TRABAJO PRÁCTICO N°1

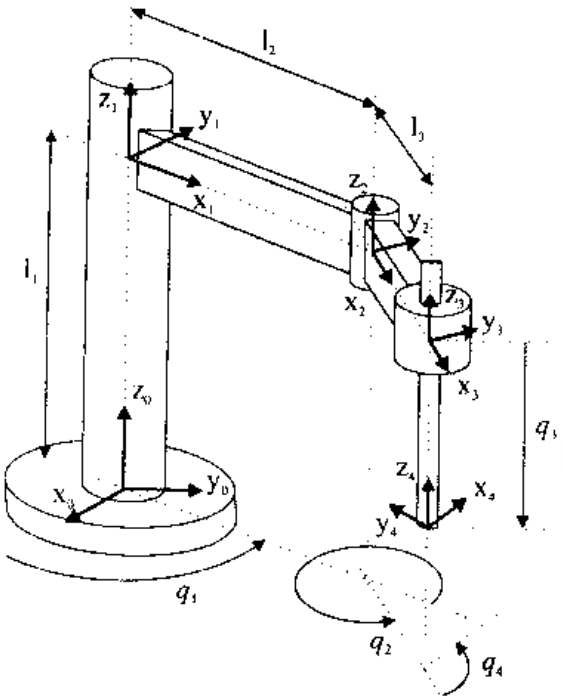
CURSO: **R6055**

DOCENTES: **Ing. Hernan Giannetta**
Ing. Damián Grazella

ALUMNO: **Alonso, Damián Ezequiel(122.476-1)**
Di Donato, Andrés Leonardo (123.664-7)
Morella, Federico Martín (126.534-9)

FECHA DE ENTREGA: **18-06-2012**

Arquitectura elegida: Robot SCARA



Introducción histórica

La palabra SCARA corresponde al acrónimo en inglés Selective Compliant Articulated Robot Arm, es decir “Brazo robótico articulado de obediencia selectiva”
Fue diseñado en los años ochenta como una alternativa a los robots de ensamble ya existentes. Es muy apto para realizar perforaciones sobre el plano XY.
Una de las características fundamentales del robot consiste en su estructura de dos piezas enlazadas, de diseño similar a un brazo humano. De aquí proviene su calificativo de “articulado”. Esta característica permite al brazo extenderse hacia áreas confinadas, así como retraerse para volver a su posición inicial. Lo anterior constituye una gran ventaja para el transporte de partes, su carga y su descarga en ámbitos donde el espacio es reducido.

Los robots SCARA pueden operar más rápidamente que los sistemas cartesianos.

Cinemática directa del robot

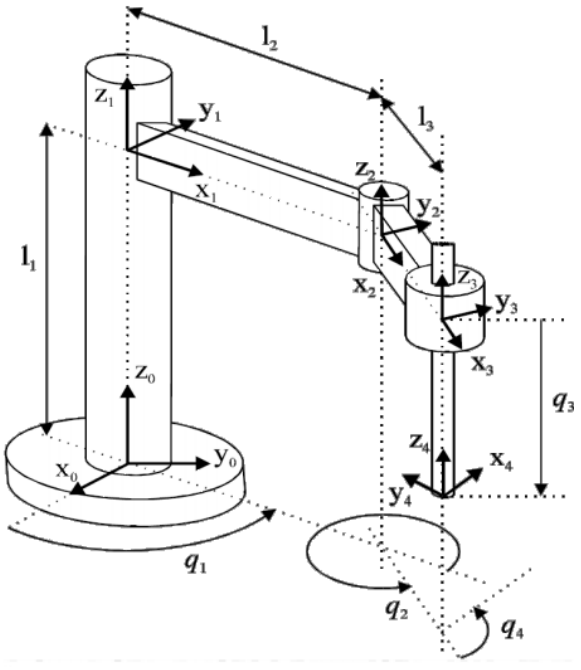
El objetivo del análisis cinemático que realizaremos a continuación es el de obtener una relación funcional matemática entre la localización del extremo del robot respecto de un sistema de referencia, y los valores asociados a cada una de sus articulaciones.
Posteriormente emplearemos las ecuaciones de cinemática directa para así poder graficar el espacio de trabajo de nuestro robot.

Método utilizado

Utilizaremos como método el modelo basado en el producto de matrices de transformación homogéneas.
Consideraremos a nuestro robot como una cadena cinemática conformada por eslabones relacionados por articulaciones. Plantearemos como sistema de referencia la base del robot.

Algoritmo de Denavit-Hartemberg

Utilizaremos el algoritmo de Denavit-Hartemberg para obtener el modelo cinemático directo.
Representamos geométricamente al robot y referenciamos el sistema de referencia (x0,y0,z0) así como los eslabones y las articulaciones



	θ (Rotación en z)	d (Traslación en z)	a (Traslación en x)	α (Rotación en x)
0	90°+q1	l1	0	0
1	q2	0	l2	0
2	0	0	l3	0
3	q4	-q3	0	0

A partir de la tabla anterior, construimos las matrices de transmoración homogénea. Utilizaremos MATLAB para simplificar la operatoria matemática y para permitir posteriores simulaciones.

```
clc;
close;
clear all;
%%
%Script que calcula la matriz de transformacion del robot Scara
disp('Calculando matriz T...');
%1) Matriz homogenea de la primera transformacion
syms q1 real; %Primer grado de libertad: rotacion de la base en torno a Z0
syms q2 real ; %Segundo grado de libertad: articulación del brazo
syms q3 real ; %Tercer grado de libertad: altura de la pinza
syms q4 real ; %Cuarto grado de libertad: orientación de la pinza. No se considera para el trazado.
syms L1 real; %Longitud del tronco
syms L2 real; %Longitud del brazo
syms L3 real; %Longitud del ante-brazo

T01=homog(0,0,q1+pi/2,[0 0 L1]); %Rotamos en Z un angulo q1+90° y nos desplazamos en Z L1
T12=homog(0,0,q2,[L2 0 0]); %Nos trasladamos en x L2 y giramos en torno a Z
T23=homog(0,0,0,[L3 0 0]); %Solo nos trasladamos en x
T34=homog(0,0,0,[0 0 -q3]);


T=simple(T01*T12*T23*T34);
disp(T);
%%
%Definimos dimensiones y rango de movimiento del robot
% El angulo de la base q1 puede rotar 270°, de 0 a pi*3/2
% El angulo de la articulación puede variar de -135° a +135°, de -pi*3/4 a
% +pi*3/4
% El descenso de la pinza q3, lo limitamos entre 0 y L1, siendo L1 la
% altura del tronco del robot

L3=0.3;
L2=0.3;
L1=0.5;

N=20;  %cantidad de pasos por eje

%Rango de movimiento de cada eje
q1_min=0;
q1_max=3*pi/2;

q2_min=-pi*3/4;
q2_max=pi*3/4;

q3_min=0;
q3_max=L1;

rango_q1=[q1_min:(q1_max-q1_min)/(N-1):q1_max];
```

```
rango_q2=[q2_min:(q2_max-q2_min)/(N-1):q2_max];
rango_q3=[q3_min:(q3_max-q3_min)/(N-1):q3_max];

%%
disp('Calculando movimientos del robot...');

a=0;
i=1;
h=waitbar(0,'Calculando...');
x=zeros(1,N^3);y=zeros(1,N^3);z=zeros(1,N^3);
  for q3=rango_q3
    for q2=rango_q2
      waitbar(a/(N*N),h);
      a=a+1;
      for q1=rango_q1
        punto=eval(T*[0,0,0,1]');
        x(i)=punto(1);
        y(i)=punto(2);
        z(i)=punto(3);
        i=i+1;
      end
    end
  end
figure;
plot3(x,y,z);
grid on;
close h;
delete h;
```

Con el script anterior, obtenemos la siguiente matriz de transformación homogénea:

T =

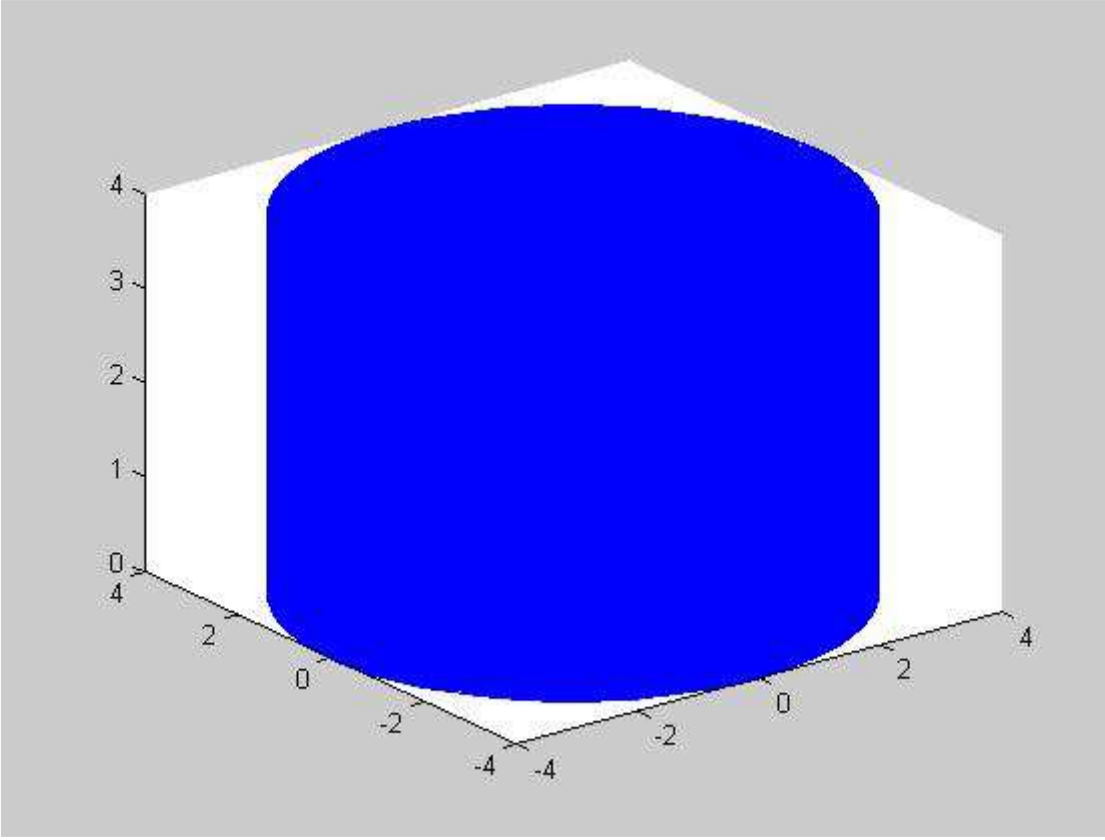
[-sin(q1+q2),	-cos(q1+q2),	0,	-sin(q1+q2)*L3-sin(q1)*L2	]
[cos(q1+q2),	-sin(q1+q2),	0,	cos(q1+q2)*L3+cos(q1)*L2	]
[0,	0,	1,	-q3+L1	]
[0,	0,	0,	1	]

Espacio de trabajo

Idealmente y si no considerásemos las restricciones mecánicas, el espacio de trabajo de un robot SCARA correspondería a un cilindro perfecto.

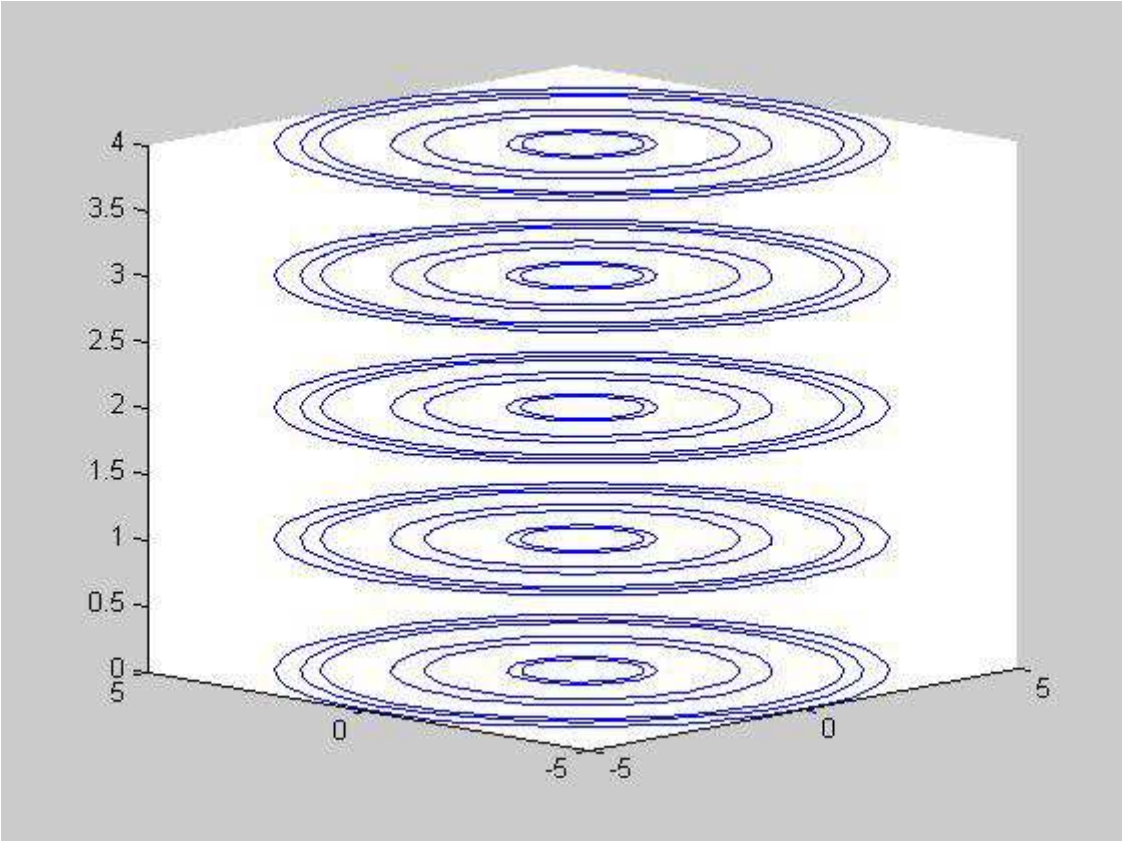
A los efectos de probar la validez de la matriz de transformación homogénea hallada en el apartado anterior, pasamos a graficar empleando MATLAB el espacio de trabajo del robot punto a punto, considerando que los ángulos pueden barrerse desde 0° a 360°.

Obtenemos la siguiente representación:



Verificamos entonces la validez de las ecuaciones obtenidas.

Pasamos ahora a repetir el gráfico anterior, pero con menor cantidad de puntos.



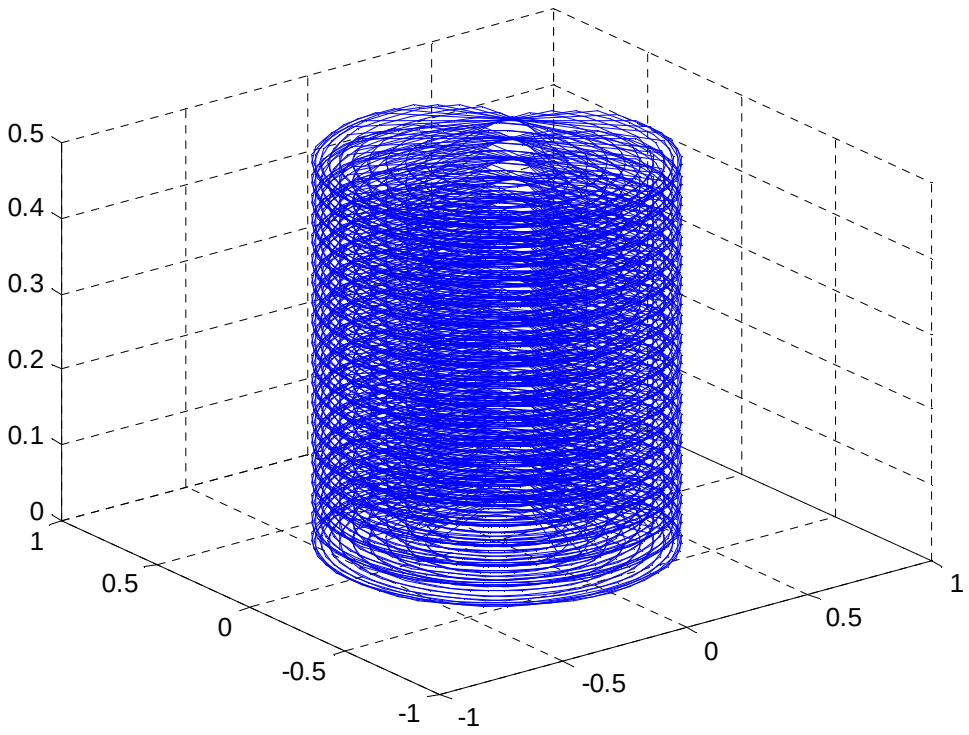
De la figura anterior podemos concluir que el robot se comporta en forma muy alineal. La gráfica anterior fue construida con incrementos constantes de los valores de las articulaciones, mientras que observamos saltos no-lineales en las circunferencias graficadas.

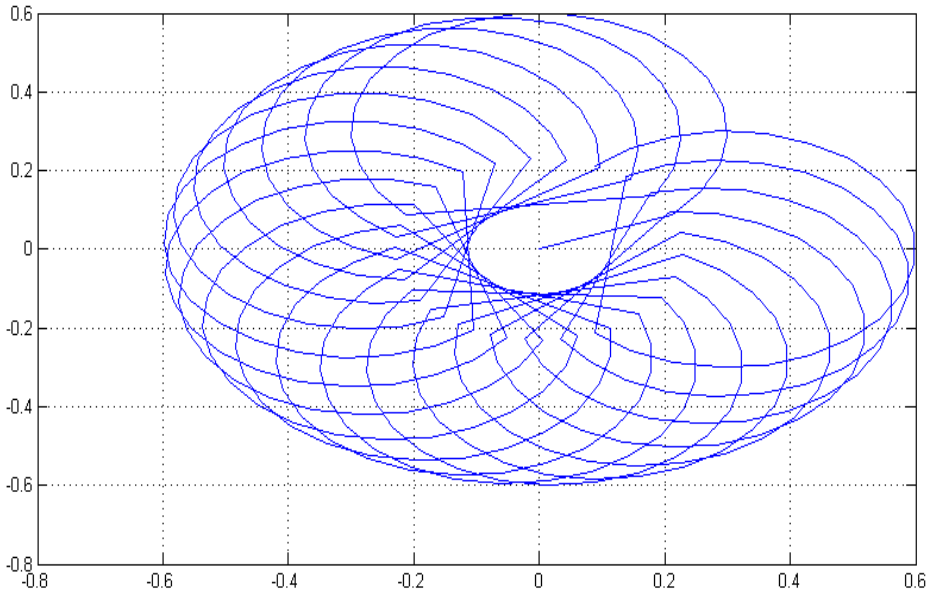
Restricción de los ángulos

Considerando la construcción física real del robot, acotamos los ángulos que las articulaciones podrían barrer en una representación más realista.

Rotación de q1: de 0° a 270°
Rotación de q2: de -135° a +135°

Repetimos entonces la representación del espacio de trabajo, obteniendo:



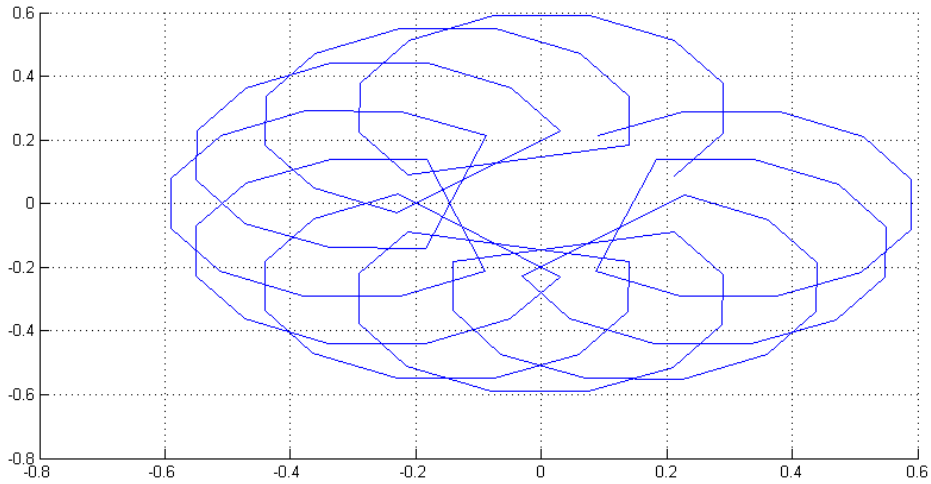


Observamos entonces que la representación no corresponde a un cilindro completo sino a un sector cilíndrico.

Implementación empleando un DSP

Repetimos ahora los cálculos implementando las ecuaciones anteriores sobre un DSP. Empleamos el DSP de Analog Devices Blackfin BF 533, para iguales rangos de variación de las articulaciones.

Simularemos el comportamiento con Visual DSP++, entorno desarrollado por Analog Devices para dicha familia de procesadores.
Efectuamos la simulación y exportamos los resultados a un archivo binario.
Posteriormente abrimos el archivo desde MATLAB para comparar ambos resultados.



Código implementado DSP

```
/*****************************************************************************
*
*
*       UTN - FRBA
*       Robótica
*       Año: 2012
*
*
*       Programas de Cálculo Cinemático de Robot Scara
*       main.c
*****************************************************************************/


#include <stdio.h>
#include "configuracion.h"
#include "MyDSPLibrary.h"


char * FileName_x =  "..\\Puntosx.dat";
char * FileName_y =  "..\\Puntosy.dat";
char * FileName_z =  "..\\Puntosz.dat";
```

```
float  x[N*N*2];
float  y[N*N*2];
float  z[N*N*2];

int main( void )
{
	float q1,q2,q3;
	long i;

	i=0;
	//barre todos los valores de q1,q2 y q3
	for(q1=q1_min;q1<=q1_max;q1+=q1_inc)
		for(q2=q2_min;q2<=q2_max;q2+=q2_inc)
			for(q3=q3_min;q3<=q3_max;q3+=q3_inc)
				{
				calc_xyz(&x[i],&y[i],&z[i],q1, q2, q3);
				i++;
				}


	// guardo los puntos calculados
	MyWriteFile( FileName_x, x , (N*N*2) );
	MyWriteFile( FileName_y, y , (N*N*2) );
	MyWriteFile( FileName_z, z , (N*N*2) );
	return 0;
}


/*************************************************************************
*
*
*	UTN - FRBA
*	Robótica
*	Año: 2012
*
*
*	Programas de Cálculo Cinemático de Robot Scara
*	MyDSPLibrary.c
*************************************************************************/

#include <stdio.h>
#include <math.h>
#include "MyDSPLibrary.h"
```

```
/************************************************************
Calcula x,y,z para un los valores de q1,q2 y q3 dados
************************************************************/
//float l1=L1;
//float l2=L2;
//float l3=L3;

int     calc_xyz(float *X,float *Y, float *Z,float q1, float q2, float q3)
        {

                *X=(-L3) * sinf(q1 + q2) - L2 * sinf(q1);
                *Y=L3 * cosf(q1 + q2) + L2 * cosf(q1);
                *Z=L1 - q3;
                return 0;
        }


/************************************************************
funciones de entrada y salida de archivo
************************************************************/
int MyReadFile(char* FileName,float *ptrSignal , int n){
int ret ;
FILE * ptrFile ;

        ptrFile = fopen( FileName , "r+b" ) ;

        if(ptrFile == (void*)0){
                printf("No se pudo abrir el archivo: %s",FileName) ;
                return -1 ;
        }


        ret = fread(ptrSignal,sizeof(float),n,ptrFile) ;

        if(ret != n ) printf("Error al leer el archivo: %s",FileName) ;


        fclose(ptrFile);

        return ret ;
}
```

```
int MyWriteFile(char* FileName,float *ptrSignal , int n){
int ret ;
FILE * ptrFile ;

	ptrFile = fopen( FileName , "w+b" ) ;

	if(ptrFile == (void*)0){
		printf("No se pudo crear el archivo: %s",FileName) ;
		return -1 ;
	}


	ret = fwrite(ptrSignal,sizeof(float),n,ptrFile) ;

	if(ret != n ) printf("Error al leer el archivo: %s",FileName) ;


	fclose(ptrFile);

	return ret ;
}
```