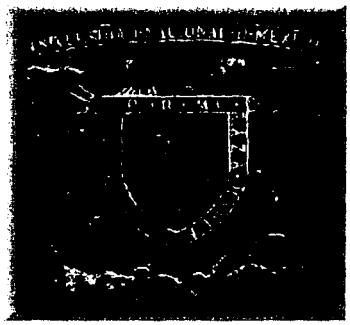


UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CONTROL DE UN BRAZO MECANICO UTILIZANDO PC's CONECTADAS A TRAVES DE INTERNET

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A

AGUSTIN PADILLA CABALLERO

DIRECTOR: DR. JESUS SAVAGE CARMONA

MEXICO, D. F 2003

A mi mamá y
mis hermanos

Agradecimientos

Deseo agradecer:

A mi mamá por el ánimo, el apoyo y la paciencia que me ha otorgado durante mis estudios.

A todos mis profesores de la Facultad de Ingeniería de todas las asignaturas cursadas durante mi estancia en la misma como alumno, así como a mi Director de Tesis, por sus enseñanzas y ejemplos a seguir.

A la Facultad de Ingeniería por el esmero en la enseñanza a sus alumnos.

Indice

1 Introducción

Una definición de Robótica es la siguiente:

Robótica es un conjunto de conocimientos con los cuales es posible construir un sistema formado de estructuras mecánicas móviles. Las estructuras mecánicas móviles se controlan con la ayuda de circuitos electrónicos programados incluidos en el sistema de robot. En ocasiones el sistema de robot se construye con la finalidad de utilizarlo en un ambiente de producción industrial para sustituir al hombre en diversas actividades. En la actualidad algunos robots pueden emplearse en tareas físicamente difíciles ó peligrosas para el ser humano, por ejemplo en ambientes donde la elevada temperatura no es adecuada para el ser humano.

Los robots de primera generación son incapaces de detectar estímulos del medio ambiente y actúan de acuerdo a una serie de instrucciones predeterminadas. Los robots de segunda generación poseen capacidad de detección sensorial, utilizando para ello sensores adecuados. Los robots de la tercera generación realizan análisis digital de imágenes, reconocen modelos, son programados para tomar decisiones, etc.

Un robot industrial esta formado por los siguientes elementos:

1. Un brazo mecánico.
2. Un dispositivo controlador. Esta parte del sistema en ocasiones está formada por un sistema electrónico el cual incluye, entre otros componentes, a uno ó más microprocesadores.
3. Actuadores. Esta parte del sistema está formada en ocasiones por motores eléctricos de corriente directa ó de pasos.
4. Elemento terminal. En ocasiones es posible tener más de un elemento terminal en un mismo robot industrial.

Es posible controlar algunos robots remotamente utilizando el sistema Internet, tal es el caso con el Brazo de Robot utilizado en el actual trabajo de tesis.

El término Internet hace referencia a una gran cantidad de redes interconectadas comunicando millones de computadoras en todo el planeta utilizando la familia de protocolos TCP/IP el cual tiene sus orígenes en el proyecto ARPANET del Departamento de la Defensa de Estados Unidos de Norteamérica en los años 60 y 70's. La red Internet es quizá la red de Area Amplia más grande del mundo, haciendo posible que cualquier computadora se comunique con cualquier otra computadora del planeta tan solo con el requisito de estar conectadas al sistema Internet.

El sistema Internet es un conjunto de redes interconectadas. Estas redes se encuentran distribuidas alrededor del mundo. En esta gran red internacional de comunicaciones por computadora se incluyen diversos servicios informáticos como el correo electrónico, la transmisión de archivos, la conexión con equipos lejanos, y también la transmisión de páginas de hipertexto las cuales se visualizan por medio de un programa especial denominado navegador.

En este trabajo de tesis el objetivo es el control de un Brazo Mecánico, específicamente el Brazo de Robot Rhino. El control se efectúa transmitiendo las posiciones indicadas por el usuario, utilizando un archivo conteniendo las posiciones.

El archivo se transmite hasta el lugar donde se encuentra el Brazo de Robot. Esto puede hacerse de diversas formas, en este caso se eligió la transmisión de archivos utilizando el sistema Internet. Los datos son transmitidos a través de Internet utilizando conectores software según el protocolo TCP/IP de comunicaciones. También se eligió el sistema operativo Windows por ser uno de los sistemas operativos más utilizados en las computadoras personales actualmente. El ambiente de utilización del sistema es precisamente el de PC's conectadas a través de Internet.

En la siguiente figura se muestra el sistema utilizado.

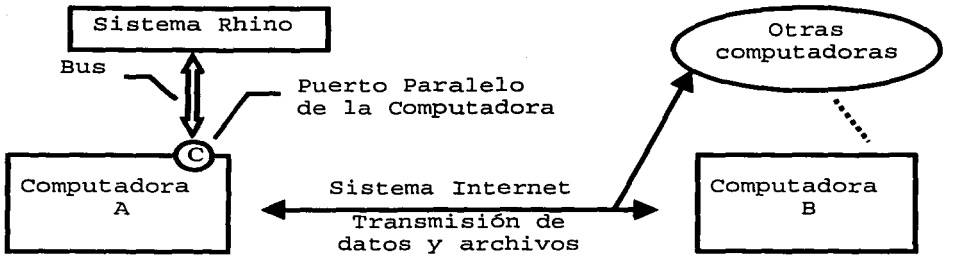


Figura 1.1. Sistema de comunicación del Brazo de Robot.

En una computadora A se conecta el Brazo de Robot Rhino, utilizando el conector del puerto paralelo de la computadora. El puerto paralelo de la computadora es entonces el único medio utilizado para controlar al Brazo de Robot desde la computadora personal.

En una computadora B el usuario del Brazo de Robot indica en un programa el cual muestra una ventana, las posiciones deseadas del Brazo de Robot. Los datos se conservan en un archivo. Posteriormente el archivo es transmitido hasta donde se encuentra la computadora A, utilizando para la transmisión de los datos, el sistema Internet. En la computadora A el archivo es utilizado de forma que el Brazo de Robot será accionado según las posiciones indicadas por el usuario en la computadora B. El sistema puede dar atención a cualquier número de usuarios, tan sólo limitado por el tiempo de transmisión de los datos a través del sistema Internet y por el límite de sobrecalentamiento por uso continuo excesivo del Brazo de Robot.

El Brazo de Robot Rhino está considerado como de tipo educacional, no es de tipo Industrial. Un uso práctico posible sería entonces utilizar el Brazo de Robot Rhino en un laboratorio de Robótica donde los usuarios son los alumnos del curso de Robótica. Cada alumno ó grupo de alumnos teniendo asignada una computadora con el sistema operativo Windows, introduce las posiciones deseadas del Brazo de Robot utilizando un programa el cual conserva las posiciones indicadas por el usuario, en un archivo. Y posteriormente utilizando

otros programas se envía el archivo a través de Internet hasta la computadora donde se encuentra el Brazo de Robot.

De acuerdo con los resultados obtenidos, el instructor decide enviar una indicación al alumno acerca del desempeño del movimiento del Brazo de Robot. Esta indicación sería enviada utilizando los programas del trabajo de tesis actual, diseñados para tal fin, utilizando el sistema Internet.

El alumno puede consultar al instructor, utilizando entonces otros programas diseñados para comunicarse utilizando el sistema Internet, también presentados en esta tesis.

Se eligió utilizar el puerto paralelo de la computadora pues el puerto serie ha sido utilizado en otras tesis sobre el mismo tema y se evita el uso de dispositivos especializados más avanzados como microprocesadores, tarjetas de captura de datos, etc. Una razón sería la de cuantificar la eficiencia, el costo, el grado de dificultad en la utilización, etc. de los diversos dispositivos y compararlos con la computadora y viceversa.

En el capitulo 1 se explica el desarrollo de esta tesis.

En el capítulo 2 se tratan los aspectos Cinemáticos y Dinámicos del Brazo de Robot. En el aspecto cinemático se aborda el problema de la Cinemática Directa e Inversa y se presentan los resultados obtenidos. En cuanto a la Dinámica se utiliza el Método de Newton-Euler.

En el capítulo 3 se abordan los aspectos relacionados con la comunicación vía Internet. Se presentan los principales protocolos utilizados en la comunicación por computadora: TCP e IP. También se da un breve resumen del protocolo HTTP utilizado para la transmisión de páginas de Internet. Se aborda el tema de las comunicaciones por computadora utilizando la tecnología de los conectores ó sockets y se presentan los algoritmos utilizados en el sistema diseñado para accionar el Brazo de Robot desde computadoras lejanas.

En el capítulo 4 se aborda el diseño electrónico del controlador utilizado para accionar el Brazo de Robot. También se presenta el diseño electrónico de la etapa

de potencia utilizada en el sistema. El controlador y la etapa de potencia se implementaron físicamente como parte del trabajo de tesis.

En este capítulo también se presenta un sistema de control para un motor de corriente directa y los diagramas electrónicos para realizarlo. Este sistema se elaboró teóricamente, con la ayuda de un programa especial para el desarrollo de sistemas de control.

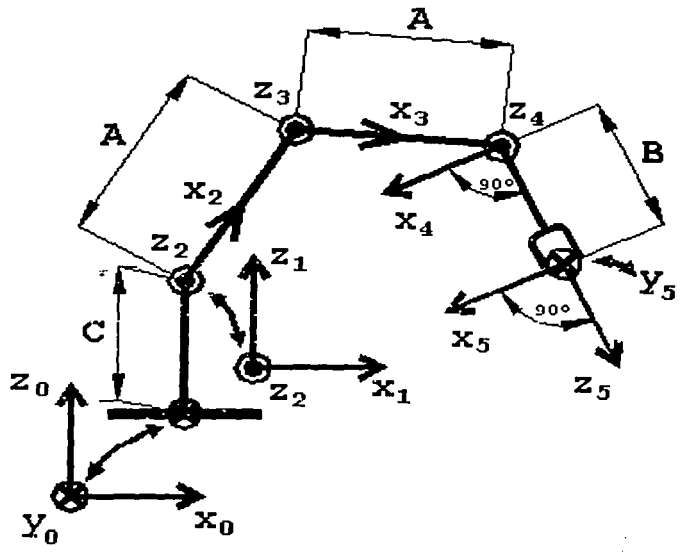
En el capítulo 5 se aborda el tratamiento de los diferentes algoritmos utilizados para accionar el Brazo de Robot desde la computadora. Estos programas se utilizan bajo el sistema operativo DOS. También se muestran los resultados obtenidos en la simulación del sistema de control de motor de C. D.

En el capítulo 6 se presentan las conclusiones obtenidas durante el actual trabajo de tesis.

Al final se muestra la bibliografía utilizada en el desarrollo de la misma.

2 Descripción del Brazo de Robot

La siguiente figura muestra los ejes utilizados en el análisis cinemático del Brazo de Robot Rhino.



A = 22.86 cm
B = 13.02 cm
C = 24.77 cm

Figura 2.1. Ejes de referencia asociados al Brazo de Robot.

En la Tabla 2.1 se tienen los parámetros de eslabón, donde las variables del primer renglón indican lo siguiente:

1. a_i es la distancia del eje Z_i al eje Z_{i+1} medida a lo largo del eje X_i.
2. α_i es el ángulo entre el eje Z_i y el eje Z_{i+1} medido con respecto al eje X_i.
3. d_i es la distancia del eje X_{i-1} al eje X_i medida a lo largo del eje Z_i.

4. θ_i es el ángulo entre el eje X_{i-1} y el eje X_i medido con respecto al eje Z_i.

La tabla de Parámetros de eslabón para el manipulador es la siguiente:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	C	θ_1
2	90 °	0	0	θ_2
3	0	A	0	θ_3
4	0	A	0	θ_4
5	-90 °	0	B	θ_5

Tabla 2.1. Tabla de Parámetros de eslabón.

La siguiente figura indica la notación utilizada en la descripción del Brazo de Robot. La notación es análoga a la utilizada en la referencia [3].

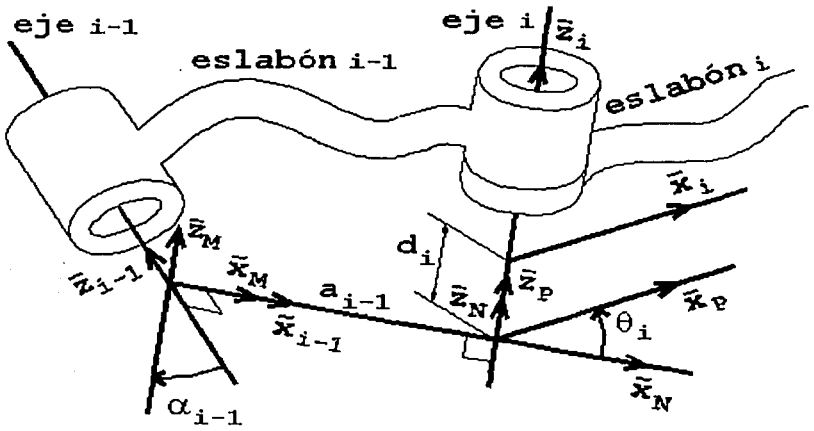


Figura 2.2. Notación cinemática utilizada.

De acuerdo con la notación de la referencia [3], el sistema Rhino posee articulaciones rotacionales descritas por la siguiente ecuación:

$$T_{i}^{i-1} = \begin{bmatrix} \cos\theta_i & -\operatorname{sen}\theta_i & 0 & a_{i-1} \\ \cos\alpha_{i-1}\operatorname{sen}\theta_i & \cos\alpha_{i-1}\cos\theta_i & -\operatorname{sen}\alpha_{i-1} & -\operatorname{sen}\alpha_{i-1}d_i \\ \operatorname{sen}\alpha_{i-1}\operatorname{sen}\theta_i & \operatorname{sen}\alpha_{i-1}\cos\theta_i & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 2.1

La matriz T_5^0 describe la posición y orientación del eslabón cinco, con respecto al eslabón cero del Brazo de Robot.

$$T_5^0 = T_1^0 \times T_2^1 \times T_3^2 \times T_4^3 \times T_5^4$$

Ecuación 2.2

Los elementos de la Matriz de Transformación T_5^0 [i , j] del Brazo de Robot Rhino son los siguientes, donde:

1. i representa el número de renglón
2. j representa el número de columna

T_5^0 [1 , 1] =

$\cos\theta_1 \cos\theta_2 \cos\theta_3 \cos\theta_4 \cos\theta_5 -$

$\cos\theta_1 \operatorname{sen}\theta_2 \operatorname{sen}\theta_3 \cos\theta_4 \cos\theta_5 -$

$\cos\theta_1 \cos\theta_2 \operatorname{sen}\theta_3 \operatorname{sen}\theta_4 \cos\theta_5 -$

$\cos\theta_1 \operatorname{sen}\theta_2 \cos\theta_3 \operatorname{sen}\theta_4 \cos\theta_5 -$

$\operatorname{sen}\theta_1 \operatorname{sen}\theta_5$;

$T_5^0\ [1\ ,\ 2] =$

$\cos\theta_1\ \text{sen}\,\theta_2\ \text{sen}\,\theta_3\ \cos\theta_4\ \text{sen}\,\theta_5\ -$

$\cos\theta_1\ \cos\theta_2\ \cos\theta_3\ \cos\theta_4\ \text{sen}\,\theta_5\ +$

$\cos\theta_1\ \text{sen}\,\theta_2\ \cos\theta_3\ \text{sen}\,\theta_4\ \text{sen}\,\theta_5\ +$

$\cos\theta_1\ \cos\theta_2\ \text{sen}\,\theta_3\ \text{sen}\,\theta_4\ \text{sen}\,\theta_5\ -$

$\text{sen}\,\theta_1\ \cos\theta_5\ ;$

$T_5^0\ [1\ ,\ 3] =$

$\cos\theta_1\ \text{sen}\,\theta_2\ \text{sen}\,\theta_3\ \text{sen}\,\theta_4\ -$

$\cos\theta_1\ \cos\theta_2\ \cos\theta_3\ \text{sen}\,\theta_4\ -$

$\cos\theta_1\ \text{sen}\,\theta_2\ \cos\theta_3\ \cos\theta_4\ -$

$\cos\theta_1\ \cos\theta_2\ \text{sen}\,\theta_3\ \cos\theta_4\ ;$

$T_5^0\ [1\ ,\ 4] =$

$A\ (\cos\theta_1\ \cos\theta_2\ +\ \cos\theta_1\ \cos\theta_2\ \cos\theta_3\ -$

$\cos\theta_1\ \text{sen}\,\theta_2\ \text{sen}\,\theta_3\ )\ +$

$B\ (\cos\theta_1\ \text{sen}\,\theta_2\ \text{sen}\,\theta_3\ \ \text{sen}\,\theta_4\ -$

$\cos\theta_1\ \text{sen}\,\theta_2\ \cos\theta_3\ \cos\theta_4\ -$

$\cos\theta_1\ \cos\theta_2\ \text{sen}\,\theta_3\ \cos\theta_4\ -$

$\cos\theta_1\ \cos\theta_2\ \cos\theta_3\ \text{sen}\,\theta_4\ )\ ;$

$T_5^0\ [2\ ,\ 1] =$

$\text{sen}\ \theta_1\ \cos\theta_2\ \cos\theta_3\ \cos\theta_4\ \cos\theta_5\ -$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \text{sen}\ \theta_3\ \cos\theta_4\ \cos\theta_5\ -$

$\text{sen}\ \theta_1\ \cos\theta_2\ \text{sen}\ \theta_3\ \text{sen}\ \theta_4\ \cos\theta_5\ -$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \cos\theta_3\ \text{sen}\ \theta_4\ \cos\theta_5\ +$

$\cos\theta_1\ \text{sen}\ \theta_5\ ;$

$T_5^0\ [2\ ,\ 2] =$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \text{sen}\ \theta_3\ \cos\theta_4\ \text{sen}\ \theta_5\ -$

$\text{sen}\ \theta_1\ \cos\theta_2\ \cos\theta_3\ \cos\theta_4\ \text{sen}\ \theta_5\ +$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \cos\theta_3\ \text{sen}\ \theta_4\ \text{sen}\ \theta_5\ +$

$\text{sen}\ \theta_1\ \cos\theta_2\ \text{sen}\ \theta_3\ \text{sen}\ \theta_4\ \text{sen}\ \theta_5\ +$

$\cos\theta_1\ \cos\theta_5\ ;$

$T_5^0\ [2\ ,\ 3] =$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \text{sen}\ \theta_3\ \text{sen}\ \theta_4\ -$

$\text{sen}\ \theta_1\ \cos\theta_2\ \cos\theta_3\ \text{sen}\ \theta_4\ -$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \cos\theta_3\ \cos\theta_4\ -$

$\text{sen}\ \theta_1\ \cos\theta_2\ \text{sen}\ \theta_3\ \cos\theta_4\ ;$

$T_5^0\ [2\ ,\ 4] =$

$A\ (\ \text{sen}\ \theta_1\ \cos\theta_2\ +\ \text{sen}\ \theta_1\ \cos\theta_2\ \cos\theta_3\ -$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \text{sen}\ \theta_3\ )\ +$

$B\ (\ \text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \text{sen}\ \theta_3\ \text{sen}\ \theta_4\ -$

$\text{sen}\ \theta_1\ \text{sen}\ \theta_2\ \cos\theta_3\ \cos\theta_4\ -$

$\text{sen}\ \theta_1\ \cos\theta_2\ \text{sen}\ \theta_3\ \cos\theta_4\ -$

$\text{sen}\ \theta_1\ \cos\theta_2\ \cos\theta_3\ \text{sen}\ \theta_4\ )\ ;$

$T_5^0\ [3\ ,\ 1] =$

$\text{sen}\ \theta_2\ \cos\theta_3\ \cos\theta_4\ \cos\theta_5\ +$

$\cos\theta_2\ \text{sen}\ \theta_3\ \cos\theta_4\ \cos\theta_5\ +$

$\cos\theta_2\ \cos\theta_3\ \text{sen}\ \theta_4\ \cos\theta_5\ -$

$\text{sen}\ \theta_2\ \text{sen}\ \theta_3\ \text{sen}\ \theta_4\ \cos\theta_5\ ;$

$T_5^0\ [3\ ,\ 2] =$

$-\ \text{sen}\ \theta_2\ \cos\theta_3\ \cos\theta_4\ \text{sen}\ \theta_5\ -$

$\cos\theta_2\ \text{sen}\ \theta_3\ \cos\theta_4\ \text{sen}\ \theta_5\ +$

$\text{sen}\ \theta_2\ \text{sen}\ \theta_3\ \text{sen}\ \theta_4\ \text{sen}\ \theta_5\ -$

$\cos\theta_2\ \cos\theta_3\ \text{sen}\ \theta_4\ \text{sen}\ \theta_5\ ;$

$T_5^0[3,3] =$

$-\sin\theta_2 \cos\theta_3 \sin\theta_4 -$

$\cos\theta_2 \sin\theta_3 \sin\theta_4 +$

$\cos\theta_2 \cos\theta_3 \cos\theta_4 -$

$\sin\theta_2 \sin\theta_3 \cos\theta_4 ;$

$T_5^0[3,4] =$

$A(\sin\theta_2 + \sin\theta_2 \cos\theta_3 +$

$\cos\theta_2 \sin\theta_3) + B(\cos\theta_2 \cos\theta_3 \cos\theta_4 -$

$\sin\theta_2 \sin\theta_3 \cos\theta_4 -$

$\sin\theta_2 \cos\theta_3 \sin\theta_4 -$

$\cos\theta_2 \sin\theta_3 \sin\theta_4) + C ;$

$T_5^0[4,1] = 0.0 ;$

$T_5^0[4,2] = 0.0 ;$

$T_5^0[4,3] = 0.0 ;$

$T_5^0[4,4] = 1.0 ;$

2.1 Cinemática Directa

El problema de la Cinemática Directa en el caso del Brazo de Robot, es el siguiente:

Dados los valores de θ_1, θ_2, θ_3, θ_4, θ_5, hallar la posición cartesiana correspondiente (x, y, z), del Brazo de Robot, tomando como base el marco de referencia cero.

En el cálculo de la Cinemática Directa se utiliza el siguiente algoritmo:

1. Leer los ángulos θ_1, θ_2, θ_3, θ_4, θ_5, y la posición del gripper.
2. Calcular T_5^0.
3. Archivar T_5^0 y la posición del gripper.

Durante el cálculo de la Matriz de Transformación T_5^0, se utilizan los valores de los ángulos θ_1, θ_2, θ_3, θ_4, θ_5, así como el valor de la longitud de los eslabones A, B, C. El tamaño de la matriz T_5^0 es de cuatro renglones, cuatro columnas.

2.2 Cinemática Inversa

El problema de la Cinemática Inversa, en relación con el Brazo de Robot, es el siguiente:

Dada la posición cartesiana (x, y, z), y la orientación indicada por el usuario (ρ, t5) del Brazo de Robot, hallar los valores de las variables de unión θ_1, θ_2, θ_3, θ_4, θ_5, correspondientes.

En el cálculo de la Cinemática Inversa se leen los siguientes valores:

x_0, y_0, z_0, ρ, t5, gripper.

Cuando los valores x_0, y_0, son muy pequeños, es necesario proporcionar los valores x_2, y_2, donde cuando menos uno de los valores absolutos de x_2, y_2, debe ser mayor de 1.0×10^{-6}. Esto es necesario para poder calcular el ángulo θ_1.

El valor de ρ varía en el rango [-180..180] grados; y el valor de gripper varía en el rango [-185..185], t5 corresponde a la rotación del gripper.

El sentido positivo del ángulo ρ se mide como se indica en la siguiente figura:

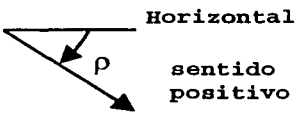


Figura 2.3. Sentido positivo del ángulo ρ.

Los demás ángulos se miden como se indica a continuación:

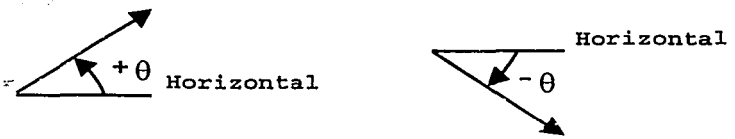


Figura 2.4. Medición de ángulos, excepto ρ.

El esquema del Brazo de Robot es el siguiente:

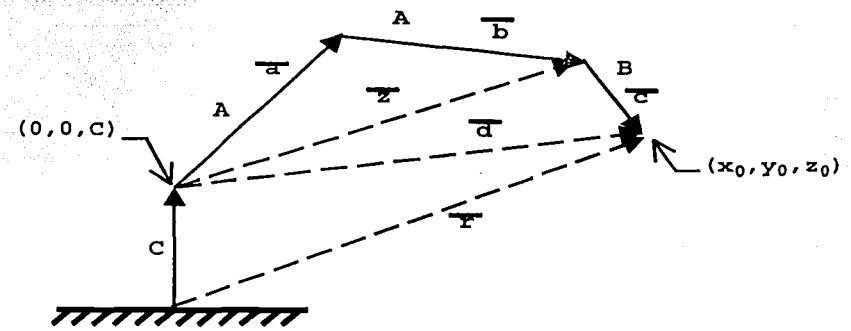


Figura 2.5. Esquema del Brazo de Robot.

El marco de referencia 0 se relaciona con la posición angular de la base del Brazo de Robot de la siguiente manera.

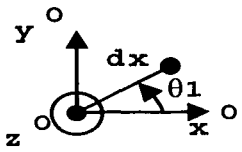


Figura 2.6. Relación entre el marco de referencia 0 y el Brazo de Robot.

Las variables utilizadas en la determinación de la Cinemática Inversa se relacionan geométricamente como se indica en la siguiente figura.

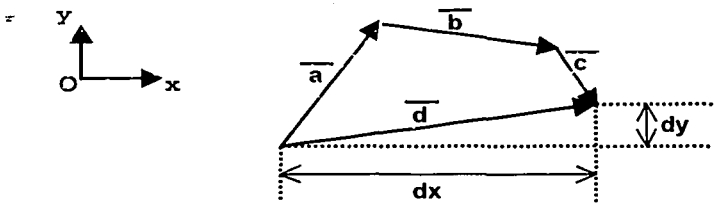


Figura 2.7. Relación geométrica entre las variables, en el caso de la Cinemática Inversa.

De la Figura 2.6 y la Figura 2.7 se obtienen las siguientes expresiones:

$$dx = \sqrt{x_0^2 + y_0^2}$$

Ecuación 2.3

$$dy = z_0 - C$$

Ecuación 2.4

$$\bar{z} = \bar{d} - \bar{c}$$

Ecuación 2.5

La siguiente figura muestra la relación de $\bar{z}$ con los ángulos α y β.

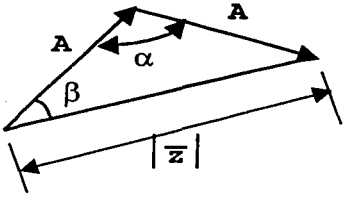


Figura 2.8. Relación entre $\bar{z}$ y los ángulos α y β.

En la determinación de los diversos ángulos se obtienen los siguientes valores.

$$\theta_1 = \text{atan2}(y_0, x_0)$$

Ecuación 2.6

ó cuando los valores x_0, y_0 sean muy pequeños:

$$\theta_1 = \text{atan2}(y_2, x_2)$$

Ecuación 2.7

también:

$$\theta_5 = t5, \ \angle \bar{c} = -\rho$$

Ecuación 2.8

Del triángulo isósceles de la Figura 2.8 se observa lo siguiente:

$$\beta = \cos^{-1}\left(\frac{|\bar{z}|}{2A}\right)$$

Ecuación 2.9

$$\alpha = \pi - 2\beta$$

Ecuación 2.10

Con el eslabón de longitud "B" de la Figura 2.1 se visualiza el ángulo del vector "c" de la siguiente manera:

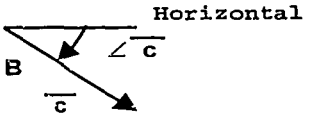


Figura 2.9. Angulo del vector "c", ∠ c.

En la Figura 2.9, el ángulo del vector "c" es de valor negativo. A continuación se tratan dos casos posibles para elegir los ángulos: ∠ a, ∠ b y ∠ c, Figura 2.5, de los vectores correspondientes a, b, c. Estos ángulos se miden con respecto a un plano horizontal como se indica en la Figura 2.4.

Caso 1:

En este caso se tiene la siguiente condición:

Ángulo del vector "c" < 0.

Tenemos la siguiente figura.

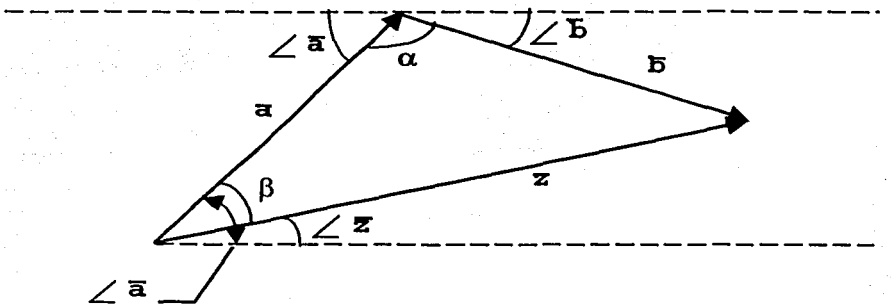


Figura 2.10. Angulo del vector "c" < 0.

Se obtuvieron las siguientes ecuaciones:

$$\angle \bar{a} = \angle \bar{z} + \beta$$

Ecuación 2.11

$$\angle \bar{b} = \angle \bar{a} - (\pi - \alpha)$$

Ecuación 2.12

Caso 2:

En este caso la condición es la siguiente:

Ángulo del vector "c" ≥ 0.

Para esta situación utilizaremos la siguiente figura.

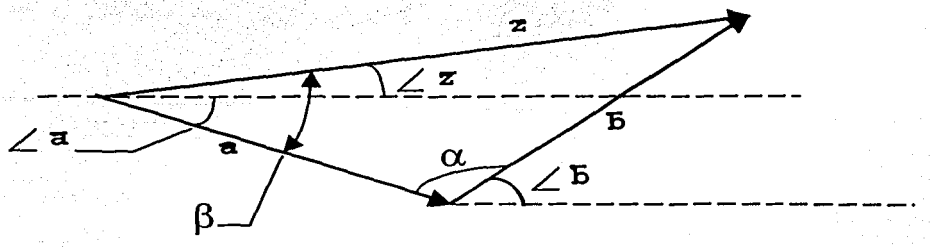


Figura 2.11. Angulo del vector "c" ≥ 0.

En este caso las ecuaciones son las que se muestran a continuación:

$$\angle \bar{a} = \angle \bar{z} - \beta$$

Ecuación 2.13

$$\angle \bar{b} = \angle \bar{a} + (\pi - \alpha)$$

Ecuación 2.14

2.3 Dinámica

La siguiente figura muestra la disposición de los ejes en el tratamiento de la Dinámica de un Brazo de Robot planar de tres eslabones, de uniones rotacionales.

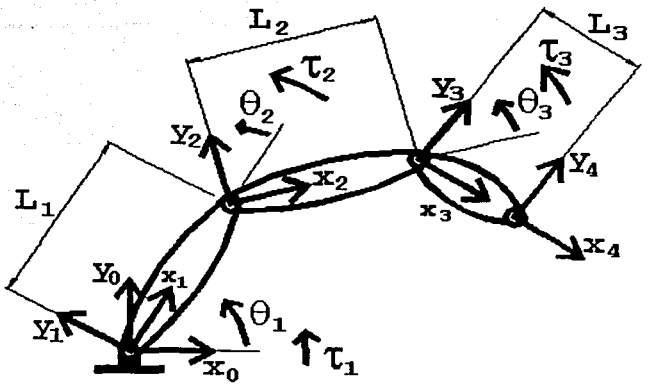


Figura 2.12. Ejes utilizados en el tratamiento dinámico del Brazo de Robot.

El método utilizado para calcular la Dinámica del Brazo de Robot planar de tres eslabones es el algoritmo de Newton-Euler de tipo iterativo, analizado en la referencia [3] y cuyas ecuaciones se presentan a continuación para el caso en que todas las uniones son rotacionales:

Iteraciones hacia adelante,

$$i: 0 .. n - 1.$$

$$\omega_{i+1}^{i+1} = R_{i}^{i+1} \omega_{i}^{i} + \dot{\theta}_{i+1} \overline{z}_{i+1}^{i+1}$$

Ecuación 2.15

La Ecuación 2.15 permite calcular el vector de velocidad angular del eslabón i+1.

$$\dot{\omega}_{i+1}^{i+1} = R_{i}^{i+1} \dot{\omega}_{i}^{i} + R_{i}^{i+1} \omega_{i}^{i} \times \dot{\theta}_{i+1} \overline{z}_{i+1}^{i+1} + \ddot{\theta}_{i+1} \overline{z}_{i+1}^{i+1}$$

Ecuación 2.16

La Ecuación 2.16 proporciona el vector de aceleración angular del eslabón i+1.

$$\dot{v}_{i+1}^{i+1} = R_{i}^{i+1}(\dot{\omega}_{i}^{i} \times P_{i+1}^{i} + \omega_{i}^{i} \times (\omega_{i}^{i} \times P_{i+1}^{i}) + \dot{v}_{i}^{i})$$

Ecuación 2.17

La Ecuación 2.17 nos proporciona el vector de aceleración lineal del eslabón i+1

$$\dot{v}_{C_{i+1}}^{i+1} = \dot{\omega}_{i+1}^{i+1} \times P_{C_{i+1}}^{i+1} + \omega_{i+1}^{i+1} \times (\omega_{i+1}^{i+1} \times P_{C_{i+1}}^{i+1}) + \dot{v}_{i+1}^{i+1}$$

Ecuación 2.18

La Ecuación 2.18 permite calcular el vector de aceleración lineal del centro de masa del eslabón i+1. El marco C_{i+1} está ubicado en el centro de masa del eslabón i+1, con la misma orientación del marco i+1.

$$F_{i+1}^{i+1} = m_{i+1} \dot{v}_{C_{i+1}}^{i+1}$$

Ecuación 2.19

La Ecuación 2.19 permite el cálculo del vector de fuerza inercial ejercida en el centro de masa del eslabón i+1.

$$N_{i+1}^{i+1} = I_{i+1}^{C_{i+1}} \dot{\omega}_{i+1}^{i+1} + \omega_{i+1}^{i+1} \times I_{i+1}^{C_{i+1}} \omega_{i+1}^{i+1}$$

Ecuación 2.20

La Ecuación 2.20 nos permite calcular el vector de torque ejercido respecto al centro de masa del eslabón i+1.

Iteraciones hacia atrás,

$$i: n \ldots 1.$$

$$f_i^i = R_{i+1}^i f_{i+1}^{i+1} + F_i^i$$

Ecuación 2.21

La Ecuación 2.21 permite evaluar el vector de fuerza ejercida en el eslabón i, con respecto al marco i.

$$n_i^i = N_i^i + R_{i+1}^i n_{i+1}^{i+1} + P_{C_i}^i \times F_i^i + P_{i+1}^i \times R_{i+1}^i f_{i+1}^{i+1}$$

Ecuación 2.22

Con la Ecuación 2.22 se calcula el vector de torque ejercido en el eslabón i, respecto al marco i.

$$\tau_i = (n_i^i)^T \bar{z}_i^i$$

Ecuación 2.23

La Ecuación 2.23 da el vector de torque necesario en cada eslabón.

Para el caso planar de tres eslabones rotacionales se tienen las siguientes ecuaciones, basadas en los resultados de la referencia [3].

Las condiciones iniciales son la siguientes:

$$P_{C_1}^{1} = L_1 \bar{x}_1 \qquad I_1^{C} 1 = 0$$

$$P_{C_2}^{2} = L_2 \bar{x}_2 \qquad I_2^{C} 2 = 0$$

$$P_{C_3}^{3} = L_3 \bar{x}_3 \qquad I_3^{C} 3 = 0$$

$$f_4 = 0 \qquad \omega_0 = 0$$

$$n_4 = 0 \qquad \dot{\omega}_0 = 0$$

$$\dot{v}_0^0 = g \; \bar{y}_0$$

$$P_1^0 = (0, 0, 0)^T$$

$$P_2^1 = (L_1, 0, 0)^T$$

$$P_3^2 = (L_2, 0, 0)^T$$

$$P_4^3 = (L_3, 0, 0)^T$$

También se tienen las siguientes matrices genéricas utilizadas en el desarrollo del cálculo de diversos términos.

$$R_{i+1}^{i} = \begin{bmatrix} \cos\theta_{i+1} & -\operatorname{sen}\theta_{i+1} & 0 \\ \operatorname{sen}\theta_{i+1} & \cos\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 2.24

$$R\,{}^{i+1}_{i} = \begin{bmatrix} \cos\theta_{i+1} & \operatorname{sen}\theta_{i+1} & 0 \\ -\operatorname{sen}\theta_{i+1} & \cos\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 2.25

Iteraciones hacia adelante en el eslabón 1:

$$\omega\,{}^{1}_{1} = R\,{}^{1}_{0}\;\omega\,{}^{0}_{0} + \dot{\theta}_1\;\bar{z}\,{}^{1}_{1} = 0 + \dot{\theta}_1\;\bar{z}\,{}^{1}_{1} = \dot{\theta}_1\;\bar{z}\,{}^{1}_{1}$$

$$\omega\,{}^{1}_{1} = \dot{\theta}_1\;\bar{z}\,{}^{1}_{1} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}$$

$$\dot{\omega}\,{}^{1}_{1} = R\,{}^{1}_{0}\;\dot{\omega}\,{}^{0}_{0} + R\,{}^{1}_{0}\;\omega\,{}^{0}_{0} \times \dot{\theta}_1\;\bar{z}\,{}^{1}_{1} + \ddot{\theta}_1\;\bar{z}\,{}^{1}_{1}$$
$$= 0 + 0 + \ddot{\theta}_1\;\bar{z}\,{}^{1}_{1} = \ddot{\theta}_1\;\bar{z}\,{}^{1}_{1}$$

$$\dot{\omega}\,{}^{1}_{1} = \ddot{\theta}_1\;\bar{z}\,{}^{1}_{1} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix}$$

$$\dot{v}\,{}^{1}_{1} = R\,{}^{1}_{0}\,(\,\dot{\omega}\,{}^{0}_{0} \times P\,{}^{0}_{1} + \omega\,{}^{0}_{0} \times (\,\omega\,{}^{0}_{0} \times P\,{}^{0}_{1}\,) + \dot{v}\,{}^{0}_{0}\,)$$
$$= R\,{}^{1}_{0}\,(\,0 + 0 + \dot{v}\,{}^{0}_{0}\,) = R\,{}^{1}_{0}\,\dot{v}\,{}^{0}_{0}$$

$$\dot{v}_1^1 = \begin{bmatrix} g \ \text{sen} \ \theta_1 \\ g \ \cos \ \theta_1 \\ 0 \end{bmatrix}$$

$$\dot{v}_{C_1}^1 = \dot{\omega}_1^1 \times P_{C_1}^1 + \omega_1^1 \times (\omega_1^1 \times P_{C_1}^1) + \dot{v}_1^1$$

$$\dot{v}_{C_1}^1 = \begin{bmatrix} -L_1 \dot{\theta}_1^2 + g \ \text{sen} \ \theta_1 \\ -L_1 \ddot{\theta}_1 + g \cos \ \theta_1 \\ 0 \end{bmatrix}$$

$$F_1^1 = m_1 \dot{v}_{C_1}^1$$

$$F_1^1 = \begin{bmatrix} -m_1 L_1 \dot{\theta}_1^2 + m_1 g \ \text{sen} \ \theta_1 \\ m_1 L_1 \ddot{\theta}_1 + m_1 g \cos \ \theta_1 \\ 0 \end{bmatrix}$$

$$N_1^1 = I_1^{C_1} \dot{\omega}_1^1 + \omega_1^1 \times I_1^{C_1} \omega_1^1$$

$$N_1^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Iteraciones hacia adelante en el eslabón 2:

$$\omega \,{}^{2}_{2} = R \,{}^{2}_{1}\, \omega \,{}^{1}_{1} + \dot{\theta}_2 \, \bar{z} \,{}^{2}_{2}$$

$$\omega \,{}^{2}_{2} = R \,{}^{2}_{1}\, \omega \,{}^{1}_{1} + \dot{\theta}_2 \, \bar{z} \,{}^{2}_{2}$$

$$\dot{\omega} \,{}^{2}_{2} = R \,{}^{2}_{1}\, \dot{\omega} \,{}^{1}_{1} + R \,{}^{2}_{1}\, \omega \,{}^{1}_{1} \times \dot{\theta}_2 \, \bar{z} \,{}^{2}_{2} + \ddot{\theta}_2 \, \bar{z} \,{}^{2}_{2}$$

$$\dot{\omega} \,{}^{2}_{2} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 + \ddot{\theta}_2 \end{bmatrix}$$

$$\dot{v} \,{}^{2}_{2} = R \,{}^{2}_{1} (\dot{\omega} \,{}^{1}_{1} \times P \,{}^{1}_{2} + \omega \,{}^{1}_{1} \times (\omega \,{}^{1}_{1} \times P \,{}^{1}_{2}) + \dot{v} \,{}^{1}_{1})$$

$$\dot{v} \,{}^{2}_{2} = \begin{bmatrix} L_1 \ddot{\theta}_1 \operatorname{sen} \theta_2 - L_1 \dot{\theta}_1^2 \cos \theta_2 + g \operatorname{sen} (\theta_1 + \theta_2) \\ L_1 \ddot{\theta}_1 \cos \theta_2 + L_1 \dot{\theta}_1^2 \operatorname{sen} \theta_2 + g \cos (\theta_1 + \theta_2) \\ 0 \end{bmatrix}$$

$$\dot{v} \,{}^{2}_{C_2} = \dot{\omega} \,{}^{2}_{2} \times P \,{}^{2}_{C_2} + \omega \,{}^{2}_{2} \times (\omega \,{}^{2}_{2} \times P \,{}^{2}_{C_2}) + \dot{v} \,{}^{2}_{2}$$

$$\dot{v}^2_{C_2} = \begin{bmatrix} L_1 \ddot{\theta}_1 \operatorname{sen}\theta_2 - L_1 \dot{\theta}_1^2 \cos\theta_2 + g \operatorname{sen}(\theta_1+\theta_2) - L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ L_1 \ddot{\theta}_1 \cos\theta_2 + L_1 \dot{\theta}_1^2 \operatorname{sen}\theta_2 + g \cos(\theta_1+\theta_2) + L_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ 0 \end{bmatrix}$$

$$F^2_2 = m_2 \dot{v}^2_{C_2}$$

$$F^2_2 = \begin{bmatrix} m_2 L_1 \ddot{\theta}_1 \operatorname{sen}\theta_2 - m_2 L_1 \dot{\theta}_1^2 \cos\theta_2 \\ + m_2 g \operatorname{sen}(\theta_1+\theta_2) - m_2 L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ m_2 L_1 \ddot{\theta}_1 \cos\theta_2 + m_2 L_1 \dot{\theta}_1^2 \operatorname{sen}\theta_2 \\ + m_2 g \cos(\theta_1+\theta_2) + m_2 L_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ 0 \end{bmatrix}$$

$$N^2_2 = I^C_2 {}_2 \, \dot{\omega}^2_2 + \omega^2_2 \times I^C_2 {}_2 \, \omega^2_2$$

$$N^2_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Iteraciones hacia adelante en el eslabón 3:

$$\omega^3_3 = R^3_2 \, \omega^2_2 + \dot{\theta}_3 \, \bar{z}^3_3$$

$$\omega \,{}^{3}_{3} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix}$$

$$\dot{\omega} \,{}^{3}_{3} = R \,{}^{3}_{2}\; \dot{\omega} \,{}^{2}_{2} + R \,{}^{3}_{2}\; \omega \,{}^{2}_{2} \times \dot{\theta}_3 \;\; \bar{z} \,{}^{3}_{3} + \ddot{\theta}_3 \;\; \bar{z} \,{}^{3}_{3}$$

$$\dot{\omega} \,{}^{3}_{3} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3 \end{bmatrix}$$

$$\dot{v} \,{}^{3}_{3} = R \,{}^{3}_{2} \left(\dot{\omega} \,{}^{2}_{2} \times P \,{}^{2}_{3} + \omega \,{}^{2}_{2} \times \left(\omega \,{}^{2}_{2} \times P \,{}^{2}_{3} \right) + \dot{v} \,{}^{2}_{2} \right)$$

$$\dot{v} \,{}^{3}_{3} = \begin{bmatrix} L_1 \ddot{\theta}_1 \operatorname{sen}(\theta_2+\theta_3) - L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \cos\theta_3 \\ +L_2(\ddot{\theta}_1+\ddot{\theta}_2)\operatorname{sen}\theta_3 - L_1\dot{\theta}_1^2\cos(\theta_2+\theta_3)+g\operatorname{sen}(\theta_1+\theta_2+\theta_3) \\ L_1\ddot{\theta}_1\cos(\theta_2+\theta_3)+L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ +L_2(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3 + L_1\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3)+g\cos(\theta_1+\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

$$\dot{v} \,{}^{3}_{C_3} = \dot{\omega} \,{}^{3}_{3} \times P \,{}^{3}_{C_3} + \omega \,{}^{3}_{3} \times \left(\omega \,{}^{3}_{3} \times P \,{}^{3}_{C_3} \right) + \dot{v} \,{}^{3}_{3}$$

$$\dot{v}_{C_3}^{3} = \begin{bmatrix} L_1 \ddot{\theta}_1 \operatorname{sen}(\theta_2+\theta_3) - L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 \\ - L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \cos\theta_3 - L_1\dot{\theta}_1^2 \cos(\theta_2+\theta_3) \\ + g \operatorname{sen}(\theta_1+\theta_2+\theta_3) + L_2(\ddot{\theta}_1+\ddot{\theta}_2)\operatorname{sen}\theta_3 \\ L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) + L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ + L_1\ddot{\theta}_1 \cos(\theta_2+\theta_3) + L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \operatorname{sen}\theta_3 \\ + L_1\dot{\theta}_1^2 \operatorname{sen}(\theta_2+\theta_3) + g\cos(\theta_1+\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

$$F_3^3 = m_3 \dot{v}_{C_3}^{3}$$

$$F_3^3 = \begin{bmatrix} m_3 L_2(\ddot{\theta}_1+\ddot{\theta}_2)\operatorname{sen}\theta_3 + m_3 L_1\ddot{\theta}_1 \operatorname{sen}(\theta_2+\theta_3) \\ - m_3 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 - m_3 L_2(\dot{\theta}_1+\dot{\theta}_2)^2\cos\theta_3 \\ - m_3 L_1\dot{\theta}_1^2\cos(\theta_2+\theta_3) + m_3 g \operatorname{sen}(\theta_1+\theta_2+\theta_3) \\ m_3 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) + m_3 L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ + m_3 L_1\ddot{\theta}_1\cos(\theta_2+\theta_3) + m_3 L_2(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3 \\ + m_3 L_1\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3) + m_3 g\cos(\theta_1+\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

$$N_3^3 = I_{C_3}^{3}\,\dot{\omega}_3^3 + \omega_3^3 \times I_{C_3}^{3}\,\omega_3^3$$

$$N\ {}^{3}_{3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

A continuación se incluyen los resultados de las iteraciones hacia atrás.

Iteraciones hacia atrás en el eslabón 3:

$$f\ {}^{3}_{3} = R\ {}^{3}_{4}\ f\ {}^{4}_{4} + F\ {}^{3}_{3}$$

$$f\ {}^{3}_{3} = F\ {}^{3}_{3} = \begin{bmatrix} m_3 L_2(\ddot{\theta}_1+\ddot{\theta}_2)\operatorname{sen}\theta_3 + m_3 L_1 \ddot{\theta}_1 \operatorname{sen}(\theta_2+\theta_3) \\ -m_3 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2 - m_3 L_2(\dot{\theta}_1+\dot{\theta}_2)^2\cos\theta_3 \\ -m_3 L_1 \dot{\theta}_1^2 \cos(\theta_2+\theta_3) + m_3 g \operatorname{sen}(\theta_1+\theta_2+\theta_3) \\ m_3 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3) + m_3 L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ +m_3 L_1 \ddot{\theta}_1 \cos(\theta_2+\theta_3) + m_3 L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \operatorname{sen}\theta_3 \\ +m_3 L_1 \dot{\theta}_1^2 \operatorname{sen}(\theta_2+\theta_3) + m_3 g \cos(\theta_1+\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

$$n\ {}^{3}_{3} = N\ {}^{3}_{3} + R\ {}^{3}_{4}\ n\ {}^{4}_{4} + P\ {}^{3}_{C_3} \times F\ {}^{3}_{3} + P\ {}^{3}_{4} \times R\ {}^{3}_{4}\ f\ {}^{4}_{4}$$

$$n\,{}^{3}_{3} = \begin{bmatrix} 0 \\ 0 \\ m_3 L_3^2(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ +m_3 L_1 L_3\ddot{\theta}_1\cos(\theta_2+\theta_3)+m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3 \\ +m_3 L_1 L_3\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3)+m_3 L_3 g\cos(\theta_1+\theta_2+\theta_3) \end{bmatrix}$$

Iteraciones hacia atrás en el eslabón 2:

$$f\,{}^{2}_{2} = R\,{}^{2}_{3}\,f\,{}^{3}_{3} + F\,{}^{2}_{2}$$

$$f\,{}^{2}_{2} = \begin{bmatrix} -m_3 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\operatorname{sen}\theta_3+(m_2+m_3)L_1\ddot{\theta}_1\operatorname{sen}\theta_2 \\ -m_3 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\cos\theta_3-(m_2+m_3)L_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ -(m_2+m_3)L_1\dot{\theta}_1^2\cos\theta_2+(m_2+m_3)g\operatorname{sen}(\theta_1+\theta_2) \\ m_3 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos\theta_3+(m_2+m_3)L_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ +(m_2+m_3)L_1\ddot{\theta}_1\cos\theta_2-m_3 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}\theta_3 \\ +(m_2+m_3)L_1\dot{\theta}_1^2\operatorname{sen}\theta_2+(m_2+m_3)g\cos(\theta_1+\theta_2) \\ 0 \end{bmatrix}$$

$$n\,{}^{2}_{2} = N\,{}^{2}_{2} + R\,{}^{2}_{3}\,n\,{}^{3}_{3} + P\,{}^{2}_{C_2} \times F\,{}^{2}_{2} + P\,{}^{2}_{3} \times R\,{}^{2}_{3}\,f\,{}^{3}_{3}$$

$$n_2^2 = \begin{bmatrix} 0 \\ 0 \\ m_3 L_3^2(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos\theta_3 \\ +(m_2+m_3)L_2^2(\ddot{\theta}_1+\ddot{\theta}_2)+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ +m_3 L_1 L_3\ddot{\theta}_1\cos(\theta_2+\theta_3)+(m_2+m_3)L_1 L_2\ddot{\theta}_1\cos\theta_2 \\ -m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}\theta_3+m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3 \\ +m_3 L_1 L_3\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3)+(m_2+m_3)L_1 L_2\dot{\theta}_1^2\operatorname{sen}\theta_2 \\ +m_3 L_3 g\cos(\theta_1+\theta_2+\theta_3)+(m_2+m_3)L_2 g\cos(\theta_1+\theta_2) \end{bmatrix}$$

Iteraciones hacia atrás en el eslabón 1:

$$f_1^1 = R_2^1 f_2^2 + F_1^1$$

$$f_1^1 = \begin{bmatrix} -m_3 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\operatorname{sen}(\theta_2+\theta_3)-(m_2+m_3)L_2(\ddot{\theta}_1+\ddot{\theta}_2)\operatorname{sen}\theta_2 \\ -m_3 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\cos(\theta_2+\theta_3)-(m_2+m_3)L_2(\dot{\theta}_1+\dot{\theta}_2)^2\cos\theta_2 \\ -(m_1+m_2+m_3)L_1\dot{\theta}_1^2+(m_1+m_2+m_3)g\operatorname{sen}\theta_1 \\ -m_3 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos(\theta_2+\theta_3)+(m_2+m_3)L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_2 \\ +(m_1+m_2+m_3)L_1\ddot{\theta}_1-m_3 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}(\theta_2+\theta_3) \\ -(m_2+m_3)L_2(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_2+(m_1+m_2+m_3)g\cos\theta_1 \\ 0 \end{bmatrix}$$

$$n_1^1 = N_1^1 + R_2^1 n_2^2 + P_{C_1}^1 \times F_1^1 + P_2^1 \times R_2^1 f_2^2$$

$$n_1^1 = \begin{bmatrix} 0 \\ 0 \\ m_3 L_3^2(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+m_3 L_1 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos(\theta_2+\theta_3) \\ +m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos\theta_3+(m_2+m_3)L_2^2(\ddot{\theta}_1+\ddot{\theta}_2) \\ +(m_2+m_3)L_1 L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_2+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3 \\ +(m_1+m_2+m_3)L_1^2\ddot{\theta}_1+m_3 L_1 L_3\ddot{\theta}_1\cos(\theta_2+\theta_3) \\ +(m_2+m_3)L_1 L_2\ddot{\theta}_1\cos\theta_2-m_3 L_1 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}(\theta_2+\theta_3) \\ -m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}\theta_3-(m_2+m_3)L_1 L_2(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_2 \\ +m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3+m_3 L_1 L_3\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3) \\ +(m_2+m_3)L_1 L_2\dot{\theta}_1^2\operatorname{sen}\theta_2+m_3 L_3 g\cos(\theta_1+\theta_2+\theta_3) \\ +(m_2+m_3)L_2 g\cos(\theta_1+\theta_2)+(m_1+m_2+m_3)L_1 g\cos\theta_1 \end{bmatrix}$$

Los torques hallados son los siguientes:

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

$$\tau_1 = (n_1^1)^T \bar{z}_1^1$$

$$\tau_1 = m_3 L_3^2(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+m_3 L_1 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos(\theta_2+\theta_3)$$
$$+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos\theta_3+(m_2+m_3)L_2^2(\ddot{\theta}_1+\ddot{\theta}_2)$$
$$+(m_2+m_3)L_1 L_2(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_2+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3$$
$$+(m_1+m_2+m_3)L_1^2\ddot{\theta}_1+m_3 L_1 L_3\ddot{\theta}_1\cos(\theta_2+\theta_3)$$
$$+(m_2+m_3)L_1 L_2\ddot{\theta}_1\cos\theta_2-m_3 L_1 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}(\theta_2+\theta_3)$$
$$-m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}\theta_3-(m_2+m_3)L_1 L_2(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_2$$
$$+m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3+m_3 L_1 L_3\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3)$$
$$+(m_2+m_3)L_1 L_2\dot{\theta}_1^2\operatorname{sen}\theta_2+m_3 L_3 g\cos(\theta_1+\theta_2+\theta_3)$$
$$+(m_2+m_3)L_2 g\cos(\theta_1+\theta_2)+(m_1+m_2+m_3)L_1 g\cos\theta_1$$

$$\tau_2 = (n\,\tfrac{2}{2})^T\,\bar{z}\,\tfrac{2}{2}$$

$$\tau_2 = m_3 L_3^2(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)\cos\theta_3$$
$$+(m_2+m_3)L_2^2(\ddot{\theta}_1+\ddot{\theta}_2)+m_3 L_2 L_3(\ddot{\theta}_1+\ddot{\theta}_2)\cos\theta_3$$
$$+m_3 L_1 L_3\ddot{\theta}_1\cos(\theta_2+\theta_3)+(m_2+m_3)L_1 L_2\ddot{\theta}_1\cos\theta_2$$
$$-m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2+\dot{\theta}_3)^2\operatorname{sen}\theta_3+m_3 L_2 L_3(\dot{\theta}_1+\dot{\theta}_2)^2\operatorname{sen}\theta_3$$
$$+m_3 L_1 L_3\dot{\theta}_1^2\operatorname{sen}(\theta_2+\theta_3)+(m_2+m_3)L_1 L_2\dot{\theta}_1^2\operatorname{sen}\theta_2$$
$$+m_3 L_3 g\cos(\theta_1+\theta_2+\theta_3)+(m_2+m_3)L_2 g\cos(\theta_1+\theta_2)$$

$$\tau_3 = (n\,\tfrac{3}{3})^T\,\bar{z}\,\tfrac{3}{3}$$

$$\tau_3 = m_3 L_3^2 (\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3) + m_3 L_2 L_3 (\ddot{\theta}_1 + \ddot{\theta}_2) \cos\theta_3$$
$$+ m_3 L_1 L_3 \ddot{\theta}_1 \cos(\theta_2 + \theta_3) + m_3 L_2 L_3 (\dot{\theta}_1 + \dot{\theta}_2)^2 \operatorname{sen}\theta_3$$
$$+ m_3 L_1 L_3 \dot{\theta}_1^2 \operatorname{sen}(\theta_2 + \theta_3) + m_3 L_3 g \cos(\theta_1 + \theta_2 + \theta_3)$$

2.4 Sistema Rhino

El sistema Rhino contiene seis motores los cuales constituyen los actuadores del Brazo de Robot y funcionan de acuerdo a los voltajes indicados en la siguiente figura:

Los signos aritméticos corresponden al tipo de voltaje aplicado necesario para activar el giro del motor en el sentido indicado. El círculo con punto corresponde al giro en el sentido hacia afuera del plano de la hoja, mientras que el círculo con una equis dentro, corresponde al giro en el sentido hacia adentro del plano de la hoja.

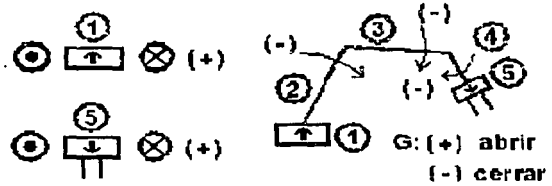


Figura 2.13. Motores del Brazo de Robot.

La tenaza del Brazo de Robot se abre aplicando un voltaje positivo, y se cierra al aplicar un voltaje negativo. El rango permisible de voltaje aplicable es de ± 20 Volts en cada motor, cada uno de los cuales contiene un conector formado por diez terminales con la siguiente disposición.

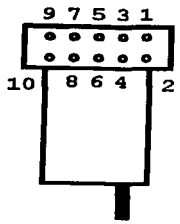


Figura 2.14. Terminales del motor.

1. Tierra lógica
2. Encodificador A

Los diez terminales del motor corresponden con las siguientes conexiones:

3. 5 Volts
4. Encodificador B
5. N. C.
6. Switch. No utilizado
7, 8. Tierra del motor
9, 10. Voltaje ± del motor.

3 *Interfaces para controlar el Brazo de Robot*

3.1 Sistema Internet

Actualmente el sistema Internet utiliza ampliamente al sistema WWW como un lenguaje de comunicación.

Tim Barners creó las bases del sistema WWW cuando trabajaba en el CERN, Centro Europeo de Investigaciones Nucleares con sede en Ginebra.

El sistema WWW, La Gran Red Mundial (World Wide Web), se basa en tres conceptos:

1. HTTP, Protocolo de Transferencia de Hipertexto. El cliente lo utiliza para solicitar las páginas y elementos de WWW, y el servidor utiliza este protocolo para proporcionar esas páginas y elementos.
2. HTML, Lenguaje de Marcas de Hipertexto. Utilizado para describir y dar formato a las páginas WWW.
3. URLs, Ubicadores de recursos. Utilizado para dar nombre y ruta de acceso a las páginas y elementos de WWW, y a las referencias dentro de las páginas HTML.

El modo de accionar del protocolo HTTP es el siguiente:

Establece una conexión TCP segura entre cliente y servidor utilizando el puerto 80 en el lado del servidor, aunque se puede utilizar otro número de puerto si el servidor lo permite (no es obligatoria la utilización del protocolo TCP si bien es el comportamiento por omisión de HTTP). El cliente envía su solicitud al servidor pidiéndole alguna página ó documento. Entonces el servidor recibe la solicitud, la analiza y en el mejor de los casos retorna al cliente el documento solicitado ó en caso de error regresa un código de error, y después el servidor cierra la conexión TCP.

Un mensaje HTTP consta de un encabezado, una línea en blanco, y una entidad cuerpo. Las páginas y/o documentos solicitados por el cliente se incluyen en la entidad cuerpo del mensaje de respuesta del servidor.

El URI_de_solicitud identifica la página ó documento solicitado por el cliente.

A continuación se incluyen los conceptos básicos del protocolo HTTP, utilizando como referencia la RFC 1945.

HTTP es un protocolo del nivel de aplicación.

En la práctica la conexión es establecida por el cliente antes de cada solicitud y el servidor la cierra después de enviar la respuesta. Los clientes y servidores deberían hacer lo necesario para prever el caso en el cual una de las partes cierra la conexión prematuramente debido p. ej. a una acción del usuario, ó algún temporizador, ó falla de programa, y deberían manejar la situación de forma predecible. En caso de cierre por cualquiera de las partes, siempre termina la solicitud actual, sin importar su estado.

Si el cliente envía una solicitud-completa HTTP/1.0 y recibe una respuesta la cual no comienza con una línea de estado, debería suponer que la respuesta es una respuesta-simple y analizarla. La respuesta-simple consiste solo de la entidad cuerpo y el servidor la termina cerrando la conexión.

Los métodos más comunes utilizados por el protocolo HTTP son tres:

1. Método OBTENER (GET).
2. Método ENCABEZADO (HEAD).
3. Método ENVIO (POST).

El método OBTENER (GET) significa recuperar cualesquier información (en la forma de entidad) identificada por el URI_de_solicitud. Si el URI_de_solicitud se refiere a un proceso de producción de datos, son los datos producidos lo que se regresará como entidad, en la respuesta, y no el texto fuente del proceso, a menos que el texto sea la salida del proceso.

La semántica del método OBTENER (GET) cambia a OBTENER_condicional (conditional GET) si el mensaje de solicitud incluye un encabezado Si_se_modificó_después_de (If_Modified_Since). Un método OBTENER_condicional solicita que el recurso identificado sea transferido sólo si el recurso ha sido modificado después de la fecha indicada en el encabezado Si_se_modificó_después_de.

El método OBTENER_condicional está diseñado para reducir el uso de la red permitiendo que las entidades en el escondite (cache) sean actualizadas sin exceso de solicitudes ó transferencias de datos innecesarias.

El método ENCABEZADO (HEAD) es idéntico al método OBTENER (GET) excepto que el servidor no debe regresar una entidad-cuerpo en la respuesta. La metainformación contenida en los encabezados HTTP en respuesta a una solicitud ENCABEZADO debería ser idéntica a la información enviada en respuesta a una solicitud OBTENER. Este método se puede utilizar para obtener metainformación acerca del recurso identificado en el URI_de_solicitud sin transferir la entidad-cuerpo. Este método a menudo se utiliza para pruebas de validez, accesibilidad, y modificaciones recientes en ligas de hipertexto.

No existe ninguna solicitud ENCABEZADO_condicional análoga al OBTENER_condicional. Si se incluye un campo de encabezado Si_se_modificó_después_de (If_Modified_Since) en una solicitud ENCABEZADO, debería ser ignorado.

EL método ENVIO (POST) se utiliza para solicitar que el servidor destino acepte el componente incluido en la solicitud, como un nuevo subordinado del recurso identificado por el URI_de_solicitud en la línea de solicitud. El método ENVIO (POST) se destina para permitir un método uniforme en el tratamiento de las siguientes funciones:

1. Anotación de recursos existentes;
2. Poner un mensaje en una pizarra de boletines, grupo de noticias, lista de correos, ó grupo de artículos similares;

3. Proveer un bloque de datos, tal como el resultado del llenado de una forma, a un proceso de manejo de datos;
4. Ampliar una base de datos a través de una operación de "agregar" a la base de datos.

La función real del método ENVIO es determinada por el servidor y por lo regular depende del URI_de_solicitud. La entidad enviada está subordinada al URI en la misma forma que un archivo está subordinado al subdirectorio en el que se encuentra contenido, ó como un artículo de noticias está subordinado al grupo de noticias al cual pertenece, ó como un registro subordinado a una base de datos.

Un ENVIO exitoso no necesita que la entidad sea creada como un recurso en el servidor origen ó que sea accesible para futuras referencias. Esto es, la acción realizada por el método ENVIO pudiera resultar en un recurso no identificable por un URI. En este caso, la respuesta de estado adecuada sería ó 200 (bien) ó 204 (ningún contenido), dependiendo de sí ó no la respuesta incluye una entidad que describa el resultado.

Si un recurso ha sido creado en el servidor origen, la respuesta debería ser 201 (creado) y contener una entidad (preferiblemente de tipo "texto/html" {"text/html"}) la cual describa el estado de la solicitud y se refiera al nuevo recurso.

Una longitud-de-contenido válida es necesaria en todas las solicitudes ENVIO HTTP/1.0. Un servidor HTTP/1.0 debería responder con un mensaje 400 (solicitud errónea) si no puede determinar la longitud del contenido del mensaje en la solicitud.

Las aplicaciones no deberían guardar en el escondite las respuestas a una solicitud ENVIO pues la aplicación no tiene manera de saber que el servidor regresará una respuesta equivalente en alguna solicitud en el futuro.

3.2 Características de Protocolos

Se tratan dos protocolos: IP y TCP. La presentación de estos protocolos está basada en las RFC 791 y 793 respectivamente.

3.2.1 Protocolo IP

El Protocolo IP se diseñó para sistemas interconectados de redes de computadoras de conmutación de paquetes utilizando bloques de datos denominados datagramas. Los anfitriones (hosts) se identifican por direcciones de longitud fija y este protocolo IP aporta la fragmentación y el reensamble necesarios para datagramas extensos.

El protocolo IP se limita a la entrega de paquetes de bits (datagrama interred {internet}) de fuente a destino sin aportar:

1. Confiabilidad de transmisión en los datos.
2. Control de flujo.
3. Secuenciamiento, etc.

IP es utilizado por los protocolos anfitrión_a_anfitrión en ambientes interred (internet); IP es llamado por los protocolos de red para enviar el datagrama interred a la siguiente puerta (gateway) ó anfitrión_destino p. ej., el módulo TCP llama al módulo interred para que éste acepte un segmento TCP (encabezado TCP y datos del usuario) y lo incluya como datos en un datagrama interred. El módulo TCP proporciona al módulo interred las direcciones IP y otros parámetros del encabezado interred, en forma de argumentos de la llamada. Luego el módulo interred crea un datagrama interred y llama a la interfaz de red local para transmitir el datagrama interred.

Modelo de Operación.

El protocolo IP implementa dos funciones básicas:

1. Direccionamiento.
2. Fragmentación.

Los módulos interred utilizan las direcciones del encabezado interred para transmitir los datagramas interred hacia el destino. La selección de una trayectoria para una transmisión se denomina enrutaje.

Los módulos interred utilizan campos del encabezado interred para fragmentar y reensamblar datagramas interred, cuando es necesario, para la transmisión de los datagramas.

Un módulo interred reside en cada anfitrión ocupado en la comunicación interred y en cada puerta (gateway) que interconecta las redes. Estos módulos incluyen reglas para interpretar campos de dirección y para fragmentar y ensamblar datagramas interred. En especial las puertas (gateways) incluyen procedimientos de enrutaje.

El protocolo interred trata a cada datagrama interred como unidad independiente sin relación con otros datagramas interred. No existen conexiones, circuitos lógicos, virtuales ó cualesquiera otra clase de conexión.

El protocolo IP utiliza cuatro mecanismos de base para proporcionar su servicio; son los siguientes:

1. Tipo de Servicio. La indicación de Tipo de Servicio es utilizada por las puertas (gateways) para elegir: a) Los parámetros de transmisión reales en una red específica, b) La red utilizada para llegar al siguiente nodo, c) La siguiente puerta (gateway) cuando se enruta un datagrama interred.
2. Tiempo de vida. Indica un límite superior, es establecido por el emisor del datagrama y se reduce en los nodos durante el trayecto, en los cuales se

procesa de la siguiente forma: si el tiempo de vida llega a cero antes de que el datagrama interred llegue a su destino, el datagrama interred es destruido por la entidad donde este límite superior se establezca en cero.

3. Opciones. Las opciones incluyen hora, fecha, seguridad y enrutaje especial.
4. Suma de Verificación de encabezado. Con esta suma se analiza que el encabezado del datagrama no contenga errores, no se incluye en esta suma al área de datos del datagrama. Si la suma de verificación indica algún error, el datagrama interred es eliminado por la entidad donde se detecte el error.

El protocolo IP no representa una forma de comunicación confiable. No hay reconocimientos extremo_a_extremo ó de nodo_a_nodo. No proporciona control de errores para datos, aunque sí para el encabezado. Tampoco existen retransmisiones ó control de flujo. Los errores se reportan a través del protocolo ICMP (Protocolo de Mensajes de Control de Interred) el cual está implementado en el módulo de protocolo interred.

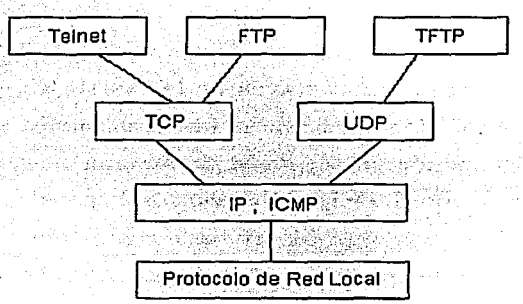


Figura 3.1. Interrelación de Protocolos.

La transmisión de un datagrama de un programa de aplicación a otro se describe a continuación suponiendo que existen puertas (gateways) intermedias y que se trata de una conexión TCP.

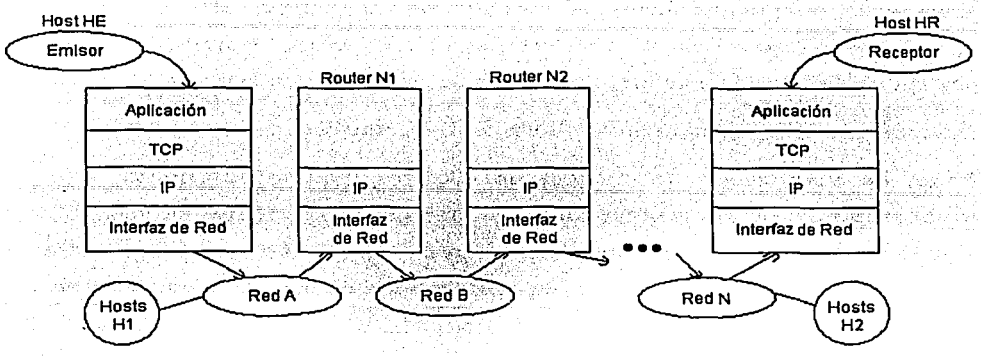


Figura 3.2. Transmisión de un datagrama IP.

El datagrama IP del anfitrión emisor HE pasa a la interfaz de red local de HE y viaja por la red A llegando a la interfaz de red local A del enrutador (router) N1 la cual pasa el datagrama a la región interred de N1, la cual analiza la dirección IP y lo envía a la interfaz de red local B de N1 la cual lo envía por la red B al siguiente nodo. El proceso se repite hasta que un enrutador detecta que la dirección IP está en su ámbito de conexiones directas, entonces envía el datagrama al anfitrión receptor HR a través de la red N. En el anfitrión HR el datagrama pasa de la región de interfaz de red local del anfitrión HR a la región interred del anfitrión HR donde se acepta el datagrama porque la dirección IP destino en el datagrama coincide con la dirección IP del anfitrión HR. La región interred pasa el segmento, el cual está incorporado como área de datos (load) en el datagrama IP, a la región de Transporte donde este segmento es recibido por el protocolo TCP.

El Datagrama IP.

La unidad de transferencia básica del protocolo IP es el datagrama IP el cual está formado de un encabezado y una área de datos.

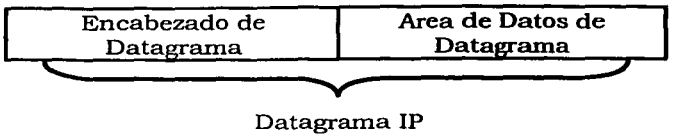


Figura 3.3. Datagrama IP.

El encabezado del datagrama contiene las direcciones IP fuente y destino a diferencia de un encabezado de un marco (frame) el cual contiene direcciones físicas.

									1										2										3		
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Versión				IHL				Tipo de Servicio								Longitud Total															
Identificación																Flags*				Distancia de fragmento											
Tiempo de vida								Protocolo								Suma de verificación de encabezado															
Dirección Fuente																															
Dirección Destino																															
Opciones																												Relleno			

Figura 3.4. Encabezado del datagrama IP.

* Banderas (Flags)

Formato de Encabezado del Datagrama IP.

-> Versión. Cuatro bits. La versión IP actual es 4.

-> Longitud de Encabezado Interred, IHL. Cuatro bits. Indica la longitud del encabezado del datagrama medido en palabras de 32 bits. Todos los campos en el encabezado IP tienen longitud fija excepto los campos Opciones y Relleno. El tamaño mínimo del encabezado IP es 5.

-> Tipo de Servicio. Ocho bits. Está formado por los campos Precedencia, D, T, y R.

0	3	4	5	6	7
Precedencia	D	T	R	0	0

Figura 3.5. Tipo de Servicio.

Precedencia. 3 bits. Este campo indica una prioridad en el tratamiento del datagrama durante su transmisión a través del sistema interred. Se utiliza efectuando mapeos Precedencia_Servicios con los servicios reales que proporcionan las redes.

Bit D. 1 bit. 0= Retraso Normal, 1= Retraso Mínimo.

Bit T. 1 bit. 0= Rendimiento Normal, 1= Alto Rendimiento.

Bit R. 1 bit. 0= Confiabilidad Normal, 1= Confiabilidad Alta.

Los bits 6 y 7 no se utilizan.

-> Longitud Total. 16 bits. Este campo muestra la longitud total del datagrama IP medida en octetos (8 bits), incluidos el encabezado y los datos.

Tamaño del área de datos= Longitud Total - Longitud del Encabezado.

El tamaño máximo de un datagrama IP es 2^16 – 1 = 65535 octetos.

-> Identificación. 16 bits. Este campo contiene un número entero único para identificar al datagrama. En caso de fragmentación este campo se copia en los fragmentos.

-> Banderas. 3 bits. El bit cero debe ser cero.

0	1	2
0	DF	MF

Figura 3.6. Banderas del encabezado IP.

Bit 1. DF. 0= El datagrama se puede fragmentar; 1= No se permite la fragmentación.

Bit 2. MF. 0= Ultimo fragmento; 1= Hay más fragmentos.

-> Distancia de Fragmento. 13 bits. Indica la posición de los datos de este fragmento, en el datagrama original, medido en unidades de ocho octetos (64 bits). El primer fragmento tiene una distancia igual a cero.

-> Tiempo de Vida. 8 bits. Indica un límite (tiempo en segundos) máximo de permanencia de un datagrama IP en una interred. Enrutadores y anfitriones que procesan datagramas deben reducir el Tiempo de Vida el cual inicialmente es establecido por el emisor original del datagrama.

Cada enrutador por donde pasa el datagrama reduce en una unidad el tiempo de vida y cuando el enrutador está sobrecargado resta además el número de segundos que permaneció el datagrama en el enrutador. Cuando el tiempo de vida llega a cero en un enrutador, el enrutador elimina el datagrama y envía un mensaje de error al emisor del datagrama. Esta técnica garantiza que un datagrama no permanezca por siempre en una interred.

-> Protocolo. 8 bits. Este campo se puede interpretar como una especificación del formato del área de datos del datagrama, de acuerdo a estándares internacionales.

-> Suma de Verificación de Encabezado. 16 bits. Asegura la integridad del encabezado IP. Se calcula tomando al encabezado IP como una secuencia de enteros de 16 bits sumándolos y utilizando aritmética de complemento a uno, y después tomando el complemento a uno del resultado, interpretando durante este proceso al campo Suma de Verificación de Encabezado como igual a cero.

-> Dirección Fuente IP. 32 bits. Contiene la dirección IP del emisor.

-> Dirección Destino IP. 32 bits. Contiene la dirección IP del receptor.

-> Relleno. Tamaño variable. Contiene ceros y se utiliza sólo cuando es necesario de forma que el encabezado del datagrama sea de un tamaño múltiplo de 32 bits.

-> Opciones. Tamaño variable. Las opciones deben estar implementadas en enrutadores y anfitriones. Su transmisión es opcional en un datagrama pero su implementación es obligatoria. Puede haber cero ó más opciones. Según su tamaño las opciones pueden ser:

1. Opciones de un octeto. Octeto de Tipo de Opción.
2. Opciones de más de un octeto. Octeto de Tipo de Opción, Octeto de Longitud de Opción, Octetos de datos de Opción.

El octeto de tipo de opción tiene la siguiente estructura:

1 bit	Bandera de copia.
2 bits	Clase de opción.
5 bits	Número de opción.

Figura 3.7. Octeto de opción.

Bandera de Copia. 1 bit. Esta bandera indica si esta opción se copia en todos los fragmentos cuando ocurre una fragmentación.

0= No copiar; 1= Copiar en los fragmentos.

Clase de Opción. 2 bits.

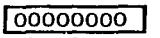
Bits		Significado
0	0	Control
0	1	Reservado
1	0	Depuración y Medición
1	1	Reservado

Figura 3.8. Clase de opción en el encabezado IP.

CLASE	NUMERO	LONGITUD	DESCRIPCION
0	0	-	Fin de lista de opciones. Esta opción ocupa un octeto, no tiene octeto de longitud.
0	1	-	Ninguna operación. Esta opción ocupa un octeto, no tiene octeto de longitud.
0	2	11	Seguridad. Utilizada con fines de Seguridad.
0	3	variable	Enrutaje Fuente Débil. Utilizada para enrutar el datagrama interred basado en información proporcionada por el usuario.
0	9	variable	Enrutaje Fuente Estricto. Utilizada para enrutar el datagrama interred basado en información proporcionada por el usuario.
0	7	variable	Registrar Ruta. Utilizada para rastrear el camino que toma un datagrama interred.
0	8	4	ID de Flujo. Utilizada para llevar el identificador del Flujo.
2	4	variable	Hora_fecha interred.

Tabla 3.1. Descripción de opciones del encabezado IP.

>>> Fin de Lista de Opciones.

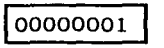


Tipo 0

Figura 3.9. Fin de lista de opciones.

Se utiliza al final de todas las opciones y no al final de cada opción. Utilizada sólo si el final de las opciones no coincide con el final del encabezado IP. Se puede copiar, introducir ó borrar debido a la fragmentación ó por cualquier otra razón.

>>> Ninguna Operación.

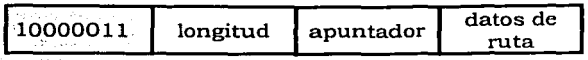


Tipo 1

Figura 3.10. Ninguna operación.

Esta opción se utiliza entre dos opciones adyacentes para alinear los octetos.

>>> Enrutaje Fuente Débil.



Tipo 131

Figura 3.11. Opción Enrutaje Fuente Débil.

*** Longitud. 8 bits. Incluye el tamaño de toda la opción desde el octeto Tipo de Opción hasta el final de los Datos de Ruta.

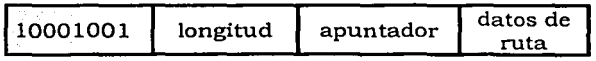
*** Apuntador. 8 bits. Indica la distancia relativa a esta opción, con valor inicial igual a cuatro, del octeto donde inicia la siguiente dirección IP que se procesará.

*** Datos de Ruta. variable. Está compuesta de una serie de direcciones IP de 4 octetos.

Si el datagrama está llegando al nodo indicado en el campo Dirección Destino y el Apuntador no es mayor que la Longitud entonces la dirección en Dirección Destino es remplazada por la siguiente dirección indicada en Datos de Ruta, el Apuntador se incrementa en cuatro y Dirección Fuente es remplazada por la dirección IP del nodo actual como se conoce en la red dentro de la cual el datagrama será enviado.

La longitud de esta opción es una constante durante el trayecto del datagrama por la interred.

>>> Enrutaje Fuente Estricto.



Tipo 137

Figura 3.12. Opción Enrutaje Fuente Estricto.

Longitud, Apuntador, Datos de Ruta y el tratamiento de esta opción es como en la opción Enrutaje Fuente Débil. La longitud de esta opción es una constante durante el trayecto del datagrama por la interred.

La denominación de Enrutaje Estricto se debe a que el nodo actual debe enviar directamente el datagrama hacia la siguiente dirección indicada en Datos de Ruta sin utilizar cualesquier otro nodo intermedio.

>>> Registrar Ruta.

Tipo 7

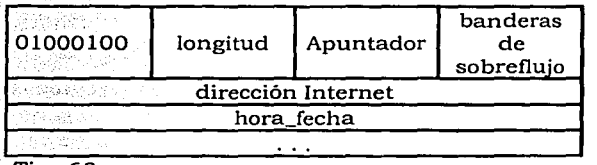
Figura 3.13. Opción Registrar Ruta.

*** Longitud es como en la opción Enrutaje Fuente Débil.

*** Apuntador. 8 bits. Indica la distancia, de valor inicial igual a cuatro, dentro de esta opción donde inicia la siguiente área para guardar una dirección de ruta. Una dirección de ruta ocupa cuatro octetos. El anfitrión original decide el tamaño de la opción. El contenido inicial de Datos de Ruta debe ser cero.

Cuando un datagrama llega a un módulo interred el nodo registra su dirección IP, como se le conoce en la red de salida por donde este datagrama será enviado, en el área de Datos de Ruta si todavía se dispone del espacio necesario para registrarla.

>>> Hora_Fecha interred.



Tipo 68

Figura 3.14. Opción Hora_Fecha interred.

Longitud y Apuntador son como en las opciones anteriores. El valor inicial de Apuntador es cinco. El campo banderas de sobreflujo está formado por los subcampos OFW (4 bits) y Banderas (4 bits).

Banderas de sobreflujo: [OFW | Banderas]

OFW, 4 bits. Contiene el número de nodos que no pudieron registrarse por falta de espacio en el área de datos.

Banderas	Significado
0	Registrar sólo Hora_Fecha omitiendo direcciones IP.
1	Registrar Dirección IP y Hora_Fecha.
3	Las direcciones IP son indicadas por el emisor original. Un nodo registra su Hora_Fecha si la siguiente dirección IP en la lista es igual a la dirección IP del nodo actual.

Figura 3.15. Banderas en la opción Hora_Fecha interred.

La Hora_Fecha se registra en milisegundos desde medianoche. El contenido inicial de Hora_Fecha debe ser cero ó pares:

Dirección IP	cero

Figura 3.16. Contenido inicial de Hora_Fecha. Este contenido es una alternativa.

En caso de sobreflujo en el campo OFW ó de no haber suficiente espacio para registrar la Hora_Fecha, se elimina el datagrama y se envía un mensaje de error, ICMP, al emisor original. La opción en caso de fragmentación sólo se utiliza en el primer fragmento, pues debe aparecer cuando más una vez en un datagrama.

Fragmentación.

Los marcos de red son reconocidos por el hardware, los datagramas son tratados por el software.

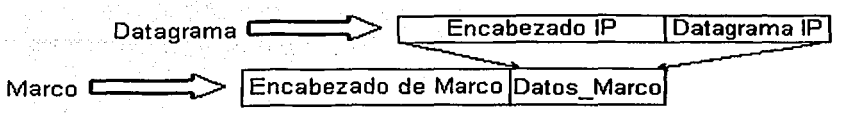


Figura 3.17. Marco y Datagrama.

Una característica de red es el MTU (Unidad de Transferencia Máxima) e indica el máximo tamaño de un datagrama que una red en particular puede transportar sin fragmentar al datagrama. Cuando el tamaño del datagrama es mayor que el MTU de una red en particular entonces el datagrama es fragmentado para poder transportarlo a través de esa red.

Encabezado de fragmento 1	Datos 1	Fragmento 1, Distancia 0
Encabezado de fragmento 2	Datos 2	Fragmento 2, Distancia 600
Encabezado de fragmento 3	Datos 3	Fragmento 3, Distancia 1200

Figura 3.18. Fragmentación de datagrama.

El campo Longitud Total del encabezado indica el tamaño del fragmento y no el tamaño del datagrama original. El campo Distancia de Fragmento especifica la distancia de los datos en el datagrama original. Con la ayuda de la bandera MF y de los campos Distancia de Fragmento y Longitud Total es posible deducir la longitud del datagrama original y también reensamblar el datagrama original.

Cuando el bit de bandera DF= 1, y un enrutador necesita fragmentar el datagrama, este datagrama se descarta y se notifica al emisor del datagrama.

Los enrutadores y anfitriones deben ser capaces de tratar adecuadamente datagramas de 576 octetos como mínimo.

3.2.2 Protocolo TCP

El Protocolo de Control de la Transmisión, TCP es un protocolo de propósito general y es posible utilizarlo sobre un protocolo diferente a IP.

TCP se diseñó para utilizarse como un protocolo altamente confiable anfitrión_a_anfitrión en redes de computadoras de conmutación de paquetes.

TCP es un protocolo confiable, orientado a la conexión, extremo_a_extremo diseñado para aplicaciones multired. TCP proporciona la comunicación entre pares de procesos en computadoras_anfitrión que se encuentran conectadas a más de una red. TCP supone que el servicio de datagramas es potencialmente no confiable en la región inmediatamente debajo de la región de Transporte. TCP puede utilizarse dentro de una amplia variedad de sistemas de comunicación variando desde conexiones establecidas por hardware hasta redes de conmutación de paquetes o redes de conmutación de circuitos.

Ubicación del Protocolo TCP:

1. Nivel_Superior
2. TCP
3. Protocolo interred
4. Red de comunicación

La interfaz entre TCP y el Nivel_Superior consiste de un conjunto de llamadas p. ej., abrir y cerrar conexiones, y enviar y recibir datos.

TCP utiliza operaciones básicas para proporcionar sus servicios:

1. Transferencia Básica de Datos.
2. Confiabilidad.
3. Control de Flujo.
4. Multiplexaje.
5. Conexiones.
6. Precedencia y Seguridad.

1. Transferencia Básica de Datos.

TCP puede transferir un flujo continuo de octetos en ambas direcciones formando segmentos de datos. En general TCP decide el momento de bloquear y transmitir los datos. TCP define una función "empujar" ("push") para permitir que el emisor obligue a TCP a transmitir y entregar los datos hasta el momento, al receptor. TCP no preserva los límites de estructuras de datos.

2. Confiabilidad.

TCP se puede recuperar de errores tales como datos perdidos, duplicados, dañados, ó entregados fuera de orden por el sistema interred. Para ello TCP asigna un número de secuencia a cada octeto transmitido y solicita un reconocimiento positivo (ACK) del TCP receptor. Si no se recibe el reconocimiento dentro de un intervalo de tiempo, los datos se retransmiten. Los números de secuencia son utilizados por el receptor para ordenar secuencialmente datos recibidos en desorden y para descartar duplicados. TCP incluye una suma de verificación para cada segmento transmitido, utilizándole el receptor para verificación del segmento y también para descartar segmentos dañados. TCP se puede recuperar, en caso de errores de transmisión en el sistema de comunicación interred.

3. Control de Flujo.

TCP posee un mecanismo de control para la cantidad de datos que el emisor puede enviar en determinado momento durante una conexión. Para ello TCP utiliza una "ventana" cuyo tamaño es regresado al emisor en cada reconocimiento, ACK, con la cual se indica un rango de números de secuencia aceptables, posteriores al último número de segmento recibido sin errores. La ventana indica la cantidad de octetos que el emisor puede transmitir al receptor antes de recibir el siguiente tamaño de ventana.

4. Multiplexaje.

TCP proporciona un conjunto de direcciones (números) de puertos dentro de cada anfitrión. Un conector (socket) es la concatenación de la dirección de un puerto con las direcciones de red y de anfitrión de la región interred. Una conexión se identifica por un par de conectores. Un mismo conector se puede utilizar en más de una conexión.

5. Conexiones.

TCP inicializa y mantiene información de estado para cada corriente de datos. Una conexión es la combinación de la información de estado, conectores, números de secuencia, y tamaños de ventana. Una conexión se especifica de manera única por un par de conectores los cuales identifican los extremos de la conexión.

Las conexiones deben establecerse aún entre anfitriones no confiables y sobre un sistema interred no confiable. TCP utiliza un mecanismo de saludo (handshake) con números de secuencia basado en reloj para evitar errores al inicializar conexiones.

6. Precedencia y Seguridad.

TCP permite a los usuarios especificar diferentes niveles de seguridad y precedencia, proporcionando valores por omisión cuando el usuario no especifica alguna opción en particular.

Filosofía.

Un proceso es un programa en ejecución. Toda comunicación se puede idealizar como una comunicación entre-procesos, incluyendo dispositivos I/O, terminales y archivos comunicándose entre sí a través de procesos. Cada proceso puede tener más de un puerto a través de el(los) cual(es) se comunica con los puertos de otros procesos.

Modelo de Operación.

En una transmisión de datos los procesos llaman a TCP y le pasan áreas (buffers) de datos como argumentos. TCP empaqueta los datos en segmentos TCP y luego llama al módulo interred para transmitir los datos al TCP receptor. El TCP receptor coloca los datos del segmento en el buffer del usuario receptor y notifica al usuario receptor. El mecanismo TCP asegura una transmisión de datos ordenada y confiable. En la transmisión del datagrama a través de una red local, el datagrama se introduce en un paquete de red local. Durante el trayecto en la transmisión puede haber puertas (gateways) intermedias. Las puertas (gateways) pueden fragmentar, empaquetar, ó realizar cualesquier otra operación necesaria con la finalidad de entregar el paquete local al módulo interred destino. Una puerta (gateway) interred "desenvuelve" el datagrama interred de su paquete local y lo examina para determinar la siguiente red por donde el datagrama debería enviarse. Entonces el datagrama interred se "reenvuelve" en un paquete local adecuado para esa siguiente red y se envía al siguiente enrutador, puerta, ó al anfitrión destino. Cualquier puerta ó enrutador puede fragmentar el datagrama interred si es necesario. La fragmentación puede ocurrir en más de una puerta ó enrutador. El módulo interred destino desenvuelve el segmento contenido en el datagrama, lo reensambla (si es necesario) y lo pasa al módulo TCP destino.

Considerando a TCP como un módulo del sistema operativo, es posible accesarlo como cualquier otro módulo a través de llamadas al sistema. A su vez TCP puede hacer llamadas a otros módulos del sistema operativo. Se puede suponer que la interfaz real a la red está controlada por un módulo controlador (driver) de dispositivo pero TCP no se comunica directamente con este controlador, más bien TCP llama al módulo interred el cual puede llamar al controlador de dispositivo.

La transmisión TCP es confiable. Utiliza para ello números de secuencia y reconocimientos. Cada octeto de datos tiene un número de secuencia. El número de secuencia del primer octeto de datos del segmento actual se indica en un campo del encabezado del segmento, este campo se denomina Número de

Secuencia del segmento. Otro campo en el encabezado indica el Número de Reconocimiento, el cual es el número de secuencia del siguiente octeto de datos esperado por este lado de la conexión; octeto el cual debería ser enviado por el otro extremo de la conexión.

En caso de que TCP transmita un segmento conteniendo datos, dispone una copia de los datos en una cola de retransmisión e inicia un temporizador; si llega un reconocimiento de los datos antes de que el temporizador llegue a cero, TCP borra los datos de la cola de retransmisión, no obstante si el reconocimiento no llega antes de que el temporizador llegue a cero, el segmento de datos se retransmite. El reconocimiento TCP no garantiza que los datos se entreguen al usuario final; más bien solo garantiza que el TCP receptor se responsabiliza de hacer lo posible para entregar los datos.

Con el fin de controlar el flujo de datos, el TCP receptor designa un tamaño de ventana al TCP emisor. La ventana señala el número de octetos, iniciando desde el Número de Reconocimiento, que el TCP receptor está preparado para recibir.

Las conexiones TCP son bidireccionales (full duplex). Los procesos pueden poseer varios puertos. Los procesos pueden iniciar conexiones solamente en los puertos que poseen. Una conexión se especifica al llamar a la rutina ABRIR (OPEN) con argumentos: puerto local y conector (socket) lejano, entonces TCP proporciona una identificación de conexión local que el usuario utiliza para referirse a la conexión posteriormente al hacer otras llamadas al módulo TCP diferentes a OPEN. Existen conexiones pasivas y activas. Una conexión pasiva significa que el proceso está dispuesto a aceptar solicitudes de conexión más que intentar el iniciar una conexión. Las conexiones pasivas están dispuestas a aceptar solicitudes de conexión de cualquiera que llame con los parámetros adecuados. En la inicialización de una conexión se utiliza la bandera de control SYN y un intercambio de tres mensajes como sincronía, denominado saludo de tres-vías. El bloque de Control de la Transmisión, TCB, es la estructura de datos donde se registra el estado de una conexión. La conexión se inicia cuando se recibe un segmento con las banderas SYN=1, ACK=0, y cuando existe un renglón en el bloque TCB creado por el usuario al llamar a la rutina ABRIR (OPEN). La

conexión pasa al estado "establecida" cuando los números de secuencia se han sincronizado en ambas direcciones. Para terminar una conexión se intercambian segmentos donde se utiliza la bandera de control FIN.

Segmento TCP

Un segmento TCP es la unidad de transferencia de datos entre dos módulos TCP.

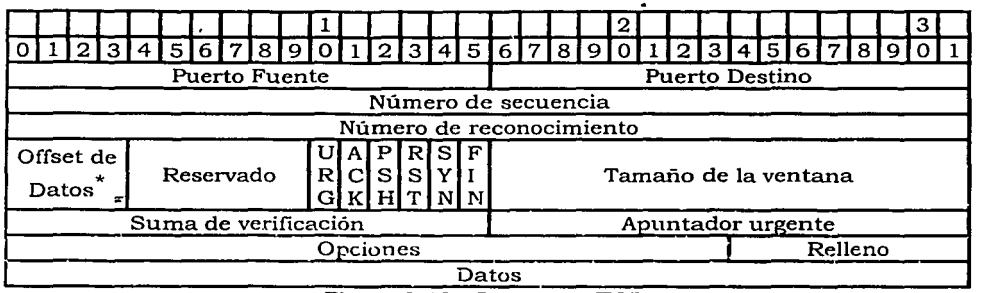


Figura 3.19. Segmento TCP.

* Distancia de Datos (Offset de datos)

Encabezado del Segmento TCP.

>> Puerto Fuente. 16 bits. Es el número del puerto fuente.

>> Puerto Destino. 16 bits. Número del puerto destino.

>> Número de Secuencia. 32 bits. Es el número de secuencia del primer octeto de datos en este segmento excepto si la bandera SYN está activada, pues si esta bandera se encuentra activada, el Número de Secuencia indica el Número de Secuencia Inicial (ISN) y entonces el primer octeto de datos tiene un número de secuencia igual a ISN+1.

>> Número de Reconocimiento. 32 bits. Cuando el bit ACK se encuentra activo este campo contiene el valor del siguiente número de secuencia que el emisor de este segmento espera recibir. Después de establecida una conexión este campo siempre se utiliza.

>> Distancia de Datos. 4 bits. Muestra el número de palabras de 32 bits contenidas en el encabezado TCP y también indica la distancia desde donde inician los datos. El encabezado ocupa un espacio de algún tamaño múltiplo de 32 bits, incluyendo las opciones.

>> Reservado. 6 bits. Este campo debe contener ceros.

>> Bits de Control. 6 bits.

Bit	Descripción
URG	El campo Apuntador urgente es significativo.
ACK	El campo Número de reconocimiento es significativo.
PSH	Función Empujar (Push).
RST	Reajustar la conexión.
SYN	Sincronizar los números de secuencia.
FIN	No hay más datos por parte del emisor.

Figura 3.20. Bits de control del encabezado TCP.

>> Tamaño de la Ventana. 16 bits. Contiene el número de octetos de datos, iniciando con el indicado en el campo Número de Reconocimiento, los cuales el emisor de este segmento se encuentra dispuesto a aceptar.

>> Suma de Verificación. 16 bits. En el cálculo de la suma de verificación se utiliza un seudo_encabezado con el siguiente formato:

Dirección Fuente IP		
Dirección Destino IP		
00000000	Protocolo 6	Longitud de segmento TCP

Figura 3.21. Seudo_encabezado utilizado en el cálculo de la suma de verificación.

El Campo Suma de Verificación se toma como igual a cero mientras se calcula la suma de verificación. Esta suma es el complemento a uno con 16 bits, de la suma complemento a uno de todas las palabras de 16 bits en el encabezado y datos del segmento. Si no es par el número de octetos de encabezado y datos, se rellena con ceros a la derecha para formar la última palabra de 16 bits y poder efectuar el cálculo de la suma de verificación.

El seudo_encabezado y el relleno no se transmiten.

El seudo_encabezado incluye la Dirección Fuente IP, Dirección Destino IP, Protocolo y la Longitud TCP. Parte de esta información la posee el protocolo interred, IP. TCP obtiene esta información a través de la interfaz entre las regiones Transporte/interred. Para ello TCP utiliza argumentos ó resultados de invocaciones a rutinas de la región interred.

En datagramas IP transportando segmentos TCP, el valor del campo Protocolo es igual a seis. El campo Longitud del segmento TCP es la longitud del encabezado TCP más la longitud de los datos expresada en octetos sin contabilizar los doce octetos en el seudo_encabezado.

El TCP receptor extrae del datagrama IP que transportó al segmento TCP hasta el receptor, la información necesaria para formar el seudo_encabezado y calcula la suma de verificación para comprobar que el segmento llegó intacto.

>> Apuntador Urgente. 16 bits. Representa una distancia positiva desde el número de secuencia en este segmento. Apunta al número de secuencia del octeto que se encuentra inmediatamente después de los datos urgentes. Este campo solo tiene significado en segmentos con el bit URG activado.

>> Opciones. variable. Las opciones ocupan un espacio cuya longitud es múltiplo de ocho bits. Las opciones sí se contabilizan en el cálculo de la Suma de Verificación. Se tienen dos casos en el formato de una opción:

1. Un solo octeto: tipo de opción.
2. Octeto tipo de opción, octeto longitud de opción, octetos de datos de la opción.

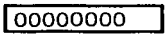
La longitud de la opción contabiliza todos los octetos de la opción incluyendo el propio octeto de longitud. El contenido del encabezado después de la opción Fin de Opción debe ser cero. Un módulo TCP debe implementar todas las opciones.

Opciones:

Tipo	Longitud	Significado
0	-	Fin de lista de opciones.
1	-	Ninguna operación.
2	4	Tamaño máximo de segmento.

Figura 3.22. Tipos de opciones TCP.

** Fin de lista de opciones.

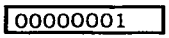


Tipo 0

Figura 3.23. Opción fin de lista de opciones.

Se utiliza al final de todas las opciones, no al final de cada opción, y sólo cuando el final de las opciones no coincide con el final del encabezado TCP.

** Ninguna operación.



Tipo 1

Figura 3.24. Ninguna operación.

Se puede utilizar entre opciones contiguas con la finalidad de alinear octetos para que inicien en límites de palabras.

** Tamaño máximo de segmento. 16 bits.

00000010	00000100	Tamaño máximo de segmento.
Tipo 2	Longitud 4	

Figura 3.25. Tamaño máximo de segmento.

Muestra el tamaño máximo de segmento, admisible, en el módulo TCP que está enviando este segmento. Este campo solo se envía al inicio de una solicitud de conexión, es decir en segmentos con la bandera SYN activada. Si no se utiliza esta opción, cualquier tamaño de segmento es admisible.

>> Relleno. variable. Se utiliza para que el final del encabezado y el inicio de los datos, ocurra en algún límite múltiplo de 32 bits. Este campo contiene ceros.

3.3 Regiones de Modelos

Esta sección está basada en la referencia [1].

Modelos de Referencia.

Se tratan brevemente los siguientes modelos:

1. Modelo de Referencia OSI.
2. Modelo de Referencia TCP/IP.

3.3.1 Modelo de Referencia OSI

Este modelo está basado en una propuesta de ISO (International Standards Organization). Este modelo se denomina Modelo de Referencia ISO OSI (Open System Interconnection). Un sistema es abierto si está dispuesto a comunicarse con otros sistemas. Comúnmente se le denomina a este modelo simplemente Modelo OSI.

El modelo OSI consta de siete capas y se fundamenta en los siguientes principios.

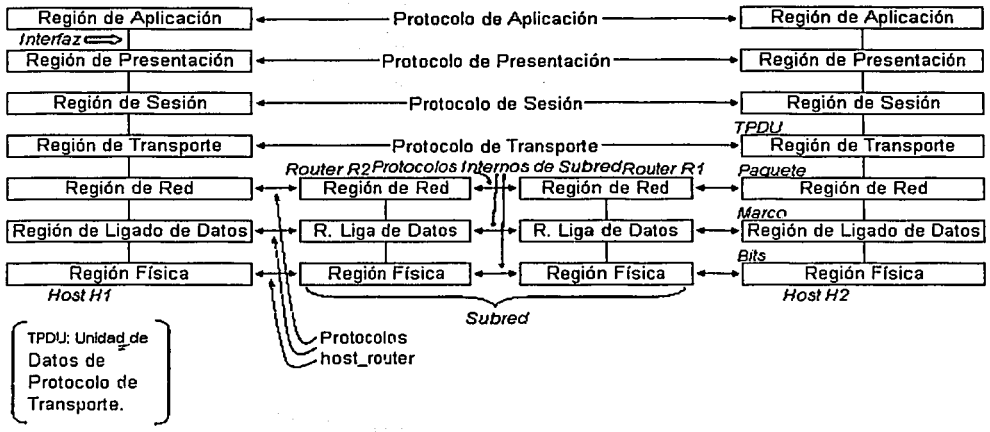


Figura 3.26. El modelo OSI.

1. A cada nivel de abstracción diferente le corresponde una región diferente.
2. Cada región tiene asignada una función bien definida.
3. La función de cada región debe considerar la definición de protocolos internacionalmente estandarizados.
4. Los límites de cada región deben ser tales que se minimice el flujo de información a través de las interfaces.
5. El número de regiones debe ser lo suficientemente alto tal que no se incluyan funciones distintas en la misma región innecesariamente; y debe ser lo suficientemente pequeño tal que la arquitectura sea adecuadamente tratable.

El modelo OSI no es una arquitectura de Red pues no especifica los servicios y protocolos en cada región. Sólo trata acerca de lo que cada región debería contener.

7	Aplicación
6	Presentación
5	Sesión
4	Transporte
3	Red
2	Ligado de datos
1	Física

Figura 3.27. Regiones del modelo OSI.

La Región Física.

Esta región tiene asignadas diversas tareas:

1. Transmitir bits a través de un canal de comunicación, revisando que el nivel lógico de cada bit no se altere al transmitirlo y/ó recibirlo.
2. Tomar una decisión en cuanto al nivel de voltaje a utilizar para cada nivel lógico.
3. Determinar la frecuencia (Hertz) de transmisión.
4. Determinar el tipo de transmisión: en una dirección (simplex); en ambos sentidos no simultáneo (half duplex); ó bidireccional (full duplex).
5. Determinar la forma de iniciar y terminar una conexión.
6. Determinar cuántos terminales (pines) debe tener el conector de red y la función que desempeña cada terminal (pin).
7. Determinar las interfaces mecánicas, eléctricas y de procedimientos.
8. Determinar el medio físico de transmisión.

Región de Ligado de Datos.

La tarea principal de esta región es tomar un medio de transmisión y transformarlo en una línea tal que la Región de Red vea esta línea como una línea

que no tiene errores de transmisión. Para ello esta región hace que el emisor divida los datos de entrada en marcos de datos (data frames) de tamaño de cientos ó unos cuantos miles de bytes. Luego el transmisor envía secuencialmente los marcos y el transmisor entonces procesa los marcos de reconocimiento (acknowledgement frames) que le envía el receptor como respuesta. La Región de Ligado de Datos se encarga de crear y reconocer los límites de los marcos. Para este reconocimiento esta región agrega unas secuencias especiales de bits al inicio y final de cada marco.

Cuando un marco se altera durante la transmisión el emisor retransmite el marco. Es también responsabilidad de esta región resolver los problemas causados por marcos dañados, perdidos y duplicados. La Región de Ligado de Datos tiene diferentes clases de servicio de diferente calidad y precio.

Otras características de esta región son:

1. Mecanismos de Regulación de Flujo. El transmisor debe conocer el tamaño del área (buffer) de datos del receptor.
2. Manejo de errores.
3. Manejo inteligente de reconocimientos, p. ej., piggybacking.
4. En redes de difusión amplia (broadcast): Arbitraje del canal compartido. La subregión MAC (Control de Acceso al Medio) de la Región de Ligado de Datos se encarga del arbitraje en este caso.

Región de Red.

Esta región se encarga del control de la operación de la subred. Determina la manera de enrutar los paquetes de fuente a destino (emisor a receptor).

Las rutas se pueden determinar en diversas formas:

1. Con tablas estáticas. Las tablas estáticas cambian solo en ocasiones.
2. Las rutas se determinan al inicio de cada conversación.
3. Las rutas se determinan para cada paquete de datos.

Cuando la cantidad de paquetes en la subred es elevada, puede haber embotellamientos y el control de esta situación está a cargo de la Región de Red.

Características de esta región lo son también las siguientes:

1. Contabilizar los paquetes, caracteres y bits enviados por cada usuario para crear información de precios y tarifas.
2. Resolver problemas como los siguientes:
   2.1. Al viajar un paquete de una a otra red puede ocurrir que el direccionamiento en las redes sea diferente.
   2.2. La segunda red tal vez rechace un paquete cuando este sea muy grande.
   2.3. Los protocolos pueden ser diferentes en cada red.
3. En redes de difusión amplia (broadcast) el problema de enrutaje es sencillo de tratar. En este caso la Región de Red es diminuta ó tal vez no existe.

Región de Transporte.

Desde la Región Física hasta e incluyendo la Región de Red la comunicación entre capas homólogas se efectúa entre el anfitrión y la máquina vecina (tal vez un enrutador) y no precisamente entre anfitriones. En la Región de Transporte las conversaciones son realmente de extremo_a_extremo, entre fuente y destino (emisor y receptor) es decir un programa en la máquina fuente conversa con un programa en la máquina destino con la ayuda de los encabezados de mensaje y mensajes de control.

La función más importante de la Región de Transporte es aceptar datos de la Región de Sesión, dividirlos tal vez en unidades más pequeñas, pasarlos a la Región de Red y asegurar que lleguen correctamente al otro extremo y de manera eficiente tal que se aíslen las capas superiores de cambios en la tecnología de hardware.

Comúnmente la Región de Transporte crea una conexión de red distinta para cada conexión de transporte solicitada por la Región de Sesión.

Si la conexión de transporte requiere un alto rendimiento entonces se tienen dos posibilidades:

1. La Región de Transporte puede crear varias conexiones de red dividiendo los datos entre las conexiones de red.
2. Si la creación y mantenimiento de las conexiones de red son de costo considerable, la Región de Transporte combina ("multiplexa") varias conexiones de transporte en una misma conexión de red.

En cualquier caso la Región de Transporte debe hacer transparente (invisible), la combinación de conexiones (multiplexaje) a la Región de Sesión.

La Región de Transporte decide sobre el tipo de servicio proporcionado a la Región de Sesión.

El tipo más conocido de conexión de transporte es el de un canal punto_a_punto libre de errores que entrega mensajes ó bytes en el mismo orden en que fueron enviados.

También existen otros tipos de servicio brindados por esta región:

1. Mensajes aislados sin garantía del orden de entrega.
2. Mensajes de difusión amplia (broadcasting) a múltiples destinos.

El tipo de servicio se determina al inicio de la conexión.

El encabezado de transporte es el lugar adecuado para incluir la información que especifica la conexión a la que pertenece cada mensaje siendo ésta una determinación no trivial en caso de haber múltiples conexiones entrando y saliendo de cada anfitrión.

También son tareas de esta región las siguientes:

1. Establecer y terminar conexiones de la red.

2. Dar el mecanismo de regulación del flujo de información tal que un anfitrión veloz no agobie a un anfitrión lento. El control de flujo entre anfitriones es diferente al control de flujo entre enrutadores.

Región de Sesión.

La Región de Sesión permite a los usuarios establecer sesiones. Una sesión permite p. ej., que el usuario tenga acceso a un sistema de tiempo compartido lejano, ó una transferencia de archivos entre dos máquinas.

Uno de los servicios de esta región es la administración del control de diálogo; el flujo puede ser unidireccional ó bidireccional. Cuando es unidireccional esta región administra el control del turno para comunicarse, p. ej., el Servicio de Administración de Turno (Token) de algunos protocolos donde solo el poseedor del turno (token) puede efectuar una operación crítica.

Otro servicio en esta región es la Sincronía donde la Región de Sesión inserta puntos de observación (checkpoints) en el flujo de datos para repetir solamente los datos posteriores al último punto de observación en caso de ocurrir alguna falla durante la transmisión.

Región de Presentación.

La Región de Presentación se ocupa de la sintaxis y la semántica de la información transmitida, p.ej., la codificación de datos de acuerdo a algún estándar. Los programas de usuario utilizan nombres de personas, cantidades numéricas, fechas, colores, etc. Estos elementos a su vez se representan como cadenas de caracteres, enteros, números de punto flotante, estructuras de datos, etc. Las diferentes computadoras pueden codificar de manera diferente a estos elementos. Para efectuar una comunicación estos elementos se definen de acuerdo a estándares abstractos de red y son traducidos al mismo estándar abstracto de red. La Región de Presentación administra estos estándares convirtiendo de la representación interna de cada computadora a la representación estándar de red y viceversa.

Región de Aplicación.

La Región de Aplicación contiene una gran variedad de protocolos que son necesarios p. ej., existe una enorme cantidad de tipos de terminal incompatibles en todo el planeta. Una solución es emplear una "terminal virtual de red" abstracta estándar.

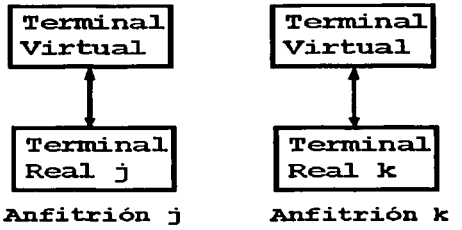


Figura 3.28. Terminal virtual de red.

El emisor 'j' crea una parte software donde se mapean los comandos de la terminal real 'j' a comandos de terminal virtual estándar y estos comandos se transmiten; entonces el receptor 'k' mapea los comandos de terminal virtual estándar a comandos de la terminal real 'k' de acuerdo a una parte software creada en el receptor 'k'. Todo el software de terminal virtual está en la Región de Aplicación.

La Región de Aplicación también se encarga de la transferencia de archivos resolviendo problemas como los siguientes:

1. Diferentes convenciones de nombramiento.
2. Formas diferentes para representar líneas de texto, etc.

Estos problemas se presentan cuando en la transferencia de archivos los extremos de la conversación pertenecen a diferentes sistemas de archivos.

También la Región de Aplicación tiene a su cargo las siguientes tareas:

1. Correo electrónico.
2. Tareas a distancia (Remote jobs).
3. Búsqueda en directorios.

Transmisión de los datos en el modelo OSI.

El desarrollo de la transmisión de datos inicia cuando el proceso emisor necesita enviar datos al proceso receptor; entonces pasa los datos a la Región de Aplicación la cual agrega un encabezado de aplicación AH (el cual puede ser nulo) al frente de los datos y pasa el resultado de este proceso a la Región de Presentación. La Región de Presentación repite el proceso y pasa el resultado a la Región de Sesión. Y esta secuencia se repite hasta que los datos llegan a la Región Física desde la cual los datos son transmitidos a la máquina receptora. En la máquina receptora los encabezados se extraen uno a uno mientras los datos ascienden desde la Región Física receptora dirigiéndose hacia arriba de la Región Física hasta que finalmente los datos se pasan al proceso receptor.

La idea en esta situación de la transmisión de datos es que si bien es cierto que el flujo de datos es vertical a través de las regiones, la programación de las regiones se proyecta como si la transmisión de datos fuese horizontal.

3.3.2 Modelo de Referencia TCP/IP

Algunas finalidades importantes cuando se creó el Modelo de Referencia TCP/IP fueron las siguientes:

1. Posibilidad de conectar diferentes redes.
2. La conexión entre fuente y destino debe subsistir a pesar de fallas en las máquinas ó en las líneas de transmisión existentes entre las máquinas fuente y destino.

3. Posibilidad de comunicación entre aplicaciones con requisitos muy diferentes, variando p. ej., desde aplicaciones de transferencia de archivos hasta aplicaciones de transmisión de voz en tiempo real.

El modelo de referencia TCP/IP se forma de cuatro capas:

4. Región de Aplicación.
3. Región de Transporte.
2. Región Interred.
1. Región Anfitrión_a_Red.

Región Anfitrión_a_Red.

El modelo de referencia TCP/IP indica que el anfitrión se debe conectar a la red utilizando algún protocolo de tal forma que el anfitrión pueda enviar paquetes IP sobre la red. Este protocolo no está definido por este modelo.

Región Interred.

El trabajo de la Región Interred es permitir que los anfitriones introduzcan datos (paquetes) a cualquier red haciéndolos viajar de forma independiente (probablemente sobre diferentes redes) hacia el destino. Posiblemente llegarán al destino en un orden diferente al orden en que fueron enviados originalmente siendo trabajo de las capas de más arriba el reordenamiento de los datos si es necesaria una entrega en el mismo orden en que fueron enviados.

La Región Interred define un formato de paquete oficial, y un protocolo denominado IP (Protocolo Interred). El trabajo de la Región Interred es entregar en el lugar indicado los paquetes IP. Esta región resuelve el enrutaje de paquetes y evita congestionamientos. En funcionalidad esta región es muy parecida a la Región de Red en el Modelo OSI.

Región de Transporte.

La Región de Transporte permite la conversación de entidades homólogas en los anfitriones fuente y destino de forma análoga a como ocurre en la Región de Transporte del Modelo OSI.

Se han definido para esta región del modelo TCP/IP dos protocolos extremo_a_extremo:

1. TCP Protocolo de Control de la Transmisión.
2. UDP Protocolo de Datagramas del Usuario.

TCP es un protocolo orientado a la conexión y confiable. Permite el flujo de bytes desde una máquina origen hasta cualquier otra máquina en la interred entregando los bytes sin errores. El TCP emisor forma mensajes a partir del flujo de bytes de entrada y pasa cada mensaje a la Región Interred. Cuando los mensajes llegan a la máquina destino, el proceso TCP receptor reensambla los mensajes que recibe y de esta forma reconstruye los datos originales.

TCP realiza control de flujo para asegurar que un emisor veloz no agobie a un receptor lento con más mensajes de los que pueda hacerse cargo.

UDP es un protocolo sin conexión no confiable utilizado cuando las aplicaciones rechazan el control de flujo y el secuenciamiento proporcionados por TCP y deseen proporcionar los suyos propios. UDP se utiliza ampliamente en modelos cliente_servidor del tipo solicitud_respuesta y en aplicaciones donde es más importante una respuesta rápida que una respuesta precisa como sucede en transmisiones de video y voz.

Región de Aplicación.

La experiencia con la utilización del modelo OSI ha demostrado que las regiones de Sesión y de Presentación son de uso escaso en la mayoría de aplicaciones.

La Región de Aplicación en el modelo TCP/IP contiene todos los protocolos de nivel superior. Los primeros protocolos incluidos fueron:

1. TELNET. Manejo de Terminal Virtual. Permite a un usuario en una máquina identificarse remotamente en una máquina distante y utilizar la máquina distante como si el usuario estuviese junto a esta última.
2. FTP. Transferencia de Archivos. Permite la transferencia eficiente de datos entre dos máquinas distintas.
3. SMTP. Correo Electrónico. Originalmente el correo electrónico se trataba como un tipo de transferencia de archivos.

Otros protocolos se incluyen en la Región de Aplicación, p. ej.,

1. DNS. Servicio de Nombres de Dominio. Mapea nombres de anfitrión a direcciones de red.
2. HTTP. Protocolo para buscar páginas en WWW (World Wide Web).

etc.

3.4 Modelo Cliente_Servidor

Esta sección está basada en la referencia [2].

Según la cantidad de clientes a los cuales da atención el servidor al mismo tiempo, los servidores se clasifican en:

1. Servidores Concurrentes
2. Servidores Iterativos

Los servidores concurrentes administran diversos clientes de forma concurrente mas no significa que se tengan múltiples procesos o cuerdas concurrentes.

El servidor concurrente se puede implementar con una sola cuerda la cual utiliza entrada/salida asíncrona permitiendo el uso simultáneo de múltiples canales de comunicación.

Los servidores iterativos procesan una sola solicitud de un solo cliente a la vez. Los servidores iterativos son más lentos que los servidores concurrentes si bien son más fáciles de diseñar comparados con los servidores concurrentes.

Según el protocolo utilizado para comunicarse, la clasificación de servidores es la siguiente:

1. Servidores orientados a la conexión
2. Servidores sin conexión

Los servidores orientados a la conexión utilizan el protocolo TCP, los servidores sin conexión utilizan el protocolo UDP.

El protocolo TCP retransmite datos perdidos, verifica los datos, reordena paquetes recibidos cuando es necesario. El protocolo TCP tiene el inconveniente de utilizar un conector (socket) para cada conexión mientras los diseños sin conexión admiten la comunicación con diversos anfitriones utilizando un mismo conector.

El protocolo UDP no retransmite datos perdidos ni verifica errores, tampoco reordena paquetes recibidos, pues todos estos requerimientos serán responsabilidad del usuario del protocolo UDP. En ocasiones un diseño UDP funcionando adecuadamente en una red local, pudiera fallar cuando se utiliza en redes de área amplia. En este caso la estrategia de retransmisión debe ser adaptiva.

Las conexiones TCP son punto_a_punto lo cual significa la imposibilidad de comunicarse empleando los esquemas de difusión amplia (broadcast), y de multidifusión (multicast) con este protocolo. Para tales casos es necesario utilizar el protocolo UDP.

De acuerdo con las clasificaciones anteriores los diversos tipos de servidores son:

1. Servidor iterativo sin conexión
2. Servidor iterativo orientado a la conexión
3. Servidor concurrente sin conexión
4. Servidor concurrente orientado a la conexión

Consideremos la siguiente situación: algunos anfitriones poseen más de una dirección IP, por ejemplo anfitriones multidirección. El servidor debe utilizar una dirección IP cuando inicializa sus operaciones como servidor haciendo una llamada a la función bind() de la Interfaz Conectiva. Los clientes no conocen todas las direcciones IP del anfitrión, entonces al tratar de comunicarse con el servidor multidireccionable, el servidor rechazaría aquellas conexiones dirigidas a una dirección diferente a la especificada en la llamada a la función bind(). Esta situación se evita si el servidor utiliza como su dirección IP la constante INADDR_ANY definida en la Interfaz Conectiva (Socket Interface).

A continuación se presentan los algoritmos correspondientes a cada tipo de servidor:

Algoritmo de un servidor iterativo sin conexión:

1. Crear un conector y ligarlo a una dirección
2. Recibir al siguiente cliente
3. Dar una respuesta a la solicitud del cliente de acuerdo al protocolo convenido según el servicio proporcionado al cliente por el servidor
4. Continuar en el inciso dos

Algoritmo de un servidor iterativo orientado a la conexión:

1. Crear un conector y ligarlo a una dirección
2. Configurar el conector en modo pasivo
3. Recibir al siguiente cliente
4. Obtener un nuevo conector para el cliente recibido
5. Comunicarse con el cliente de acuerdo al protocolo convenido según el servicio proporcionado al cliente por el servidor
6. Terminar la comunicación de acuerdo al protocolo convenido utilizando las funciones proporcionadas por la Interfaz Conectiva
7. Continuar en el inciso tres

Algoritmo de un servidor concurrente sin conexión:

1. Crear un conector y ligarlo a una dirección
2. Utilizar repetidamente la función recvfrom() de la Interfaz Conectiva para recibir al siguiente cliente
3. Crear un nuevo proceso o una nueva cuerda esclavos, para dar atención al cliente aceptado
4. El nuevo esclavo da respuesta a la solicitud del cliente de acuerdo al protocolo convenido según el servicio proporcionado al cliente por el servidor utilizando la función sendto() de la Interfaz Conectiva
5. Salir del proceso o cuerda esclavos después de enviar la respuesta al cliente

En este caso de un servidor concurrente sin conexión la cuerda principal permanece en un lazo sinfín recibiendo nuevos clientes utilizando la función recvfrom(), mientras los esclavos dan respuesta a los clientes. Existe un esclavo para cada cliente aceptado por la cuerda principal.

Algoritmo de un servidor concurrente orientado a la conexión:

1. Crear un conector y ligarlo a una dirección
2. Configurar el conector en modo pasivo
3. Repetidamente recibir al siguiente cliente
4. Obtener un nuevo conector para el cliente recibido
5. Crear un nuevo proceso o una nueva cuerda esclavos, para dar atención al cliente aceptado
6. El nuevo esclavo se comunica con el cliente de acuerdo al protocolo convenido según el servicio proporcionado al cliente por el servidor
7. El nuevo esclavo también se ocupa de la terminación de la comunicación con el cliente de acuerdo al protocolo convenido utilizando las funciones proporcionadas por la Interfaz Conectiva

El sistema operativo Windows proporciona la función CreateProcess() la cual permite crear un nuevo proceso y la función _beginthread() para crear cuerdas.

Es posible utilizar una sola cuerda en la implementación de un servidor concurrente orientado a la conexión. En este caso se tiene una concurrencia aparente.

Algoritmo de un servidor concurrente orientado a la conexión utilizando una sola cuerda

1. Crear un conector y ligarlo a una dirección
2. Agregar el conector a la lista de conexiones_entrada/salida posibles
3. Utilizar la función select() para las conexiones_entrada/salida posibles
4. Si el conector principal está dispuesto, aceptar una nueva conexión

   4.1. Asignar un nuevo conector para la nueva conexión

4.2. Agregar el nuevo conector a la lista de conexiones_entrada/salida posibles

5. Si algún otro conector esta dispuesto, utilizar la función recv() para recibir la solicitud del cliente y utilizar la función send() para dar respuesta a la solicitud del cliente
6. Continuar en el inciso tres

Desde el punto de vista del cliente, un servidor concurrente orientado a la conexión y un servidor concurrente orientado a la conexión utilizando una sola cuerda, son iguales.

3.5 Interfaz Conectiva

La función socket() crea un conector:

```
SOCKET socket (
   int dirección,
   int tipo,
   int protocolo
  );
```

dirección: PF_INET.

tipo: SOCK_STREAM ó SOCK_DGRAM.

protocolo: nombre de protocolo en particular ó cero si el llamador prefiere no especificar el protocolo.

En caso de no haber errores, socket() regresa un descriptor el cual hace referencia al nuevo conector. En caso de error regresa INVALID_SOCKET. El código de error se recupera con la función WSAGetLastError().

La función bind() asocia una dirección local a un conector.

```
int bind (
```

```
SOCKET s,
const struct sockaddr FAR * dirección,
int longitud
);
```

s: descriptor de conector no enlazado.

dirección: la dirección por asignar al conector.

la estructura sockaddr tiene la siguiente forma:

```
struct sockaddr {
    u_short    sa_family;
    char       sa_data[14];
};
```

longitud: la longitud de la dirección.

En caso de no haber errores bind() regresa cero. Si hay algún error regresa SOCKET_ERROR y el código de error se recupera llamando a la función WSAGetLastError(). La función bind() se utiliza en un conector no conectado (connect())de tipo SOCK_DGRAM, ó SOCK_STREAM.

La función connect() establece una conexión con otro conector.

```
int connect (
SOCKET s,
const struct sockaddr FAR * nombre,
int longitud
);
```

s: descriptor de un conector no conectado.

nombre: el conector con el cual se va a conectar.

longitud: la longitud del nombre.

Cuando no hay errores la función connect() regresa cero, de otra forma regresa SOCKET_ERROR y el código de error se recupera con la función WSAGetLastError(). La función connect() se utiliza en un conector no conectado (connect()), de tipo SOCK_DGRAM, ó SOCK_STREAM.

La función accept() acepta una conexión de un conector.

```
SOCKET accept (
  SOCKET s,
  struct sockaddr FAR * dirección,
  int FAR * longitud
 );
```

s: un descriptor de conector el cual está escuchando (listen()).

dirección: un apuntador opcional el cual recibe la dirección de la entidad que solicita la conexión.

longitud: apuntador a un entero el cual contiene la longitud de la dirección. Inicialmente contiene la longitud supuesta de la dirección, y al regreso contiene la verdadera longitud en bytes de la dirección retornada. Si dirección y/o longitud es igual a NULL, accept() no regresa información de la dirección remota.

Si no ocurre algún error, accept() retorna un valor de tipo conector (socket) el cual es un descriptor de un nuevo conector diferente a "s". En caso de error la función accept() retorna SOCKET_ERROR y el código de error se recupera con WSAGetLastError(). accept() se utiliza en conectores de tipo SOCK_STREAM.

La función listen() prepara a un conector y lo deja esperando a futuras conexiones.

```
int listen (
  SOCKET s,
  int longitud
 );
```

s: descriptor de un conector enlazado, no conectado.

longitud: longitud máxima hasta la que puede crecer la cola de conexiones pendientes. Esta función sólo es aplicable a conectores de tipo SOCK_STREAM. La llamada a esta función pone al conector "s" en modo pasivo. El máximo aceptable actualmente es cinco y valores menores a uno ó mayores a cinco son aproximados al valor más cercano, 1 ó 5. Cuando todo sale bien listen() regresa cero, en caso de error retorna SOCKET_ERROR y el código de error se recupera con WSAGetLastError().

La función closesocket() cierra un conector.

```
int closesocket (
   SOCKET s
  );
```

s: descriptor de conector.

Si todo está bien closesocket() regresa cero. En caso de error retorna SOCKET_ERROR y el código de error se obtiene con WSAGetLastError().

La función recv() recibe datos desde un conector.

```
int recv (
   SOCKET s,
   char FAR * buffer,
   int longitud,
   int banderas
  );
```

s: descriptor de un conector conectado (connect()).

buffer: un espacio para recibir los datos de entrada.

longitud: la longitud en bytes del área (buffer) de datos.

banderas: especifica la forma en que se hace la llamada.

Las banderas son las siguientes.

MSG_PEEK los datos se copian al área (buffer) de datos pero no se quitan de la cola de entrada.

MSG_OOB especifica datos fuera-de-banda en conectores tipo SOCK_STREAM. Si el conector está configurado con la opción SO_OOBINLINE y existen datos fuera-de-banda, con la opción MSG_OOB especificada se recibirán sólo datos fuera-de-banda y con la función ioctlsocket(,SIOCATMARK,) es posible saber si existen más datos fuera-de-banda.

En caso de éxito recv() regresa el número de bytes recibidos. Si la conexión se ha cerrado regresa cero. En caso de errores retorna SOCKET_ERROR y el código de error se obtiene con WSAGetLastError(). recv() se puede utilizar en conectores de tipo SOCK_DGRAM, ó SOCK_STREAM.

La función recvfrom() recibe datos y guarda la dirección fuente.

```
int recvfrom (
   SOCKET s,
   char FAR * buffer,
   int longitud,
   int banderas,
   struct sockaddr FAR * origen,
   int FAR * longitud2
  );
```

s: descriptor de un conector enlazado.

buffer: un espacio para los datos de entrada.

longitud: la longitud en bytes del área (buffer) de datos .

banderas: especifica la forma en que se hace la llamada.

Las banderas son las siguientes.

MSG_PEEK los datos se copian al área (buffer) de datos pero no se quitan de la cola de entrada.

MSG_OOB especifica datos fuera-de-banda en conectores tipo SOCK_STREAM. Si el conector está configurado con la opción SO_OOBINLINE y existen datos fuera-de-banda, con la opción MSG_OOB especificada se recibirán solo datos fuera-de-banda y con la función ioctlsocket(,SIOCATMARK,) es posible saber si existen más datos fuera-de-banda.

origen: apuntador a datos conteniendo la dirección de origen de los datos recibidos.

longitud2: apuntador al tamaño en bytes del origen.

En caso de éxito recvfrom() regresa el número de bytes recibidos. Si la conexión se ha cerrado regresa cero. Si existe algún error retorna SOCKET_ERROR y la información de error se obtiene con WSAGetLastError(). recvfrom() se utiliza en un conector posiblemente conectado de tipo SOCK_DGRAM ó SOCK_STREAM.

En conectores tipo SOCK_STREAM los parámetros origen y longitud2 se ignoran.

La función send() envía datos en un conector conectado (connect()).

```
int send (
   SOCKET s,
   const char FAR * buffer,
   int longitud,
   int banderas
  );
```

s: descriptor de un conector conectado.

buffer: un espacio donde se encuentran los datos a transmitir.

longitud: la longitud en bytes de los datos en el área (buffer) de datos.

banderas: especifica la forma en que se hace la llamada.

Las banderas son:

MSG_DONTROUTE indica que los datos por enviar no deberían estar sujetos a enrutaje, aunque en realidad esta bandera podría ser ignorada.

MSG_OOB indica que los datos por enviar son datos fuera-de-banda. Sólo se aplica en conectores tipo SOCK_STREAM.

Cuando no hay errores send() regresa el número de caracteres enviados, el cual puede ser menor al número especificado en longitud. En caso de error regresa SOCKET_ERROR y la descripción del error se obtiene con WSAGetLastError(). send() se puede utilizar en un conector conectado (connect()), de tipo SOCK_DGRAM ó SOCK_STREAM.

La función sendto() envía datos a un destino específico.

```
int sendto (
    SOCKET s,
    const char FAR * buffer,
    int longitud,
    int banderas,
    const struct sockaddr FAR * destino,
    int longitud2
  );
```

s: descriptor de un conector.

buffer: un espacio el cual contiene los datos a enviar.

longitud: el tamaño en bytes del área (buffer) de datos.

banderas: especifica la forma en que se hace la llamada.

Las banderas son:

MSG_DONTROUTE indica que los datos por enviar no deberían estar sujetos a enrutaje, aunque en realidad esta bandera podría ser ignorada.

MSG_OOB indica que los datos por enviar son datos fuera-de-banda. Sólo se aplica en conectores tipo SOCK_STREAM.

destino: un apuntador a la dirección del conector destino.

longitud2: el tamaño en bytes de destino.

Si tiene éxito, sendto() regresa el número total de caracteres enviados. Este número puede ser menor al especificado en longitud. Si existe alguna falla sendto() regresa SOCKET_ERROR y la información acerca del error se obtiene con la función WSAGetLastError(). sendto() se puede utilizar en conectores de tipo SOCK_DGRAM ó SOCK_STREAM. Normalmente se utiliza en conectores del tipo SOCK_DGRAM y en este caso se debe tener cuidado de no exceder el tamaño máximo de paquete IP indicado en el campo iMaxUdpDg de la estructura WSADATA regresada al llamar a la función WSAStartup(). En conectores del tipo SOCK_STREAM los parámetros destino y longitud2 se ignoran.

Al enviar un mensaje de difusión amplia (broadcast) (solo en conectores tipo SOCK_DGRAM) la dirección en el parámetro destino debería ser construida con INADDR_BROADCAST (definida en WINSOCK.H) y un número de puerto. El tamaño del datagrama no debería ser mayor a 512.

3.6 Programas de Control del Sistema

Con el programa "xyz.exe" se introducen los puntos donde debe situarse el Brazo de Robot. La salida de "xyz.exe" es la entrada para el programa "datoposc.exe" el cual proporciona como salida los datos necesarios para los programas de la interfaz Computadora-Brazo_de_Robot.

La secuencia de utilización de programas del sistema es la siguiente:

xyz.exe -> datoposc.exe -> transmisión de datos vía Internet -> salir al DOS -> programas de la Interfaz Computadora-Brazo_de_Robot.

3.6.1 Descripción de los Programas que utilizan la Interfaz Conectiva

A continuación se muestra la secuencia de instrucciones de un servidor iterativo cuya implementación se efectúa con un conector UDP, el servidor hace eco de los datos enviados por el cliente.

Servidor UDP:

1. Inicialización()
2. socket()
3. bind()
4. recvfrom()
5. sendto(), continuar en el inciso 2.

La ventana del servidor UDP es la siguiente:

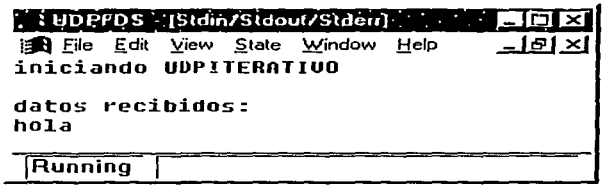


Figura 3.29. Servidor UDP.

Enseguida aparece la secuencia de instrucciones del cliente UDP correspondiente al servidor de eco UDP.

Cliente UDP:

1. Inicialización()
2. socket()
3. bind()
4. sendto()
5. recvfrom(), continuar en el inciso 2.

La ventana correspondiente del cliente UDP es la siguiente:

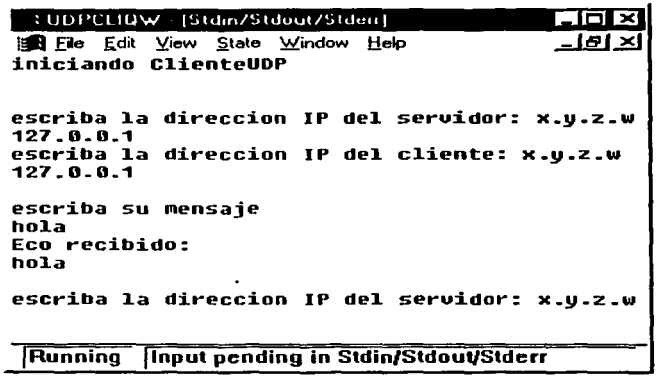


Figura 3.30. Cliente UDP.

A continuación se presenta la secuencia de instrucciones de un servidor de eco, iterativo, orientado a la conexión.

Servidor TCP:

1. Inicialización()
2. socket()
3. bind()
4. listen()
5. accept()
6. recv()
7. send(), continuar en el inciso 5.

La ventana correspondiente al servidor TCP tiene el siguiente aspecto:

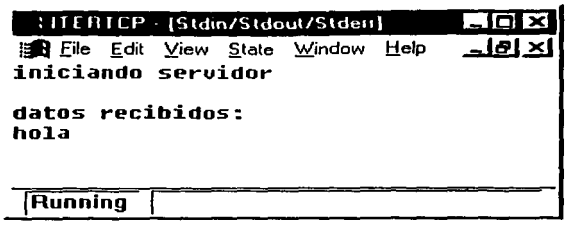


Figura 3.31. Servidor TCP.

La siguiente secuencia de instrucciones corresponde al cliente del servidor TCP anterior.

Cliente_eco TCP:

1. Inicialización()
2. socket()
3. connect()
4. send()
5. recv(), continuar en el inciso 2.

Enseguida se muestra la ventana correspondiente a este cliente TCP.

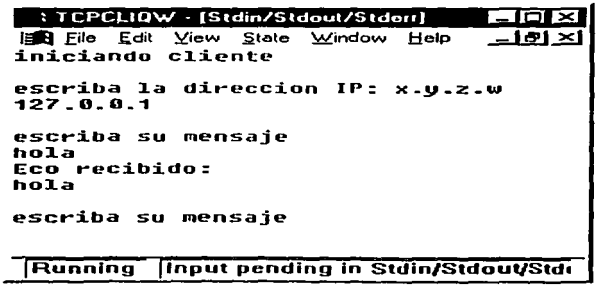


Figura 3.32. Cliente TCP.

3.6.2 *Uso de los Programas interactuando con la Interfaz Conectiva*

Para la comunicación vía Internet se tienen los siguientes grupos de programas:

El programa cli3dumi.exe llama al programa cliente tcpcli.exe, el servidor es serxyz2.exe; este grupo de programas sirve para conversar. El servidor se ubica donde se encuentra el Brazo de Robot Rhino.

El cliente inicia con la siguiente ventana:

Figura 3.33. Inicio de cliente_1.

El servidor muestra la siguiente ventana inicial:

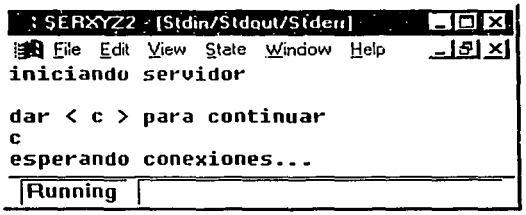


Figura 3.34. Inicio de servidor_1.

Durante la conversación el cliente utiliza el caracter "c", y al finalizar la conversación, los caracteres "xy".

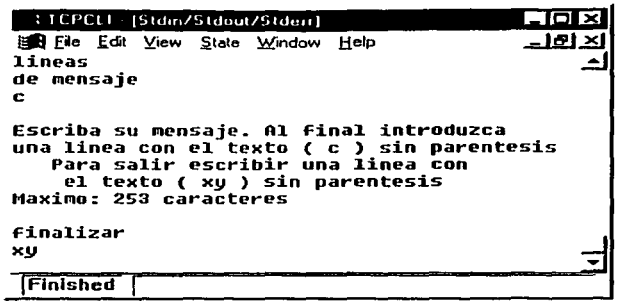


Figura 3.35. Conversación de cliente_1.

Cuando el servidor acepta la comunicación con algún cliente, en el lado del servidor aparece la dirección IP del cliente en un cuadro de mensajes.

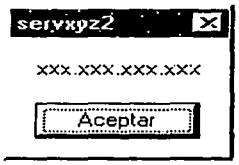


Figura 3.36. Aceptación del servidor_1.

El servidor espera los mensajes del usuario como se muestra en la siguiente ventana:

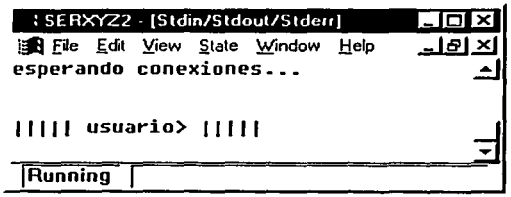


Figura 3.37. Servidor_1 esperando a usuarios.

Al término de la comunicación con algún cliente en particular, es necesario indicar al servidor la disposición de dar atención a más clientes:

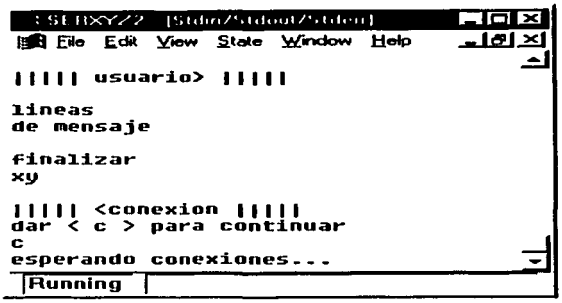


Figura 3.38. Servidor_1 esperando más usuarios.

Para salir del servidor se introduce un caracter diferente a "c".

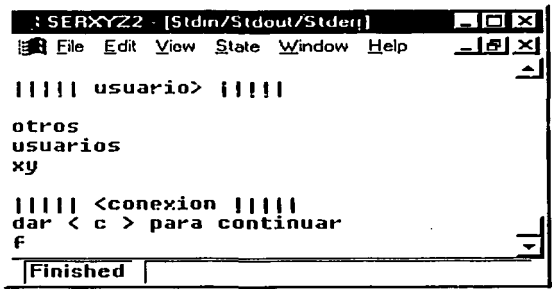


Figura 3.39. Salida del servidor_1.

Otro grupo de programas está formado por el programa usrdumi.exe el cual llama al programa cliente usrcli.exe, el servidor es usrsrv.exe; este grupo de programas sirve para conversar. El servidor se activa del lado lejano del Brazo de Robot Rhino.

El servidor inicia con la siguiente ventana:

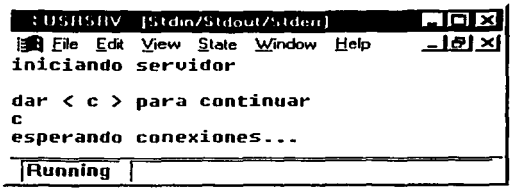


Figura 3.40. Inicio de servidor_2.

El cliente inicia de la siguiente manera:

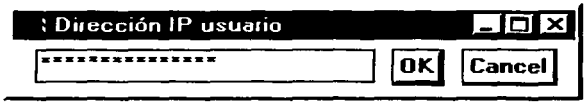


Figura 3.41. Inicio de cliente_2.

Después de introducir la dirección IP del servidor, el cliente obtiene el siguiente recuadro:

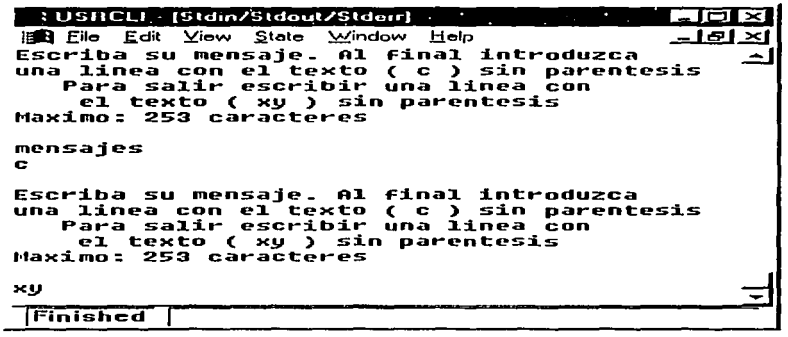


Figura 3.42. Conversación de cliente_2.

El servidor termina cuando se da como respuesta un caracter diferente a la letra "c" :

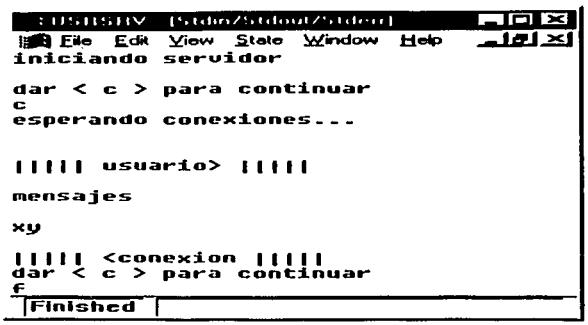


Figura 3.43. Conversación de servidor_2.

El siguiente grupo de programas se utiliza en la transmisión de archivos:

arv2dumi.exe llama al programa cliente cliarv2.exe; el servidor es servarvo.exe. El servidor se puede utilizar en ambos lados de la comunicación, es decir tanto del lado del usuario del Brazo de Robot, como del lado donde se encuentra el Brazo de Robot Rhino. Esto se hace con la finalidad de que se puedan enviar archivos de solicitud por parte del usuario y archivos de respuesta desde donde se encuentra el Brazo de Robot Rhino.

El servidor de archivos presenta la siguiente configuración inicial:

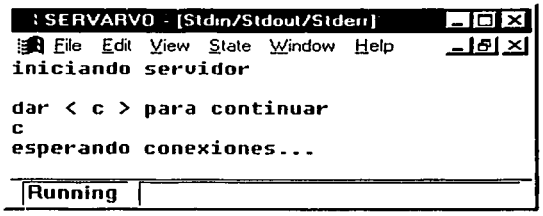


Figura 3.44. Inicio del servidor de archivos.

El cliente aparece con la siguiente ventana:

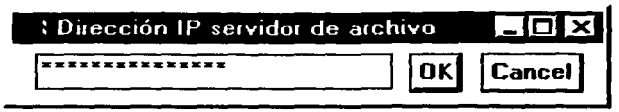


Figura 3.45. Inicio de cliente_3.

Después de introducir la dirección IP del servidor de archivos, el programa cliente muestra la siguiente ventana:

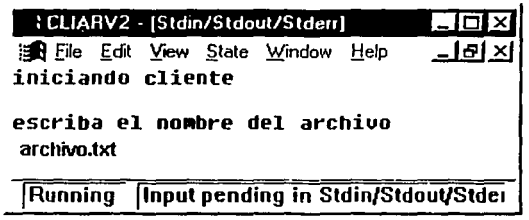


Figura 3.46. El usuario proporciona el nombre del archivo al cliente_3.

Enseguida el usuario proporciona el nombre del archivo a transmitir y la ventana cambia indicando que la transmisión del archivo está en progreso:

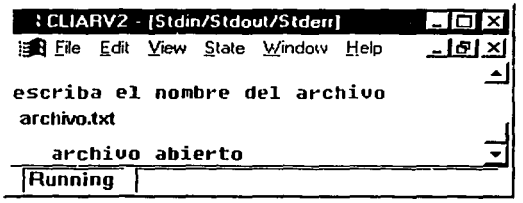


Figura 3.47. Cliente_3 transmitiendo el archivo.

Cuando termina la transmisión del archivo, el cliente muestra la siguiente ventana:

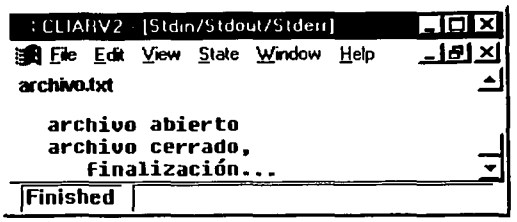


Figura 3.48. Salida del cliente_3.

Después de iniciada la comunicación, el servidor prepara el archivo local donde recibirá el archivo del cliente:

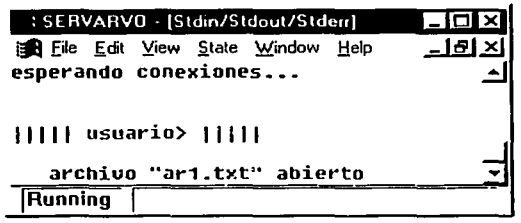


Figura 3.49. El servidor de archivos preparando el archivo de recepción.

Cuando termina la transmisión del archivo en el lado del servidor la ventana presenta la siguiente configuración:

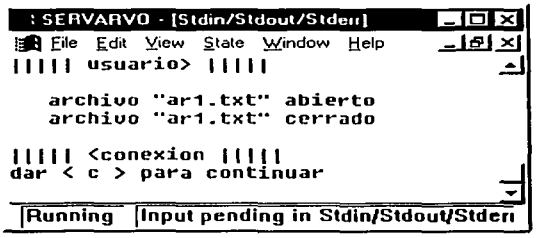


Figura 3.50. El servidor de archivos recibió el archivo del cliente.

Al indicar el caracter "c", el servidor muestra la siguiente ventana.

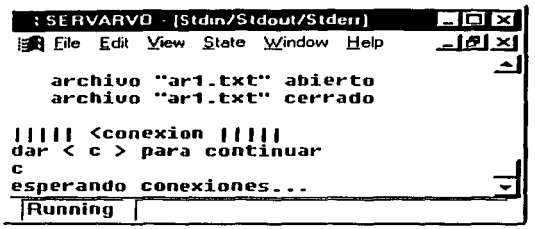


Figura 3.51. El servidor de archivos continúa recibiendo archivos de los clientes.

Para salir del servidor es necesario proporcionar un caracter diferente a "c":

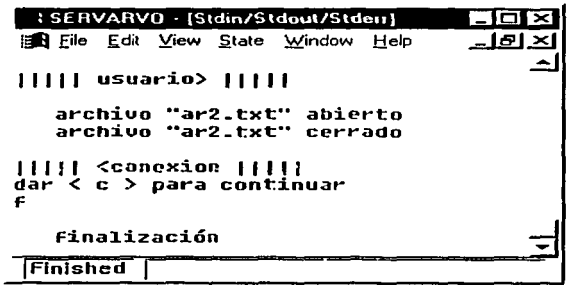


Figura 3.52. Salida del servidor de archivos.

4 Sistemas Electrónicos y de Control

4.1 Sistema Electrónico del Controlador del Brazo de Robot

La interfaz Controlador-Computadora es la siguiente:

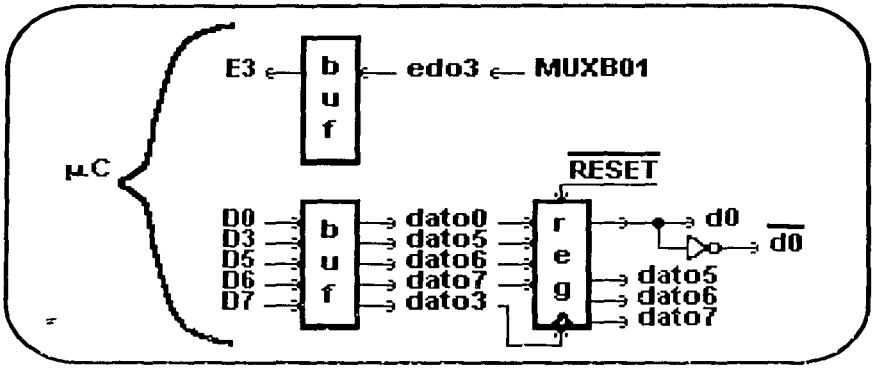


Figura 4.1. Interfaz Controlador-Computadora.

La interfaz con los encodificadores del Brazo de Robot es la siguiente:

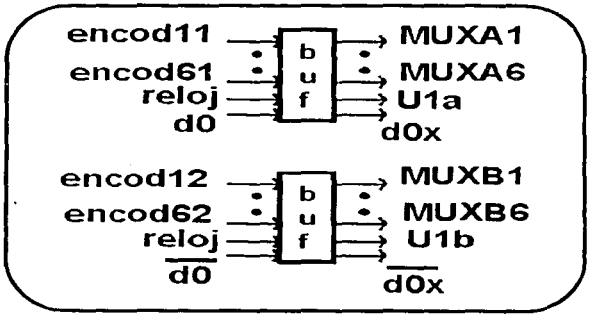


Figura 4.2. Interfaz con los encodificadores del Brazo de Robot.

La interfaz de potencia tiene la siguiente configuración en cada motor:

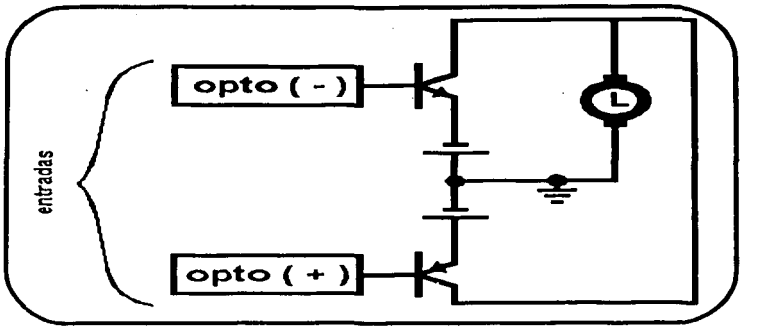


Figura 4.3. Interfaz de potencia del sistema de Brazo de Robot.

El controlador tiene la siguiente disposición:

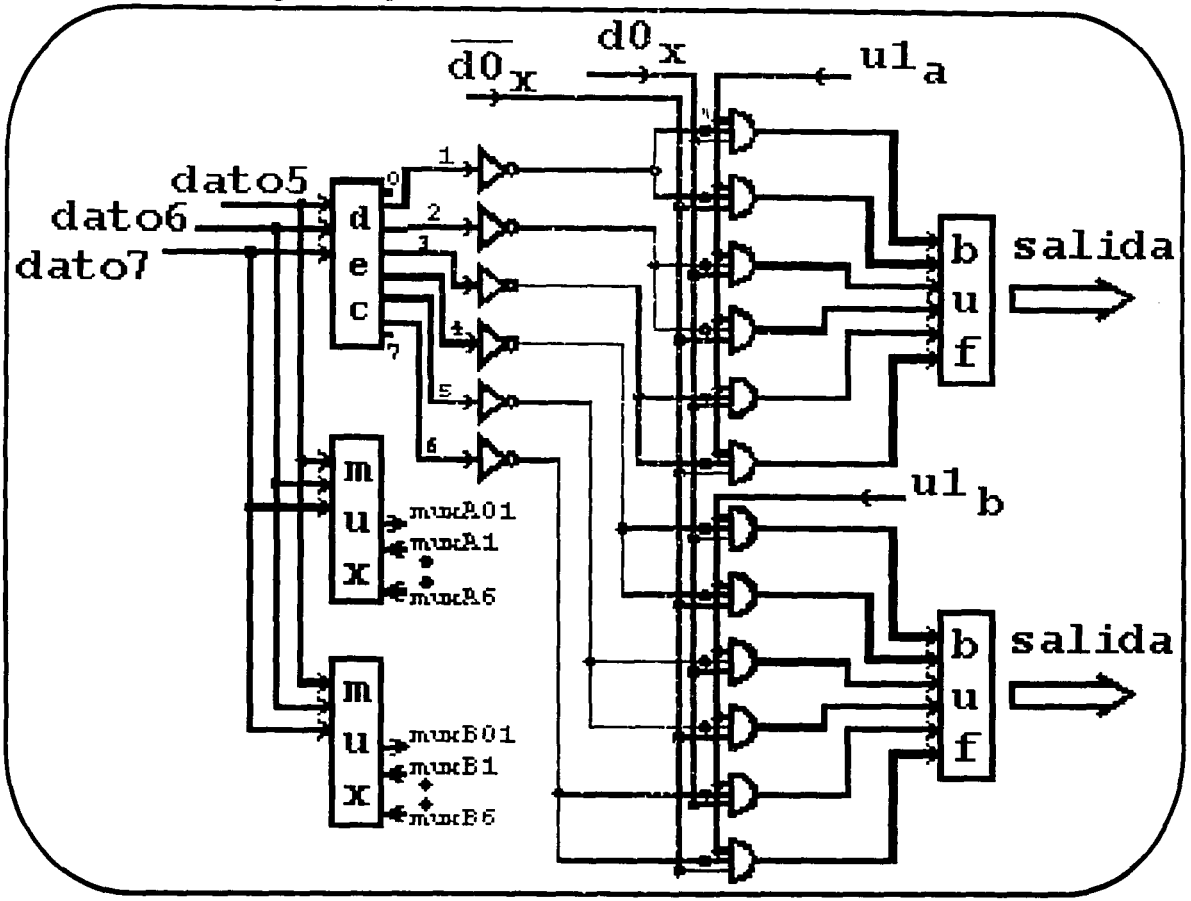


Figura 4.4. Controlador del Brazo de Robot.

La etapa de potencia es la siguiente:

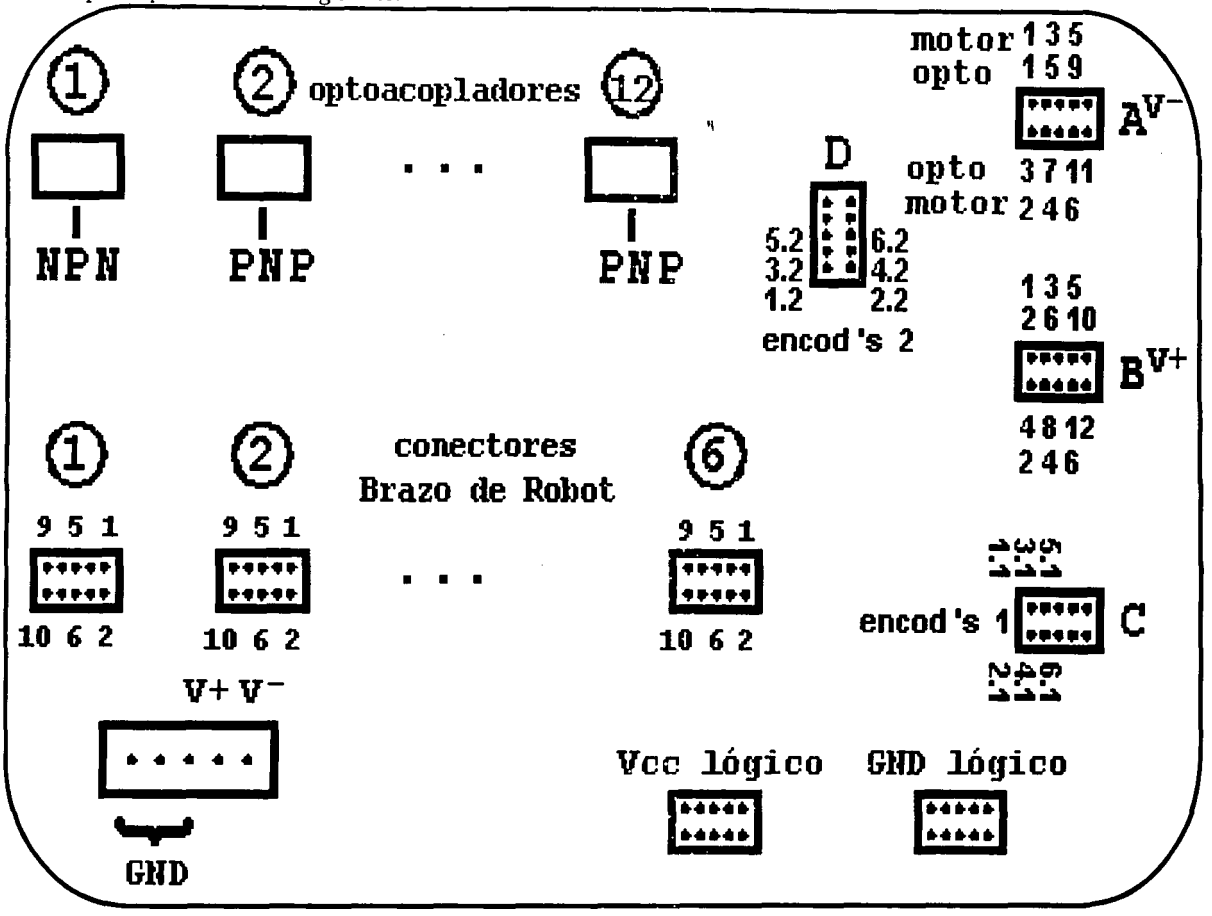


Figura 4.5. Etapa de potencia del controlador del Brazo de Robot.

4.2 Sistema de Control de un motor de corriente directa

El siguiente diagrama muestra un sistema de Control de Motor de C. D. :

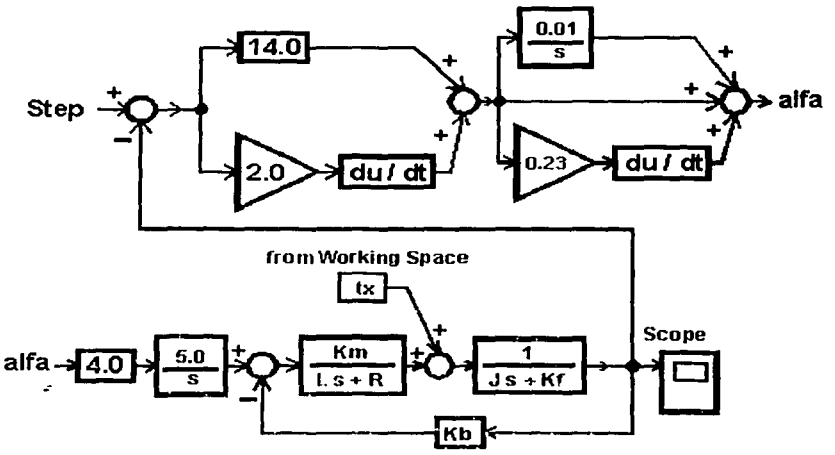


Figura 4.6. Sistema de Control de motor de corriente directa.

El sistema de Control se diseñó inicialmente siguiendo el método de Ziegler y Nichols utilizando el primer método indicado en la referencia [5]. Posteriormente se modificó hasta llegar a la configuración mostrada en la figura 4.6.

El siguiente diagrama presenta la sección PD electrónica del sistema de control.

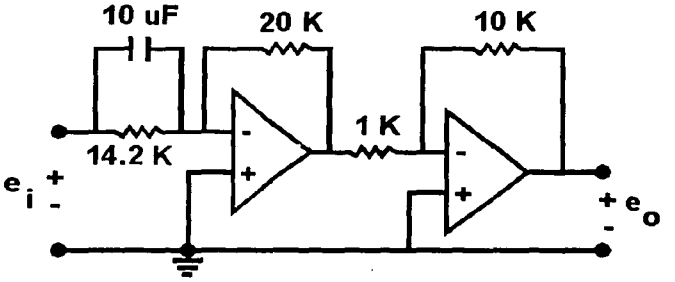


Figura 4.7. Sección PD electrónica del sistema de control.

En el siguiente diagrama se muestra la sección PID electrónica del sistema de control.

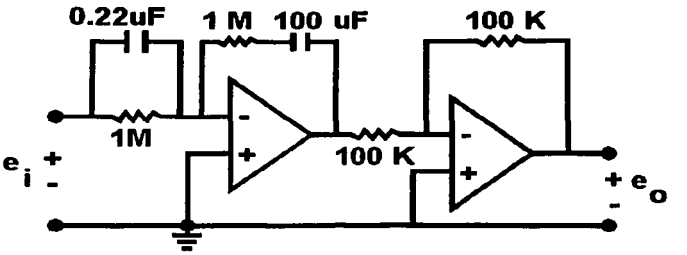


Figura 4.8. Sección PID electrónica del sistema de control.

El sistema de control presentado está basado en el ejemplo del sistema de ayuda acerca del control de Motores de C.D. del programa MATLAB.

5 *Experimentos y Resultados*

5.1 Posicionar el Brazo Mecánico en un punto (x, y, z) en el espacio tridimensional, donde x, y, z, son números reales

El programa "archi60" posiciona al Brazo de Robot en un punto indicado del espacio tridimensional.

El siguiente algoritmo corresponde al programa "archi60".

Programa "archi60"

1. Leer características del sistema:
   1.1. Número del puerto
   1.2. Constante de tiempo
2. Leer posición Home
3. Leer:
   3.1. Número de motor
   3.2. Sentido
   3.3. Posición
   En caso de no haber más datos, continuar en el inciso 11.
4. Revisar datos
5. Enviar datos a través del puerto paralelo
6. Leer encodificadores
7. ¿Es la posición deseada?
   7.1. Si. Continuar en el inciso 8.
   7.2. No. Continuar en el inciso 6.

8. Desactivar motor
9. Esperar terminación por efectos de inercia
10. Actualizar la posición del motor, continuar en el inciso 3.
11. Guardar la posición actual de cada motor en un archivo. Esta es la nueva posición Home.

Al iniciar el programa "archi60" la ventana es la siguiente:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

   Todavia NO ACTIVE EL SISTEMA      >>>>>>>>>>>>>>>>>>>>>>>>

 dar < retorno > ...
r

                              Caracteristicas de sistema
paralelo= 0x378      estado= 0x379      constante de tiempo= 100


Aceptar datos ?      s = si,      otro= no
   < retorno > = si

ESPERE,  NO ACTIVE EL SISTEMA todavia


Ahora active el sistema

   despues dar < retorno >
```

Figura 5.1. Inicio del programa "archi60".

Durante el funcionamiento del programa la ventana del programa es parecida a la siguiente:

```
Ahora active el sistema

    despues dar < retorno >

                                        POSICION HOME
sentido[1]= 0       sentido[2]= 1
sentido[3]= 0       sentido[4]= 0
sentido[5]= 0       sentido[6]= 1
 posicion[1]= 100        posicion[2]= 20
 posicion[3]= 300        posicion[4]= 200
 posicion[5]= 10        posicion[6]= 20
 dato 1
 dato 2
 dato 3
 dato 4
 dato 5
 dato 6
 dato 7
 dato 8
 dato 9
 dato 10
 dato 11
 dato 12
```

Figura 5.2. Programa "archi60" activado.

Al término de la sesión la ventana muestra la nueva posición Home.

```
 dato 2
 dato 3
 dato 4
 dato 5
 dato 6
 dato 7
 dato 8
 dato 9
 dato 10
 dato 11
 dato 12

guardando nueva posicion home, espere un momento ...

                        NUEVA POSICION HOME
sentido[1]= 1       sentido[2]= 0
sentido[3]= 1       sentido[4]= 1
sentido[5]= 1       sentido[6]= 1
 posicion[1]= 110        posicion[2]= 10
 posicion[3]= 20        posicion[4]= 200
 posicion[5]= 100        posicion[6]= 50

final, dar < retorno >...
```

Figura 5.3. Finalización del programa "archi60".

5.2 Posicionar el Brazo Mecánico según las coordenadas (θ_1, θ_2, θ_3, θ_4, θ_5) de los eslabones, donde θ_1, θ_2, θ_3, θ_4, θ_5 son números reales

El programa xyz.exe permite introducir los valores θ_1 hasta θ_5 tanto en forma absoluta con respecto a un plano horizontal como en forma relativa con el eslabón anterior.

La ventana inicial del programa xyz.exe es la siguiente:

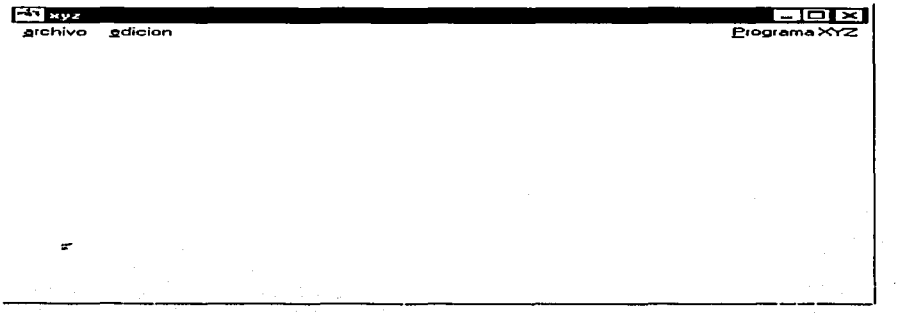


Figura 5.4. Inicio del programa "xyz".

Al entrar al menú edición aparece el siguiente cuadro de mensajes:

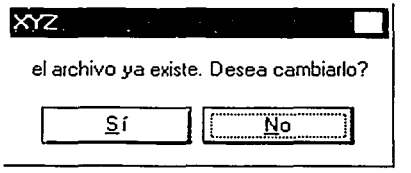


Figura 5.5. Cuadro de mensajes del programa "xyz".

Eligiendo la opción afirmativa aparece la siguiente ventana:

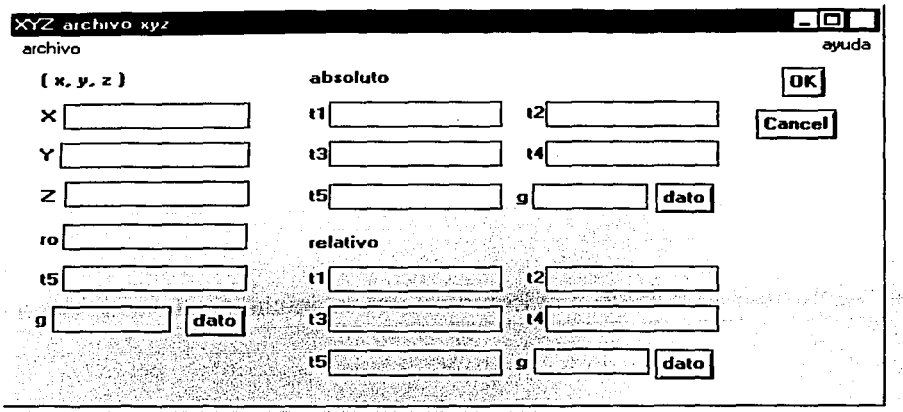


Figura 5.6. Ventana de introducción de datos del programa "xyz".

El menú de ayuda presenta el siguiente cuadro de mensajes:

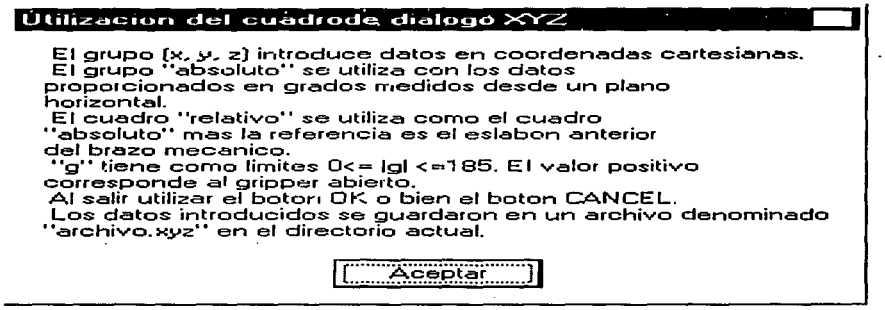


Figura 5.7. Menú de ayuda del programa "xyz".

El programa "xyz.exe" produce datos con formato "XYZ", "ABS", y "REL".

5.3 Indicar al Brazo Mecánico una nueva Posición_Base_Inicial (0, 0, 0) en cualquier momento

Se tiene un archivo especial "c:_datos__{hpos}_.txt" donde se guardan los valores iniciales correspondientes a la Posición_Base_Inicial. Para ello es necesario utilizar el programa "motor.exe" y posteriormente escribir los valores correspondientes en el mismo archivo especial.

5.4 Mover el Brazo Mecánico a la Posición_Base_Inicial (0, 0, 0) en cualquier momento

En este caso es suficiente indicar como siguiente posición a la Posición_Base_Inicial desde el programa de captura de datos "xyz.exe", en la opción de coordenadas cartesianas [x, y, z].

5.5 Posicionar el Brazo Mecánico en una posición cualquiera, accionando directamente los actuadores del Brazo Mecánico, uno a la vez

El siguiente algoritmo permite accionar directamente los motores del Brazo Mecánico. El programa "motor" se utiliza bajo el sistema operativo DOS.

Programa "Motor"

1. Leer número de puerto
2. Leer número de motor y sentido
3. Activar motor a través del puerto paralelo
4. Esperar ⟨ retorno ⟩ del teclado
5. Detener motor a través del puerto paralelo
6. ¿Continuar?

   6.1. Si. Continuar en el inciso 2.

6.2. No. Continuar en el inciso 7.

7. Finalización

La ventana inicial del programa "motor " es la siguiente:

```
TODAVIA    NO    ACTIVE EL SISTEMA

          Dar numero de puerto, ej: 0x378


    Dato obtenido:
numero de puerto: 0x378

Aceptar?  < s >      otro= salir
s
.
Active el sistema luego dar <retorno>

Dar numero de motor < 1..6 >
1
Dar sentido < 0..1 >
0
Datos obtenidos >

                                motor  1            sentido  0

Aceptar datos?  < s >  o  < n >
   < f > = salir
```

Figura 5.8. Inicio del programa "motor".

Al finalizar este programa no almacena la nueva posición Home.

```
Dar sentido < 0..1 >
0
Datos obtenidos >

                                    motor   4           sentido   0

Aceptar datos?  < s >  o  < n >
   < f > = salir
s
Dar  < retorno >  para continuar

Dar numero de motor < 1..6 >
3
Dar sentido < 0..1 >
0
Datos obtenidos >

                                    motor   3           sentido   0

Aceptar datos?  < s >  o  < n >
   < f > = salir
f

 Finalisa, dar  < retorno >
```

Figura 5.9. Finalización del programa "motor".

5.6 Memorizar las posiciones indicadas por el usuario

El siguiente algoritmo permite memorizar las posiciones indicadas por el usuario. El programa se utiliza bajo el sistema operativo DOS.

Programa "Manual"

1. Leer número de puerto y constante de tiempo
2. Leer posición Home
3. Leer número de motor y sentido desde el teclado
4. Enviar datos a través del puerto paralelo
5. Esperar ⟨ retorno ⟩ del teclado
6. Detener motor a través del puerto paralelo
7. Esperar terminación por efectos de inercia

8. Actualizar la posición del motor
9. ¿Memorizar posición?
   9.1. Si. Almacenar posición de motores 1 al 6, en archivo. Continuar en el inciso 10.
   9.2. No. Continuar en el inciso 10.
10. ¿Continuar en el programa?
    10.1. Si. Continuar en el inciso 3.
    10.2. No. Continuar en el inciso 11.
11. Guardar posición actual de cada motor en archivo. Esta es la nueva posición Home

Las ventanas iniciales del programa "manual" son las siguientes:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

  Todavia NO ACTIVE EL SISTEMA     >>>>>>>>>>>>>>>>>>>>>>>>

 dar < retorno > ...

                    Caracteristicas de sistema
paralelo= 0x378      estado= 0x379      constante de tiempo= 100

Aceptar datos ?      s = si,      otro= no
  < retorno > = si

ESPERE, NO ACTIVE EL SISTEMA todavia


Ahora active el sistema

Despues dar < retorno >
```

Figura 5.10. Ventana inicial_1 del programa "manual".

```
Ahora active el sistema

Despues dar < retorno >

                                    POSICION HOME
sentido[1]= 1      sentido[2]= 0
sentido[3]= 1      sentido[4]= 1
sentido[5]= 1      sentido[6]= 1
 posicion[1]= 110       posicion[2]= 10
 posicion[3]= 20        posicion[4]= 200
 posicion[5]= 100       posicion[6]= 50

Dar numero de motor < 1..6 >
3
   Dar sentido < 0..1 >
0
Datos obtenidos >

                              motor  3          sentido  0

Aceptar datos?  < s >  o  < n >
   < f > = salir
s
Dar < caracter > para continuar
```

Figura 5.11. Ventana inicial_2 del programa "manual".

Se debe indicar al programa las posiciones que se deben memorizar. Estas posiciones serán conservadas por el programa en un archivo.

```
s
posicion [ 1 ] ocupada

Dar numero de motor < 1..6 >
4
   Dar sentido < 0..1 >
0
Datos obtenidos >

                              motor  4          sentido  0

Aceptar datos?  < s >  o  < n >
   < f > = salir
s
Dar < caracter > para continuar

     memorizar posicion [ 2 ] ?  < s > = si
  otra letra = continuar
  < f > = salir
```

Figura 5.12. El programa "manual" conserva las posiciones indicadas en un archivo.

Al finalizar el programa muestra la nueva posición Home.

```
   < f > = salir
s
Dar < caracter > para continuar

     memorizar posicion [ 6 ] ?  < s > = si
  otra letra = continuar
  < f > = salir
■

■
f

guardando nueva posicion home, espere un momento ...

                          NUEVA POSICION HOME
sentido[1]= 1      sentido[2]= 0
sentido[3]= 1      sentido[4]= 1
sentido[5]= 1      sentido[6]= 1
 posicion[1]= 110      posicion[2]= 10
 posicion[3]= 20      posicion[4]= 200
 posicion[5]= 100      posicion[6]= 50

final, dar < retorno >...
r
```

Figura 5.13. Finalización del programa "manual".

5.7 Posicionar el Brazo Mecánico en los puntos del espacio previamente indicados por el usuario

El siguiente algoritmo permite posicionar al Brazo de Robot en los puntos indicados previamente por el usuario con el programa "manual.exe". "iterpos.exe" se debe utilizar con el sistema operativo DOS.

Programa "iterpos"

1. Leer número de puerto y constante de tiempo
2. Leer posición Home
3. Leer archivo de posiciones. Una posición en este caso se refiere a la posición de los seis motores. El máximo número de posiciones es de 100. Los datos se revisan antes de almacenarlos en los vectores correspondientes: v_motor[ ], v_sentido[ ], v_posición[ ]

4. Leer secuencia. Los números deben estar en el rango [1..100] correspondiente con la posición de los datos en el archivo de posiciones.

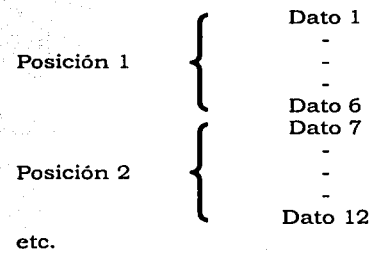


Figura 5.14. Posiciones conservadas en el archivo de posiciones.

Ejemplo de secuencia:

2 4 10 3 4 3 9

el máximo número de números de secuencia es 100, ej. :

3, 9, 4, 9, ... 3

máximo cien números

5. Leer número de veces de repetición de la secuencia.
6. Posicionarse al inicio de la secuencia.
7. Leer siguiente posición desde la secuencia
8. ¿Final de secuencia?

   8.1. Si. Continuar en el inciso 17.

   8.2. No. Continuar en el inciso 9.

9. Leer motor, sentido y posición desde los vectores de datos correspondientes formados con los datos del archivo de posiciones
10. Enviar datos a través del puerto paralelo

11. Leer encodificadores
12. ¿Es la posición deseada?
    - 12.1. Si. Continuar en el inciso 13.
    - 12.2. No. Continuar en el inciso 11.
13. Desactivar motor a través del puerto paralelo
14. Esperar terminación por efectos de inercia
15. Actualizar la posición del motor
16. ¿Se han utilizado los motores 1.. 6?
    - 16.1. Si. Continuar en el inciso 7.
    - 16.2. No. Continuar en el inciso 9.
17. ¿Repeticiones?
    - 17.1. Si. Continuar en el inciso 6.
    - 17.2. No. Continuar en el inciso 18.
18. ¿Recibir otra secuencia?
    - 18.1. Si. Continuar en el inciso 4.
    - 18.2. No. Continuar en el inciso 19.
19. Guardar la posición actual de cada motor, en archivo. Esta es la nueva posición Home.

Las ventanas iniciales del programa "iterpos" son las siguientes.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

   Todavia NO ACTIVE EL SISTEMA        >>>>>>>>>>>>>>>>>>>>>>

  dar < retorno > ...

                                   Caracteristicas de sistema
paralelo= 0x378        estado= 0x379       constante de tiempo= 100

Aceptar datos ?         s = si,       otro= no
   < retorno > = si

ESPERE,  NO ACTIVE EL SISTEMA todavia


Ahora active el sistema

Despues dar < retorno >
```

Figura 5.15. Ventana inicial_1 del programa "iterpos".

```
  s

                                   Caracteristicas de sistema
  paralelo= 0x378       estado= 0x379       constante de tiempo= 100


  Aceptar datos ?        s = si,       otro= no
     < retorno > = si

  ESPERE,  NO ACTIVE EL SISTEMA todavia


  Ahora active el sistema

  Despues dar < retorno >
                                         POSICION HOME
  sentido[1]= 1        sentido[2]= 0
  sentido[3]= 1        sentido[4]= 1
  sentido[5]= 1        sentido[6]= 1
   posicion[1]= 110        posicion[2]= 10
   posicion[3]= 20        posicion[4]= 200
   posicion[5]= 100        posicion[6]= 50
```

Figura 5.16. Ventana inicial_2 del programa "iterpos".

El programa solicita la secuencia de posiciones donde deberá ubicarse al Brazo de Robot.

```
ESPERE,  NO ACTIVE EL SISTEMA todavia


Ahora active el sistema

Despues dar < retorno >

                                        POSICION HOME
sentido[1]= 1       sentido[2]= 0
sentido[3]= 1       sentido[4]= 1
sentido[5]= 1       sentido[6]= 1
 posicion[1]= 110        posicion[2]= 10
 posicion[3]= 20       posicion[4]= 200
 posicion[5]= 100        posicion[6]= 50

Dar secuencia, numeros: < 1..100 >
     maximo cien numeros,  salir con < caracter_letra >
100 9

indice maximo= 5
   dar indice
3 4 3 2 4 2 3 2 1 a
```

Figura 5.17. Introducción de la secuencia de posiciones al programa "iterpos".

Después el programa solicita el número de repeticiones para la secuencia proporcionada.

```
secuencia obtenida:
3 4 3 2 4 2 3 2 1
    Aceptar secuencia? < s >= si     otro= no
            < f > = salir
s

Dar numero de repeticiones, valor absoluto < 30100

   num. negativo  = infinito
1
numero de repeticiones obtenido:
      repeticiones= 1

   Aceptar numero de repeticiones?
  < s >= si     otro= no
           < f > = salir
s

Dar < caracter_letra > para terminar repeticiones
  nodo 1
  nodo 2
```

Figura 5.18. Programa "iterpos" accionando al Brazo de Robot.

Al terminar de accionar el Brazo de Robot según la secuencia proporcionada, el programa solicita otras secuencias.

```
    num. negativo   = infinito
1
numero de repeticiones obtenido:
      repeticiones= 1

    Aceptar numero de repeticiones?
  < s >= si      otro= no
            < f > = salir
s

Dar < caracter_letra > para terminar repeticiones
   nodo 1
   nodo 2
   nodo 3
   nodo 4
   nodo 5
   nodo 6
   nodo 7
   nodo 8
   nodo 9
 rep   1
    s
    Obtener otra secuencia? < caracter_letra > = si
           < f > = salir
```

Figura 5.19. Programa "iterpos" solicitando otra secuencia.

Al finalizar, el programa "iterpos" muestra la nueva posición Home.

```
   nodo 4
   nodo 5
   nodo 6
   nodo 7
   nodo 8
   nodo 9
 rep   1

    Obtener otra secuencia? < caracter_letra > = si
           < f > = salir
f

guardando nueva posicion home, espere un momento ...

                    NUEVA POSICION HOME
sentido[1]= 1      sentido[2]= 0
sentido[3]= 1      sentido[4]= 1
sentido[5]= 1      sentido[6]= 1
 posicion[1]= 110       posicion[2]= 10
 posicion[3]= 20       posicion[4]= 200
 posicion[5]= 100       posicion[6]= 50

final, dar < retorno >...
```

Figura 5.20. Finalización del programa "iterpos".

5.8 Trazar líneas en el espacio a través de puntos próximos entre sí

El siguiente algoritmo calcula los puntos próximos pertenecientes a una línea indicada por el usuario. Se utiliza un método vectorial. Este programa funciona bajo el sistema operativo Windows. Posteriormente se utilizaría el programa "datoposc.exe" también bajo Windows para obtener los datos necesarios con el fin de accionar los motores del sistema Rhino. El sistema Rhino se utiliza bajo el sistema operativo DOS.

Programa "linea"

1. Leer $\overline{px}$ [ ], $\overline{ux}$ [ ], tL, ρ, t5, gripper
2. ¿Se tienen más datos?
   2.1. Si. Continuar en el inciso 3.
   2.2. No. Finalización.
3. Los valores obtenidos deben cumplir con las siguientes restricciones

   $$tL \geq 1.0, \qquad |\overline{ux}| \geq 1.0$$

   Utilizar la notación de la siguiente figura:

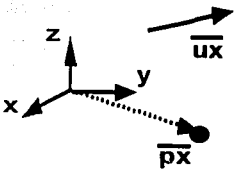


Figura 5.21. Vectores utilizados en el programa "linea".

4. Calcular puntos con la siguiente fórmula, variando el parámetro "t" según el intervalo t: 0..tL

$$\overline{p} = \overline{px} + t\frac{\overline{ux}}{|\overline{ux}|}$$

Ecuación 5.1

5. Para cada punto $\overline{p}$ calculado, almacenar en archivo con formato XYZ, los siguientes datos:

   5.1. Las coordenadas x, y, z del vector $\overline{p}$

   5.2. ρ, t5, gripper

6. Continuar en el inciso uno.

La ventana del programa "línea " es la siguiente.

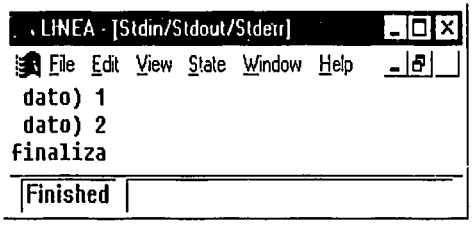


Figura 5.22. Programa "linea" activado.

Al finalizar se muestra un resumen de los datos leídos.

```
   datos obtenidos:
 px: 10.000000 10.000000 10.000000

 ux: 1.000000 2.000000 3.000000

 tL: 3.000000
ro= 40.000000  xyz5= 10.000000  xyzg= 90.000000

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   datos obtenidos:
 px: 15.000000 15.000000 15.000000

 ux: 3.000000 4.000000 2.000000

 tL: 4.000000
ro= -40.000000  xyz5= -110.000000  xyzg= -100.000000

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Figura 5.23. Resumen mostrado por el programa "linea".

También el programa presenta los resultados obtenidos, la ventana correspondiente es la siguiente. Estos datos se conservan en un archivo con formato XYZ.

```
XYZ 10.000000000 10.000000000 10.000000 40.000000 10.000000 90.000000
XYZ 10.267261242 10.534522484 10.801784 40.000000 10.000000 90.000000
XYZ 10.534522484 11.069044968 11.603567 40.000000 10.000000 90.000000
XYZ 10.801783726 11.603567451 12.405351 40.000000 10.000000 90.000000

XYZ 15.000000000 15.000000000 15.000000 -40.000000 -110.000000 -100.0
XYZ 15.557086015 15.742781353 15.371391 -40.000000 -110.000000 -100.0
XYZ 16.114172029 16.485562705 15.742781 -40.000000 -110.000000 -100.0
XYZ 16.671258044 17.228344058 16.114172 -40.000000 -110.000000 -100.0
XYZ 17.228344058 17.971125411 16.485563 -40.000000 -110.000000 -100.0
```

Figura 5.24. Los resultados del programa "linea" se conservan en un archivo.

5.9 Realizar Barridos en el espacio

Un barrido es el trazo de líneas paralelas, las cuales están igualmente separadas y se encuentran en el mismo plano.

El siguiente algoritmo permite realizar barridos después de solicitar los datos necesarios por parte del usuario. Se ejecuta bajo Windows, utilizando un método vectorial.

Programa "barrido"

1. Leer $\overline{N}$ [ ], $\overline{px}$ [ ], $\overline{ux}$ [ ], tL, sL, ρ, t5, gripper
2. ¿Se tienen más datos?
   2.1. Si. Continuar en el inciso 3.
   2.2. No. Finalización.
3. Los valores obtenidos deben cumplir con las siguientes restricciones
   3.1. [tL] ≥ 1.0, [sL] ≥ 1.0
   3.2. [$\overline{N}$] ≥ 1.0, [$\overline{ux}$] ≥ 1.0

La siguiente figura muestra las interrelaciones entre los datos:

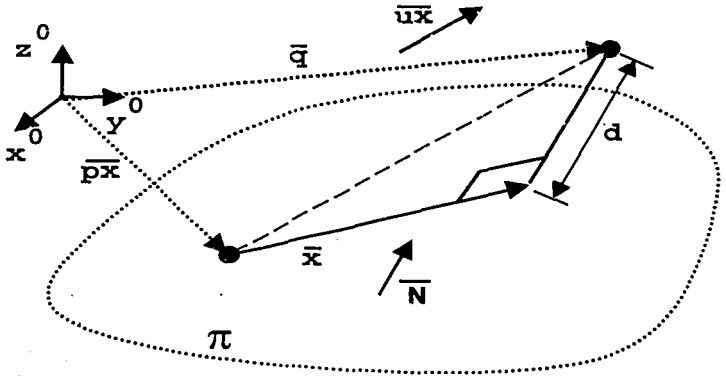


Figura 5.25. Vectores utilizados en el programa "barrido".

Se obtienen las siguientes ecuaciones:

$$\bar{q} - \overline{px} = \frac{\overline{ux}}{|\overline{ux}|}$$

Ecuación 5.2

$$d = \frac{|\overline{ux} \bullet \bar{N}|}{|\overline{ux}||\bar{N}|}$$

Ecuación 5.3

$$\bar{x} = \bar{q} - \overline{px} - d\frac{\bar{N}}{|\bar{N}|}$$

Ecuación 5.4

$$\bar{x} = \frac{\overline{ux}}{|\overline{ux}|} - \left(\frac{|\overline{ux} \bullet \bar{N}|}{|\overline{ux}||\bar{N}|}\right)\frac{\bar{N}}{|\bar{N}|}$$

Ecuación 5.5

$$\bar{x}_u = \frac{\bar{x}}{|\bar{x}|}$$

Ecuación 5.6

La ecuación de donde se obtienen los puntos de cada recta durante el barrido es la siguiente:

$$\bar{L} = \left(\overline{px} + t\ \bar{x}_u \right) + s \left(\frac{\bar{N} \times \bar{x}_u}{|\bar{N} \times \bar{x}_u|} \right)$$

Ecuación 5.7

Cada vector $\overline{L}$, pertenece al conjunto de vectores del barrido.

4. Calcular puntos de cada recta $\overline{L}$ sucesivamente variando los parámetros "t" y "s" de acuerdo con los siguientes intervalos:

   4.1. t = 0 .. tL

   4.2. s = 0 .. sL

Las iteraciones se realizan en dos lazos de programación, uno interno y otro externo. El lazo interno corresponde con la variable "t", y el lazo externo corresponde con la variable "s".

5. Para cada punto calculado de las rectas $\overline{L}$, almacenar en archivo, con formato XYZ los siguientes datos:

   5.1. Las coordenadas x, y, z del vector $\overline{L}$

   5.2. ρ, t5, gripper

6. Continuar en el inciso uno.

La ventana correspondiente al programa "barrido" es la siguiente.

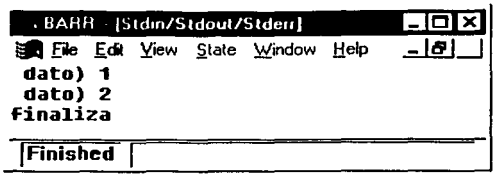


Figura 5.26. Programa "barrido" activado.

Al finalizar, el programa "barrido" muestra una ventana con los datos utilizados y también una ventana con los resultados obtenidos. Los resultados se conservan en un archivo.

```
   datos obtenidos:
 ene: 1.000000 2.000000 3.000000

 px: 15.000000 16.000000 17.000000

 ux: 3.000000 4.000000 4.000000

 tL: 3.000000  sL= -4.000000
ro= 50.000000  xyzS= -40.000000  xyzg= -100.00000

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   datos obtenidos:
```

Figura 5.27. Al finalizar, el programa "barrido" muestra los datos utilizados.

Los resultados obtenidos se conservan en un archivo con formato XYZ:

```
XYZ 15.000000000 16.000000000 17.000000 50.000000 -40.000000 -100.00
XYZ 15.596284794 15.254644088 17.298142 50.000000 -40.000000 -100.00
XYZ 16.192569588 14.509288015 17.596285 50.000000 -40.000000 -100.00
XYZ 16.788854382 13.763932023 17.894427 50.000000 -40.000000 -100.00
XYZ 17.385139176 13.018576030 18.192570 50.000000 -40.000000 -100.00
XYZ 15.756978119 16.398409536 16.482068 50.000000 -40.000000 -100.00
XYZ 16.353262913 15.653053544 16.780210 50.000000 -40.000000 -100.00
XYZ 16.949547707 14.907697551 17.078352 50.000000 -40.000000 -100.00
XYZ 17.545832501 14.162341559 17.376495 50.000000 -40.000000 -100.00
XYZ 18.142117295 13.416985566 17.674637 50.000000 -40.000000 -100.00
XYZ 16.513956238 16.796819073 15.964135 50.000000 -40.000000 -100.00
XYZ 17.110241032 16.051463080 16.262278 50.000000 -40.000000 -100.00
XYZ 17.706525826 15.306107088 16.560420 50.000000 -40.000000 -100.00
XYZ 18.302810620 14.560751095 16.858562 50.000000 -40.000000 -100.00
XYZ 18.899095414 13.815395103 17.156705 50.000000 -40.000000 -100.00
XYZ 17.270934358 17.195228609 15.446203 50.000000 -40.000000 -100.00
XYZ 17.867219152 16.449872617 15.744345 50.000000 -40.000000 -100.00
XYZ 18.463503946 15.704516624 16.042488 50.000000 -40.000000 -100.00
```

Figura 5.28. Los resultados obtenidos del programa "barrido" se conservan en un archivo.

5.10 Trazar curvas en el espacio dando como entrada los puntos sobre la curva

De manera similar al programa "linea.exe", en el caso de trazado de curvas el archivo de salida debe tener formato XYZ. Aunque también es posible incluir datos de formato ABS ó REL como en el caso de la captura de datos con el programa "xyz.exe". El siguiente fragmento de código permite el trazado de una hélice.

```
const int cthel= 10;

const int caso7= 5;

r= 4.0;

h= 0.0;

kaso7= 18.0;

for( ht= 0; ht < 188; ht += 1.0 )

ku= ht/cthel;

vu= fmod(ku, 3.1416);

p1[0]= r * cos(vu) + h;
```

```
p1[1]= r * sin(vu) + kaso7;

p1[2]= caso7 * (ku/3.1416) - 5;
```

las componentes del vector p1 serían conservadas en el archivo de salida con formato XYZ en este caso. El fragmento de código debe ser parte de un programa fuente.

5.11 Realizar acciones de manera repetitiva

El programa "iterpos.exe" permite posicionarse de manera repetitiva en los puntos memorizados indicados por el usuario.

5.12 Gráficas obtenidas con la simulación del sistema de control de motor de C. D.

La respuesta del sistema de control al paso unitario es la siguiente:

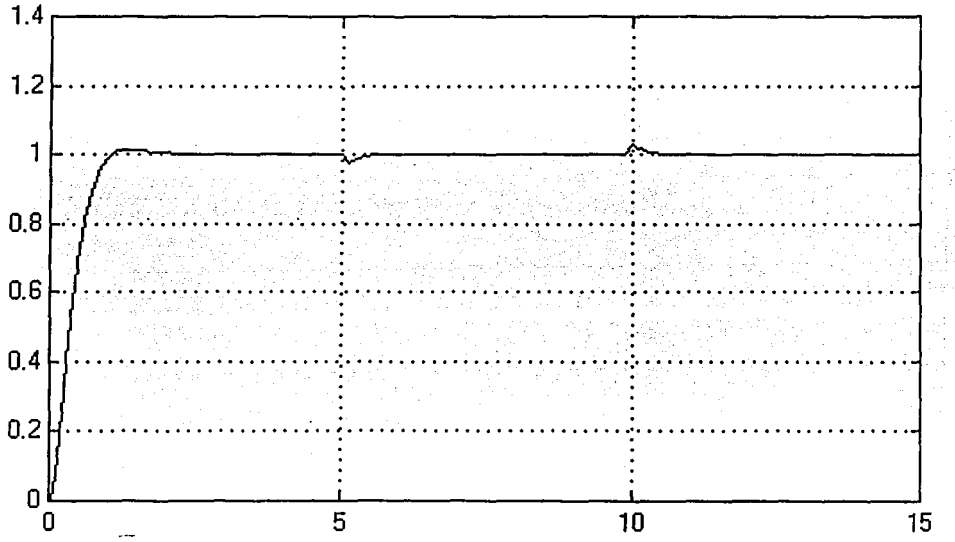


Figura 5.29. Respuesta en la salida SCOPE del sistema de control de motor de C. D.

En la figura 5.29, el eje horizontal representa el tiempo en milisegundos.

La perturbación "tx" es nula excepto en el intervalo de 5 a 10 milisegundos donde tiene un valor de —0.1, según se muestra en la siguiente figura.

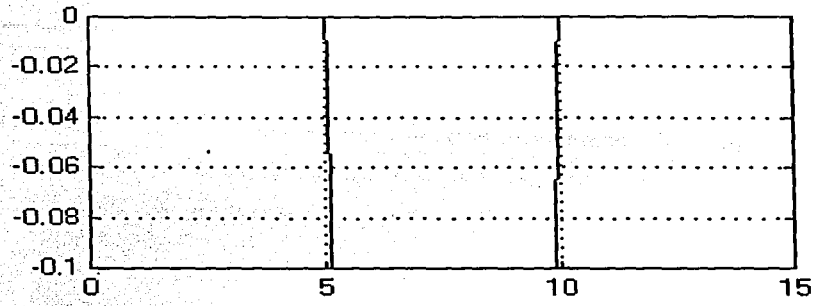


Figura 5.30. Perturbación "tx" utilizada en el sistema de control de motor de C. D.

Algunas zonas de interés de la respuesta del sistema se muestran a continuación.

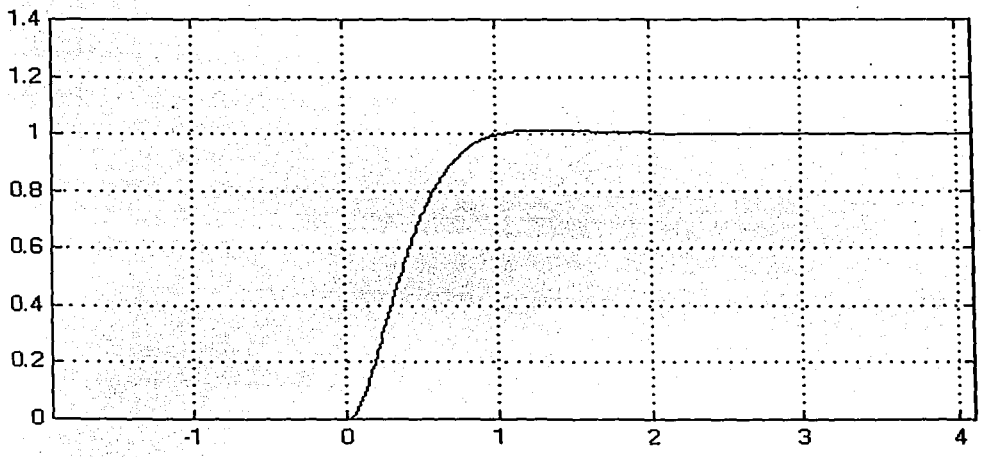


Figura 5.31. Zona inicial de la respuesta al paso unitario del sistema de control.

A continuación se muestra la respuesta del sistema alrededor del milisegundo cinco:

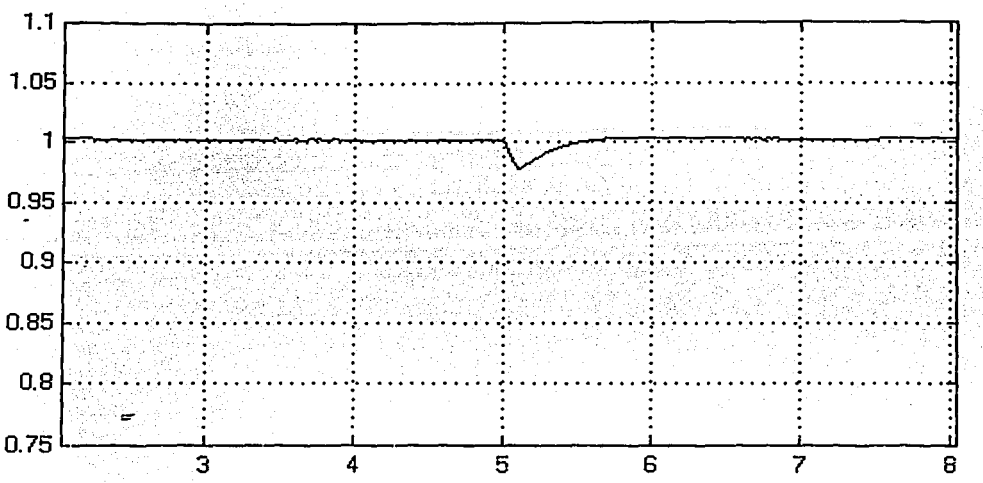


Figura 5.32. Respuesta alrededor del milisegundo cinco del sistema de control.

A continuación se muestra la respuesta del sistema de control alrededor del milisegundo diez:

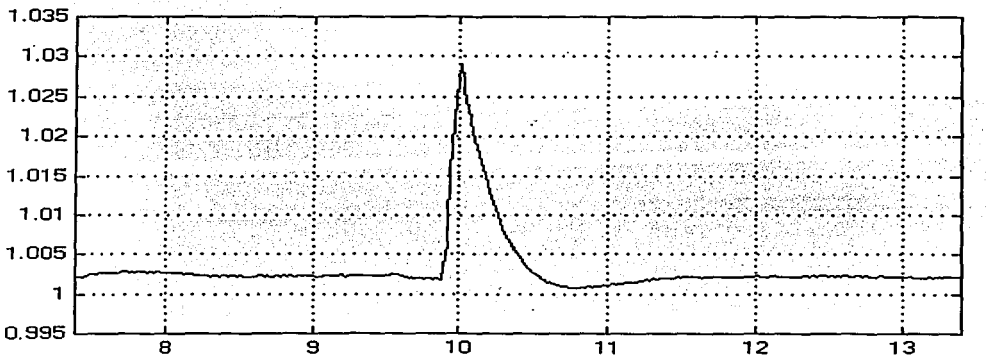


Figura 5.33. Respuesta alrededor del milisegundo diez del sistema de control.

6 *Conclusiones*

El desarrollo de la Robótica depende de otras disciplinas como la Inteligencia Artificial, la Electrónica, etc. La utilización de robots industriales, en actividades anteriormente exclusivas del ser humano ó de otras máquinas, tiene diversas finalidades una de las cuales es mejorar la calidad en la producción.

Por ejemplo: el colocar, pegar, soldar, etc. una pieza mecánica ó electrónica en una determinada ubicación, de manera muy precisa, justifica la utilización de robots en vez de utilizar la mano de obra humana.

Ahora existe un nuevo "ente" artificial creado por el hombre:

EL ROBOT.

Solo el hombre, el tiempo y el futuro decidirán qué hacer con este nuevo ente en nuestro planeta. Las nuevas situaciones creadas por la presencia de robots cambian el comportamiento del ser humano. A este respecto es necesario modificar algunas ideas por ejemplo el creer que en este planeta Tierra no habría nunca una máquina que pudiera desplazarse con ayuda de las fuentes de energía adecuadas, de manera parecida al ser humano.

El diseño y fabricación de un robot se puede considerar como un arte pues es posible admirar el funcionamiento del mismo.

El objetivo de esta tesis fue el control de un Brazo Mecánico, específicamente el control del Brazo de Robot Rhino. El control se efectuó transmitiendo las posiciones indicadas por el usuario, utilizando àrchivos los cuales contenían las posiciones.

Los archivos se transmitían hasta el lugar donde se encontraba el Brazo de Robot. Los datos se enviaron a través de Internet utilizando conectores software según el protocolo TCP/IP de comunicaciones. Se eligió el sistema operativo Windows por ser uno de los sistemas operativos más utilizados en las

computadoras personales. Las PC's empleadas incluían procesadores modelo 486, conectadas a través de Internet.

Se alcanzaron los objetivos propuestos para el desarrollo de esta tesis, con las siguientes observaciones:

1. En el tratamiento de la Cinemática Directa la matriz T_5^0 permitió obtener la posición espacial (x, y, z) del Brazo de Robot. En el caso de la Cinemática Inversa se dieron dos opciones para elegir la posición del Manipulador las cuales dependen de la orientación espacial del eslabón de longitud B en la Figura 2.1 del Capítulo 2.
2. En el caso de los conectores software, se utilizó el lenguaje Visual C++ con el sistema operativo Windows sin manejo de mensajes, pues la comunicación es breve durante la transmisión de los archivos. La interfaz de usuario para capturar los datos del archivo a transmitir se diseñó utilizando Visual C++.
3. El sistema pudo dar atención a más de un usuario tan solo limitado por el tiempo de transmisión de los datos y archivos a través del sistema Internet. Se dio atención a solo un usuario a la vez durante la transmisión de archivos pues no se utilizaron cuerdas para dar atención a más de un usuario al mismo tiempo. En el sistema de software diseñado no se utilizaron cuerdas.
4. Los sistemas electrónicos se construyeron utilizando componentes los cuales se pueden conseguir en las tiendas de componentes electrónicos de la Ciudad de México, no fue necesario importar ningún componente utilizado.
5. El sistema de control de motor de corriente directa no se implementó físicamente. Fue simulado utilizando MATLAB. Los sistemas electrónicos de este sistema de control de motor de C.D. simulado, cuyos diagramas aparecen en las figuras 4.7 y 4.8 del capítulo 4, representan una posible implementación basada en la simulación del sistema.

Referencias

[1] Computer Networks

Third Edition

Andrew S. Tanenbaum

Prentice Hall, 1996

[2] Internetworking with TCP/IP

Client_Server Programming and Applications

Windows Socket Version

Volume III

Douglas E. Comer and David L. Stevens

Prentice Hall, 1997

[3] Introduction to Robotics

Mechanics and Control

Second Edition

John J. Craig

Addison Wesley, 1989

[4] Robotics

Control, Sensing, Vision, and Intelligence

K. S. Fu, R. C. González

C. S. G. Lee

Mc Graw Hill, 1987

[5] Modern Control Engineering

Third Edition

Katsuhiko Ogata

University of Minnesota

Prentice Hall, 1997

[6] Feedback Control System Analysis & Synthesis

John J. D'Azzo, Constantine H. Houpis

Second Edition

Mc Graw Hill, 1996

[7] An Engineering Approach to Digital Design

William I. Fletcher

Prentice Hall, 1980

[8] Microelectronics

Second Edition

Jacob Millman, Arvin Grabel

Mc Graw Hill, 1998

[9] Visual C++ Microsoft ®

Aplicaciones para Windows

Francisco Javier Ceballos

Computec, RA-MA,

Alfaomega Grupo Editor S.A. de C.V. 1997

[10] Enciclopedia del Lenguaje C

Francisco Javier Ceballos

Computec, RA-MA,

Alfaomega Grupo Editor S.A. de C.V. 1997

[11] Programming Windows

Second Edition

Charles Petzold

Microsoft Press ™

[12] The Student Edition of MATLAB

The Ultimate Computing Environment for Thecnical Education

User's Guide

Version 4

Prentice Hall

The Math Works Inc. 1995