Robótica – Entregable 2

Versión básica

El robot inicia los módulos « torres » en el cual se elige el punto de origen de la torre y el numero de piezas a desplaza. A partir de ahí, desplaza todas las piezas hacia un lado de la torre inicial para crear una torre intermedia y finalmente vuelve a desplazar todas las piezas al otro lado del punto de inicio para volver a poner las piezas en el orden inicial. Cuando termina de desplazar todas las piezas vuelve a su posición de reposo.

Código de la primera parte :

```
MODULE Module1
CONST robtarget
pReposo:=[[1808.910161514,0,1855],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,
9E+09,9E+09]];
CONST robtarget
pBaseTorre1:=[[1500,500,750],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget pBaseTorre2:=[[1500,-
500,750],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST num alt := 100;

    PROC main()
        VAR num nPiezas1;
        VAR num nPiezas2;

        nPiezas1 := 6;
        nPiezas2 := 3;

        torre nPiezas1, pBaseTorre1;
        torre nPiezas2, pBaseTorre2;
    ENDPROC

    PROC torre(num piezas, robtarget pBase)
        VAR robtarget pOrigenAprox;
        VAR robtarget pOrigen;
        VAR robtarget pInterAprox;
        VAR robtarget pInter;
        VAR robtarget pDestAprox;
        VAR robtarget pDest;

        pOrigenAprox := pBase;
        pOrigen := pBase;
        pInterAprox := pBase;
        pInter := pBase;
        pDestAprox := pBase;
        pDest := pBase;

        pOrigenAprox.trans.z := pOrigenAprox.trans.z + alt*piezas + 50;
        pOrigen.trans.z := pOrigen.trans.z + alt*piezas;

        pInterAprox.trans.y := pInterAprox.trans.y + 3*alt;
        pInterAprox.trans.z := pInterAprox.trans.z + piezas*alt + 50;
        pInter.trans.y := pInter.trans.y + 3*alt;
        pInter.trans.z := pInter.trans.z + alt;

        pDestAprox.trans.y := pDestAprox.trans.y - 3*alt;
        pDestAprox.trans.z := pDestAprox.trans.z + piezas*alt + 50;
        pDest.trans.y := pDest.trans.y - 3*alt;
        pDest.trans.z := pDest.trans.z + alt;
```

```
        MoveJ pReposo, v1000, z100, Ventosa\WObj:=wobj0;

        FOR i FROM 1 TO piezas DO
            SetDO SD_ActivaVentosa, 0;

            MoveJ pOrigenAprox, v1000, z0, Ventosa\WObj:=wobj0;
            MoveL pOrigen, v100, fine, Ventosa\WObj:=wobj0;
            SetDO SD_ActivaVentosa, 1;
            WaitDI ED_PiezaCogida, 1;
            MoveL pOrigenAprox, v100, z100, Ventosa\WObj:=wobj0;

            MoveJ pInterAprox, v500, z0, Ventosa\WObj:=wobj0;
            MoveL pInter, v100, fine, Ventosa\WObj:=wobj0;
            SetDO SD_ActivaVentosa, 0;
            WaitDI ED_PiezaCogida, 0;
            MoveJ pInterAprox, v500, z100, Ventosa\WObj:=wobj0;

            pOrigen.trans.z := pOrigen.trans.z - alt;
            pInter.trans.z := pInter.trans.z + alt;
        ENDFOR

        pInter.trans.z := pInter.trans.z - alt;
        SetDO SD_ActivaVentosa, 0;

        FOR i FROM 1 TO piezas DO
            MoveL pInterAprox, v500, z0, Ventosa\WObj:=wobj0;
            MoveL pInter, v100, fine, Ventosa\WObj:=wobj0;
            SetDO SD_ActivaVentosa, 1;
            WaitDI ED_PiezaCogida, 1;
            MoveJ pInterAprox, v100, z100, Ventosa\WObj:=wobj0;

            MoveJ pDestAprox, v500, z0, Ventosa\WObj:=wobj0;
            MoveL pDest, v100, fine, Ventosa\WObj:=wobj0;
            SetDO SD_ActivaVentosa, 0;
            WaitDI ED_PiezaCogida, 0;
            MoveJ pDestAprox, v500, z100, Ventosa\WObj:=wobj0;

            pInter.trans.z := pInter.trans.z - alt;
            pDest.trans.z := pDest.trans.z + alt;
        ENDFOR

        MoveJ pReposo, v1000, z100, Ventosa\WObj:=wobj0;
    ENDPROC

ENDMODULE
```

Versión avanzada(I)

El robot inicia los módulos “torres”, en primer lugar el robot desactiva su ventosa y vuelve a la posición de reposo. Se calcula las posiciones de las torres gracias a la circunferencia y al numero de torres definidos. A partir de ahí empieza a desplazar las piezas gracias a un bucle sobre las piezas y las desplaza hacia sus posiciones de destino. He utilizado el resultado de la división euclidiana de la iteración en la que nos encontramos por el numero de torres ya que su resultado nos da la altura a la que debe dejar la pieza el robot y su resto permite saber a que torre hay que llevar la pieza.

Código de la segunda parte :

```
MODULE Module1
    CONST robtarget pBaseOrigen:=[[1500,
500,750],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget
pCentroDestino:=[[1700,1000,750],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]
```

```
;
    CONST robtarget
pReposo:=[[1808.910161514,0,1855],[0.5,0,0.866025404,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,
9E+09,9E+09]];
    CONST num alt := 50;

    PROC main()
        VAR num nPiezas;
        VAR num nTorres;
        VAR num circuferencia;

        nTorres := 3;
        nPiezas := 10;
        circuferencia := 200;

        torres nPiezas, nTorres, circuferencia;
    ENDPROC

    PROC torres(num nPiezas, num nTorres, num circ)
        VAR num ang;
        VAR num rest;
        VAR num ent;

        VAR robtarget pOrigenAprox;
        VAR robtarget pOrigen;

        VAR robtarget pDestino;
        VAR robtarget pDestinoAprox;

        ang := 360/nTorres;

        MoveJ pReposo, v1000, z100, Ventosa\WObj:=wobj0;

        pOrigen := pBaseOrigen;
        pOrigenAprox := pBaseOrigen;
        pOrigen.trans.z := pOrigen.trans.z + alt*nPiezas;
        pOrigenAprox.trans.z := pOrigenAprox.trans.z + alt*nPiezas + 50;

        SetDO SD_ActivaVentosa, 0;

        FOR i FROM 0 TO nPiezas-1 DO
            ent := Trunc (i/nTorres);
            rest := i - nTorres*ent;

            pDestino := pCentroDestino;
            pDestinoAprox := pCentroDestino;
            pDestino.trans.z := pDestino.trans.z + alt*(ent+1);
            pDestinoAprox.trans.z := pDestinoAprox.trans.z + alt*nPiezas + 50;
            pDestino.trans.x := pDestino.trans.x + circ*cos(rest*ang);
            pDestino.trans.y := pDestino.trans.y + circ*sin(rest*ang);
            pDestinoAprox.trans.x := pDestinoAprox.trans.x + circ*cos(rest*ang);
            pDestinoAprox.trans.y := pDestinoAprox.trans.y + circ*sin(rest*ang);

            MoveJ pOrigenAprox, v500, fine, Ventosa\WObj:=wobj0;
            MoveL pOrigen, v100, fine, Ventosa\WObj:=wobj0;
            SetDO SD_ActivaVentosa, 1;
            WaitDI ED_PiezaCogida, 1;

            MoveL pOrigenAprox, v100, fine, Ventosa\WObj:=wobj0;

            MoveL pDestinoAprox, v500, fine, Ventosa\WObj:=wobj0;
            MoveL pDestino, v100, fine, Ventosa\WObj:=wobj0;
            SetDO SD_ActivaVentosa, 0;
            WaitDI ED_PiezaCogida, 0;
            MoveL pDestinoAprox, v100, fine, Ventosa\WObj:=wobj0;
```

```
            pOrigen.trans.z := pOrigen.trans.z - alt;
        ENDFOR
    ENDPROC
ENDMODULE
```