# PRÁCTICA #5

**TEMA:** Creación de trayectorias parametrizables.

## 1. Objetivos

- Determinar el algoritmo de movimiento del robot para ciertas trayectorias definidas.
- Analizar la utilidad de las variables locales y globales para la creación de funciones.
- Crear un programa sencillo donde se aplique la creación de trayectorias dependiendo de los parámetros ingresados por el usuario.

## 2. Marco teórico

### 2.1. Variables

Existen distintos tipos de variables que podemos utilizar en las rutinas de los programas del robot, entre las más importantes tenemos números reales (Real), cadena de caracteres (String), booleanos (Bool), y de posición (Trans).

Cada una de estas variables puede ser definida como global o local, una variable global permite ser llamado desde cualquier subprograma del robot mientras que las variables locales solo pueden utilizarse en el programa donde se crearon. Se recomienda utilizar variables globales solo para valores fijos del ambiente del robot o de comunicación, mientras que las variables locales pueden ser modificadas a lo largo de la ejecución del programa más eficientemente.

Las variables reales se definen simplemente con un signo igual mientras que las variables String deben escribirse entre “”:

```
x = 1   global x
name = “prueba” Para
definir una variable de
```

posición con coordenadas rectangulares se antepone transl.

```
point1 = transl(20, 100, 50)
```

Para definir una variable de ángulos de cada articulación del robot se lo hace de esta forma:

```
point2 =  [90, 100, 50, 80, 45, 60]
```

(Para información detallada https://robodk.com/doc/en/PythonAPI/robolink.html )

### 2.2. Funciones

A una función que se llama dentro del programa principal pero se la declara antes de la misma, se utiliza para ejecutar instrucciones de parámetros cambiantes o para líneas de instrucciones que se repiten constantemente a lo largo de la rutina principal, permitiendo una presentación más limpia del programa.

Una función se crea igual que una rutina principal paro se deben definir sus parámetros de entrada como variables locales en caso de necesitarlas:

```
def letraA (a, b, c)
```

Es recomendable que una función no llame a variables globales, sino que trabaje con las variables locales de su argumento. Para llamar a la subrutina dentro del programa principal se debe colocar el nombre de la función con sus parámetros en caso de tenerlos, como parámetros se pueden utilizar variables locales pero el tipo de variable debe ser el correcto:

```
letraA (320, 350, 50)
```

### 2.3. Bucles de control

Para que la estructura del programa esté ordenada se utilizan instrucciones de control ya conocidas como IF…ELSE, FOR …., WHILE …. . También permiten que nuestro programa realice distintos movimientos dependiendo de las condiciones del ambiente o de las variables asignadas. El uso de estas instrucciones es primordial para una programación eficaz del robot.

Todas las instrucciones disponibles y su estructura se deben revisar en https://robodk.com/doc/en/PythonAPI/intro.html

### 2.4. Parametrización de trayectorias

La parametrización de trayectorias consiste en relacionar las posiciones de una curva con respecto a una variable cuyo valor se recorre en un intervalo definido. Para parametrizar una curva primero debemos tener su ecuación, luego se define una de sus componentes en función de una variable de parametrización y se encuentra la relación entre las demás componentes y esta nueva variable.

Por ejemplo, para una recta en el plano y = 10, la ecuación tiene la forma z = a*x + b, donde a y b son constantes mientras que x y z son las componentes de las posiciones. Una opción de parametrización seria asignar una variable t a x y darle valores a a y b para obtener lo siguiente:

$$\begin{Bmatrix} x = t \\ y = 10 \\ z = 2 * t + 5 \end{Bmatrix}$$

Ahora podemos realizar un lazo FOR para t = 0 a 100 con la instrucción LMOVE para graficar la recta.

```
FOR t in range(100):
    pos = transl(t, 10, 2*t + 5)
    robot.MOVEJ (pos)
```

Aunque el ejemplo muestre una recta, para el robot solo utilizaremos la parametrización para curvas más complejas, esto debido a que líneas rectas y arcos de circunferencia se pueden lograr con las instrucciones MOVEL, MOVEC. Como un ejemplo extra podemos utilizar la parametrización para realizar curvas elípticas para dibujar letras.

La ecuación de una elipse es:

$$\frac{(x - x_0)^2}{a_2} \quad \frac{(y - y_0)^2}{b^2} += 1$$

Donde $x_0$ y $y_0$ son el centro de la elipse mientras que a y b son los diámetros. Parametrizando la ecuación obtenemos:

$$\{x = x_0 + a * \cos\theta\}$$
$$y = y_0 + b * \sin\theta \text{ Donde}$$

Ɵ será la variable parametrizada.

Para dibujar una elipse realizamos un barrido en un lazo while de la variable Ɵ desde –π/2 hasta π/2.

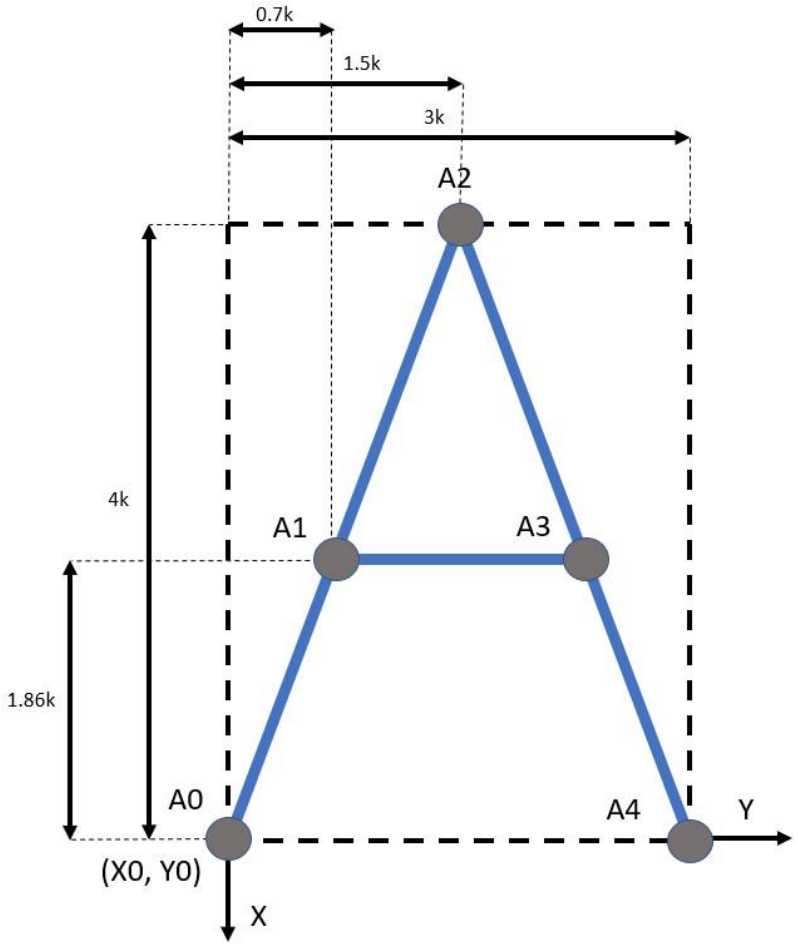
```
t = -pi/2
x0= -450
y0= 200
a=75
b=150
robot.MOVEJ(transl(x0, y0, 50)
while (t <=pi):
    pos = transl(x0 + a*cos(t), y0 + b*sin(t), 50)
    robot.MOVEJ (pos)
```

**3. Descripción.**

En esta práctica se dibujarán una serie de letras de las cuales se pueda modificar su tamaño y su ubicación. Al robot se le acoplará un soporte con un marcador y se escribirá sobre papel A3 en una mesa al costado del robot.

Para que las letras puedan ser modificadas en tamaño y posición, debemos establecer normas generales:

- Las letras mayúsculas deberán estar inscritas en un área rectangular de relación 4:3
- Las letras minúsculas deberán estar inscritas en un área cuadrada y se permitirá extenderse según sea el caso
- El punto de referencia de precisión de cada letra será su esquina inferior izquierda
- Se debe utilizar las funciones para levantar el marcador del papel
- Todos los movimientos del robot para escribir deberán ser con MOVEL, o MOVEJ.




Tomando en cuenta la letra A mayúscula de la imagen anterior, podemos definir sus puntos como:

A0 = transl(X0 , Y0 , Z)

A1 = transl(X0 – 1.86*k, Y0 + 0.7*k, Z)

A2 = transl(X0 – 4*k , Y0 + 1.5*k, Z)

A3 = transl(X0 – 1.86*k, Y0 + 2.3*k, Z)

A4 = transl(X0 , Y0 + 3*k , Z)

Siendo variables globales que no van a cambiar:

Z: posición en Z donde el robot escribe sobre el papel encima de la mesa.

Mientras que las variables que pueden variar a menudo:

X0 y Y0: posición de la letra en el papel con respecto al robot. k: escalamiento de la letra, sirve para cambiar su tamaño.

**SE PUEDE REALIZAR CAMBIOS DE BASE COMO EL EJEMPLO ADJUNTADO EN EL SIDWEB.**

Una vez establecidos los puntos tenemos que escribir la secuencia de movimientos, utilizaremos JAPPRO para el primer punto, LMOVE siempre que escribamos en el papel, y LDEPART Y LAPPRO para levantar y volver a bajar el marcador.

```
APROX = transl(0,0,-100)
RETRACT = transl(0,0,-50)
robot.MoveJ = (A0)*APROX
robot.MoveL = (A0)
robot.MoveL = (A1)
robot.MoveJ = (A1)*RETRACT
robot.MoveJ = (A2)*APROX
robot.MoveL = (A2)
robot.MoveL = (A3)
robot.MoveL = (A4)
robot.MoveJ = (A4)*RETRACT
```

Se puede comprimir tanto la definición de los puntos como la secuencia de movimiento en una sola función que tenga como parámetros la posición y la escala de la letra.

```
def dibujarA (x, y, k)
        …
```

## 4. Adquirir parámetros de la estación

Para adquirir los targets creados en la estación de trabajo se debe realizar la siguiente programación:

```
from robolink import *    # RoboDK API
from robodk import *     # Robot toolbox

RDK = Robolink()
robot = RDK.ItemUserPick('',ITEM_TYPE_ROBOT)          #adquirir todos
los parámetros del robot if not robot.Valid():
   quit()
reference = robot.Parent()  # devuelve el artículo
robot.setPoseFrame(reference)#establece el marco de referencia de un robot
pose_ref=robot.Pose()  #devuelve la posición actual del robot con matriz
b=robot.pose_2_xyzrpw(pose_ref) # devuelve la posición actual del robot
print("b = ",b)
```

Basta con crear un target como referencia en nuestra estación para poder trabajar alrededor de ese target, y para adquirir la ubicación del mismo requerimos el siguiente código.

```
frameletras = RDK.Item("Frame 2",ITEM_TYPE_FRAME)
robot.setPoseFrame(frameletras)
tarletras = RDK.Item("letras",ITEM_TYPE_TARGET)
tarletras = tarletras.Pose()
```

Con esto ya podremos realizar el movimiento alrededor de este target además de poder crear nuevos targets como por ejemplo:

```
for i in range(5):
   if I < 1:
        robot.MoveJ(tarletras)
   robot.MoveJ(tarletras*transl(-20*i,20,0))
```

Otro ejemplo para moverse con aproximación y retracción con targets existentes en la estación:

```
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
from time import sleep
RDK = Robolink()
rob = robodk

robot = RDK.ItemUserPick('',ITEM_TYPE_ROBOT)    #adquirir todos los parámetros
d el robot
if not robot.Valid():
   quit()
reference = robot.Parent()  # devuelve el artículo
robot.setPoseFrame(reference)#establece el marco de referencia de un
robot pose_ref=robot.Pose()  #devuelve la posición actual del robot con matriz

def elipse():
   Ref1 = RDK.Item("Target 2",ITEM_TYPE_TARGET)
   RefTar = Ref1.Pose()
   b1 = rob.pose_2_xyzrpw(RefTar)
   t = -pi/2
   x0= b1[0]
   y0= b1[1]
   z0= b1[2]
```

```
    a=20
    b=10
    RDK.RunProgram('WeldOff(-1)')
    robot.MoveJ(RefTar)
    robot.setSpeed(2050,2000)
    while (t <=3*pi/2):
        pos = transl(x0 + a*cos(t), y0 + b*sin(t), z0)*rotx(pi)*rotz(-pi/2)
        robot.MoveJ(pos)
        if (t==-pi/2):
            sleep(2)
            RDK.RunProgram('WeldOn')
        t +=pi/360

elipse()
RDK.RunProgram('WeldOff(0)')
```

Como extra puede agregar un Mbox para pedir al usuario que ingrese lo valores de a y b, determinado la amplitud de la elipse:

```
    valores = mbox('Ingrese valores de a y b',entry="50,100")
a,b = [float(x.replace(' ','')) for x in valores.split(',')]
```

## 5. Procedimiento

a) Agregar el soporte del marcador al elemento terminal del robot.

b) Crear un programa para dibujar distintas letras de distintos tamaños, las letras y los tamaños quedan a criterio del profesor y de los estudiantes.

c) Al menos una letra debe contener arcos de elipses o alguna otra curva parametrizada.

d) Ejecutar el programa.