

Table of Contents

- [1 Introducción](#)
- [2 Configuración](#)
- [3 Definición de una herramienta en RAPID](#)
- [4 Definición de WorkObjects](#)
- [5 Desarrollo de una aplicación\(Parte I\)](#)
- [6 Desarrollo de una aplicación\(Parte II\)](#)
- [7 Conclusión](#)

Introducción

El objetivo de este trabajo es aprender a programar una aplicación para controlar un robot de seis ejes; en este caso el ABB-IRB2400. Para ello se utilizara el entorno de RobotStudio y el lenguaje de programación RAPID.

Configuración

Ignorar este apartado, solo permite la inclusión de código MATLAB en el cuaderno. Saltar a la siguiente sección.

In [1]:

```
import matlab.engine
import io
from IPython.core.magic import register_cell_magic
ip = get_ipython()

out = io.StringIO()
err = io.StringIO()

# Setup matlab cell magic #
@register_cell_magic
def matlab_magic(line, cell):
    out.truncate(0)
    out.seek(0)
    err.truncate(0)
    err.truncate(0)
    raw = r'{line}.eval("""{cell}""", nargout=0, stdout=out, stderr=err)'
    ip.run_cell(raw.format(line=line, cell=cell))
    print(out.getvalue())
    print(err.getvalue())
```

In [2]:

```
eng = matlab.engine.start_matlab() # start matlab session
eng.eval("format compact", nargout=0) # compact formatting in matlab's stdout
```

Definición de una herramienta en RAPID

Para definir la herramienta, podemos utilizar el tipo de dato `tooldata`. Como el robot sostiene la herramienta `robhold` será `TRUE`. Para la componente `tframe` definiremos la posición del TCP (en mm), expresada en el sistema de coordenadas de la muñeca `tool0`, así como su orientación. Por último la componente `loaddata`, que caracteriza los parámetros dinámicos de la herramienta, viene definida a su vez por cuatro subcomponentes: `mass` para la masa(en kg), `cog` para el centro de gravedad(en mm), `aom` para la orientación de los ejes de inercia e `ix` `iy` `iz` para los valores de dichas inercias(en kgm^2).

Así pues, la estructura desglosada de `tooldata` tiene el siguiente aspecto:

```
< dataobject of tooldata >
  < robhold of bool >
    < tframe of pose >
      < trans of pos >
        < x of num >
        < y of num >
        < z of num >
      < rot of orient >
        < q1 of num >
        < q2 of num >
        < q3 of num >
        < q4 of num >
    < tload of loaddata >
      < mass of num >
      < cog of pos >
        < x of num >
        < y of num >
        < z of num >
      < aom of orient >
        < q1 of num >
        < q2 of num >
        < q3 of num >
        < q4 of num >
      < ix of num >
      < iy of num >
      < iz of num >
```

Se nos pide lo siguiente:

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.



donde se han resaltao algunos parámetros que usaremos para la definición de la herramienta. La varilla, supuesta infinitamente delgada a efectos de cálculos, tiene por lo tanto inercia cero en el eje longitudinal z y, en los ejes transversales x e y viene dada por:

$$I_x = I_y = \frac{m\ell^2}{12} = \frac{0.140 \times 0.02^2}{12} = 4.66 \times 10^{-6} \text{ kgm}^2$$

donde m y ℓ son la masa y la longitud del centro de masa a la base, respectivamente.

RAPID pide al usuario expresar la orientación en forma de cuaternios, por lo tanto, distinguiendo entre las dos opciones:

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

Que podemos traducir en una rotación de $\frac{\pi}{2}$ sobre el eje y . El cuaternio viene dado por:

$$\mathbf{Q}_1 = \left[\cos \frac{\theta}{2}, \mathbf{k} \sin \frac{\theta}{2} \right] = \left[\cos \frac{\pi}{4}, (0, 1, 0) \sin \frac{\pi}{4} \right] = [0.707, 0, 0.707, 0]$$

- Invirtiendo la orientación del eje Z del TCP por defecto e intercambiando los ejes X e Y.

Que podríamos expresar como composición de cuaternios. Sin embargo, es cómodo expresar una matriz de rotación y hallar las componentes de \mathbf{Q}_2 :

$$\mathbf{R} = [\mathbf{n} \ \mathbf{o} \ \mathbf{a}] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

y sabiendo que

$$q_0 = \frac{1}{2} \sqrt{n_x + o_y + a_z + 1}$$

$$q_1 = \text{sign}(o_z - a_y) \frac{1}{2} \sqrt{n_x - o_y - a_z + 1}$$

$$q_2 = \text{sign}(a_x - n_z) \frac{1}{2} \sqrt{-n_x + o_y - a_z + 1}$$

$$q_3 = \text{sign}(n_y - o_x) \frac{1}{2} \sqrt{-n_x - o_y + a_z + 1}$$

donde sign toma el signo de su argumento. Sustituyendo obtenemos

$$q_0 = \frac{1}{2} \sqrt{0 + 0 - 1 + 1} = 0$$

$$q_1 = \text{sign}(0 - 0) \frac{1}{2} \sqrt{0 - 0 + 1 + 1} = 0.707$$

$$q_2 = \text{sign}(0 - 0) \frac{1}{2} \sqrt{-0 + 0 + 1 + 1} = 0.707$$

$$q_3 = \text{sign}(1 - 1) \frac{1}{2} \sqrt{-0 - 0 - 1 + 1} = 0$$

Y por lo tanto,

$$\mathbf{Q}_2 = [0, 0.707, 0.707, 0]$$

Verifiquemos con MATLAB:

In [3]:

```
%%matlab_magic eng

Q1 = eul2quat([0 pi/2 0], 'XYZ')

R = [0 1 0; 1 0 0; 0 0 -1];
Q2 = rotm2quat(R)
```

```
Q1 =
    0.7071         0    0.7071         0
Q2 =
         0    0.7071    0.7071         0
```

Como queríamos demostrar.

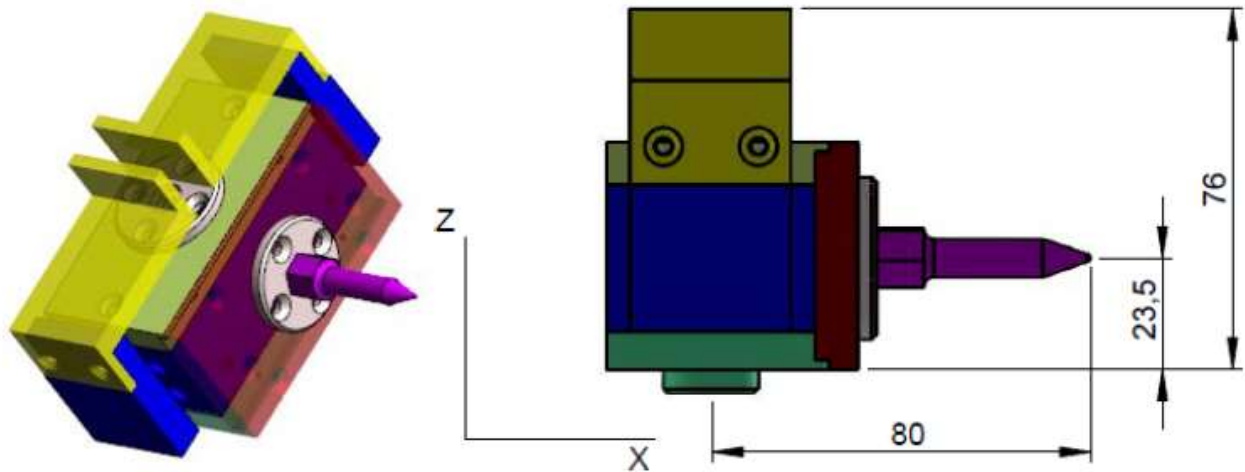
Con los datos obtenidos podemos definir ambas herramientas en RAPID:

```
PERS tooldata puntero1 := [ TRUE, [[0, 0, 105], [0.707, 0, 0.707, 0]], [0.14, [0, 0, 20], [1, 0, 0, 0], 4.66e-6, 4.66e-6, 0]];
```

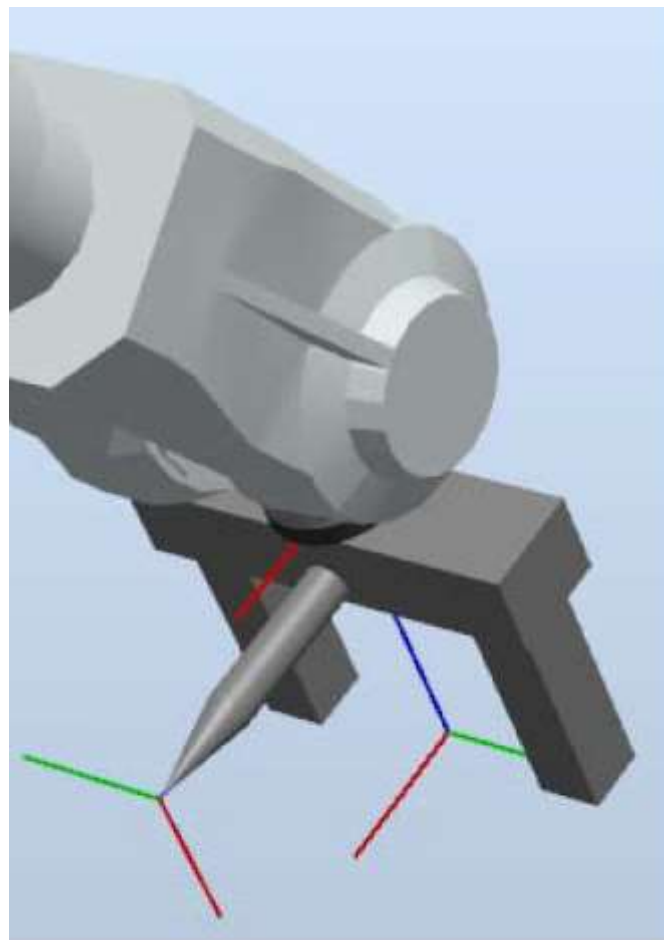
```
PERS tooldata puntero2 := [ TRUE, [[0, 0, 105], [0, 0.707, 0.707, 0]], [0.14, [0, 0, 20], [1, 0, 0, 0], 4.66e-6, 4.66e-6, 0]];
```

donde cabe destacar que `aom` usa el quaternion unidad `[1, 0, 0, 0]` porque hemos calculado los momentos de inercia para los ejes originales.

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).



donde buscamos unos TCP con la orientación que se presenta en la figura:



El momento de inercia en una masa puntual viene dado por $I = mr^2$. Así pues, si el CDG esta a (5, 0, 25) respecto de tool0 , se tiene para cada eje:

$$I_x = 0.68 \times 0.005^2 = 1.7 \times 10^{-5} \text{ kgm}^2$$

$$I_y = 0.68 \times 0^2 = 0 \text{ kgm}^2$$

$$I_z = 0.68 \times 0.025^2 = 4.25 \times 10^{-4} \text{ kgm}^2$$

Los cuaternios que expresan orientación de ambos TCP se puede obtener analíticamente de manera idéntica a como se ha explicado anteriormente; Es por ello que usaremos MATLAB para transformar las matrices de rotación:

$$R_{garra} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad R_{puntero} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Y, en forma de cuaternios:

In [4]:

```
%%matlab_magic eng
```

```
Rgarra = [1 0 0; 0 -1 0; 0 0 -1];
```

```
Q3 = rotm2quat(Rgarra)
```

```
Rpuntero = [0 0 -1; 0 1 0; 1 0 0];
```

```
Q4 = rotm2quat(Rpuntero)
```

```
Q3 =
```

```
    0    1    0    0
```

```
Q4 =
```

```
 0.7071    0 -0.7071    0
```

Formateando,

$$\begin{aligned} \mathbf{Q}_3 &= [0, 1, 0, 0] \\ \mathbf{Q}_4 &= [0.707, 0, -0.707, 0] \end{aligned}$$

para la garra y el puntero, respectivamente.

Con estos resultados y basándonos en la figura que describe las dimensiones de la herramienta podemos definirlos en RAPID de la siguiente forma:

```
PERS tooldata garra := [ TRUE, [[0, 0, 76], [0, 1, 0, 0]], [0.68, [5, 0, 25], [1, 0, 0, 0], 1.7e-5, 0, 4.25e-4]];
```

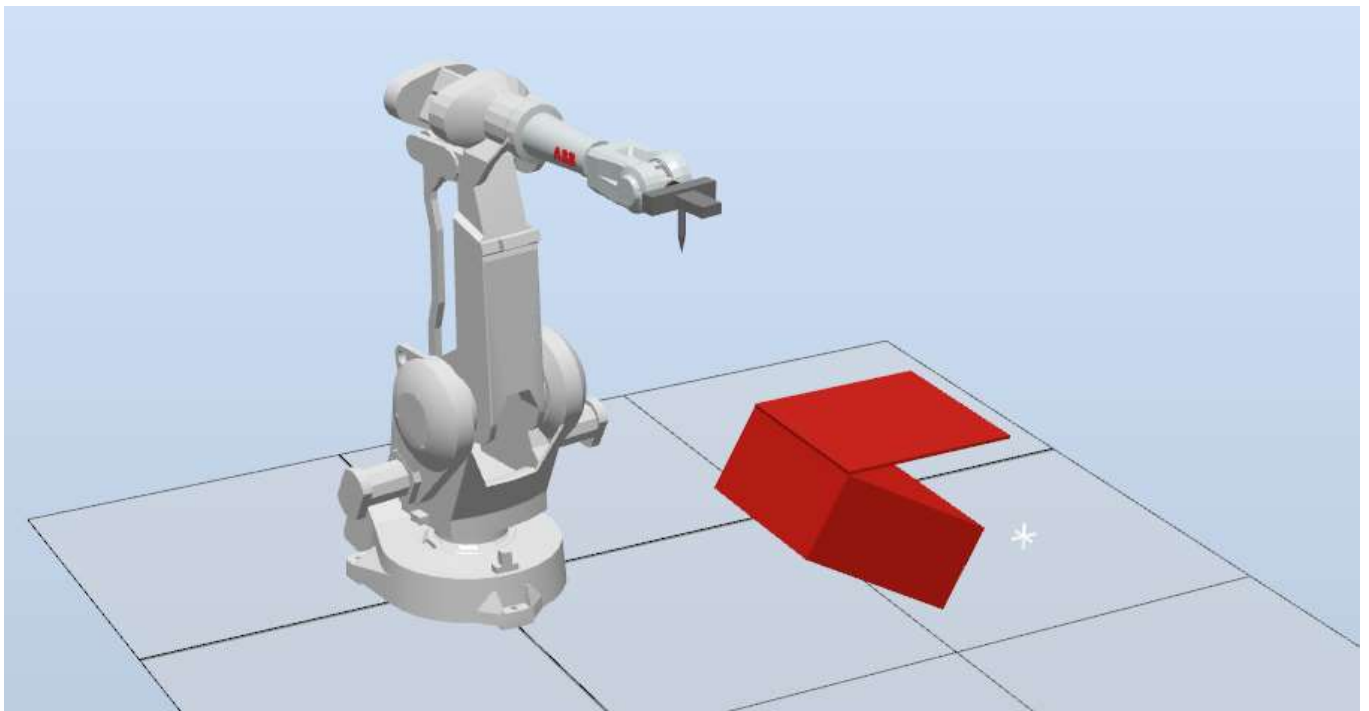
```
PERS tooldata puntero := [ TRUE, [[80, 0, 23.5], [0.707, 0, -0.707, 0]], [0.68, [5, 0, 25], [1, 0, 0, 0], 1.7e-5, 0, 4.25e-4]];
```

Definición de WorkObjects

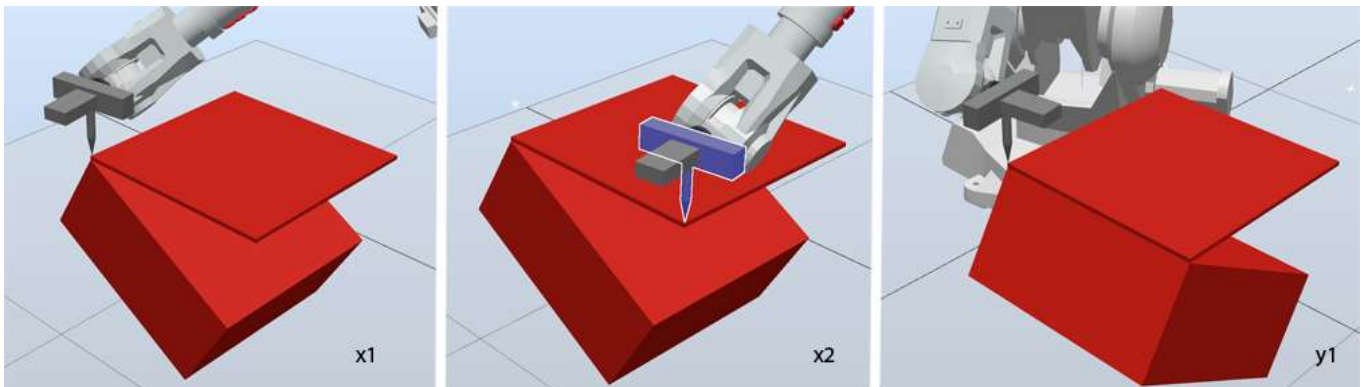
Definición de WorkObjects (a realizar sobre el Flexpendant): RAPID da soporte a una estructuración básica del entorno del robot en 2 niveles por medio de los WorkObjects. Estos elementos incorporan la localización de una referencia de usuario respecto del mundo y la localización de una referencia de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

RAPID da soporte a una estructura de localización en dos niveles, el primero de una referencia de usuario respecto del mundo y el segundo una referencia de objeto respecto de la de usuario. Podemos definirlos mediante el tipo de dato `wobjdata`, que está formado por algunos parámetros funcionales y por supuesto dos `pose`s, una para cada nivel(arco). Sin embargo, es interesante definirlo en el FlexPendant de forma interactiva.

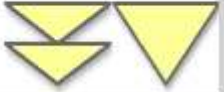
Como ejemplo, escogiendo una posición y orientación arbitraria para la caja con tapa del escenario de prueba, procedemos a definir un *WorkObject* solidario a la tapa.



Podemos definir la referencia mediante tres puntos: dos de ellos en el eje x y otro en el eje y , tal y como se muestra en las siguientes figuras:



Una vez creado, podemos ver el resultado en el FlexPendant:

		1 a 5 de 9
Mét.usuario:	WobjFrameCalib	
X:	1016.764 mm	
Y:	-794.7482 mm	
Z:	799.7984 mm	
Cuaternio 1	0.70710676908493	
Cuaternio 2	-1.70165918689236E-07	

O también en el código fuente de RAPID:

```
...
TASK PERS wobjdata plano_trazado:=[FALSE,TRUE,"",[[1016.76,-794.748,799.798],[0.
707107,-1.70166E-07,-1.68899E-10,0.707107]],[[0,0,0],[1,0,0,0]]];
...
```

Desarrollo de una aplicación(Parte I)

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: ...

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

(Ver [enunciado \(.Enunciado.pdf\)](#) para más información sobre las directrices.)

Se ha elaborado el siguiente código de RAPID, donde se hace abundante uso de la función `RelTool()` para formar los `robtarg` necesarios para dibujar tanto el cuadrado como la circunferencia.

```

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];
    CONST num distancia:=150;
    TASK PERS wobjdata plano_trazado:=[FALSE,TRUE,"",[[101.172,952.795,450.836],
[2.49104E-07,0,0,1]],[[0,0,0],[1,0,0,0]]];

    VAR intnum emergencia;

    PROC Pinta_cuadrado(robtarget centro, PERS wobjdata objeto_trabajo, PERS too
ldata tool)
        VAR robtarget Paprox;
        VAR robtarget P1;
        VAR robtarget P2;
        VAR robtarget P3;
        VAR robtarget P4;

        Paprox:= RelTool (centro, 0, 0, 50);
        MoveJ Paprox, v100, fine, tool\WObj:=objeto_trabajo;

        P1:= RelTool (centro, 50, 50, 0);
        MoveL P1, v100, fine, tool\WObj:=objeto_trabajo;

        P2:= RelTool (centro, 50, -50, 0);
        MoveL P2, v100, fine, tool\WObj:=objeto_trabajo;

        P3:= RelTool (centro, -50, -50, 0);
        MoveL P3, v100, fine, tool\WObj:=objeto_trabajo;

        P4:= RelTool (centro, -50, 50, 0);
        MoveL P4, v100, fine, tool\WObj:=objeto_trabajo;

        !Move to complete square
        MoveL P1, v100, fine, tool\WObj:=objeto_trabajo;

        !Move to approximation position
        MoveL Paprox, v100, fine, tool\WObj:=objeto_trabajo;
    ENDPROC

    PROC Pinta_circunferencia(robtarget centro, PERS wobjdata objeto_trabajo, PE
RS tooldata tool)
        VAR robtarget Paprox;
        VAR robtarget P1;
        VAR robtarget P2;

```

```

VAR robtarget P3;
VAR robtarget P4;

Paprox:= RelTool (centro, 0, 0, 50);
MoveJ Paprox, v100, fine, tool\WObj:=objeto_trabajo;

P1:= RelTool (centro, 0, 100*Sqrt(2), 0);
P2:= RelTool (centro, 100*Sqrt(2), 0, 0);
P3:= RelTool (centro, 0, -100*Sqrt(2), 0);
P4:= RelTool (centro, -100*Sqrt(2), 0, 0);

MoveL P1, v100, fine, tool\WObj:=objeto_trabajo;
MoveC P2, P3, v100, fine, tool\WObj:=objeto_trabajo;
MoveC P4, P1, v100, fine, tool\WObj:=objeto_trabajo;

MoveJ Paprox, v100, fine, tool\WObj:=objeto_trabajo;

ENDPROC

PROC cerrar()
    ! Completa el código fuente
ENDPROC

PROC abrir()
    ! Completa el código fuente
ENDPROC

PROC main()
    VAR robtarget ptest := [[150, 150, 0],[1,0,0,0], config_fun, [0,0,0,0,9E
9,9E9] ];

    MoveAbsJ casa, v100, fine, Puntero_TCP;
    WHILE DI1=0 DO
        IF DI2=0 THEN
            Pinta_cuadrado ptest, plano_trazado, Puntero_TCP;
        ELSE
            Pinta_circunferencia ptest, plano_trazado, Puntero_TCP;
        ENDIF
    ENDWHILE
    MoveAbsJ casa, v100, fine, Puntero_TCP;
ENDPROC

TRAP emergency_routine
    StopMove \Quick;
    ! Completa el código fuente
    StartMove;
ENDTRAP

ENDMODULE

```

Repítase el apartado anterior de forma que el trazado de la figura se lleve a cabo en un plano inclinado respecto al plano de trazado original (por ejemplo 30°). Para ello se sugiere modificar la definición del `robtarg` 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.

Para ello, podemos añadir la siguientes líneas que modifica el `robtarg` `pctest` :

```
pctest:= RelTool(pctest, 0, 0, 0\Ry:=30);  
pctest:= Offs(pctest, 0, 0, 25);
```

o de forma mas compacta y quizá mas correcta(convención):

```
pctest:= RelTool(pctest, 0, 0, 25\Ry:=30);
```

Que efectivamente inclina el plano que se va a dibujar y aplica un desplazamiento en el eje z **del plano de trazado**.

Cuando se usa `Offs()` en el primer ejemplo, la modificación se lleva a cabo según los ejes de referencia **respecto de los cuales se ha definido el punto que se le pasa como parámetro** y **no** es por lo tanto una traslación relativa del punto a lo largo de sus ejes. En el caso de que el punto y el objeto desde el cual esta referenciado compartan **la misma orientación** entonces el resultado es **equivalente**. En nuestro caso, no se cumple esta condición, por lo que aplicar `RelTool()` tras modificar la orientación levantaría el plano de trazado con respecto a los ejes de ese `robtarg`, y no del `plano_trazado` . Sin embargo, al usar `Offs()` el plano de trazado se levanta con respecto al sistema de referencia del que está definido `pctest` , que es `plano_trazado` , tal y como se desea. En el segundo ejemplo, con `RelTool()` tanto las traslaciones y rotaciones se llevan a cabo en torno a la orientación del punto que se le pasa como parámetro y, como `pctest` por ahora solo era una traslación desde `plano_trazado` podemos usar una sola línea que haga las dos transformaciones.

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:

Declarar una interrupción como dato de tipo intnum

```
intnum emergencia;
```

Asociar el disparo de dicha interrupción con la ejecución de la correspondiente rutina de servicio de interrupción mediante CONNECT

```
CONNECT emergencia WITH emergency_routine;
```

Definir las condiciones de disparo de la interrupción, asociándola al nivel alto de la entrada binaria nº 3, por medio de ISignalDI

```
ISignalDI DI1, 1, emergencia;
```

Programar la rutina de servicio de interrupción (TRAP), de forma que se detenga el movimiento del robot (útese 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 (útese 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)

```
TRAP emergency_routine
  StopMove \Quick;
  TPWrite "Emergencia, esperando DI3==0";
  WaitDI DI3, 0;
  TPWrite "Reanudando ejecución";
  StartMove;
ENDTRAP
```

Se muestra a continuación el código completo con las modificaciones ya explicadas:

```

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];
    CONST num distancia:=150;
    TASK PERS wobjdata plano_trazado:=[FALSE,TRUE,"", [[101.172,952.795,450.836],
[2.49104E-07,0,0,1]], [[0,0,0],[1,0,0,0]]];

    VAR intnum emergencia;

    PROC Pinta_cuadrado(robtarget centro, PERS wobjdata objeto_trabajo, PERS too
ldata tool)
        VAR robtarget Paprox;
        VAR robtarget P1;
        VAR robtarget P2;
        VAR robtarget P3;
        VAR robtarget P4;

        Paprox:= RelTool (centro, 0, 0, 50);
        MoveJ Paprox, v100, fine, tool\WObj:=objeto_trabajo;

        P1:= RelTool (centro, 50, 50, 0);
        MoveL P1, v100, fine, tool\WObj:=objeto_trabajo;

        P2:= RelTool (centro, 50, -50, 0);
        MoveL P2, v100, fine, tool\WObj:=objeto_trabajo;

        P3:= RelTool (centro, -50, -50, 0);
        MoveL P3, v100, fine, tool\WObj:=objeto_trabajo;

        P4:= RelTool (centro, -50, 50, 0);
        MoveL P4, v100, fine, tool\WObj:=objeto_trabajo;

        !Cerrar el cuadrado
        MoveL P1, v100, fine, tool\WObj:=objeto_trabajo;

        !Volver a posicion de aprox
        MoveL Paprox, v100, fine, tool\WObj:=objeto_trabajo;
    ENDPROC

    PROC Pinta_circunferencia(robtarget centro, PERS wobjdata objeto_trabajo, PE
RS tooldata tool)
        VAR robtarget Paprox;
        VAR robtarget P1;
        VAR robtarget P2;
        VAR robtarget P3;
        VAR robtarget P4;

```



```

Paprox:= RelTool (centro, 0, 0, 50);
MoveJ Paprox, v100, fine, tool\WObj:=objeto_trabajo;

P1:= RelTool (centro, 0, 100*Sqrt(2), 0);
P2:= RelTool (centro, 100*Sqrt(2), 0, 0);
P3:= RelTool (centro, 0, -100*Sqrt(2), 0);
P4:= RelTool (centro, -100*Sqrt(2), 0, 0);

MoveL P1, v100, fine, tool\WObj:=objeto_trabajo;
MoveC P2, P3, v100, fine, tool\WObj:=objeto_trabajo;
MoveC P4, P1, v100, fine, tool\WObj:=objeto_trabajo;

MoveJ Paprox, v100, fine, tool\WObj:=objeto_trabajo;

ENDPROC

PROC cerrar()
    ! Completa el código fuente
ENDPROC

PROC abrir()
    ! Completa el código fuente
ENDPROC

PROC main()
    VAR robtarget ptest := [[150, 150, 0],[1,0,0,0], config_fun, [0,0,0,0,9E
9,9E9] ];
    ptest:= RelTool(ptest, 0, 0, 71\Ry:=30);

    CONNECT emergencia WITH emergency_routine;
    ISignalDI DI3, 1, emergencia;

    MoveAbsJ casa, v100, fine, Puntero_TCP;
    WHILE DI1=0 DO
        IF DI2=0 THEN
            Pinta_cuadrado ptest, plano_trazado, Puntero_TCP;
        ELSE
            Pinta_circunferencia ptest, plano_trazado, Puntero_TCP;
        ENDIF
    ENDWHILE
    MoveAbsJ casa, v100, fine, Puntero_TCP;
ENDPROC

TRAP emergency_routine
    StopMove \Quick;
    TPWrite "Emergencia, esperando DI3==0";
    WaitDI DI3, 0;
    TPWrite "Reanudando ejecución";
    StartMove;
ENDTRAP

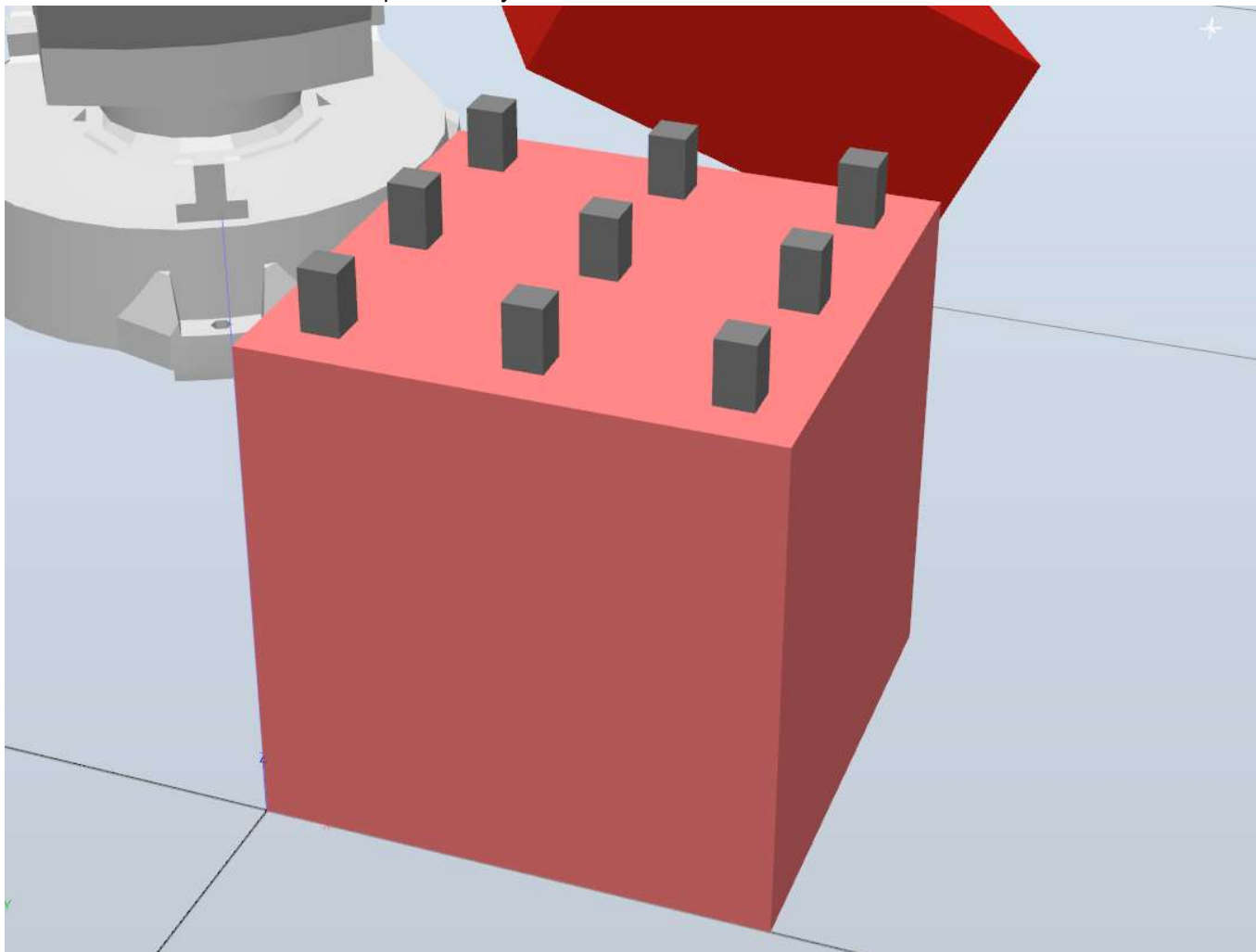
```

ENDMODULE

Desarrollo de una aplicación(Parte II)

Generación del escenario de partida

Creamos la mesa con las nueve piezas tal y como se describe en el enunciado:



Habilítase el uso de la garra

Para ello, usamos nuestra salida digital DO1 para activar o desactivar el actuador y las entradas DI4 y DI5 para los sensores de pinza abierta o cerrada. Así pues. Podemos crear dos procedimientos que abran o cierren la pinza:

```
PROC cerrar()  
    SetDo DO1, 1;  
    WaitDI DI4, 1;  
ENDPROC
```

```
PROC abrir()  
    SetDo DO1, 0;  
    WaitDI DI5, 1;  
ENDPROC
```

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).

Con un WorkObject `basemesa` en la esquina de la base de nuestra mesa, es sencillo implementar dicho procedimiento teniendo en cuenta los parámetros geométricos y la localización de las piezas que se apoyan en ésta. Así pues, mediante el uso de la garra que acabamos de implementar y la definición de ciertos destinos de paso para evitar colisiones y asegurar un movimiento óptimo, se elabora el siguiente código; donde se han incluido ciertos comentarios que describen el movimiento que recorre el robot:

```

PROC coger_y_dejar(num fila, num columna)
    VAR num lado:= 400;
    VAR robtarget origen;
    VAR robtarget aproxCoger;
    VAR robtarget aproxPieza;
    VAR robtarget enPieza;
    VAR robtarget esquina;
    VAR robtarget aproxDejar;
    VAR robtarget dejando;
    VAR num xcentro;
    VAR num ycentro;
    xcentro:= 50 + 150*(fila-1);
    ycentro:= 50 + 150*(columna-1);
    origen := [[0, 0, 0],[1,0,0,0], [0,0,0,0], [0,0,0,0,9E9,9E9] ];

    !Aproximacion a cara_coger
    aproxCoger:= RelTool(origen, lado/2, lado/2,550);
    MoveJ aproxCoger, v500, fine, Garra_TCP\Wobj:=basemesa;

    !Aproximacion a pieza
    aproxPieza := RelTool(origen, xcentro, ycentro, 400+100);
    MoveL aproxPieza, v500, fine, Garra_TCP\Wobj:=basemesa;
    abrir;

    !Bajar coger y subir
    enPieza := RelTool(aproxPieza, 0, 0, -80);
    MoveL enPieza, v500, fine, Garra_TCP\Wobj:=basemesa;
    cerrar;
    MoveL aproxPieza, v500, fine, Garra_TCP\Wobj:=basemesa;

    !Ir a otra cara
    esquina:=RelTool(aproxPieza, 0, 100+50+150*(3-columna), 0);
    MoveL esquina, v500, fine, Garra_TCP\Wobj:=basemesa;
    aproxDejar:=RelTool(esquina, 0, 0, -100-50-150*(columna-1)\Rx:=-90);
    MoveJ aproxDejar, v500, fine, Garra_TCP\Wobj:=basemesa;

    !Bajar Dejar y subir
    dejando := RelTool(aproxDejar, 0, 0, -80);
    MoveL dejando, v500, fine, Garra_TCP\Wobj:=basemesa;
    abrir;
    WaitTime 0.5;
    MoveL RelTool(aproxDejar,0,0,100), v500, fine, Garra_TCP\Wobj:=basemesa;

    !Volver a la primera cara
    MoveJ esquina, v500, fine, Garra_TCP\Wobj:=basemesa;
    MoveJ aproxCoger, v500, fine, Garra_TCP\Wobj:=basemesa;

```

ENDPROC

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

En el main, recorreremos las filas y columnas con dos bucles iterativos:

```
FOR fila FROM 1 TO 3 DO
    FOR columna FROM 1 TO 3 DO
        coger_y_dejar fila, columna;
    ENDFOR
ENDFOR
```

Se presenta el código completo a continuación:

```

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];
  CONST num distancia:=150;
  TASK PERS wobjdata plano_trazado:=[FALSE,TRUE,"",[[101.172,952.795,450.836],
[2.49104E-07,0,0,1]],[[0,0,0],[1,0,0,0]]];

  VAR intnum emergencia;
  TASK PERS wobjdata basemesa:=[FALSE,TRUE,"",[[997.701,-0.29706,-0.479568],
[0.867482,-2.47042E-08,2.31523E-09,0.497469]],[[0,0,0],[1,0,0,0]]];

  PROC Pinta_cuadrado(robtargtarget centro, PERS wobjdata objeto_trabajo, PERS too
ldata tool)
    VAR robtarget Paprox;
    VAR robtarget P1;
    VAR robtarget P2;
    VAR robtarget P3;
    VAR robtarget P4;

    Paprox:= RelTool (centro, 0, 0, 50);
    MoveJ Paprox, v500, fine, tool\WObj:=objeto_trabajo;

    P1:= RelTool (centro, 50, 50, 0);
    MoveL P1, v500, fine, tool\WObj:=objeto_trabajo;

    P2:= RelTool (centro, 50, -50, 0);
    MoveL P2, v500, fine, tool\WObj:=objeto_trabajo;

    P3:= RelTool (centro, -50, -50, 0);
    MoveL P3, v500, fine, tool\WObj:=objeto_trabajo;

    P4:= RelTool (centro, -50, 50, 0);
    MoveL P4, v500, fine, tool\WObj:=objeto_trabajo;

    !Move to complete square
    MoveL P1, v500, fine, tool\WObj:=objeto_trabajo;

    !Move to approximation position
    MoveL Paprox, v500, fine, tool\WObj:=objeto_trabajo;
  ENDPROC

  PROC Pinta_circunferencia(robtargtarget centro, PERS wobjdata objeto_trabajo, PE
RS tooldata tool)
    VAR robtarget Paprox;

```

```

VAR robtarget P1;
VAR robtarget P2;
VAR robtarget P3;
VAR robtarget P4;

Paprox:= RelTool (centro, 0, 0, 50);
MoveJ Paprox, v500, fine, tool\WObj:=objeto_trabajo;

P1:= RelTool (centro, 0, 100*Sqrt(2), 0);
P2:= RelTool (centro, 100*Sqrt(2), 0, 0);
P3:= RelTool (centro, 0, -100*Sqrt(2), 0);
P4:= RelTool (centro, -100*Sqrt(2), 0, 0);

MoveL P1, v500, fine, tool\WObj:=objeto_trabajo;
MoveC P2, P3, v500, fine, tool\WObj:=objeto_trabajo;
MoveC P4, P1, v500, fine, tool\WObj:=objeto_trabajo;

MoveJ Paprox, v500, fine, tool\WObj:=objeto_trabajo;

```

ENDPROC

```

PROC coger_y_dejar(num fila, num columna)
  VAR num lado:= 400;
  VAR robtarget origen;
  VAR robtarget aproxCoger;
  VAR robtarget aproxPieza;
  VAR robtarget enPieza;
  VAR robtarget esquina;
  VAR robtarget aproxDejar;
  VAR robtarget dejando;
  VAR num xcentro;
  VAR num ycentro;
  xcentro:= 50 + 150*(fila-1);
  ycentro:= 50 + 150*(columna-1);
  origen := [[0, 0, 0],[1,0,0,0], [0,0,0,0], [0,0,0,0,9E9,9E9] ];

  !Aproximacion a cara_coger
  aproxCoger:= RelTool(origen, lado/2, lado/2,550);
  MoveJ aproxCoger, v500, fine, Garra_TCP\WObj:=basemesa;

  !Aproximacion a pieza
  aproxPieza := RelTool(origen, xcentro, ycentro, 400+100);
  MoveL aproxPieza, v500, fine, Garra_TCP\WObj:=basemesa;
  abrir;

  !Bajar coger y subir
  enPieza := RelTool(aproxPieza, 0, 0, -80);
  MoveL enPieza, v500, fine, Garra_TCP\WObj:=basemesa;
  cerrar;
  MoveL aproxPieza, v500, fine, Garra_TCP\WObj:=basemesa;

  !Ir a otra cara

```



```

esquina:=RelTool(aproxPieza, 0, 100+50+150*(3-columna), 0);
MoveL esquina, v500, fine, Garra_TCP\Wobj:=basemesa;
aproxDejar:=RelTool(esquina, 0, 0, -100-50-150*(columna-1)\Rx:=-90);
MoveJ aproxDejar, v500, fine, Garra_TCP\Wobj:=basemesa;

!Bajar Dejar y subir
dejando := RelTool(aproxDejar, 0, 0, -80);
MoveL dejando, v500, fine, Garra_TCP\Wobj:=basemesa;
abrir;
WaitTime 0.5;
MoveL RelTool(aproxDejar,0,0,100), v500, fine, Garra_TCP\Wobj:=basemesa;

!Volver a la primera cara
MoveJ esquina, v500, fine, Garra_TCP\Wobj:=basemesa;
MoveJ aproxCoger, v500, fine, Garra_TCP\Wobj:=basemesa;

```

```
ENDPROC
```

```

PROC cerrar()
    SetDo D01, 1;
    WaitDI DI4, 1;
ENDPROC

```

```

PROC abrir()
    SetDo D01, 0;
    WaitDI DI5, 1;
ENDPROC

```

```

PROC main()
    VAR robtarget ptest := [[150, 150, 0],[1,0,0,0], config_fun, [0,0,0,0,9E
9,9E9] ];
    ptest:= RelTool(ptest, 0, 0, 71\Ry:=30);

    CONNECT emergencia WITH emergency_routine;
    ISignalDI DI3, 1, emergencia;

    MoveAbsJ casa, v1000, fine, Puntero_TCP;
    WHILE DI1=0 DO
        IF DI2=0 THEN
            Pinta_cuadrado ptest, plano_trazado, Puntero_TCP;
        ELSE
            Pinta_circunferencia ptest, plano_trazado, Puntero_TCP;
        ENDIF
    ENDWHILE

    !coger_y_dejar 2,1;
    FOR fila FROM 1 TO 3 DO
        FOR columna FROM 1 TO 3 DO
            coger_y_dejar fila, columna;
        ENDFOR
    ENDFOR

```

```
ENDFOR
```

```
!Programa terminado, parar  
StopMove \Quick;  
WaitTime 3600;
```

```
ENDPROC
```

```
TRAP emergency_routine  
  StopMove \Quick;  
  TPWrite "Emergencia, esperando DI3==0";  
  WaitDI DI3, 0;  
  TPWrite "Reanudando ejecución";  
  StartMove;  
ENDTRAP
```

```
ENDMODULE
```

Conclusión

En resumen, se ha abordado un reducido problema de diseño; desde la creación de la escena y de la herramienta del robot hasta la programación de su tarea, incluyendo sensores, accionadores e interrupciones. RAPID permite la programación de robots de una manera que permite al usuario no preocuparse de los problemas cinemáticos, dinámicos y de servocontrol del robot. Sin embargo, tiene otros aspectos contraintuitivos, como es la incapacidad de definir el sistema de referencia de un punto cuando se define.

In []: