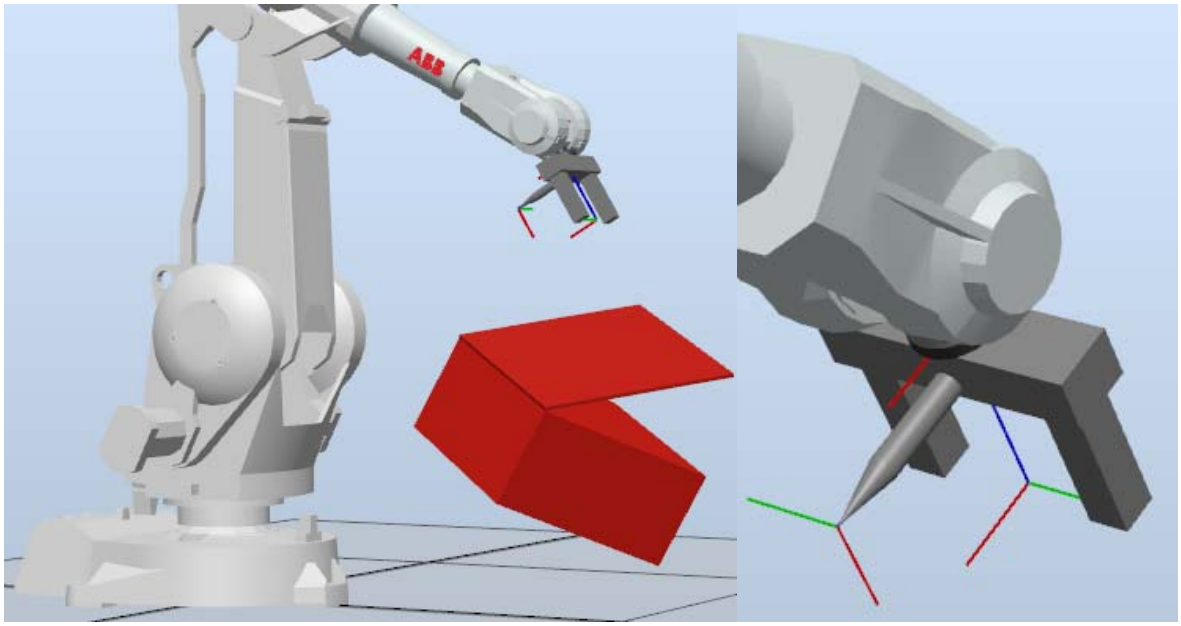
Programación y manejo de un robot ABB en un escenario virtual con RobotStudio

Objetivo:

La pretensión de esta actividad plasmar los conocimientos adquiridos sobre lenguajes de tipo textual explícito y de las herramientas de localización espacial. En el actual contexto de docencia virtual, esta actividad sustituye (yendo más allá) a las prácticas 3 y 4, que hubiesen tenido lugar de forma presencial con los robots del laboratorio. Se utilizará *RobotStudio* de ABB como herramienta de programación, así como la emulación que este ofrece de su *FlexPendant* (terminal de explotación de los robots de ABB). Para ello, deberéis descargaros el software desde https://new.abb.com/products/robotics/es/robotstudio/descargas. En la situación actual de pandemia por el Covid-19, ABB ha relajado la restricción de un mes en la disponibilidad de la funcionalidad *premium* desde la instalación, por lo que no hay que observar la precaución al respecto de la que os hablé en su día en clase.

Preliminares: descripción del punto de partida

Se suministra un archivo de extensión *rspag* (*RobotStudio Pack & Go*), realizado con la versión 2019.5.2 de *RobotStudio* (y 6.05.03 de *RobotWare*). Desempaquetado el fichero, aparecerá un sencillo escenario, inicialmente compuesto por un robot ABB IRB 2400 dotado de una herramienta (una garra que incorpora un puntero) similar a la disponible en los robots del laboratorio y un objeto arbitrariamente ubicado en el volumen de trabajo del robot:



La ubicación del sólido que aparece “flotando” en el espacio deberá ser modificada (aunque sólo sea ligeramente) antes de realizar los apartados 3 y 4. Para ello, se usará la opción “Fijar posición” empleada en la práctica 1. El objetivo es que cada uno de vosotros tenga que guiar el robot a posiciones diferentes para llevar a cabo los mencionados apartados.

En lo relativo a la herramienta suministrada, sus principales características son:

- Doble funcionalidad (garra+puntero), lo que obliga a contemplar dos TCP diferentes.
- Dimensiones: 200 x 50 x 150 mm, con un puntero de 150 mm. La apertura máxima de la garra es 120 mm.
- Apertura y cierre sincronizada de los dedos.
- Funcionalidad plena de cara al agarre/bliberación de objetos, por medio de la detección + conexión/desconexión automática de piezas al TCP de la herramienta como respuesta al cierre/apertura de dedos.
- Interfaz de usuario consistente en una única entrada (*Cerrar*) y 2 salidas (*Abierta* y *Cerrada*) que indican respectivamente la finalización de la apertura/cierre.
- Ha sido modelada por medio de un “*mecanismo*” incrustado en una “*componente inteligente*” de *RobotStudio*.

Para poder utilizarla en cualquier escenario hay que llevar a cabo los siguientes pasos (alguno de ellos ya realizados para el escenario a utilizar como punto de partida):

1. Crear una nueva solución / estación. Añadir un robot con su controlador (HECHO).
2. Copiar el archivo de librería a la carpeta donde esté la librería de usuario asociada a la solución sobre la que se esté trabajando (HECHO)
3. Importarla desde la librería de usuario y añadirla al escenario. Aparecerá un componente inteligente, que incorpora un objeto de tipo mecanismo (HECHO).
4. Conectar la componente inteligente al robot, contestando afirmativamente a la pregunta de si se desea actualizar la posición (HECHO).
5. Crear las señales de E/S binarias en el controlador del robot, con el fin de establecer posteriormente el conexionado de algunas de ellas con la herramienta HECHO: el escenario ya incorpora 5+5 E/S binarias (*DI1*, *DO1*, *DI2*, *DO2*, etc).
6. Las E/S de la interfaz de la herramienta deben ser conectadas a las correspondientes S/E del robot. Para ello, desde el menú *Simulación*, abrir la lógica de la estación y añadir las conexiones correspondientes entre el robot y la herramienta Esto puede hacerse desde la subpestaña *diseño* por medio del ratón, o bien mediante su definición directa desde la subpestaña *Señales y conexiones*. Conéctense las salidas *Cerrada* y *Abierta* de la garra a las estradas *DI4* y *DI5* del robot respectivamente. Conéctese asimismo la salida *DO1* del robot a la entrada *Cerrar* de la herramienta.
7. Desde el programa RAPID ya se pueden usar, tanto *SetDO* para abrir/cerrar, como *WaitDI* a fin de esperar el fin de cada una de las operaciones de apertura/cierre antes de proseguir con otros movimientos. Nótese que no es imprescindible el uso de las salidas Abierta/Cerrada, pues pueden ser sustituidas por sendas esperas temporales (*WaitTime*) suficientemente largas.

Documentación a elaborar / fecha de entrega

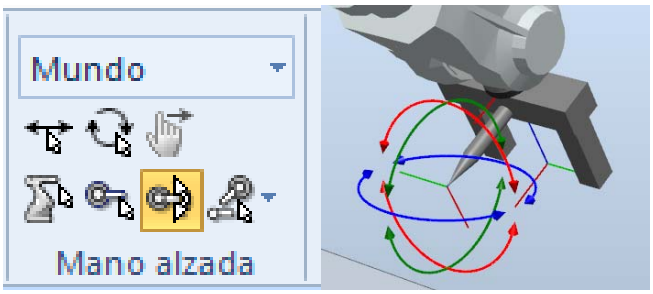
Antes del **29 de mayo**, se entregará a través de la tarea de Moodle creada a tal efecto, un informe con la resolución de los apartados 2, 3, 4 y 5, así como un archivo *Pack & Go* de *RobotStudio* con la aplicación final solicitada en los apartados 4 y 5.

Apartados a resolver

En la memoria final deberá incluirse la solución para todos los apartados, excepto para el 1º de ellos.

1.- Toma de contacto con las herramientas de guiado del robot, En *RobotStudio* existen varias formas de acometer el guiado de un robot virtual:

a) Mediante arrastre directo (tal y como se hizo en la práctica 1), utilizando las herramientas del bloque "*Mano alzada*" del menú "*Posición inicial*".



b) Mediante la emulación del *FlexPendant* (el terminal físico de explotación de ABB, que incorpora la botonera de guiado del robot real). El guiado requiere que el robot esté en modo manual. Tanto el *FlexPendant* como el interruptor con el modo de operación están accesibles desde el menú "Controlador":

El aspecto que tiene la emulación del *FlexPendant* es el que muestra la figura siguiente. Se puede seleccionar, tanto el espacio de guiado, como (en caso de guiado cartesiano) la referencia de guiado utilizada y la referencia TCP cuya localización se está modificando. También se muestran los ejes/variables afectados por el movimiento del *joystick*. Más información en las páginas 127-152 del *Manual de operador IRC5 con FlexPendant*.

Recuérdese habilitar el guiado por medio del correspondiente botón "*Enable*". Reseñar por último que se pueden mantener activas las dos formas de guiado, si bien, con el fin de tener una experiencia de usuario más cercana a la que se tendría con un robot real, se sugiere emplear el *FlexPendant*, pese a que en un principio pueda resultar más incómodo. Algunas ideas que surgen tras el guiado del robot en los espacios articular y cartesiano:

- El modelo geométrico del robot es fuertemente acoplado: la modificación de una sola coordenada articular, supone en general la modificación de todas las coordenadas de usuario. Al contrario sucede lo mismo: la modificación de una sola coordenada en el espacio de usuario suele requerir la variación de todas las coordenadas articulares.
- El guiado en el espacio de usuario es más amigable e intuitivo, no sólo por nuestra condición de usuarios, sino además porque en el objetivo de alcanzar una determinada localización, permite desacoplar el problema de alcanzar la posición, del problema de alcanzar la orientación, resolviéndolos de forma independiente.
- Normalmente, la referencia según la cual se lleva a cabo el guiado es la referencia "Mundo". Sin embargo, puede haber ocasiones en las que resulte interesante guiar el robot según la referencia TCP.

2.- Definición de una herramienta en RAPID (a realizar sobre el papel): la definición de la herramienta del robot puede llevarse a cabo por medio de varios procedimientos, en función del conocimiento disponible acerca de su geometría y sus parámetros dinámicos. En esta actividad, vamos a abordar su definición directa en RAPID mediante la instrucción de declaración / asignación de valores a una variable de tipo *tooldata*. Para ello, debe tenerse en cuenta la estructura de los tipos de dato *tooldata* y *loaddata* (pg. 1732-1738 y 1638-1643 de la referencia técnica de RAPID). *tooldata* recoge los parámetros geométricos (la localización de la referencia TCP respecto de su ubicación por defecto) y dinámicos (masa, localización del CDG, orientación de los ejes principales de inercia y valor de las inercias según dichos ejes) de la herramienta:

Como ejemplo de herramienta, se muestra a continuación la definición de la herramienta predefinida (*tool0*), cuyo origen está en el centro de la brida de montaje de herramienta *tool0* está definida en un módulo de sistema y no puede modificarse:

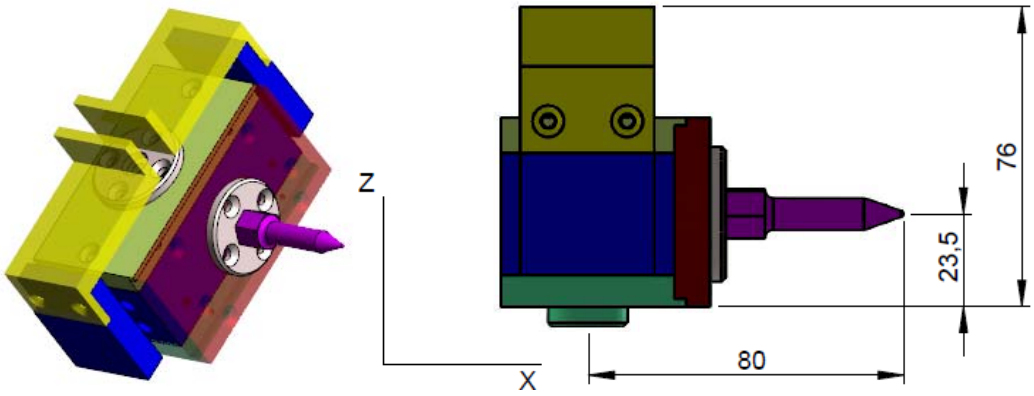
```
PERS tooldata tool0 := [ TRUE, [ [0, 0, 0], [1, 0, 0 ,0] ], [0.001,
      [0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0] ];
```

Se pide:

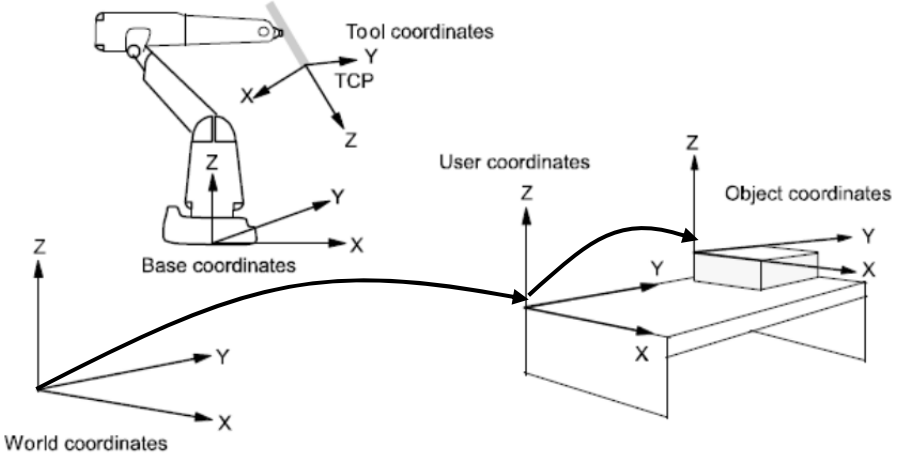
a) A la vista de la estructura del tipo de dato *tooldata*, defínase sobre el papel una herramienta de tipo puntero, considerando una punta de 105 mm de longitud y 140 gramos de peso, con su centro de gravedad situado a 20 mm de la base y con unas inercias en su cdg equivalentes a las que tendría una varilla del mismo peso y longitud respecto de su cdg. En cuanto a la orientación del TCP, se pide definirla a través del correspondiente cuaternio de 2 formas diferentes:

- Invirtiendo la orientación de los ejes *X* y *Z* del TCP por defecto, pero manteniendo la orientación del eje *Y*.
- Invirtiendo la orientación del eje Z del TCP por defecto e intercambiando los ejes X e Y.

b) En cuanto a la herramienta "Garra con puntero", establézcanse sobre el papel sendos TCP diferenciados, uno para la funcionalidad de garra (punto medio entre los dedos en su extremo) y otro asociado al puntero. La localización de su CDG (680 gramos) es 5, 0, 25 respecto de *tool0* (se considerará como una masa puntual).



3.- Definición de *WorkObjects* (a realizar sobre el *Flexpendant*): RAPID da soporte a una estructuración básica del entorno del robot en 2 niveles por medio de los *WorkObjects*. Estos elementos incorporan la localización de una referencia de usuario respecto del mundo y la localización de una referencia de objeto respecto de la de usuario:



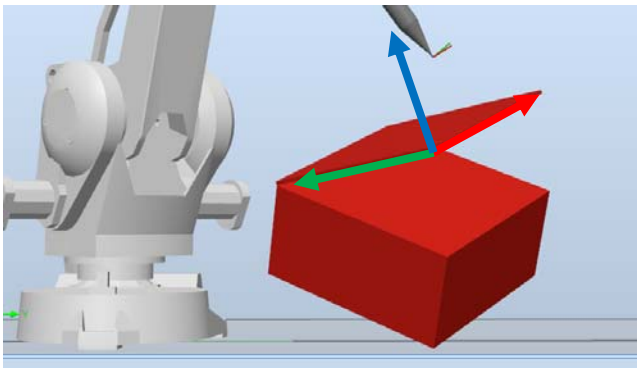
Esta estructuración en 2 niveles permite especificar localizaciones de destino en coordenadas de objeto, por medio de la inclusión del objeto de trabajo en las instrucciones de movimiento. La definición de los objetos de trabajo puede llevarse a cabo por medio de varios procedimientos, en función del conocimiento disponible del entorno del robot.

- Por medio de la definición directa en RAPID mediante la instrucción de declaración / asignación de valores a una variable de tipo *wobjdata*.
- Por medio de la definición interactiva desde el *FlexPendant.*

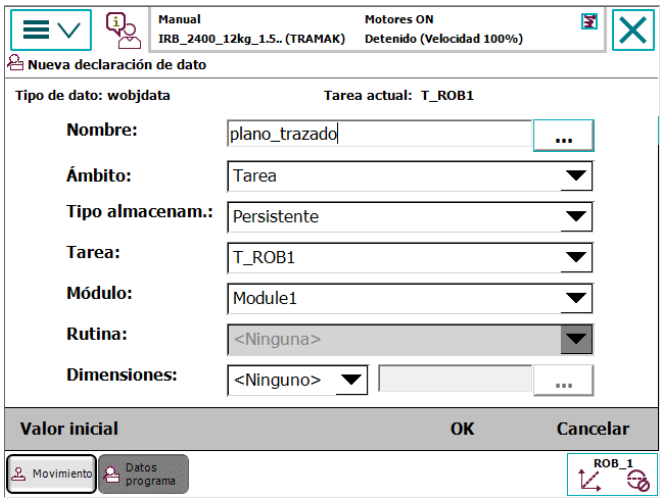
Para la **definición directa en RAPID**, debe tenerse en cuenta la estructura del tipo de dato *wobjdata* (pg. 1759-1762 del manual de referencia técnica de RAPID). Como ejemplo, se muestra a continuación la definición del objeto de trabajo predefinido (*wobj0*), que coincide con la referencia Mundo.

```
PERS wobjdata wobj0 := [ FALSE, TRUE, "", [ [0, 0, 0], [1, 0, 0
     ,0] ], [ [0, 0, 0], [1, 0, 0 ,0] ] ];
```

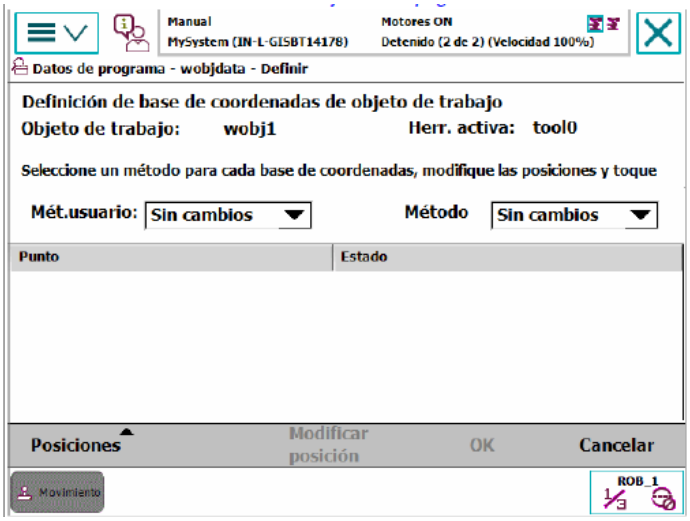
Existen no obstante procedimientos de referenciado basados en guiado del robot, que permiten obtener **de forma interactiva** la localización de las referencias de usuario y de objeto. **Se pide**: crear un objeto de trabajo sobre una determinada superficie del sólido de color rojo arbitrariamente localizado en la escena, con el fin de poder trabajar sobre ella posteriormente. La siguiente figura muestra la referencia de objeto del *WorkObject* pretendido. Como se ha comentado anteriormente, es obligatorio realizar un cambio de orientación del sólido, antes da acometer el procedimiento de referenciado.



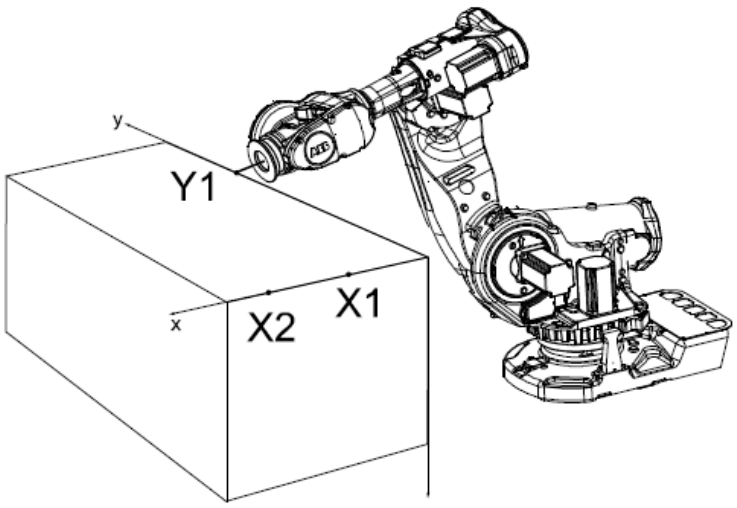
El procedimiento de trabajo a seguir está descrito en las pgs. 204-207 del manual de operador IRC5 con FlexPendant. Desde el menú *Datos de programa*, pulsamos sobre *wobjdata* para ver la lista de WorkObjects disponibles y pulsamos en *Nuevo*. Ello dará lugar a un cuadro de diálogo que iremos rellenando con los valores adecuados (asignarle el nombre "*plano_trazado*". En cuanto al módulo en el que va a ser almacenado, se sugiere emplear el mismo módulo en el que estén las aplicaciones que hagan uso del objeto de trabajo (*Module1*):



Una vez introducidos los valores, pulsamos *OK*, y en la siguiente pantalla (en la que aparece seleccionado el nuevo WorkObject, desde el menú *Editar*, seleccionamos *Definir*.



En lo relativo a las localizaciones de las referencias de usuario y de objeto, emplearemos el método de los tres puntos de ABB, parecido al que ya fue implementado sobre Matlab en la práctica 2. Las siguientes instrucciones están directamente extraídas del manual de usuario:



	Acción	Información
1	En el menú Método de usuario, toque 3 puntos.	
2	Presione el dispositivo de habilitación de tres posiciones y mueva el robot hasta el primer punto (X1, X2 o Y1) que desee definir.	El uso de una distancia elevada entre X1 y X2 resulta preferible y permite obtener una definición más exacta.
3	Seleccione el punto en la lista.	
4	Toque Modificar posición para definir el punto.	
5	Repita los pasos del 2 al 4 con los demás puntos.	

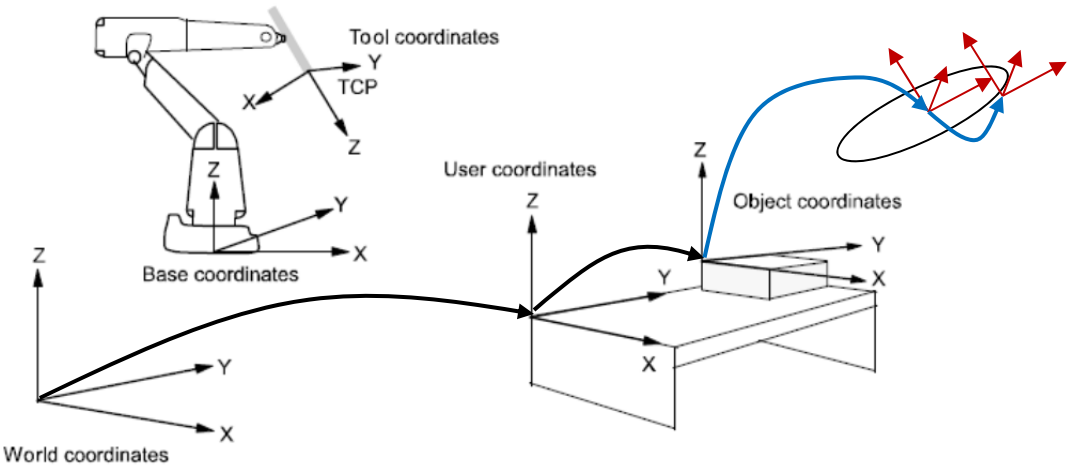
Por simplicidad, se sugiere prescindir de uno de los dos niveles de estructuración, definiendo únicamente la localización de la referencia de usuario ("*método de usuario*") y dejando la localización de la referencia de objeto en sus valores por defecto (transformación nula), por lo que la referencia de objeto será la misma que la de usuario. Tendréis que ir combinando el guiado con sucesivas pulsaciones de "*Modificar posición*". Una vez introducido el punto 3, pulsad OK: se llevará a cabo el cálculo de la localización correspondiente, obteniendo una ventana como la que sigue. **¡Atención!** hay que liberar el dispositivo de guiado (botón "*Enable*", que estaba en verde), antes de volver a pulsar OK.

Tras ello, aparecerá una línea en el código fuente, con la definición de *plano_trazado*.

Tras esos 2 últimos pasos, automáticamente habrá aparecido una línea en el código fuente de RAPID, con la definición del WorkObject denominado *plano_trazado*.

4.- Desarrollo de una aplicación (parte I): uso del puntero (a realizar sobre el editor de RAPID de *RobotStudio*). Los objetos de trabajo de RAPID aportan una estructuración básica del entorno del robot, que en muchos casos puede resultar suficiente para nuestros propósitos. Sin embargo, en aquellos casos de mayor complejidad, es necesario incrementar el número de niveles de la estructura, recurriendo a las posibilidades que brinda el lenguaje. Por ejemplo, supóngase un escenario que incluye un objeto de trabajo, respecto del cual vamos a trazar una figura geométrica plana (una circunferencia) a partir de la localización de una referencia asociada a su centro geométrico. Dicha referencia "centro", cuya localización vendrá definida en coordenadas locales al objeto de trabajo, permitirá especificar, no sólo la posición del centro de la figura, sino también la orientación del plano de trazado (el plano definido por sus ejes X e Y). Los puntos que definen la figura se pueden especificar

fácilmente en coordenadas locales a dicha referencia. La incorporación de esta referencia supone la adición de facto de un nivel más a la estructura que había sido creada en el apartado anterior, tal y como muestra el siguiente esquema:



Se pide:

4.1.- Créense 2 procedimientos en RAPID (*Pinta_cuadrado* y *Pinta_circunferencia*) que dibujen respectivamente, un cuadrado de 100 mm de lado o su circunferencia circunscrita (de $100\sqrt{2}$ mm de diámetro). Cada procedimiento tendrá al menos como argumento de llamada, la localización (de tipo *robtarget*) del centro geométrico de la figura respecto del *WorkObject* definido en el punto anterior (*plano_trazado*). El procedimiento el trazado de la figura correspondiente deberá atenerse a las siguientes directrices:

- Todos los movimientos deberán realizarse a una velocidad limitada (se sugiere emplear como velocidad máxima V100).
- El primer movimiento (coordinado articular) permitirá alcanzar una aproximación de 50 mm al primer punto del trazado, definida según la perpendicular al plano de trazado. La orientación alcanzada se corresponderá con la que se llevará a cabo el trazado de la figura.
- El segundo movimiento alcanzará en línea recta el primer punto de trazado con terminación *fine*.
- Tras el trazado de la figura geométrica, el último movimiento permitirá alcanzar (en línea recta) de nuevo la localización utilizada como aproximación.
- El procedimiento incluirá la declaración y definición de las localizaciones necesarias (como variables). Úsese una de las siguientes funciones proporcionadas por el lenguaje (funciones *RelTool* o *PoseMult*).

4.2.- Escríbase el programa principal (*main*), considerando los procedimientos anteriores y de forma que se atenga a las siguientes directrices:

- Será explícitamente cíclico (mediante estructura iterativa incluida en el código fuente), repitiendo el trazado de una figura geométrica por medio de la llamada a uno de los dos procedimientos generados en el apartado anterior. La condición de salida de la estructura cíclica, vendrá determinada por la activación de la entrada binaria número 1 (*DI1*).
- El tipo de figura a trazar vendrá determinado por el valor de la entrada binaria *DI2*.

- Elíjase una localización para la figura a una cierta distancia del plano de trazado, con el fin de asegurar la ausencia de contacto.
- La entrada a la estructura cíclica vendrá precedida de un movimiento coordinado articular a una configuración de reposo ya definida (casa). Del mismo modo, tras salir de la estructura cíclica, el robot será llevado a dicha configuración de reposo antes de finalizar la ejecución del programa.

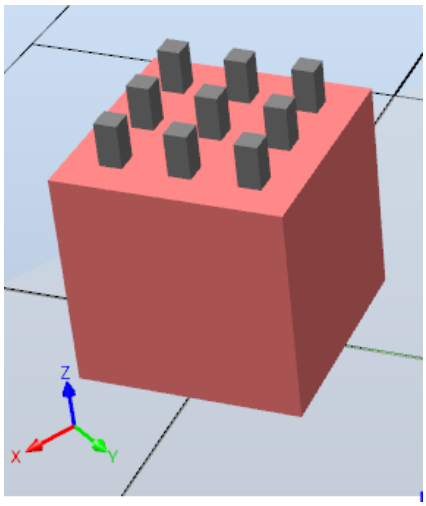
4.3.- Repítase el apartado anterior de forma que el **trazado** de la figura se lleve a cabo **en un plano inclinado** respecto el plano de trazado original (por ejemplo 30º). Para ello se sugiere modificar la definición del *robtarget* que define la localización del centro, levantándolo además respecto del objeto, con el fin de no colisionar con él. ¿Qué impacto tiene el uso de *Offs* en lugar de *RelTool*? Explica el porqué de las diferencias.

4.4.- Incorpórese una rutina de interrupción, con el fin de procurar una parada de emergencia ante la activación de la entrada binaria nº 3 (*DI3*). Para ello, será necesario:

- Declarar una interrupción como dato de tipo *intnum* (pg 1631 del manual de referencia técnica de RAPID).
- Asociar el disparo de dicha interrupción con la ejecución de la correspondiente rutina de servicio de interrupción mediante *CONNECT* (pg 139 del manual de referencia técnica de RAPID).
- Definir las condiciones de disparo de la interrupción, asociándola al nivel alto de la entrada binaria nº 3, por medio de *ISignalDI* (pg 332 del manual de referencia técnica de RAPID).
- Programar la rutina de servicio de interrupción (*TRAP*), de forma que se detenga el movimiento del robot (úsese *StopMove* para ello). La ejecución de la rutina de servicio deberá tener lugar hasta que la señal binaria que activó la interrupción pase a valer 0 (úsese *WaitDI*). La reanudación del movimiento del robot tendrá lugar por medio de la instrucción *StartMove* (será la última instrucción de la rutina). Con el fin de hacer la espera más amena, se sugiere incluir la activación de una salida binaria y/o la escritura de un mensaje en la ventana de explotación del *FlexPendant* (por medio de *TPWrite*).

5.- Desarrollo de una aplicación (parte II): uso de la garra (a realizar sobre el editor de RAPID de *RobotStudio*). El objetivo de este apartado es completar la aplicación realizada hasta ahora, de forma que al salir del bucle de trazado de cuadrados/circunferencias (mediante la activación de la entrada *DI1*) y tras el retorno del robot a su configuración de reposo, se proceda a mover individualmente determinados objetos desde una distribución organizada en una cara de un paralepípedo a una distribución similar en otra cara del mismo. El alcance del mencionado objetivo se plantea en dos etapas diferenciadas: la creación de la parte de la escena correspondiente y la implementación de la parte de la aplicación en RAPID.

5.1.- Generación del escenario de partida: Añádase a la escena con la que se ha estado trabajando un objeto poliédrico apoyado en el suelo a modo de mesa, de forma que las aristas de su base no sean paralelas a los ejes principales. Ubíquense sobre la cara superior de dicho objeto una matriz 3 x 3 de elementos de dimensiones compatibles con la garra, con vistas a su posterior manipulación. La siguiente figura muestra un ejemplo ilustrativo de la escena:



La forma que se sugiere de llevar a cabo la creación de esta parte de la escena pasa por asociar un sistema de coordenadas a la base de la mesa, convirtiéndolo posteriormente en referencia UCS. De esta forma, resulta muy sencillo ubicar otros elementos en su parte superior, suministrando sus correspondientes coordenadas UCS. Conviene asimismo recordar que es posible copiar y pegar objetos, lo que sin duda resulta muy útil en este caso. Al elemento "mesa" **únicamente se le podrá asociar un objeto de trabajo RAPID** (ubicado preferentemente en su base).

5.2.- Habilítese el uso de la garra, siguiendo los pasos 6 y 7 del apartado "preliminares". Una vez realizado el conexionado, inclúyanse sendos procedimientos *abrir* / *cerrar* que encapsulen las mencionadas operaciones, de forma que no se dé por finalizada su ejecución hasta que no haya concluido el correspondiente movimiento de apertura / cierre.

5.3.- Inclúyase un procedimiento (*coger_y_dejar*), de forma que en cada llamada se recoja un elemento de la matriz del plano de origen, depositándolo en la correspondiente localización del plano de destino (otra cara de la mesa). Obsérvense las siguientes directrices:

- Establézcanse los argumentos que se consideren necesarios para tal fin. Pueden ser variados: desde los índices de fila y columna de la matriz, o de forma alternativa, las localizaciones desde las que coger y dejar, o cualesquiera otros que consideren oportunos.
- Se deberá considerar la estructuración adecuada de la escena, de forma que se facilite la selección del plano de la mesa en la que se toman y depositan los objetos. En este sentido, deberá considerarse como parte de la estructura el *WorkObject* asociado a la base de la mesa (el único que la mesa tendrá, como se ha especificado anteriormente).
- La definición de la orientación de las localizaciones involucradas deberá facilitar el alcance cómodo por parte del robot.
- Los movimientos de aproximación / despegue deberán ser rectilíneos (sin cambio de orientación) y a una velocidad reducida.
- Se sugiere emplear cuantas localizaciones de paso resulten necesarias, con el fin de evitar colisiones.

5.4.- Complétese el programa principal, para mover toda la distribución organizada de objetos desde la cara superior de la mesa (en la que se encuentran inicialmente) a una de las caras laterales de la misma. Obsérvense las siguientes directrices:

- Al igual que para el procedimiento anterior, deberá ser tenida en cuenta la adecuada estructuración de la escena, de forma que se facilite la selección del plano de la mesa en la que se toman y depositan los objetos (mediante sendas transformaciones que, desde el objeto de trabajo, permitan localizar el centro de las caras de la mesa origen y destino).
- Inclúyanse las estructuras iterativas y los tipos de datos adecuados para la tarea a realizar.
- Se valorará, tanto la estructuración del entorno, como la estructuración del código fuente, su modularidad / mantenibilidad / legibilidad.

ANEXOI: tipos de datos, instrucciones y funciones de RAPID a emplear en la práctica

Tipos de datos	Instrucciones		Funciones
tooldata	*MoveAbsJ*	*StopMove*	*TestDI*
wobjdata	*MoveJ*	*SetDO*	*Offs*
jointtarget	*MoveL*	*TPWrite*	*RelTool*
robtarget	*MoveC*	*WaitDI*	*PoseMult*
intnum	*CONNECT*	*StartMove*	

ANEXO II: Plantilla con código fuente

```
MODULE Module1
    PERS tooldata Garra_TCP:=[TRUE,[[0,0,150],[0,1,0,0]],[1,[0,0,10],[1,0,0,0],0,0,0]];
    PERS tooldata Puntero_TCP:=[TRUE,[[150,0,30],[0.707106781,0,-
0.707106781,0]],[1,[0,0,10],[1,0,0,0],0,0,0]];
    CONST confdata config_fun:=[0,0,0,0];
    CONST confdata config_fuf:=[0,0,0,1];
    CONST extjoint ejes_externos:=[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09];
    CONST jointtarget casa:=[[0,0,0,0,0,0],ejes_externos];

    PROC Pinta_cuadrado(robtarget centro, PERS wobjdata objeto_trabajo)
        ! Completa el código fuente
    ENDPROC

    PROC Pinta_circunferencia(robtarget centro, PERS wobjdata objeto_trabajo)
        ! Completa el código fuente
    ENDPROC

    PROC cerrar()
        ! Completa el código fuente
    ENDPROC

    PROC abrir()
        ! Completa el código fuente
    ENDPROC

    PROC main()
        MoveAbsJ casa, v1000, fine, Puntero_TCP;
        WHILE DI1=0 DO
            ! Completa el código fuente
        ENDWHILE
        MoveAbsJ casa, v1000, fine, Puntero_TCP;
    ENDPROC

    TRAP emergency_routine
        ! Completa el código fuente
    ENDTRAP

ENDMODULE
```