudio debemos ir a Posición Inicial → Otros → Crear objeto de trabajo (Figura 2).

En un robot real el objeto de trabajo se crea desde el FlexPendant, utilizando un método basado en la posición de tres puntos (lo estudiarás en el Ejercicio Entregable 1)

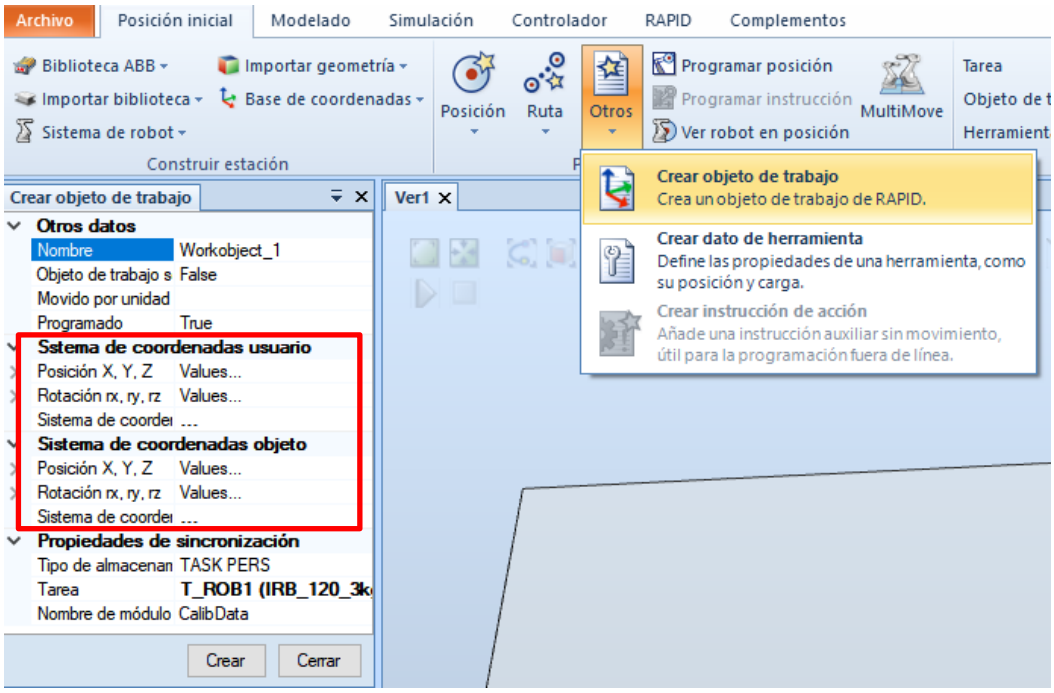

Figura 2. Creación de un objeto de trabajo en RobotStudio

Una vez creado el objeto de trabajo, **podemos definirlo** dándole valores a la posición y orientación del sistema de coordenadas del objeto y del usuario (recuadro de color rojo en la Figura 2). No es resulta necesario dar valores a ambos, uno de los dos (objeto o usuario) puede quedase en la posición (0,0,0) y sin rotación si no es necesario.

Tras crearlo, se puede seleccionar en el menú "Parámetros" y **utilizarlo para definir posiciones y trayectorias** en el árbol "Trayectorias y puntos" del menú izquierdo.

Una vez que hayamos creado los puntos y las trayectorias asociadas al nuevo objeto de trabajo, si modificamos la posición de este último (a mano alzada o utilizando las herramientas de posicionamiento de RobotStudio) comprobaremos que todas las posiciones asociadas a dicho WorkObject se actualizan automáticamente.

3.2. Configuración de ejes del robot en cada posición.

Cuando el controlador calcula el ángulo de los ejes del robot para alcanzar un objetivo (una posición con una orientación de la herramienta dada), con frecuencia es posible encontrar más de una solución posible para el ángulo concreto de alguno de los ejes del robot. En la figura 3 se puede apreciar cómo el robot es capaz de llegar a la misma posición y orientación de la herramienta con tres configuraciones de ejes diferentes.

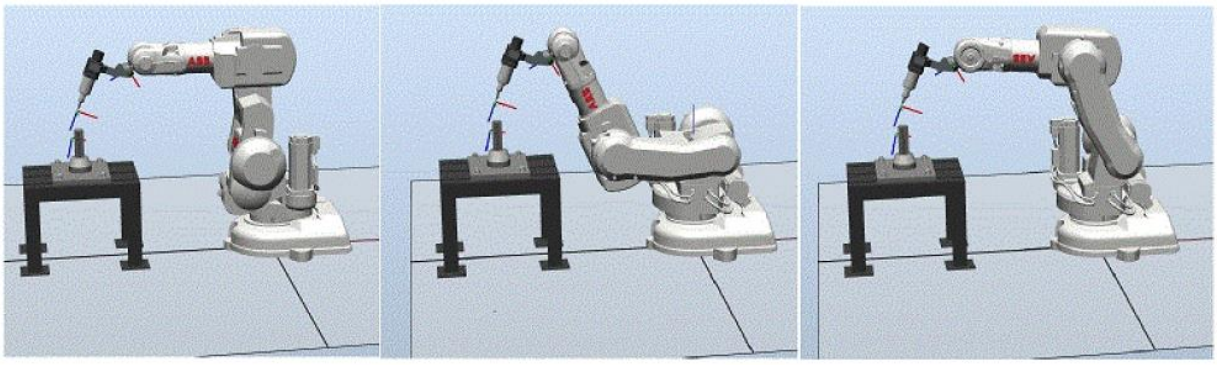

Figura 3. Tres configuraciones de ejes diferentes para la misma posición y orientación de la herramienta

Para distinguir entre las distintas configuraciones, todas las posiciones llevan asociadas unos **valores de configuración** que especifican en qué cuadrante debe situarse cada eje.

¿Cómo se designan las configuraciones de ejes? Para designar una configuración determinada, **CFGi,** se utiliza una serie de cuatro números enteros **(0,1,2,-1)** que especifican en qué cuadrante de una revolución completa se encuentran los ejes significativos. Los cuadrantes están numerados de cero en adelante para una rotación positiva (en el sentido contrario a las agujas del reloj) y de -1 en adelante para la rotación negativa (en el sentido de las agujas del reloj). En el caso de los ejes lineales, el entero especifica el rango (en metros) en el que se encuentra el eje desde la posición neutral.

Por ejemplo, una configuración para un robot industrial de seis ejes como el IRB 120 puede parecerse a la siguiente: **(0,-1,2,1)**, cuyo significado es:

- El **primer entero** (0) especifica la posición del **eje 1**: en algún punto del primer cuadrante positivo (en una rotación de entre 0 y 90 grados).
- El **segundo entero** (-1) especifica la posición del **eje 4**: en algún punto del primer cuadrante negativo (en una rotación de entre 0 y -90 grados).
- El **tercer entero** (2) especifica la posición del **eje 6**: en algún punto del tercer cuadrante positivo (en una rotación de entre 180 y 270 grados).
- El **cuarto entero** (1) especifica la posición del **eje X**, un eje virtual utilizado para especificar el centro de la muñeca respecto de los demás ejes.

La figura 4 muestra la manera de seleccionar la configuración de ejes en una posición mediante RobotStudio, utilizando el botón derecho del ratón sobre cada posición:

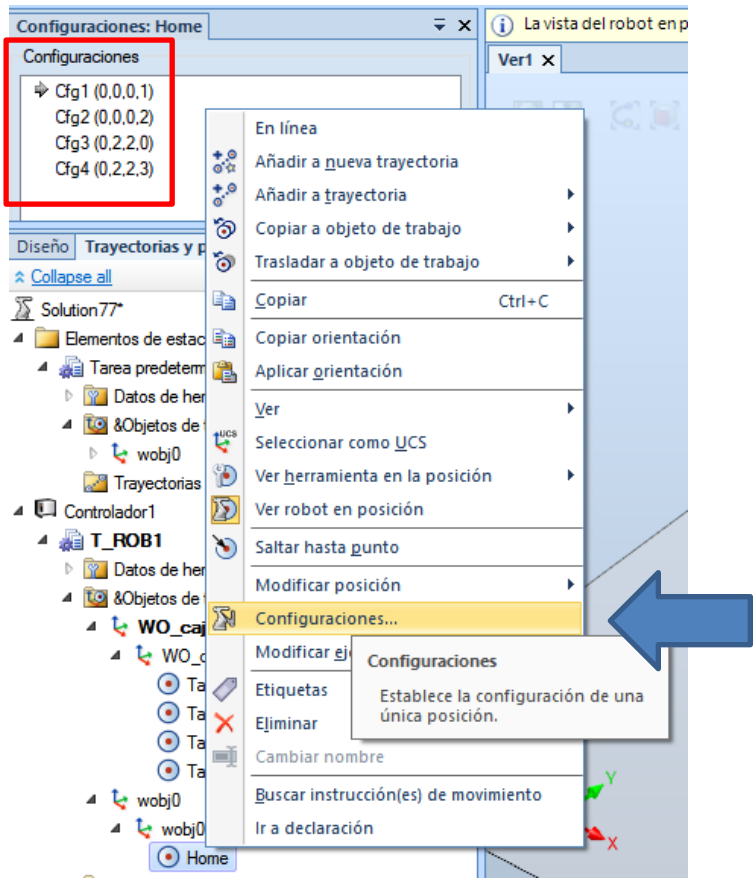

Figura 4. Selección de configuraciones para la posición Home de un robot en RobotStudio

A la hora de ejecutar una trayectoria, es posible decidir **si el controlador tiene que tener en cuenta o no los valores de las configuraciones de cada punto**. Si el control de configuraciones está **desactivado**, los valores de configuración almacenados en los objetivos no se tienen en cuenta y el controlador utilizará la configuración más cercana a su configuración actual para alcanzar el objetivo. Si se **activa**, solo utilizará la configuración especificada para alcanzar los objetivos. El control de configuraciones puede desactivarse y activarse de forma independiente para los movimientos de ejes y los movimientos lineales, y se controla mediante las instrucciones de acción **ConfJ y ConfL**

La ejecución de un programa sin el control de configuraciones puede dar lugar a **configuraciones diferentes cada vez que se ejecuta un ciclo:** cuando el robot regresa a la posición inicial tras completar un ciclo, puede elegir una configuración distinta de la original. La ejecución de un programa con el control de configuraciones fuerza al robot a utilizar las configuraciones almacenadas en los objetivos. Con ello se consiguen ciclos y movimientos predecibles.

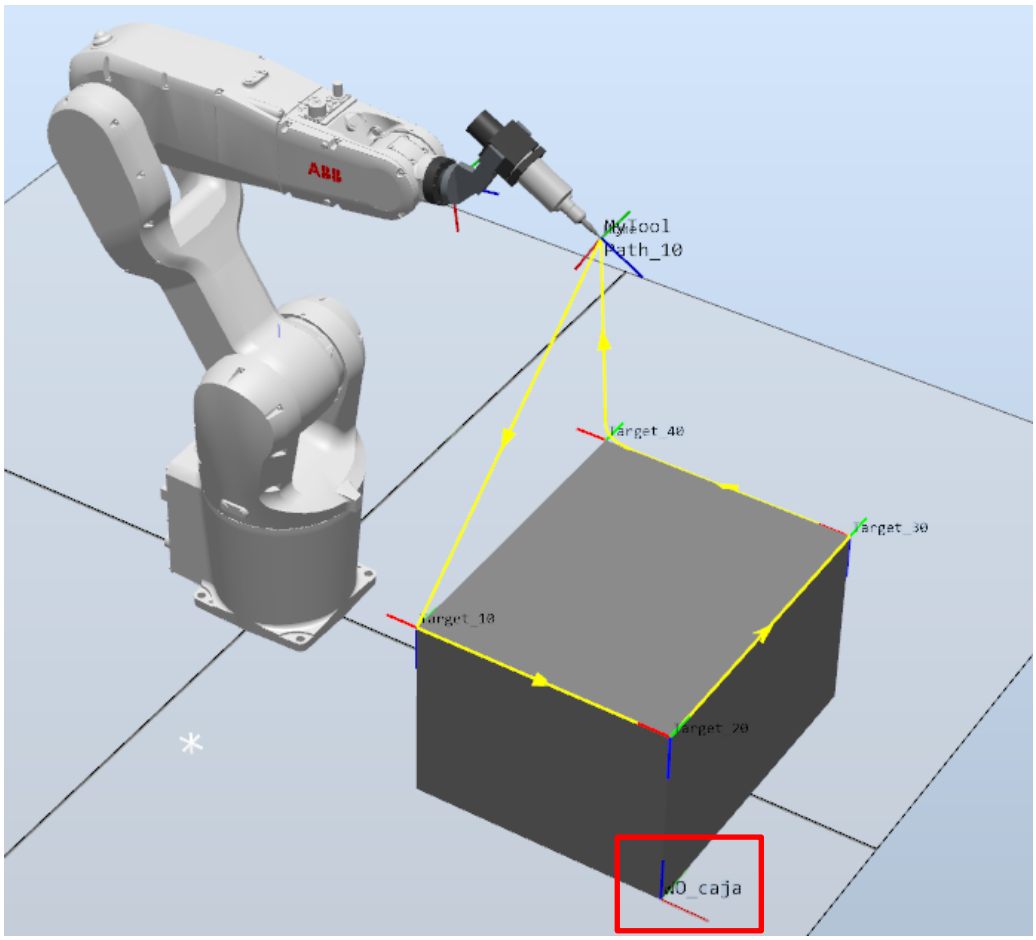

Figura 5. Ejercicio práctico 1. La posición inicial del WO_Caja está marcada en el recuadro rojo

Ejercicio práctico 1. Creación de WorkObject. Configuraciones de ejes del robot

1. Crea una solución con un robot ABB IRB1200 y un controlador. Inserta la herramienta *My_Tool* de la biblioteca y posiciónala en la brida del robot
2. Inserta una caja de dimensiones y posición parecidas a las que aparece en la Figura 5
2. Crea un objeto de trabajo "**WO_Caja"** y posiciónalo en un vértice inferior de la caja
3. Crea una posición inicial llamada **"Home"** que esté situada sobre la caja y asócíala asociándola al objeto de trabajo "WorkObject0" (wobj0). Revisa y ajusta su orientación.
4. Crea posiciones sobre los vértices superiores asociadas a WO_Caja. Reorienta la herramienta hasta que tenga una orientación adecuada para el robot y visualiza el resultado. Comprueba que el robot puede acceder a esa posición y orientación de la herramienta (opción "ver robot en posición")

5. Establece una trayectoria simple utilizando las posiciones creadas. Usa ordenes de tipo MoveL, a una velocidad máxima de v200 y con zona "Fine". Ejecuta la trayectoria.
6. Modifica la configuración de ejes para alguno de los puntos de la trayectoria y vuelve a ejecutarla, observando las diferencias con respecto a la ejecución anterior.
7. Inserta una instrucción de acción de tipo "ConfL \Off" al inicio de la trayectoria y vuelve a ejecutarla, fijándote en las diferencias con respecto a la ejecución anterior

   NOTA: Si obtienes errores de tipo "Error de configuración de robot" o "Eje fuera de rango" debes seleccionar otra configuración de ejes en el punto problemático.

Ejercicio práctico 2. Modificación de un WorkObject y de una trayectoria asociada

1. Abre la solución del ejercicio práctico anterior y guárdala con otro nombre.
2. Mueve la caja de sitio, modifica la posición del WO_Caja para volverla a hacer coincidir con la misma esquina que antes y comprueba que la trayectoria se actualiza automáticamente.
3. ¿Qué ocurre con la posición Home? ¿Por qué no se ha actualizado?

   IMPORTANTE: Es posible que al mover la caja aparezca un símbolo de "warning" o "stop" en alguna de las instrucciones de movimiento. Sitúa el ratón sobre la instrucción y aparecerá una descripción del problema. Posiblemente tengas que actualizar alguna de las configuraciones de ejes o incluso modificar la orientación de la herramienta en alguna de las posiciones para que la trayectoria se ejecute sin errores.

Ejercicio práctico 3. Visualización del código RAPID asociado a una trayectoria. Ejecución de rutina desde el FlexPendant

1. Abre la solución del ejercicio práctico 1. Sincroniza la estación con el controlador, es decir, con RAPID (Posición Inicial → Sincronizar → Sincronizar con RAPID, eligiendo todo lo que aparece)
2. Estudia el código y trata de identificar el objetivo de cada instrucción. Recuerda que el código se encuentra en Controlador → Estación actual → RAPID → T_ROB1 → Module1). ¿Qué son los robtargets, y qué estructura tienen? ¿Dónde está la herramienta configurada? ¿Y el WorkObject caja?
3. Abre el FlexPendant e intenta ejecutar el programa desde el mismo. Para ello, configura el robot en modo Manual, ve al Editor de Programa del Flexpendant, y lleva el Puntero de Programa (PP) a la rutina que quieras ejecutar (Debug → PP a rutina). Por último, ejecútala pulsando el botón Play del FlexPendant.

2. Programas en lenguaje RAPID. Instrucciones

2.1. Formato de las instrucciones de movimiento.

Pasemos ahora a estudiar las diferentes instrucciones de movimiento. Para mover el TCP de la herramienta del robot de un punto a otro del espacio existen tres instrucciones básicas, que aparecen en la Figura 6 y que se basan en la cinemática inversa y diferencial del robot:

```
MoveJ Punto, Velocidad, Zona, Herramienta
```

Se mueve el robot hacia un punto usando coordenadas articulares.

Cuando no tiene que seguir ninguna trayectoria determinada.

```
MoveL Punto, Velocidad, Zona, Herramienta
```

Se mueve el robot hacia un punto usando la línea recta.

```
MoveC Punto_Circulo, Punto_Destino, Velocidad, Zona, Herramienta;
```

Se mueve el extremo del robot hacia el punto de destino pasando por el punto del circulo trazando un arco de circunferencia.

Figura 6. Instrucciones básicas de movimiento en RAPID (extraído de [1])

Es muy importante que entendamos la diferencia entre las instrucciones MoveL y MoveJ.

- **MoveL (*Move Linear*)**: el TCP de la herramienta se mueve siguiendo una trayectoria recta a la velocidad especificada.
- **MoveJ (*Move Joint, eje a eje*)**: el TCP de la herramienta alcanza el punto destino moviendo todos los ejes de sus ángulos iniciales a los finales a una velocidad constante en las coordenadas articulares. Esto da lugar a que la velocidad del TCP sea aproximada y, lo que es más importante, a trayectorias del TCP no lineales, pero **es el tipo de movimiento habitual** cuando no se necesitan líneas rectas (Figura 7).

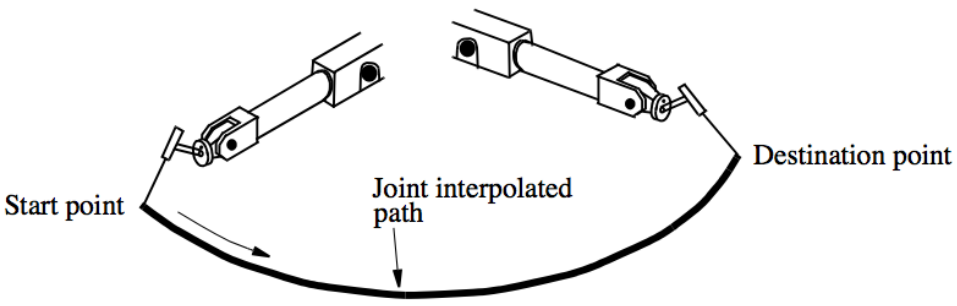

Figura 7. Trayectoria del TCP (exagerada) cuando se realiza una instrucción de movimiento tipo MoveJ

Cada una de las instrucciones anteriores tiene cuatro argumentos obligatorios (Figura 8):

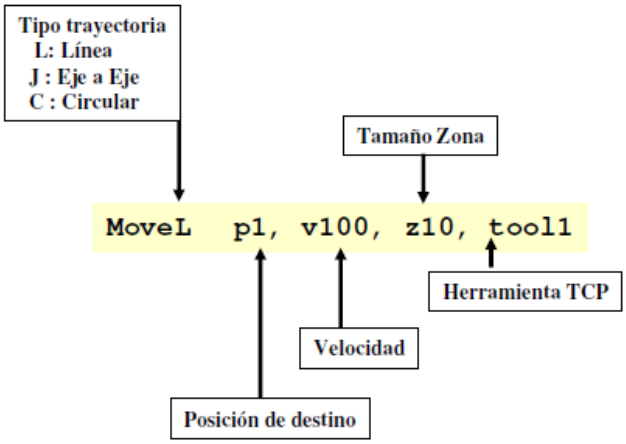

Figura 8. Argumentos obligatorios en una instrucción de movimiento en RAPID

- **Posición (y orientación) del TCP en el destino**. Se trata de un tipo de dato *robtarget*, que contiene posición, orientación, configuración de los ejes de robot (para evitar singularidades) y posición de los ejes externos (si no los hay aparace 9E+09)
- **Velocidad de desplazamiento del TCP.** Tipo de dato *speeddata*, que sirve para definir la velocidad lineal y de reorientación del TCP y de los ejes externos si los hay. Se suele trabajar con los valores predefinidos v100, v200, v1000… que se corresponden con velocidades del TCP en mm/s y llevan asociadas velocidades de reorientación de 500º/s.
- **Tamaño de la zona de paso:** Tipo de dato *zonedata* que sirve para trabajar con puntos de paso, en los que no es imprescindible que el TCP pase exactamente por el punto especificado, sino que puede hacerlo por sus inmediaciones. Se da en distancia (mm), que indica a qué distancia del punto destino se puede iniciar el movimiento hacia el siguiente punto (Figura 9). Permite que los tiempos se acorten. Cuando se necesita parar en un punto se usa "fine".

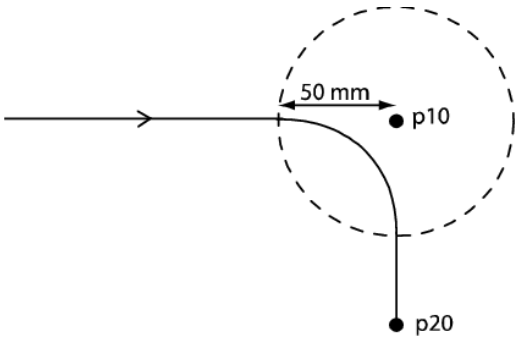

Figura 9. Concepto de tamaño de la zona de paso (zonedata). Se usan valores predefinidos y

- **Herramienta a utilizar:** por último, se especifica el tipo de dato "tooldata" que especifica la herramienta que está colocada en la brida del robot, y que aprendimos a definir en la práctica anterior.
- **WorkObject:** se puede especificar el sistema de referencia con respecto al cual están tomados los puntos como argumento final de cada instrucción (es opcional).

En el caso de las instrucciones de movimiento circular *MoveC* se deben especificar dos posiciones destino, una de ellas es la posición final y otra la posición de paso, que se utiliza para calcular el radio del círculo a trazar.

- Por ejemplo, el siguiente código se corresponde con la trayectoria que se muestra en la Figura 10. Se usará la herramienta tPen, y se pasará exactamente por todos los puntos (zona "fine"), moviéndose entre ellos a velocidad de 500 mm/s:

```
MoveL p10, v500, fine, tPen;
MoveC p20, p30, v500, fine, tPen;
MoveL p40, v500, fine, tPen;
```

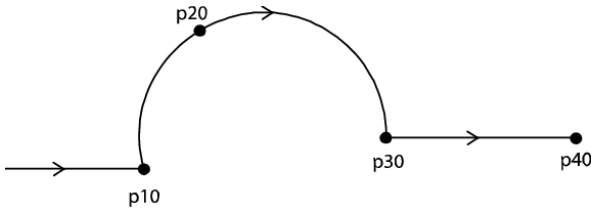

Figura 10. Ejemplo de trayectoria entre cuatro puntos p10, p20, p30 y p40.

Por último, destacar que también existe una instrucción de movimiento que permite mover el robot mediante la cinemática directa, es decir, eje a eje. Se trata de **ModeAbsJ.**

Consulta el manual de RAPID para más información sobre cada una de estas instrucciones y tipos de datos [2]. En el manual encontrarás una descripción muy detallada de cada instrucción, junto con sus argumentos obligatorios, opcionales y excluyentes (Figura 11)

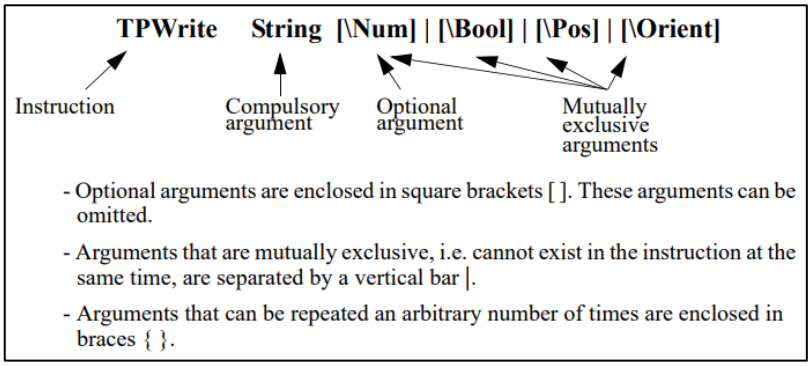

Figura 11. Sintaxis general de las instrucciones de RAPID en el manual de referencia del lenguaje [2]

Ejercicio práctico 4. Cambio de zonas de paso y velocidades. Trayectorias circulares

1. Abre la solución del ejercicio práctico 1. Sincroniza la estación con el programa RAPID y modifica el tamaño de las zonas de paso y alguna de las velocidades de movimiento directamente en el código RAPID. Ejecuta la rutina desde el FlexPendant y comprueba el efecto.

   IMPORTANTE: Cuando modificas un programa en el editor de RAPID de RobotStudio, los cambios aparecen resaltados a la izquierda en color amarillo. Debes pulsar en "Aplicar" para que los cambios se graben en el controlador (y pasan a color verde), lo que además servirá para comprobar la sintaxis del código en busca de errores. Por último, cuando el código esté libre de errores y grabado en el controlador (color verde), debes "sincronizar con estación" para ver los cambios reflejados en la simulación

2. Añade a la estación un objeto cilíndrico de 400 mm de altura y 400 mm de diámetro, situado en el punto (0, 600, 0).
3. Crea un nuevo WO llamado WO_cilindro y sitúa el sistema de coordenadas del usuario en el centro de la cara superior del cilindro. Sincroniza con RAPID.
4. Programa cuatro nuevas posiciones que se sitúen sobre la arista de la cara superior del cilindro. Para ello, utiliza la opción Posición -> Crear puntos de Arista. Comprueba la orientación correcta de la herramienta en dichos puntos y sincroniza con RAPID
5. Realiza en RAPID, dentro del Módulo 1, un nueva rutina tipo procedimiento (PROC – ENDPROC) que genere una trayectoria que, usando las posiciones programadas, recorra circularmente la arista superior completa del cilindro, partiendo desde HOME y acabando en HOME. Utiliza tantos MoveC como necesites. Al programar, puedes usar TAB para completar los argumentos de cada instrucción.
6. Ejecuta la rutina desde el FlexPendant y comprueba el resultado

2.2. Estructura de una aplicación y un programa en RAPID.

Como ya has ido viendo en los ejemplos anteriores, una aplicación robótica en RAPID está formada por dos partes fundamentales (Figura 12):

- **Módulos del sistema:** creados por el fabricante. Se guardan con extensión .mod. Contienen información específica necesaria para la ejecución de todos los programas (variables persistentes como herramientas, etc.)
- **Programa:** creados por el programador. Un programa se divide en diferentes **Módulos**, cada uno de los cuales debe tener inicialmente la **declaración de datos** (herramientas, posiciones, constantes numéricas, sistemas de coordenadas, etc.) y una serie de **rutinas**, que sirven para estructurar la ejecución en partes más pequeñas.

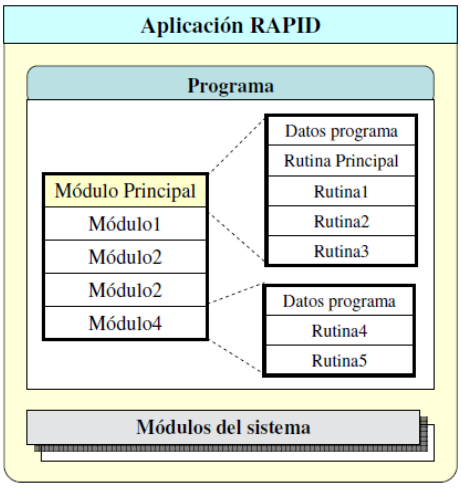

Figura 12. Estructura de una aplicación en lenguaje RAPID

El módulo principal contiene además la **rutina principal (main)**, que es el punto de inicio de la ejecución del programa (Figura 13). El nombre de los módulos y rutinas debe comenzar por una letra y ser de longitud máxima 16 caracteres.

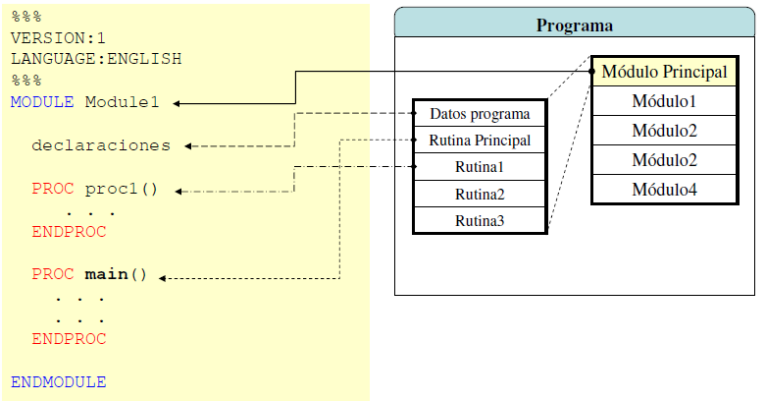

Figura 13. Aspecto del módulo principal del programa en lenguaje RAPID.

Si se tiene la opción Multitarea también existe la opción de crear "Tareas".

En los programas se pueden insertar comentarios comenzando por una exclamación ¡, y cadenas de texto entre comillas: "Cadena de texto".

En resumen, la programación básica se realiza utilizando la modularidad que proporcionan las rutinas en RAPID. Cabe destacar que hay **tres tipos de rutinas** (Figura 14):

- **Procedimientos:** rutinas ordinarias de trabajo: trayectorias, etc.
- **Funciones:** permiten realizar instrucciones y adicionalmente retornar un valor tras su ejecución.

- **Interrupciones:** tipo especial de rutinas que se definen con TRAP y que se ejecutan automáticamente cuando ocurre un evento, suspendiendo la ejecución del programa principal. Su ejecución se dispara mediante la instrucción CONNECT, que liga el evento con la interrupción.

Procedimientos:

```
PROC <nombre procedimiento> ( Lista de parámetros )
     <Lista de declaraciones de datos>;
     <Lista de instrucciones>;
     ERROR <lista instrucciones>;
ENDPROC
```

Funciones:

```
FUNC <tipo valor dato> ( Lista de parámetros)
     <Lista de declaraciones de datos>;
     <Lista de instrucciones>;
     RETURN dato;
     ERROR <lista instrucciones>;
ENDFUNC
```

Interrupciones:

```
TRAP <nombre trap>
     <Lista de declaraciones de datos>;
     <Lista de instrucciones>;
     ERROR <lista instrucciones>;
ENDTRAP
```

Figura 14. Tres tipos de rutinas en lenguaje RAPID.

Ejercicio práctico 5. Creación de un programa completo. Ejecución en modo AUTO.

1. Abre la solución del ejercicio práctico anterior (ejercicio 4). Deberás tener dos rutinas diferentes, una para cada pieza. Completa la rutina main() del Modulo 1 para que desde la misma se lance una vez cada una de las la rutinas de soldadura que tenemos.
2. Pon el robot en modo Automático usando el FlexPendant. Arma los motores (botón blanco). Abre la ventana de producción del FlexPendant para seguir la evolución del puntero de programa. Lanza la rutina main() desde el FlexPendant (puntero de programa PP to main + botón de Play) y comprueba el resultado.
4. En la pestaña "RAPID" o en el FlexPendant, busca el "modo de ejecución" del controlador, cámbialo a "continuo" y observa el resultado. Pulsa la seta de emergencia del FlexPendant para simular una parada de emergencia. Cuando haya pasado el peligro, rearma los motores y pulsa Play para continuar la ejecución
5. ¿De qué manera podemos ejecutar "instrucción a instrucción" en modo AUTO?

2.3. Instrucciones adicionales: control de flujo, E/S y esperas

Por último, existen instrucciones adicionales para controlar el flujo del programa, gestionar los valores de las entradas y salidas digitales del sistema y establecer esperas controladas por tiempo o controladas por los valores de una determinada entrada digital.

Las Figura 15 y la Figura 16 muestra las más importantes:

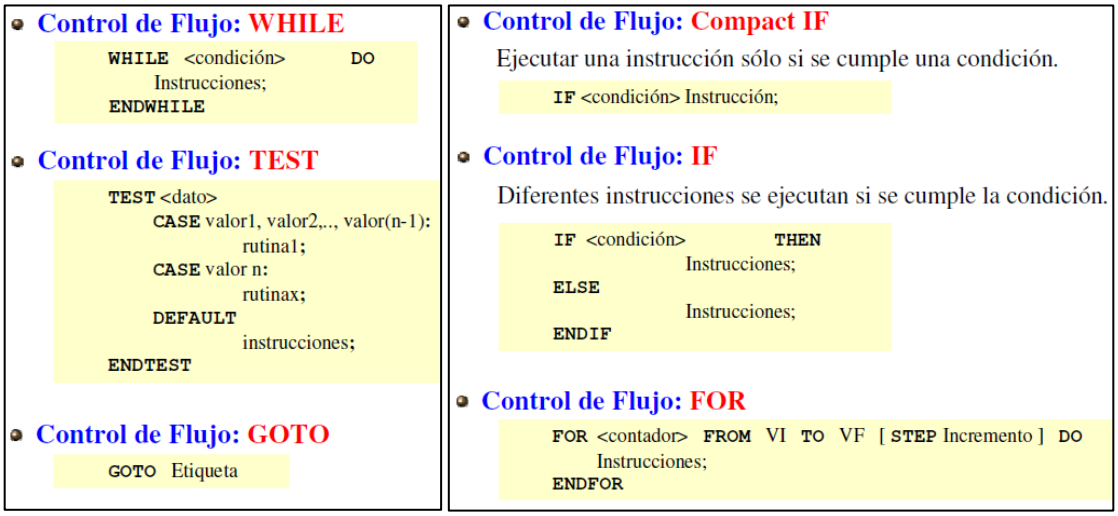

Figura 15. Instrucciones de control de flujo.

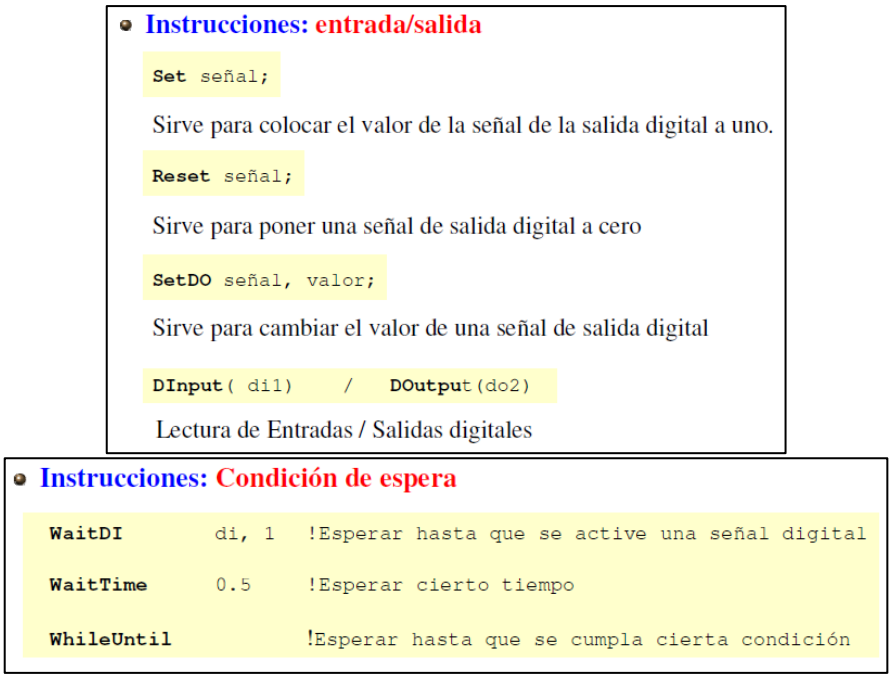

Figura 16. Instrucciones de gestión de Entradas/Salidas y de condiciones de espera.

2.4. Introducción de datos de operador desde FlexPendant

A veces resulta útil o necesario que el operador introduzca algún dato (texto plano o numérico) desde la pantalla del FlexPendant, para utilizar dicha información en la toma de decisiones durante ejecución del programa (indicación de número de piezas, selección manual de rutinas a ejecutar, etc.). También se pueden sacar datos simples o textos por pantalla (figura 17).

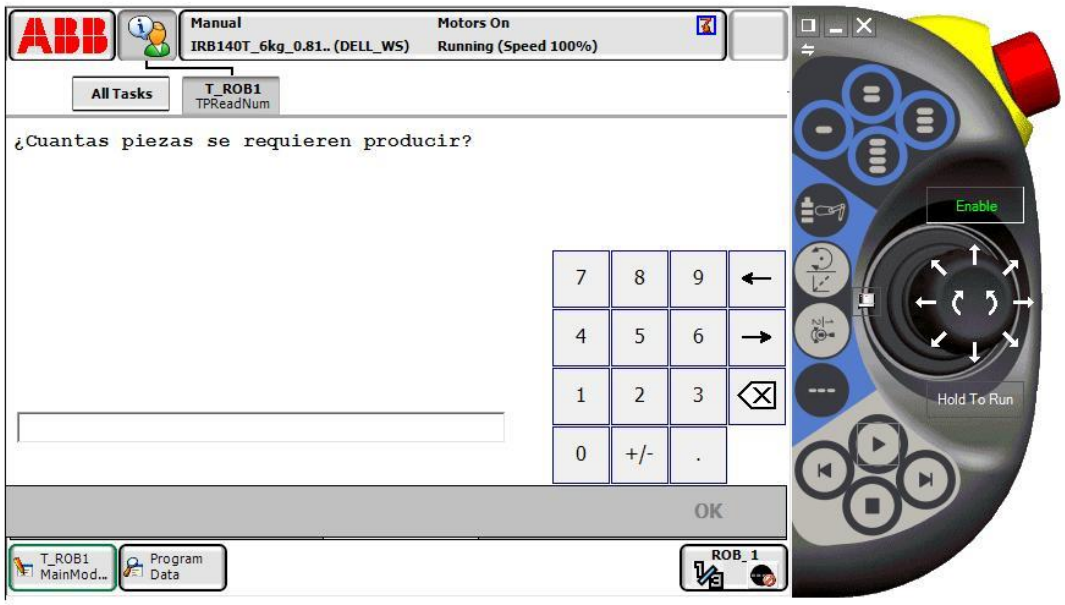

Figura 17. Visualización en la pantalla del FlexPendant de una instrucción de tipo TPReadNum

Para realizar estas acciones de manera sencilla se utilizan las siguientes instrucciones:

- **TPReadNum dato, "Pregunta";** Se muestra la pregunta en la pantalla del FlexPendant. Se para la ejecución del programa hasta que se introduce un número, que se almacena en "dato". Al inicio del módulo que definir "dato" como variable, mediante la declaración VAR num dato;
- **TPWrite "Texto";** Se muestra el "Texto" en la pantalla del FlexPendant. También permite mostrar valores de variables (para confirmar el valor introducido, por ejemplo).
- **TPReadFK (*FlexPendant Read Function Key*)** se utiliza para escribir un texto en las teclas de función y para determinar qué tecla de función se ha presionado (hay 5). Importante: devuelve un número igual a la posición de la opción.
    - VAR num opcion;
    - TPReadFK opcion, "Elige una opción", "A", "B", "C","D",stEmpty;
- **TPErase:** borrado de la pantalla del FlexPendant.

En el uso de estas instrucciones pueden ser útiles las instrucciones que permiten convertir cadenas de texto en valores numéricos (**strtoval**) y viceversa (**valtostr**).

Para más información, consulta la sintaxis y ejemplos de estas instrucciones en el manual de RAPID. Hay opciones más potentes (ProductionScreen o ScreenMaker) para la creación de pantallas con alto contenido gráfico basado en widgets, como la que aparece en la figura 18, que no vamos a ver en este curso.

Figura 18. Pantalla del FlexPendant creada con ScreenMaker

Ejercicio práctico 6. Programación de trayectorias en RAPID

1. Abre la solución del ejercicio práctico 5, que ya tendrá rutinas.
2. Usando el editor de RAPID, reprograma el main para que se pregunte al usuario a través del FlexPendant qué rutina quiere ejecutar (cubo o cilindro) y comprueba el resultado. Deberás utilizar **TPReadFK** y sentencias de control de flujo.
3. Amplía el programa para que, además pregunte cuántas veces quiere ejecutar la rutina con **TPReadNum**, partiendo de HOME y llegando a HOME sólo al inicio y final de la ejecución de todas las repeticiones. La pantalla del FlexPendant debe mostrar la rutina y la iteración en la que se encuentra. Lánzalo desde el FlexPendant en Modo AUTO.

4. Ejercicios entregables: programación en RAPID

Descarga el fichero "IRS_RI_Practica2.rspag" del Aula Virtual e instálalo en RobotStudio usando la opción Archivo, "Compartir", "Unpack and Work". Te aparecerá la estación que aparece en la Figura 19. Como ves, se trata de una copia del robot con la mesa de trabajo y los objetos (6 cubos y un pedestal) que tenemos en el laboratorio. La herramienta pinza ya está creada. Se acciona mediante la salida digital D16_O del controlador.

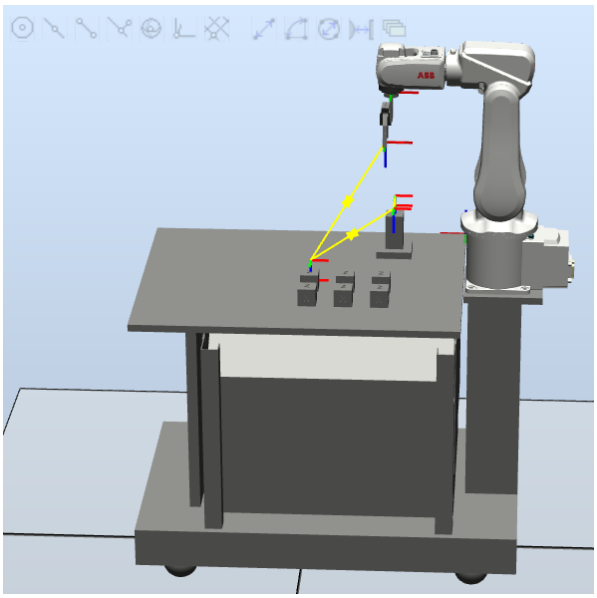

Figura 19. Estación que modela la mesa del laboratorio, que se usará para los ejercicios entregables.

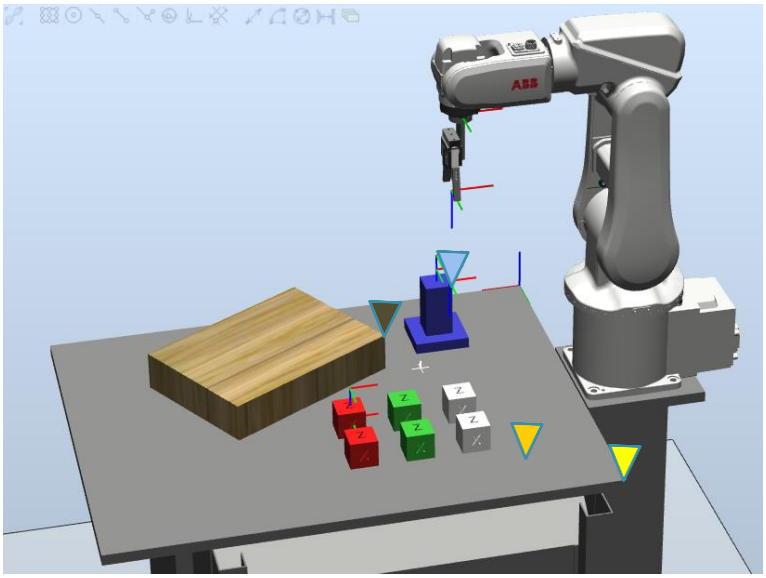

Figura 20. Setup para los ejercicios entregables. Los triángulos indican las posiciones de los tres objetos de trabajo que debes crear en el ejercicio entregable 1.

En esta práctica la mesa del robot, los cubos y del pedestal se diseñaron fuera de RobotStudio. Al importarlos como archivos STL (formato que sólo contiene una malla del volumen del objeto, no información sobre la geometría del objeto o las superficies del mismo), RobotStudio no sabe señalar los bordes ni los extremos con herramientas automáticas de ajuste y medida.

De todas formas, sí se puede determinar la posición y orientación exacta del extremo de cada objeto con Botón derecho (sobre el objeto)-> Definir posición -> Fijar Posición. Por otro lado, tened en cuenta que:

- El sistema de coordenadas "Mundo" se encuentra bajo la mesa.
- La base del robot IRB120 se encuentra en el punto (560, 0, 840)
- El extremo delantero derecho de la mesa se encuentra en el punto (452.5, -350, 840)
- Los cubos miden 50 x 50. El pedestal tiene una altura de 120 (todo en mm)

Ejercicio entregable 1 (3 puntos). Creación de objetos de trabajo con FlexPendant virtual

1. Sustituye la herramienta pinza del robot por la herramienta MyTool de la biblioteca de RobotStudio.
2. Añade una plataforma de dimensiones 350x225x90 mm, en una posición similar a la que se observa en la Figura 20, y con un giro con respecto al borde de la mesa de 45º.
3. Consulta los manuales de ABB y averigua cómo puedes crear un objeto de trabajo utilizando tres puntos pertenecientes a sistema de coordenadas usuario o u objeto.
4. Define mediante dicho método un primer objeto de trabajo “WO_Mesa” cuyo Uframe (base de coordenadas del usuario) se encuentre en la esquina de la misma marcada con un triángulo amarillo en la Figura 20, y orientada según los bordes de la mesa. El Oframe (base de coordenadas del objeto) se debe encontrar a 50 mm x 50 mm del cubo blanco (triángulo naranja), con los ejes paralelos al anterior (este segundo sistema puedes definirlo introduciendo las coordenadas manualmente).
5. Define un nuevo WO_Plataforma similar al anterior usando el método de los tres puntos donde ahora el Oframe deberá situarse sobre el triángulo marrón, y tener sus ejes paralelos a la plataforma.
6. Por último, define un último WO_Soporte para el soporte azul de la mesa.
7. Graba un vídeo donde se explique el procedimiento realizado para definir uno de los WO (no hace falta que sea de los tres), y donde se muestre la posición y orientación correcta de los tres WO creados.

ENTREGA en el AULA VIRTUAL: Vídeo del procedimiento, con audio explicativo.

Ejercicio entregable 2 (4 puntos, hasta 7 puntos considerando los extras). Generación de un programa completo en RAPID utilizando instrucciones de movimiento y de control de flujo

1. Vuelve a colocar la herramienta pinza en el robot.
2. Realiza un programa en RAPID que coloque los 6 cubos sobre el pedestal, todos con la misma orientación, y posteriormente los coloque siguiendo la distribución original que tenían sobre la mesa pero sobre la nueva plataforma de trabajo y siguiendo la orientación de la misma (WO_Plataforma).
3. Se valorará el uso de "puntos de aproximación" a las posiciones de coger y dejar cubos, el uso correcto de velocidades (rápidas hasta los puntos de aproximación, lentas de los puntos de aproximación a los de coger y dejar), la realización de programas con el mínimo número de instrucciones (uso adecuado de rutinas) y el uso de los tres WO creados en el ejercicio entregable 1. Se penalizará la existencia de colisiones.

   Puntos extra (se entregará siempre una única solución, ya sea sin extras, con el extra 1, con los extras 1 y 2 o con todos, indicándolo en el nombre de la solución):

4. **EXTRA 1 (1 punto):** Construye el programa de tal manera que se pueda especificar el número de cubos que se moverán (de 1 a 6), y que sólo realice el movimiento de dichos cubos al pedestal, y del pedestal a la plataforma. El número de cubos se especificará desde el FlexPendant.
5. **EXTRA 2 (1 punto):** Construye el programa de tal manera que si se activa una entrada digital ***DI_giro*** (que deberás definir) a 1 antes de comenzar la ejecución los cubos se suban a la plataforma incluyendo un giro de 90º en cualquiera de sus caras, de tal manera que al situarse sobre la nueva plataforma la cara superior sea distinta a la cara Z.
6. **EXTRA 3 (1 punto):** Construye el programa de tal manera que si se activa una entrada digital ***DI_retorno*** (que deberás definir) a 1 el robot espere 5 segundos al terminar y retorne todos los cubos a sus posiciones y orientaciones originales, sin pasar por el pedestal.

ENTREGA en el AULA VIRTUAL: Empaquetamiento de la solución creada que incluya los programas utilizando la opción "Pack and Go" del menú Archivo -> Compartir. El fichero .rspag se subirá al Aula Virtual. En este caso no hace falta vídeo.

REFERENCIAS Y BIBLIOGRAFÍA

[1] Lenguaje RAPID, Automatización Industrial II. Dr. Alaa Khamis, UC3M, 2006

[2] https://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK107046A8697&LanguageCode=en