UNIVERSIDAD TECNOLÓGICA NACIONAL – FRBA

INGENIERÍA ELECTRÓNICA

Asignatura: Robótica
Curso: R6051

Código: 95-0482
Año: 2011

Trabajo Práctico n°1:

Análisis cinemático de la plataforma Stewart

Alumnos:

Cignoli, Rodolfo
Pietrasanta, Agustín

Profesor: Mas. Ing. GIANNETA, Hernan

JTP: Ing. GRANZELLA, Damián

Objetivos:

- Obtención del modelo cinemático inverso de una plataforma Stewart genérica.
- Implementación en Matlab.
- Implementación en la familia DSP Freescale 56800E, mediante el IDE de CodeWarrior.

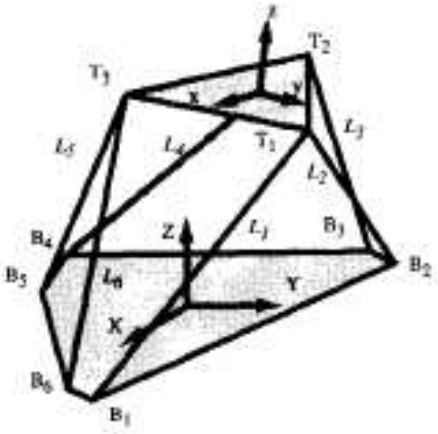
Introducción:

La plataforma Stewart tiene dos características fundamentales que la separan de los robots industriales. Posee seis actuadores lineales directamente conectados a una base móvil. Esta unión es simple y en forma cerrada, y con ella se logra un sistema mucho más rígido en proporción al tamaño y peso de cualquier manipulador de tipo cadena abierta. Por ser un sistema de cadena cerrada tiene muchas más restricciones de movimiento. Pero a la vez la fuerza de los seis actuadores se complementa obteniendo así una gran fuerza.

Cinemática Inversa y Modelado Físico:

En este trabajo, nuestro interés se centra en la configuración de tipo Stewart que utiliza seis actuadores lineales idénticos. La “Base” es un hexágono semi-regular. La parte superior “Top” es un triangulo equilátero. Los actuadores lineales son las “Piernas”, y están conectadas a los vértices de la Base y Top con articulaciones de 2 a 3 grados de libertad. Con esto se logra una sistema completo de 6 grados de libertad, es decir, traslación en los ejes X,Y,Z y rotación alrededor de los mismos.

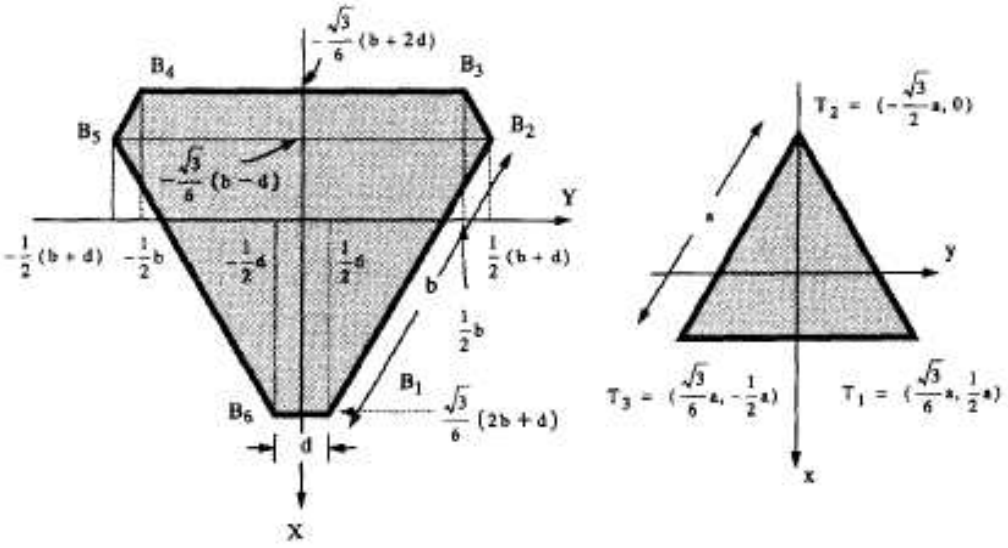
El marco de referencia (X,Y,Z) se encuentra en el centro de la Base con el eje Z apuntado verticalmente hacia arriba.



Sistema de referencia de la plataforma Stewart

Otro marco de referencia se sitúa en el centro de Top (x,y,z) en su centro de gravedad, con el eje z apuntando verticalmente.

A continuación se puede ver para la Base y para Top la relación de dimensiones.



Las longitudes de las piernas se denominarán **L_1, L_2, L_3,… L_6**, el origen del sistema Top se determina como **$[px,py,pz]^T$**, y la rotaciones respecto de cada eje **(alpha, beta, gamma),** donde alpha corresponde a rotación sobre el eje X, beta el eje Y y gamma sobre el eje Z.

Entonces la posición y orientación de la plataforma Top queda definida por el vector **X = [px, py, pz, alpha, beta, gamma].**

Mediante el estudio de la cinemática inversa se encuentran las longitudes que deberán tener las piernas, para cumplir con la posición y orientación que solicite el vector X.

Las expresiones de las longitudes de las piernas se obtienen mediante el siguiente razonamiento, que hacemos para la primera, y con la misma lógica se extiende al resto.

$$L_1 = \sqrt{(X_{T1} - X_{B1})^2 + (Y_{T1} - Y_{B1})^2 + (Z_{T1} - Z_{B1})^2}$$

Donde $X_{T1} \dots Z_{Bn}$ se obtienen del análisis geométrico de la plataforma. Ver referencia [1] para su estudio completo.

$$L_1 = \sqrt{\left(X_{T1} - d/2\sqrt{3} - b/\sqrt{3}\right)^2 + \left(Y_{T1} - d/2\right)^2 + Z_{T1}^2}$$
$$L_2 = \sqrt{\left(X_{T1} - d/2\sqrt{3} + b/2\sqrt{3}\right)^2 + \left(Y_{T1} - d/2 - b/2\right)^2 + Z_{T1}^2}$$
$$L_3 = \sqrt{\left(X_{T2} + d/\sqrt{3} + b/2\sqrt{3}\right)^2 + \left(Y_{T2} - b/2\right)^2 + Z_{T2}^2}$$
$$L_4 = \sqrt{\left(X_{T2} + d/\sqrt{3} + b/2\sqrt{3}\right)^2 + \left(Y_{T2} + b/2\right)^2 + Z_{T2}^2}$$
$$L_5 = \sqrt{\left(X_{T3} - d/2\sqrt{3} + b/2\sqrt{3}\right)^2 + \left(Y_{T3} + b/2 + d/2\right)^2 + Z_{T3}^2}$$
$$L_6 = \sqrt{\left(X_{T3} - d/2\sqrt{3} - b/\sqrt{3}\right)^2 + \left(Y_{T3} + d/2\right)^2 + Z_{T3}^2}\,.$$

Implementación en Matlab:

Script principal GraficoStewart.m:

Define los parámetros dimensionales de la plataforma y permite que el usuario determine una trayectoria, configurando las traslaciones respecto a los ejes cartesianos de la base, y sus rotaciones correspondientes.

```
clc;
clear all;
syms alpha beta gamma ; %alpha: rot en x
                        %beta: rot en y
                        %gamma: rot en z

syms px py pz;  %Distancia al origen del TOP respecto de BASE

%Parametros BASE stewart

b=925;    %mm Lado largo de la base
d=150;   %mm Lado corto de la base
a=612.5;  %mm Lado TOP

px=0;
py=0;
pz=200;

alpha=0;
beta=0;
gamma=0;

for m = 1:10

%EJEMPLO DE TRAYECTORIA:
%Configuracion para traslacion
px=0;
py=0;
pz=pz+60;
%Configuracion para rotacion
alpha=0;
beta=beta+0.1;
gamma=gamma+0.1;
```

```
%--------------------------------

%Llamo a script Stewart.m, para el cálculo cinemático.
Stewart;
```

```
%Grafico de líneas de la base
t = 1:1:7;
%Puntos de la BASE
Xb = [Xb1 Xb2 Xb3 Xb4 Xb5 Xb6 Xb1];
Yb = [Yb1 Yb2 Yb3 Yb4 Yb5 Yb6 Yb1];
Zb = [Zb1 Zb2 Zb3 Zb4 Zb5 Zb6 Zb1];
plot3(Xb(t), Yb(t), Zb(t),'-')
hold on;
%Grafico de líneas de Top
t = 1:1:4;
Xt = [Xt1 Xt2 Xt3 Xt1];
Yt = [Yt1 Yt2 Yt3 Yt1];
Zt = [Zt1 Zt2 Zt3 Zt1];
plot3(Xt(t), Yt(t), Zt(t),'-')
axis([-600 600 -600 600 0 1200]);
grid on;
hold on;
end

hold off;


Xb1 = (sqrt(3)/6)*(2*b+d);
Yb1 = (1/2)*d;
Zb1 = 0;

Xb2 = -(sqrt(3)/6)*(b-d);
Yb2 = (1/2)*(b+d);
Zb2 = 0;

Xb3 = -(sqrt(3)/6)*(b+2*d);
Yb3 = (1/2)*b;
Zb3 = 0;

Xb4 = -(sqrt(3)/6)*(b+2*d);
Yb4 = -(1/2)*b;
Zb4 = 0;

Xb5 = -(sqrt(3)/6)*(b-d);
Yb5 = -(1/2)*(b+d);
Zb5 = 0;

Xb6 = (sqrt(3)/6)*(2*b+d);
Yb6 = -(1/2)*d;
Zb6 = 0;

%Con subindice en minuscula es respecto a TOP
%Con mAyuscula respecto a la base
xt1 = (sqrt(3)/6)*a;
yt1 = (1/2)*a;
zt1 = 0;

xt2 = -(sqrt(3)/3)*a;
```

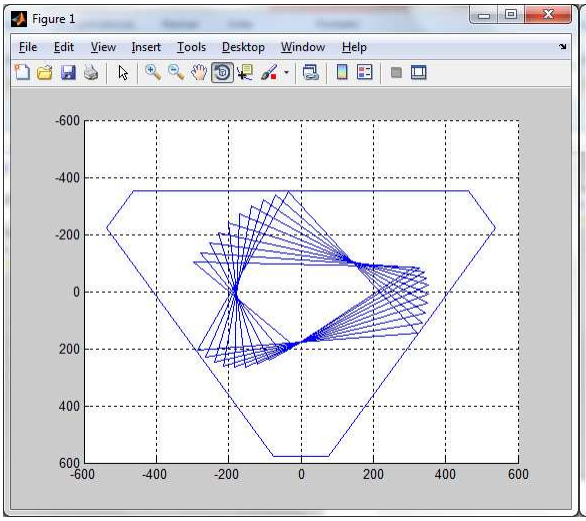
```
yt2 = 0;
zt2 = 0;
```

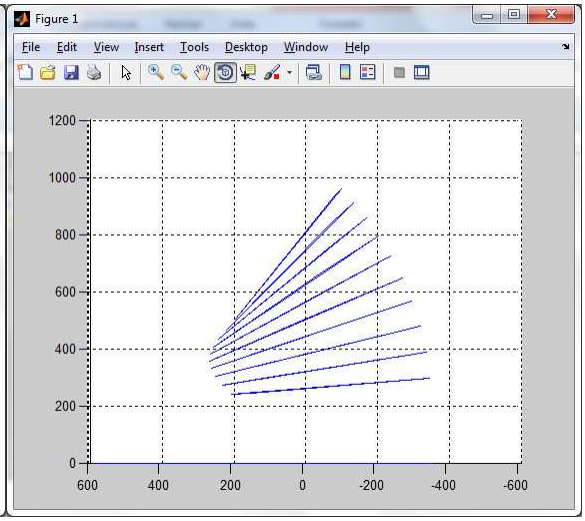
Script Stewart.m:

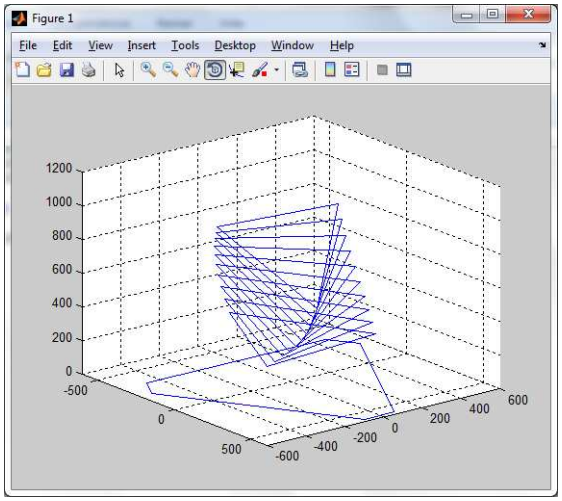
Genera la matriz transformación para obtener los desplazamientos de la plataforma TOP. Y calcula la longitudes de las piernas L1…L6 para cada punto de la trayectoria propuesta.

```
xt3 = (sqrt(3)/6)*a;
yt3 = -(1/2)*a;
zt3 = 0;

TTB = [cos(beta)*cos(gamma)+sin(alpha)*sin(beta)*sin(gamma) ...
        -cos(beta)*sin(gamma)+sin(alpha)*sin(beta)*cos(gamma) ...
        cos(alpha)*sin(beta) px;...

        cos(alpha)*sin(gamma)...
        cos(alpha)*cos(gamma)...
        -sin(alpha) py;...

        -sin(beta)*cos(gamma)+sin(alpha)*cos(beta)*sin(gamma) ...
         sin(beta)*sin(gamma)+sin(alpha)*cos(beta)*cos(gamma) ...
         cos(alpha)*cos(beta) pz;...

         0 0 0 1;...
        ];

XYZt1 = TTB * [xt1 yt1 zt1 1]';
XYZt2 = TTB * [xt2 yt2 zt2 1]';
XYZt3 = TTB * [xt3 yt3 zt3 1]';

Xt1=XYZt1(1);
Yt1=XYZt1(2);
Zt1=XYZt1(3);

Xt2=XYZt2(1);
Yt2=XYZt2(2);
Zt2=XYZt2(3);

Xt3=XYZt3(1);
Yt3=XYZt3(2);
Zt3=XYZt3(3);

L1 = sqrt((Xt1 - d/(2*sqrt(3)) - b/sqrt(3))^2 + (Yt1 - d/2)^2 + Zt1^2 );
L2 = sqrt((Xt1 - d/(2*sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt1 - d/2 - b/2)^2 +
Zt1^2 );
L3 = sqrt((Xt2 + d/(sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt2 - b/2)^2 + Zt2^2 );
L4 = sqrt((Xt2 + d/(sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt2 + b/2)^2 + Zt2^2 );
L5 = sqrt((Xt3 - d/(2*sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt3 + b/2 + d/2)^2 +
Zt3^2 );
L6 = sqrt((Xt3 - d/(2*sqrt(3)) - b/sqrt(3))^2 + (Yt3 + d/2)^2 + Zt3^2 );
```

Imágenes correspondientes a la trayectoria cargada en el script GraficoStewart.

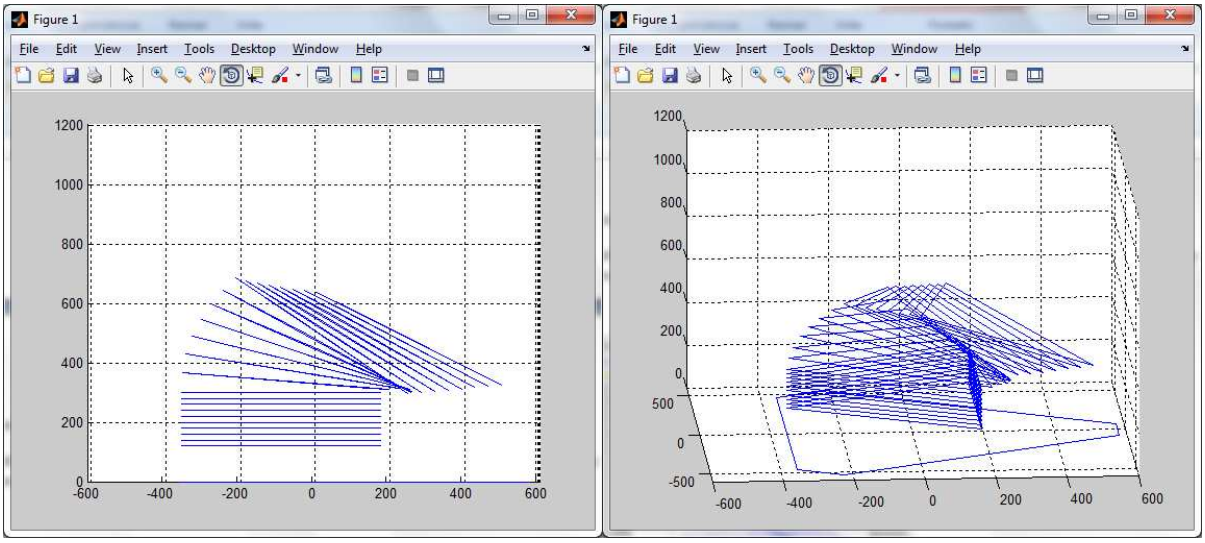




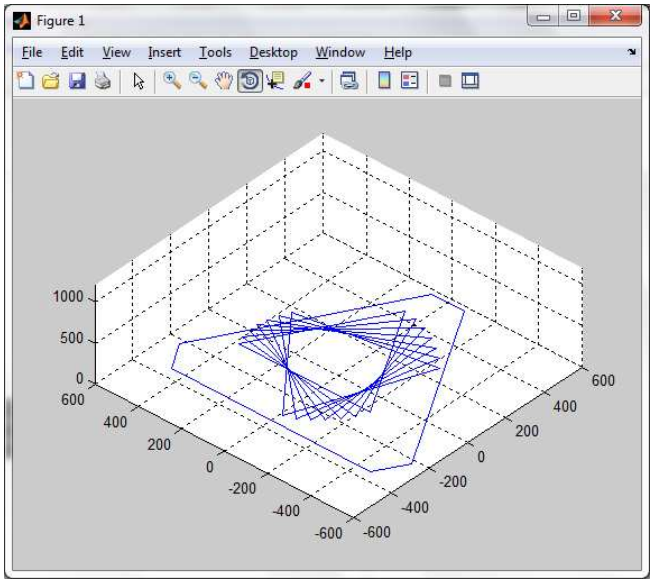


Otros ejemplos de trayectorias:

Traslación en z, luego rotación en beta y gamma, y luego traslación en x y menor rotación en beta y gamma.



Rotación simple sobre el eje Z de la base:



Implementación en DSP56800E

Se encuentra comentado el código de MATLAB correspondiente a cada instrucción del DSP.

```
/** ###################################################################
**     Filename  : tp17.C
**     Project   : tp17
**     Processor : 56F8367
**     Version   : Driver 01.14
**     Compiler  : Metrowerks DSP C Compiler
**     Date/Time : 26/04/2011, 06:28 p.m.
**     Abstract  :
**         Main module.
**         This module contains user's application code.
**     Settings  :
**     Contents  :
**         No public methods
**
** ###################################################################*/
/* MODULE tp17 */

/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "AFR1.h"
#include "MFR1.h"
#include "DFR1.h"
#include "MEM1.h"
#include "XFR1.h"
#include "TFR1.h"
#include "VFR1.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#include "stdio.h"

#define MXRAD 100
#define PULSE2RAD 32767/MXRAD

//Variables de dimensionamiento
Frac16 a,b,d;
//Variables de estado
Frac16 px, py, pz;
double alpha, beta, gama;

//Variables de actuadores
Word32 L1_32, L2_32, L3_32,
             L4_32, L5_32, L6_32;

Word16 L1, L2, L3,
             L4, L5, L6;
```

```
//Cofactores
Frac16 cosA, cosB, cosG, sinA ,sinB ,sinG;
Frac16 co11, co12, co13, co14,
       co21, co22, co23, co24,
       co31, co32,   co33, co34,
       co41, co42, co43, co44;

//Coordenadas BASE
Word16 Xb1, Xb2, Xb3, Xb4, Xb5, Xb6,
              Yb1, Yb2,Yb3, Yb4, Yb5, Yb6,
              Zb1, Zb2, Zb3, Zb4, Zb5, Zb6;

//Coordenadas TOP respecto de BASE
Word16 Xt1, Xt2, Xt3,
              Yt1, Yt2,Yt3,
              Zt1, Zt2, Zt3;

//Coordenadas TOP respecto de TOP
Word16 xt1, xt2, xt3,
              yt1, yt2,yt3,
              zt1, zt2, zt3;

Word32 aux32;
Word16 aux16;
Frac16 aux;

Frac16 c,i;
Frac16 pulse2rad=PULSE2RAD ,testResult[MXRAD],testResult2[MXRAD];
Word16 c16[MXRAD];
Word32 c32[MXRAD];

void main(void)
{
/* Write your local variable definition here */
/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
PE_low_level_init();
/*** End of Processor Expert internal initialization.                    ***/
/* Write your code here */

//Parametros BASE stewart

b=925  ;  //mm Lado largo de la base
d=150  ;  //mm Lado corto de la base
a=1225/2; //mm Lado TOP

//Traslacion de la plataforma
px=100;
py=100;
pz=100;

//Rotacion de la plataforma
//Completar en fraccion de rad.
//ej: alpha=0.1 equivale a 0.1 * pi => 18 grados
alpha = 0.1;
beta = 0.1;
gama = 0.1;

/*Calculo de coordenadas de BASE*/
```

```
//      Xb1 = (3^(1/2)/6)*(2*b+d);
        aux16 = FRAC16(0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,(2*b+d));
        Xb1 = extract_h(aux32);

//      Yb1 = (1/2)*d;
        aux16 = FRAC16(0.5);//(1/2)
        aux32 = L_mult(aux16,d);
        Yb1 = extract_h(aux32);

//      Zb1 = 0;
        Zb1 = 0;

//      Xb2 = -(sqrt(3)/6)*(b-d);
        aux16 = FRAC16(-0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,(b-d));
        Xb2 = extract_h(aux32);

//      Yb2 = (1/2)*(b+d);
        aux16 = FRAC16(0.5);//(1/2)
        aux32 = L_mult(aux16,b+d);
        Yb2 = extract_h(aux32);

//      Zb2 = 0;
        Zb2 = 0;

//      Xb3 = -(sqrt(3)/6)*(b+2*d);
        aux16 = FRAC16(-0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,(b+2*d));
        Xb3 = extract_h(aux32);

//      Yb3 = (1/2)*b;
        aux16 = FRAC16(0.5);//(1/2)
        aux32 = L_mult(aux16,b);
        Yb3 = extract_h(aux32);

//      Zb3 = 0;
        Zb3 = 0;

//      Xb4 = -(sqrt(3)/6)*(b+2*d);
        aux16 = FRAC16(-0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,(b+2*d));
        Xb4 = extract_h(aux32);

//      Yb4 = -(1/2)*b;
        aux16 = FRAC16(-0.5);//(1/2)
        aux32 = L_mult(aux16,b);
        Yb4 = extract_h(aux32);

//      Zb4 = 0;
        Zb4 = 0;

//      Xb5 = -(sqrt(3)/6)*(b-d);
        aux16 = FRAC16(-0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,(b-d));
        Xb5 = extract_h(aux32);

//      Yb5 = -(1/2)*(b+d);
```

```
        aux16 = FRAC16(-0.5);//(1/2)
        aux32 = L_mult(aux16,b+d);
        Yb5 = extract_h(aux32);

//      Zb5 = 0;
        Zb5 = 0;

//      Xb6 = (sqrt(3)/6)*(2*b+d);
        aux16 = FRAC16(0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,(2*b+d));
        Xb6 = extract_h(aux32);

//      Yb6 = -(1/2)*d;
        aux16 = FRAC16(-0.5);//(1/2)
        aux32 = L_mult(aux16,d);
        Yb6 = extract_h(aux32);

//      Zb6 = 0;
        Zb6 = 0;

/*      Con subindice en minuscula es respecto a TOP
        Con mAyuscula respecto a la base.
*/

//      xt1 = (sqrt(3)/6)*a;
        aux16 = FRAC16(0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,a);
        xt1 = extract_h(aux32);

//      yt1 = (1/2)*a;
        aux16 = FRAC16(0.5);//(1/2)
        aux32 = L_mult(aux16,a);
        yt1 = extract_h(aux32);

//      zt1 = 0;
        zt1 = 0;

//      xt2 = -(sqrt(3)/3)*a;
        aux16 = FRAC16(-0.5773);//(3^(1/2)/3)
        aux32 = L_mult(aux16,a);
        xt2 = extract_h(aux32);
//      yt2 = 0;
//      zt2 = 0;
        yt2 = 0;
        zt2 = 0;
//      xt3 = (sqrt(3)/6)*a;
        aux16 = FRAC16(0.2886);//(3^(1/2)/6)
        aux32 = L_mult(aux16,a);
        xt3 = extract_h(aux32);
//      yt3 = -(1/2)*a;
        aux16 = FRAC16(-0.5);//(1/2)
        aux32 = L_mult(aux16,a);
        yt3 = extract_h(aux32);
//      zt3 = 0;
        zt3 = 0;
```

```
for(;;) {

/*Precalculo valores de sin y cos para la matriz*/
       cosA = TFR1_tfr16CosPIx(FRAC16(alpha));
       cosB = TFR1_tfr16CosPIx(FRAC16(beta));
       cosG = TFR1_tfr16CosPIx(FRAC16(gama));

       sinA = TFR1_tfr16SinPIx(FRAC16(alpha));
       sinB = TFR1_tfr16SinPIx(FRAC16(beta));
       sinG = TFR1_tfr16SinPIx(FRAC16(gama));

//Cargo los cofactores de la matriz TTB

       co11 =
add(extract_h(L_mult(cosB,cosG)),extract_h(L_mult(extract_h(L_mult(sinA,sinB)),sinG)));
       co12 =
add(extract_h(L_mult(negate(cosB),sinG)),extract_h(L_mult(extract_h(L_mult(sinA,sinB)),co
sG)));
       co13 = extract_h(L_mult(cosA,sinB));
       co14 = px;

       co21 = extract_h(L_mult(cosA,sinG));
       co22 = extract_h(L_mult(cosA,cosG));
       co23 = negate(sinA);
       co24 = py;

       co31 =
add(extract_h(L_mult(negate(sinB),cosG)),extract_h(L_mult(extract_h(L_mult(sinA,cosB)),si
nG)));
       co32 =
add(extract_h(L_mult(sinB,sinG)),extract_h(L_mult(extract_h(L_mult(sinA,cosB)),cosG)));
       co33 = extract_h(L_mult(cosA,cosB));
       co34 = pz;

       co41 = 0;
       co42 = 0;
       co43 = 0;
       co44 = 1;

/*TTB = [cos(beta)*cos(gama)+sin(alpha)*sin(beta)*sin(gama) ...
           -cos(beta)*sin(gama)+sin(alpha)*sin(beta)*cos(gama) ...
           cos(alpha)*sin(beta) px;...

           cos(alpha)*sin(gama)...
           cos(alpha)*cos(gama)...
           -sin(alpha) py;...

           -sin(beta)*cos(gama)+sin(alpha)*cos(beta)*sin(gama) ...
            sin(beta)*sin(gama)+sin(alpha)*cos(beta)*cos(gama) ...
            cos(alpha)*cos(beta) pz;...

            0 0 0 1;...
           ];
*/

/*
       XYZt1 = TTB * [xt1 yt1 zt1 1]'
       XYZt2 = TTB * [xt2 yt2 zt2 1]';
```

```
        XYZt3 = TTB * [xt3 yt3 zt3 1]';

        Xt1=XYZt1(1);
        Yt1=XYZt1(2);
        Zt1=XYZt1(3);

        Xt2=XYZt2(1);
        Yt2=XYZt2(2);
        Zt2=XYZt2(3);

        Xt3=XYZt3(1);
        Yt3=XYZt3(2);
        Zt3=XYZt3(3);
*/

//Posicion de los vertices de la plataforma TOP respecto de la base.

        Xt1 =
add(add(add(extract_h(L_mult(co11,xt1)),extract_h(L_mult(co12,yt1))),extract_h(L_mult(co1
3,zt1))),co14);
        Yt1 =
add(add(add(extract_h(L_mult(co21,xt1)),extract_h(L_mult(co22,yt1))),extract_h(L_mult(co2
3,zt1))),co24);
        Zt1 =
add(add(add(extract_h(L_mult(co31,xt1)),extract_h(L_mult(co32,yt1))),extract_h(L_mult(co3
3,zt1))),co34);

        Xt2 =
add(add(add(extract_h(L_mult(co11,xt2)),extract_h(L_mult(co12,yt2))),extract_h(L_mult(co1
3,zt2))),co14);
        Yt2 =
add(add(add(extract_h(L_mult(co21,xt2)),extract_h(L_mult(co22,yt2))),extract_h(L_mult(co2
3,zt2))),co24);
        Zt2 =
add(add(add(extract_h(L_mult(co31,xt2)),extract_h(L_mult(co32,yt2))),extract_h(L_mult(co3
3,zt2))),co34);

        Xt3 =
add(add(add(extract_h(L_mult(co11,xt3)),extract_h(L_mult(co12,yt3))),extract_h(L_mult(co1
3,zt3))),co14);
        Yt3 =
add(add(add(extract_h(L_mult(co21,xt3)),extract_h(L_mult(co22,yt3))),extract_h(L_mult(co2
3,zt3))),co24);
        Zt3 =
add(add(add(extract_h(L_mult(co31,xt3)),extract_h(L_mult(co32,yt3))),extract_h(L_mult(co3
3,zt3))),co34);

//Longitudes de los actuadores lineales.

//      L1 = sqrt((Xt1 - d/(2*sqrt(3)) - b/sqrt(3))^2 + (Yt1 - d/2)^2 + Zt1^2 );
        aux32 = L_add(L_negate(L_mult(d, FRAC16(0.2886))),L_negate(L_mult(b,
FRAC16(0.5773))));
        aux16 = add(extract_h(aux32),Xt1);
        L1_32 = L_mult(aux16,aux16);
        aux16 = add(Yt1, negate(extract_h(L_mult(FRAC16(0.5),d))));
        L1_32 = L_add(L_mult(aux16,aux16),L1_32);
        L1_32 = L_add(L_mult(Zt1,Zt1),L1_32);
        L1 = mfr32Sqrt(L1_32);      //L1 actuador lineal
```

```
//      L2 = sqrt((Xt1 - d/(2*sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt1 - d/2 - b/2)^2 + Zt1^2 );
        aux32 = L_add(L_negate(L_mult(d, FRAC16(0.2886))),L_mult(b, FRAC16(0.2886)));
        aux16 = add(extract_h(aux32),Xt1);
        L2_32 = L_mult(aux16, aux16);
        aux16 = add(Yt1, negate(extract_h(L_mult(FRAC16(0.5),d))));
        aux16 = add(aux16, negate(extract_h(L_mult(FRAC16(0.5),b))));
        L2_32 = L_add(L_mult(aux16,aux16),L2_32);
        L2_32 = L_add(L_mult(Zt1,Zt1),L2_32);
        L2 = mfr32Sqrt(L2_32);


//      L3 = sqrt((Xt2 + d/(sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt2 - b/2)^2 + Zt2^2 );
        aux32 = L_add(L_mult(d, FRAC16(0.5773)),L_mult(b, FRAC16(0.2886)));
        aux16 = add(extract_h(aux32),Xt2);
        L3_32 = L_mult(aux16, aux16);
        aux16 = add(Yt2, negate(extract_h(L_mult(FRAC16(0.5),b))));
        L3_32 = L_add(L_mult(aux16,aux16),L3_32);
        L3_32 = L_add(L_mult(Zt2,Zt2),L3_32);
        L3 = mfr32Sqrt(L3_32);


//      L4 = sqrt((Xt2 + d/(sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt2 + b/2)^2 + Zt2^2 );
        aux32 = L_add(L_mult(d, FRAC16(0.5773)),L_mult(b, FRAC16(0.2886)));
        aux16 = add(extract_h(aux32),Xt2);
        L4_32 = L_mult(aux16, aux16);
        aux16 = add(Yt2, extract_h(L_mult(FRAC16(0.5),b)));
        L4_32 = L_add(L_mult(aux16,aux16),L4_32);
        L4_32 = L_add(L_mult(Zt2,Zt2),L4_32);
        L4 = mfr32Sqrt(L4_32);

//      L5 = sqrt((Xt3 - d/(2*sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt3 + b/2 + d/2)^2 + Zt3^2 );
        aux32 = L_add(L_negate(L_mult(d, FRAC16(0.2886))),L_mult(b, FRAC16(0.2886)));
        aux16 = add(extract_h(aux32),Xt3);
        L5_32 = L_mult(aux16,aux16);
        aux16 = add(Yt3, extract_h(L_mult(FRAC16(0.5),d)));
        aux16 = add(aux16, extract_h(L_mult(FRAC16(0.5),b)));
        L5_32 = L_add(L_mult(aux16,aux16),L5_32);
        L5_32 = L_add(L_mult(Zt3,Zt3),L5_32);
        L5 = mfr32Sqrt(L5_32);


//      L6 = sqrt((Xt3 - d/(2*sqrt(3)) - b/sqrt(3))^2 + (Yt3 + d/2)^2 + Zt3^2 );
        aux32 = L_add(L_negate(L_mult(d, FRAC16(0.2886))),L_negate(L_mult(b,
FRAC16(0.5773))));
        aux16 = add(extract_h(aux32),Xt3);
        L6_32 = L_mult(aux16,aux16);
        aux16 = add(Yt3, extract_h(L_mult(FRAC16(0.5),d)));
        L6_32 = L_add(L_mult(aux16,aux16),L6_32);
        L6_32 = L_add(L_mult(Zt3,Zt3),L6_32);
        L6 = mfr32Sqrt(L6_32);
  }
}
```

Tabla comparativa de simulaciones:

Traslación y rotación deseada	Px	0 mm	0 mm	0 mm	10 mm
	Py	0 mm	0 mm	0 mm	10 mm
	Pz	100 mm	120 mm	130 mm	140 mm
	Alpha	0°	18°	27°	36°
	Beta	0°	0°	0°	0°
	gamma	0°	0°	0°	18°
Resultados Matlab	L1	473 mm	485 mm	493 mm	531 mm
	L2	473 mm	487 mm	496 mm	479 mm
	L3	473 mm	477 mm	480 mm	505 mm
	L4	473 mm	477 mm	480 mm	457 mm
	L5	473 mm	471 mm	471 mm	521 mm
	L6	473 mm	470 mm	460 mm	419 mm
Resultados DSP	L1	475 mm	504 mm	523 mm	638 mm
	L2	471 mm	515 mm	548 mm	521 mm
	L3	473 mm	477 mm	480 mm	547 mm
	L4	473 mm	477 mm	480 mm	391 mm
	L5	471 mm	469 mm	478 mm	608 mm
	L6	475 mm	458 mm	449 mm	325 mm

Conclusiones:

Mediante el estudio de publicaciones con referencia a la plataforma Stewart, se logró la implementación de la cinemática inversa con la obtención de trayectorias. Luego al comparar los resultados de la simulación del DSP con nuestro script de Matlab, pudimos observar que si bien los resultados se comportan coherentemente, se observan errores notables cuando entran en juego las rotaciones. A efectos de la realización práctica de este Tp sería necesario utilizar un dsp con mayor precisión en el cálculo de los senos y cosenos, debido a que la propagación de los errores de los mismos es muy fuerte. Esto ocurre por la cantidad de operaciones realizadas en forma encadenada.

Referencias

[1] Kinematic Analysis of a Stewart Platform Manipulator. Kai Liu, *Member, IEEE,* John M. Fitzgerald, and Frank L. Lewis, *Senior Member, IEEE*