24

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

MOVIMIENTO DE UN ROBOT
TELEOPERADO A TRAVÉS DE SISTEMAS
DE LIBRE DISTRIBUCIÓN POR INTERNET

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECÁNICO ELECTRICISTA

P R E S E N T A :

Carlos Enrique Contreras Vara.

DIRECTOR
M.I. Víctor González Villela

CODIRECTOR DE TESIS
Dr. Jesús Manuel Dorador González

Cd. Universitaria,

México, D.F. 2002

PAGINACIÓN DISCONTINUA

AGRADECIMIENTOS

Quiero agradecer a Dios por todas y cada una de las bendiciones que me ha otorgado a lo largo de la vida y que sin Él ésta tesis no habría sido posible.

Al pueblo de México por que gracias a sus esfuerzos la educación gratuita en nuestro país es posible y con ello éste proyecto.

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería.

Agradezco a todos mis maestros por sus enseñanzas y experiencias en especial a aquellos que dejaron huella con su ejemplo, a mi director M.I. Victor González Villela por dirigir este proyecto tan importante para mi.

A mi codirector Dr. Jesús Manuel Dorador, al M.I. Álvaro Ayala Ruíz y al Ing. Noé Cruz Marín, Ignacio Juárez Campos, Eduardo Garduño por sus consejos y ayuda. A los miembros y amigos de la Unidad de Cómputo Académico que me dieron la oportunidad de crecer y aprender.

A mis compañeros de clase y generación que me apoyaron en el trayecto.

A mis amigos Ernesto Paredes, Dr. Eliseo Lima, Gerardo García, Luis García por su amistad y apoyo incondicional.

DEDICATORIA

Quiero dedicar esta tesis primeramente a Dios.

Quiero dedicársela también a la persona que ha sido un incentivo para ser un mejor hombre día a día, un mejor ingeniero, un mejor amigo, una mejor persona, pero sobre todo a ser un mejor hijo y a creer en Dios, para ti mi vida dedico este esfuerzo, gracias por tu ayuda, tu apoyo, tus energías, tu tiempo, tu alegría tu presencia, gracias por ser el amor de mi vida y estar en todo momento a mi lado, hombro con hombro en las buenas y en las malas, gracias Laura.

A ti mamá por estar siempre apoyándome en todos los aspectos de mi vida, has sido y serás siempre una parte de mi, una parte muy importante de mi ser, te quiero mucho mamá y quiero que te sientas orgullosa de este esfuerzo ya que ésta tesis también es reflejo de ti, por que compartiste conmigo desvelos, angustias y alegrías. Gracias papá por estar ahí cuando te he necesitado, ésta tesis también es logro tuyo, te quiero papá. Gracias a mis padres por darme libertad de elección, amor y comprensión.

A mi hermano, que estoy seguro que pronto alcanzará su meta, gracias por estar ahí siempre, te quiero hermano, te respeto y te admiro.

A aquellas personas que de alguna manera o en algún momento fueron incentivo, dieron un consejo sabio o estuvieron apoyándome, que su experiencia o su presencia que han hecho que hoy sea lo que soy. Con todo respeto a mi abuelita Ma. del Refugio Alvarado, al Sr. Alfredo Peña, Sra. Olga Sánchez, Rosa Elena y Alfredo Peña, a mis tíos: Carlos Vara, César Vara, a mis tías July y Almadelia, al Sr. Juan Manuel Ramírez.

A mis abuelitos, que descansen en paz gracias: Carlos Vara Cuevas, Enrique Contreras Campos, Carmen Suárez, su vida y sus consejos dejaron huella en mi corazón.

Lo imposible es el fantasma de los tímidos y el refugio de los cobardes.

Napoleón Bonaparte.

Es detestable esa avaricia espiritual que tienen los que sabiendo algo, no procuran la transmisión de esos conocimientos.

Miguel de Unamuno.

ÍNDICE

Prólogo.

Este trabajo surge de la necesidad de operar, de manera remota, una o varias máquinas automáticas comerciales o algunas desarrolladas en proyectos de investigación. El uso de robots operados por la Internet es una práctica que se ha comenzado a extender por todo el mundo debido a los beneficios que otorga el poder manipular, desde un lugar remoto a otro y sin trasladarse grandes distancias, objetos, herramientas o cualquier otro dispositivo.

Actualmente se vive en la era de la Internet, contar con una línea telefónica, un módem o enlace dedicado, resulta indispensable para poder acceder a los recursos de la red.

No hay una definición que pueda resumir la Internet; hay varios puntos de vista para entenderlo. Por un lado, la Internet podría ser concebido en relación a sus protocolos comunes, como una colección de circuitos y rutinas, como un conjunto de recursos compartidos o incluso como una disposición a intercomunicarse; es decir, como una mega red, una red de redes de computadoras. Sin embargo, otro enfoque, que parece más adecuado, es pensar en las redes como el medio a través del cual se envía y acumula información.

La Internet o el Internet o Internet, como es llamada, es tanto un conjunto de comunidades como un conjunto de tecnologías, y su éxito se puede atribuir a la satisfacción de las necesidades básicas de la comunidad y a la utilización de ésta de un modo efectivo para impulsar la infraestructura. Es a la vez una oportunidad de difusión mundial, un mecanismo de propagación de la información y un medio de colaboración e interacción entre los individuos y sus computadoras, independientemente de su localización geográfica.

El número total de usuarios de la Internet asciende hoy en día a varios millones, y su crecimiento es exponencial. Este alto nivel de conectividad ha creado un grado de comunicación, colaboración, acceso a la información e intercambio de recursos sin precedentes en la historia de la humanidad.

Este trabajo pretende establecer una base para proyectos futuros, aprovechando los recursos que la Internet, los sistemas operativos libres y la infraestructura de la Facultad de Ingeniería ofrece para los alumnos y las industrias que requieran dispositivos controlados remotamente.

INTRODUCCIÓN

La Internet comienza a finales de 1960 como un experimento para diseñar redes de computadoras robustas. La meta era construir una red de computadoras que pudiera mantenerse funcionando aún con la pérdida de muchas máquinas sin que el resto de ellas perdieran la habilidad de comunicarse. El financiamiento de este proyecto vino del Departamento de la Defensa de los Estados Unidos, el cuál tuvo el interés de construir redes de información que pudieran soportar un ataque nuclear.

El resultado fué un éxito técnico asombroso, pero era limitado en tamaño y alcance. En gran parte, sólo contratistas de defensa e instituciones educativas podían adquirir acceso a lo que fue conocido entonces como DARPAnet (Red de Proyectos de Investigación Avanzados del Departamento de Defensa por sus siglas en inglés).

Con la llegada de modems de alta velocidad para la comunicación digital sobre líneas telefónicas comunes, algunos individuos y organizaciones comenzaron a conectarse y tomar ventaja de las comunicaciones avanzadas y globales. No obstante, no fué sino alrededor de 1993 en que la Internet realmente despegó como tal.

Varios eventos fueron cruciales para encaminar el alza meteórica de popularidad de la Internet. Primero a principios de 1990, empresas e individuos, ansiosos por tomar ventaja del poder de las comunicaciones digitales globales finalmente presionaron a al gobierno, fundador de la Internet, para abrir sus sistemas al tráfico de grandes redes de computadoras sin restricción. Cabe mencionar que la red no fue diseñada para ser una vía de información en base al contenido, es decir, no era posible ver el contenido de los documentos o las imágenes o archivos que se compartían por la red hasta que se transmitían hacia la computadora del usuario.

Fieles a su tradición de libre intercambio y coparticipación, muchos de los miembros de Internet, que en su mayoría eran académicos, compartían documentos de texto con temas educativos, pero la búsqueda de esa información en Internet era difícil de encontrar. Muy poca gente tenía el conocimiento o el interés de aprender cómo usar el software arcaico o tenía el tiempo para gastarlo en buscar a través de todos los documentos buscando alguno de interés.

Así que se necesitaba otra chispa a la luz del cohete de Internet. Casi al mismo tiempo que la Internet se abría para los negocios, algunos físicos en CERN, el Laboratorio de Física de Partículas Europeo, lanzaron al público un lenguaje y sistema de distribución que desarrollaron para crear y compartir documentos electrónicos con multimedia por medio de la Internet, así nacen el Hypertext Markup Language o HTML, el software navegador y el World Wide Web. Nunca más un autor tendría que distribuir su trabajo como colecciones de imágenes, sonidos y textos. HTML unificó esos elementos. Además, los sistemas WWW permitieron el hypertext lynking o conexión de hipertexto, con el cual los documentos automáticamente hacen referencia a otros documentos localizados en cualquier parte del mundo, lo que lo hace menos rebuscado y más productivo.

El despegue sucedió cuando algunos estudiantes de facultad en la NCSA el Centro Nacional de Aplicaciones de Supercómputo, en la Universidad de Ilinois, Urbana-Champaign escribieron un programa navegador de web llamado Mosaic. A pesar de que fue diseñado para ver documentos HTML, el software también tenía herramientas para acceder a muchos recursos prolíficos en la Internet, como archivos de software FTP y colecciones organizadas de documentos como Gopher. Con versiones basadas en interfaces gráficas fáciles de utilizar, que son familiares para la mayoría de quienes poseen una computadora, Mosaic se convierte en un éxito instantáneo. Estaba disponible, como la mayor parte del software de Internet, gratis. Millones de usuarios adquirieron una copia y comenzaron a navegar por la Internet buscando páginas interesantes.

Desde entonces el desarrollo de la Internet ha sido impresionante, no sólo se pueden compartir sonido, imágenes y documentos, ahora es posible ver videos, realidad virtual, discutir temas en vivo con otras personas en el mundo entero, se puede hacer una reservación en un vuelo, comprar flores, ver televisión e inclusive ver conciertos y eventos en vivo. Hasta ahora, apenas se ha comenzado a explotar todo el potencial de Internet.

Una de las aplicaciones de la Internet que se ha comenzado a investigar es el desarrollo del manejo de robots utilizando como medio de comunicación la Internet y es la base de esta tesis que tratará de demostrar que es posible utilizar herramientas de libre distribución para la implementación de un sistema que permitirá el movimiento de un robot de manera remota.

En el capítulo 1 se dan los conceptos básicos para entender el desarrollo y se muestran algunos robots controlados actualmente por Internet.

En el capítulo 2 se describen y justifica el uso de los sistemas de libre distribución para el desarrollo de esta tecnología.

En el capítulo 3 se describe el sistema, la plataforma usada y las partes mecánicas y eléctricas del robot utilizado para la demostración.

En el capítulo 4 se registran las pruebas físicas realizadas sobre el robot y los elementos que demuestran que la tesis es factible.

Finalmente se menciona, en las conclusiones algunas de las aplicaciones que este tipo de proyectos tendrían en la docencia así como en la industria y qué proyectos a futuro se pueden derivar de este trabajo para beneficio de la Facultad. Además se proponen proyectos que se podrían implementar para el sistema de manufactura flexible de la Facultad.

Objetivo general de la Tesis.

Mover un robot a través de la Internet.

Objetivos particulares.

- Utilizar plataformas libres para el desarrollo de los sistemas de control.
- Utilizar un robot SCORBOT existente en el laboratorio de Sistemas de Manufactura Flexible en la Facultad de Ingeniería de la UNAM para el movimiento remoto por medio de conexiones Internet, con un sistema desarrollado en plataforma y lenguajes de libre distribución.

Hipótesis.

1. Es posible, mediante el uso de una computadora, conectada a la Internet y auxiliada por los controladores proporcionados por el fabricante, ordenar las acciones que realizará el robot desde otra computadora remota.
2. Es posible integrar un conjunto de programas de software libre para operar, desde la computadora remota por Internet, el robot.
3. Con un programa escrito en C es posible mandar comandos al controlador del fabricante por medio del puerto serie de la computadora.
4. El usuario podrá ser capaz de mover las partes del robot desde su computadora por Internet como si estuviera presente con un teclado multifuncional.
5. Es factible utilizar una base de datos de libre distribución para controlar el acceso a la operación remota del robot.
6. Se pueden almacenar posiciones del robot en la base de datos para posteriormente pedir al controlador que mueva el robot a esas posiciones.
7. La interacción entre el usuario en la computadora remota y el robot puede llevarse a cabo a través de una página Web desarrollada a base de instrucciones escritas en un lenguaje script en la computadora, ese lenguaje script es de libre distribución.
8. El usuario podrá mandar comandos en lenguaje del controlador directamente al robot por medio de la página Web.
9. El usuario podrá escribir programas en lenguaje del controlador y almacenarlas en la base de datos para ejecutarlos posteriormente.
10. Es posible visualizar las acciones realizadas por el robot por medio de una cámara de video y mostrar las imágenes en tiempo real.

CAPÍTULO I

ALGUNOS SISTEMAS ACTUALES DE CONTROL REMOTO POR MEDIO DE LA INTERNET.

1.1 CONCEPTOS BÁSICOS.

1.1.1 Redes.

En un comienzo las computadoras funcionaron de manera aislada. Más tarde, distintas compañías comenzaron a ofrecer formas de comunicar computadoras de distintas formas. En un principio, una forma para cada marca de computadora, de tal forma que si alguien deseaba crear una red, debía tener varias computadoras iguales cada uno equipado con tarjetas adicionales que permitieran comunicación entre ellas. Ejemplos de redes son: Ethernet, IBM Token Ring, Novell, etc. Cada red tiene un cierto lenguaje de comunicaciones (protocolo).

1.1.2 La Internet.

La Internet es una colección de redes – una red de redes – computadoras compartiendo información digital mediante un conjunto de sistemas de redes y de programas llamados protocolos. Casi cualquiera puede conectar su computadora a la Internet e inmediatamente comunicarse con otras computadoras y usuarios en la Red.

La Internet es una forma de establecer comunicación entre distintos tipos de redes. Para ello, dentro de cada red se escoge una computadora que actúe como Gateway o puerta de enlace (Figura 1.1), que se preocupa de traducir el lenguaje interno de la red en el lenguaje de la Internet, llamado TCP/IP.

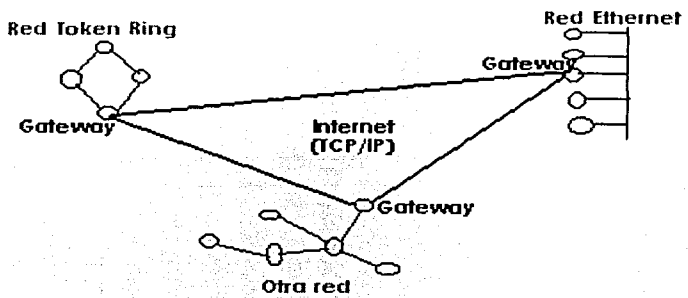


Figura 1.1 Esquema de la Red Internet

Por lo anterior, se puede decir que la Internet es una red de redes.

1.1.3 TCP/IP.

Las dos componentes del protocolo de la Internet se denominan:

TCP Transmission Control Protocol. Es el protocolo de transferencia entre computadoras, básicamente establece que la información debe ser dividida en fragmentos o "paquetes" que se propagan de manera separada y luego se juntan en el destino.

IP Internet protocol. Es el protocolo de direcciones, básicamente establece que cada computadora se identifica por su dirección IP compuesta de 4 números (Ej.: 146.83.4.57), cada computadora conectada en la red debe tener una dirección IP única.

1.1.4 Funcionamiento de las redes sobre telefonía.

El esquema anterior es válido para computadores permanentemente conectados a la Internet. En el caso de que no estén siempre conectados, probablemente se conecten vía telefónica. El teléfono establece una red cuyos componentes van cambiando (Figura1.2).

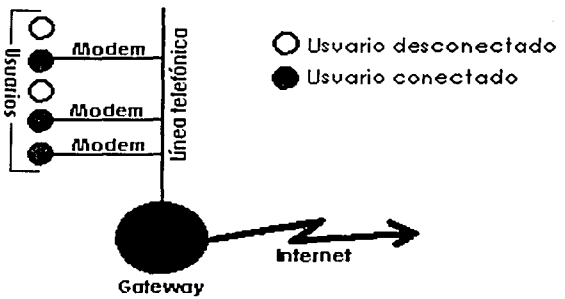


Figura 1.2 Esquema de una conexión por medio de línea telefónica

1.1.5 Los servidores en la Internet.

El tener un lenguaje unificado que comunica computadoras, abre la posibilidad de que una computadora conectada a la red ofrezca un servicio a las demás. Por ejemplo, puede ofrecer varios archivos disponibles para ser copiados en una computadora específica. A todas aquellas máquinas que ofrecen servicios a través de la red se les ha denominado como servidores y todas aquellas computadoras que utilizan esos servicios se les ha denominado clientes.

Cualquier máquina servidor proporciona sus servicios a la Internet utilizando **puertos** numerados, para cada servicio de que dispone el servidor. Por ejemplo, si una máquina servidor maneja un servidor Web y un servidor FTP, el servidor Web estará disponible típicamente en el puerto 80, y el servidor FTP estará disponible en el puerto 21. Los clientes se conectan a un servicio con una dirección IP específica y en un número de puerto específico.

Cada uno de los servicios es disponible en un "número de puerto conocido". En la Tabla 1.3 se muestran algunos de los números de puertos más conocidos.

Servicio	Puerto
Daytime	13
FTP	21
Telnet	23
SMTP (Simple Mail Transfer, para correo)	25
Gopher	70
Finger	79
WWW	80

Tabla 1.3 Servicios que se ofrecen en los servidores de la Internet y los puertos conocidos por donde el servidor usualmente los ofrece.

Si la máquina servidor acepta conexiones en un puerto desde el mundo exterior, se puede conectar a ese puerto y utilizar el respectivo servicio. Por ejemplo, un servidor Web debe estar en el puerto 80. Si se configura una máquina e instala software para servidor Web en él, podría colocarse el servicio Web en el puerto 918 (o cualquier otro puerto libre) si se desea. Si la máquina es conocida como: www.yyy.com, alguien podría conectarse a la máquina con la URL (Siglas de Uniform Resource Locator): http://xxx.yyy.com:918. EL ":918" especifica el número de puerto. Cuando el puerto no es especificado, el Navegador asume que el servidor utiliza el conocido puerto 80.

Entre los servicios que se ofrecen en la Red de Internet se encuentran:

- DNS.
- WWW.

1.1.5.1 DNS.

Debido a que los seres humanos a veces tienen problemas para recordar direcciones IP, y debido a que las direcciones IP a veces necesitan cambiar, todos los servidores en la Internet también tienen nombres que son más entendibles para los humanos llamados **nombres de dominio**. Por ejemplo, www.geocities.com es un nombre permanente. Es más fácil para la mayoría de la gente recordar www.geocities.com que su equivalente en números.

El nombre www.geocities.com tiene 3 partes:

1. El tipo de servicio ("www").
2. El nombre de dominio ("geocities").
3. El tipo de entidad ("com").

Los nombres de dominio son manejados por una corporación llamada InterNIC. Su tarea primordial es crear nombres para tipos de entidades y garantizar que todos los nombres de dominio son únicos. El nombre es creado por la compañía que maneja el servicio.

Para transformar las direcciones IP a nombres, se utilizan un grupo de servidores llamados DNS **(Domain Name System)** Servidores de Nombres de Dominio. Estos servidores tienen simples bases de datos que transforman las direcciones IP; estos están distribuidos por toda la Internet.

Si se teclea la URL http://www.geocities.com/carlosecv/cv.html en un navegador, éste fracciona el nombre "www.geocities.com" y el Servidor DNS regresará la dirección IP correspondiente.

1.1.5.2 WWW.

El (World Wide Web) es otro servicio. Mediante él, una computadora puede ofrecer documentos para ser vistos por otros usuarios en la red. A las computadoras que ofrecen este servicio también se les conoce como servidores Web. Estos documentos tienen la característica de que incluyen referencias a elementos multimedios y a otros documentos.

Se necesitan varios protocolos o acuerdos para hacer que un servicio como éste funcione. Es requerido un protocolo de direccionamiento, un protocolo de transferencia y un protocolo de especificación de documentos. Estos son respectivamente URL, HTTP y HTML.

1.1.5.2.1 URL (direccionamiento).

Este protocolo establece que para identificar un recurso en el Web, se utilizará una fragmento de texto de la forma http://COMPUTADORA/DIRECTORIO/PAGINA, donde COMPUTADORA es el nombre simbólico de una computadora conectada a la red (este nombre simbólico es transformado a una dirección IP mediante un servidor DNS), DIRECTORIO es una carpeta dentro de esa COMPUTADORA y ARCHIVO es el nombre de un archivo.

1.1.5.2.2 HTTP (transferencia).

Este protocolo establece la conexión de una computadora con otra.. Este protocolo está incorporado en todos los navegadores, y la operación es invisible para el usuario. Se encarga de hacer la petición de documentos HTML.

1.1.5.2.3 HTML (documentos).

Una página Web, o documento HTML, se construye utilizando el lenguaje HTML (Abreviatura de *Hypertext Markup Langua*ge). Para ello, existen varios programas que permiten ahorrarse el trabajo de escribir las marcas, o transformar documentos de otros formatos.

El software es opcional. Se prefiere hacer las páginas directamente a mano, utilizando un editor de texto. HTML es tan simple que no se necesita nada más.

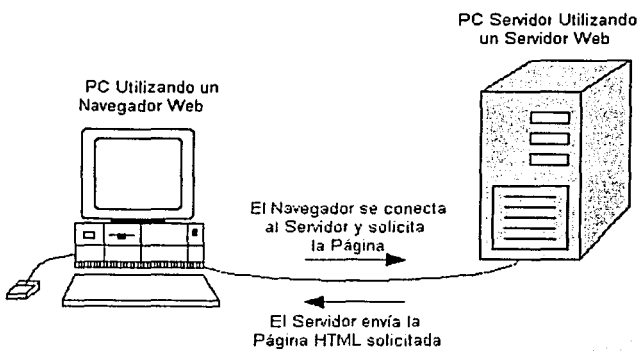


Figura 1.4 Diagrama de lo que sucede cuando se solicitan páginas Web a un Servidor

Para resumir (Figura 1.4), cuando se teclea una URL en un Navegador ocurren las siguientes cosas:

- El servidor fragmenta la URL en tres partes: 1) El protocolo ("http"), 2)El nombre del servidor ("www.geocities.com"), 3) El nombre de archivo (con la respectiva ruta, si existe).
- El navegador se comunica con un servidor de nombres para transformar la dirección en otra de números llamada dirección IP que se utiliza para la conexión al servidor antes dicho. El navegador entonces realiza una conexión al servidor en el puerto 80 (Figura 1.5). Siguiendo el protocolo HTTP, el navegador envía una propuesta de petición al servidor preguntando por el archivo. El servidor regresa el texto HTML de la página al navegador. El navegador lee los comandos HTML y muestra la página en su pantalla.

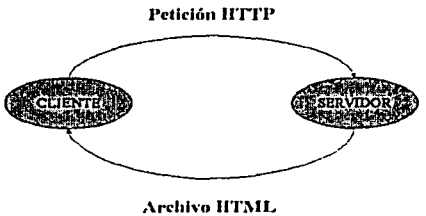


Figura 1.5 Círculo de petición de documento HTML y respuesta del servidor Web

1.1.5.3 Páginas Dinámicas.

Al desarrollar páginas HTML y colocarlas en un servidor Web no se tiene un control sobre las acciones de los usuarios, les hemos llamado páginas estáticas; tampoco es posible tomar decisiones sobre los contenidos de las páginas HTML, como mostrar cierta información sólo a aquellas personas autorizadas para hacerlo. La necesidad de controlar la información ha dado como resultado el desarrollo de herramientas para acceder al servicio de WWW. En otras palabras, si los datos que se van a mostrar requieren de restricciones o se desea mayor interactividad con los usuarios, la mejor manera de hacerlo es el uso de lenguajes script en el servidor Web. Al uso de Bases de Datos y lenguajes script para publicación en el Web lo hemos denominado como páginas dinámicas.

Si se desea hacer cosas más complicadas , es decir páginas dinámicas, hace falta la ejecución de programas. Para ello existen actualmente 2 alternativas que se muestran en la Tabla 1.6.

Ejecución en el Cliente	Ejecución en el Servidor (Lenguajes Script)
APPLETS	CGI
	PHP
	ASP
	SERVLETS
	JSP

Tabla 1.6 Alternativas para la ejecución de programas por la Internet

1.1.5.3.1 Ejecución en el cliente (APPLETS).

Un APPLET es un programa escrito en JAVA y que es ejecutado en un navegador compatible con JAVA. Estrictamente hablando, JAVA es interpretado, sin embargo realmente JAVA es interpretado y compilado, esto se debe a que únicamente cerca del 20% del código en JAVA es interpretado por el navegador. Pero este 20% es un porcentaje de mucha relevancia para las aplicaciones desarrolladas en JAVA, debido a que la seguridad de JAVA y la capacidad de correr sobre distintas plataformas nace del hecho de que los pasos finales de la compilación son manejados localmente.

Un programador primero compila el código fuente en JAVA para obtener un código llamado el "bytecode". Este proceso se hace usando el compilador de JAVA. Este bytecode de carácter binario y en su arquitectura interna es neutral (o independiente de la plataforma). Sin embargo, el bytecode no se encuentra completo hasta que se pone junto con el ambiente de ejecución o runtime, el cual es usualmente un navegador. Debido a que cada ambiente de ejecución es particular para una plataforma, los bytecode pueden ser interpretados por la plataforma especifica y el producto final puede ser ejecutado para dicha plataforma.

Esta es una de las ventajas del JAVA para los desarrolladores. Se puede escribir y compilar un applet de JAVA en un sistema UNIX e incluir el applet en una página Web. Luego tres usuarios usando diferentes ambientes pueden acceder a la página y el applet será ejecutado sin ningún tipo de problemas en el nuevo ambiente.

La característica que permite a JAVA ser ejecutado en otro ambiente es debido a su carácter interpretado, pero esto hace a su vez que los programas desarrollados en JAVA se ejecuten usando el doble o hasta seis veces el tiempo que le tomaría a un programa desarrollado en C++.

Existe una herramienta llamada JIT (Just-In-Time) el cual hace que este problema de velocidad sea superado.

1.1.5.3.2 La máquina virtual de JAVA

El corazón de JAVA es el JAVA Virtual Machine, o JVM. El JVM es un computador virtual que reside en la memoria únicamente. El JVM permite a los programas en JAVA ser ejecutados en una variedad de plataformas sin importar en que plataforma el código fue compilado. El hecho de que los programas en JAVA sean compilados solo para la JVM, hace que JAVA sea un lenguaje único. Pero para poder hacer que los programas en JAVA corran en otras plataformas, la JVM debe ser implementada para cada plataforma, es decir se instala en el navegador (Figura 1.7).

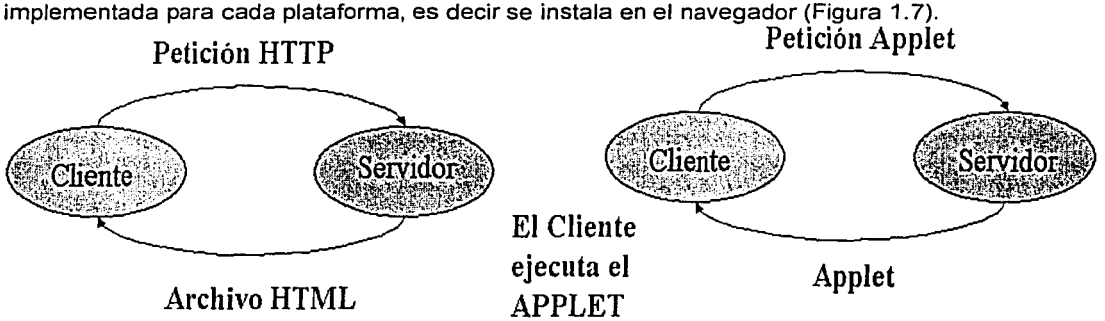


Figura 1.7 Diagrama de ejecución de un Applet, a la derecha se pide un documento HTML, a la izquierda el cliente pide al servidor el APPLET y lo ejecuta el cliente con el JVM

Los APPLETS se usan para conseguir:

- Efectos visuales y sonoros.
- Textos en movimiento.
- Utilidades (por ejemplo: relojes).
- Pequeños programas educativos.
- Juegos interactivos.
- Presentaciones multimedia, etc.

1.1.5.3.3 Lenguajes Script (Ejecución en el Servidor).

Un script es un programa simple interpretado por una aplicación, es un pequeño programa con una serie de instrucciones, las cuales ejecutan automáticamente una tarea específica. Los script son a menudo ejecutados cuando una página web se descarga. Ellos pueden ser escritos en diferentes lenguajes de programación (Figura 1.8).

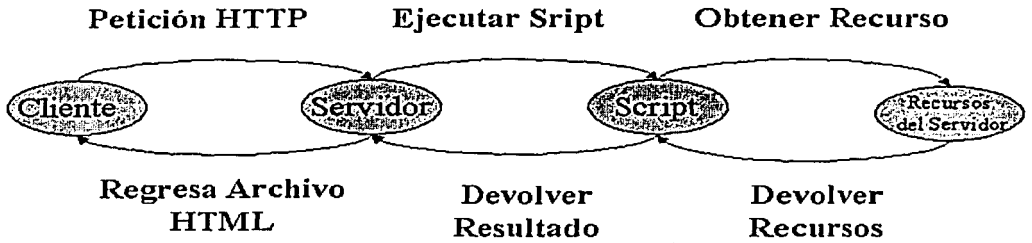


Figura 1.8 Diagrama de ejecución de un programa script en un servidor Web

Se utilizan para:

- Validar usuarios en páginas web (login, password)
- Conseguir opinión guardando la información de los formularios.
- Personalizar la información de las páginas web (ej. motores de búsqueda).
- Cuando los datos cambian frecuentemente. (ej: noticias)
- Tiendas en línea, etc.

1.1.5.3.3.1 CGI.

Abreviación de Common Gateway Interface, el CGI es un programa de interfaz que permite al servidor Web utilizar programas externos para realizar una función específica. También denominado pasarelas o CGI "scripts", estos programas consisten generalmente de una serie de instrucciones escritas en un lenguaje de programación como C o PERL que procesan la petición de un navegador, ejecutan un programa y formatean los resultados en HTML de manera que puedan ser presentados en el navegador. Se utilizan para añadir interactividad a una página web al permitir a los usuarios rellenar y enviar formularios que podrán ser procesados (como un catálogo en línea), acceder a bases de datos por medio de una búsqueda, y obtener acceso a un sitio protegido escribiendo una contraseña. Los scripts CGI también se usan para introducir una variedad de sistemas de análisis y de tráfico de medida de audiencia de un sitio

Los CGI fueron las primeras herramientas para hacer páginas interactivas en la Internet.

1.1.5.3.3.2 ASP.

Desarrollado por Microsoft es usado en conjunto con el Internet Information Server (IIS), el cual es un programa servidor Web que corre en servidores Microsoft. ASP no es soportado por otro tipo de plataforma que no sea Microsoft. Los scripts se almacenan en archivos con extensión .asp.

En los archivos .asp, las secuencias de comandos se separan del texto y de las etiquetas HTML mediante delimitadores. Un *delimitador* es un carácter o una secuencia de caracteres que marca el principio o el final de una unidad. En el caso de HTML, dichos delimitadores son los símbolos menor que (<) y mayor que (>), que enmarcan las etiquetas HTML.

ASP utiliza los delimitadores <% y %> para enmarcar los comandos. Dentro de los delimitadores puede incluir cualquier comando válido dentro del lenguaje de secuencia de comandos que esté utilizando.

El ejemplo siguiente muestra una página HTML sencilla que contiene un comando:

```
<HTML>
<BODY>
Esta página se actualizó por última vez el <%= Now ()%>.
</BODY>
</HTML>
```

La función **Now()** de VBScript devuelve la fecha y la hora actuales. Cuando el servidor Web procesa esta página, reemplaza <%= Now ()%> con la fecha y la hora actuales, y devuelve la página al explorador con el siguiente resultado:

Esta página se actualizó el 1/29/99 2:20:00 p.m.

A los comandos enmarcados por delimitadores se les llama *comandos principales de secuencias de comandos*, que se procesan mediante el lenguaje principal de secuencia de comandos. Todos los comandos utilizados dentro de los delimitadores deben ser válidos en el lenguaje principal de secuencia de comandos. De forma predeterminada, el lenguaje principal de secuencia de comandos es VBScript, pero también puede establecer un lenguaje diferente.

1.1.5.3.3.3 PHP.

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

PHP es una extensión para servidores web. Lo que hace es colocarse "entre" el servidor y el cliente (Figura 1.9).

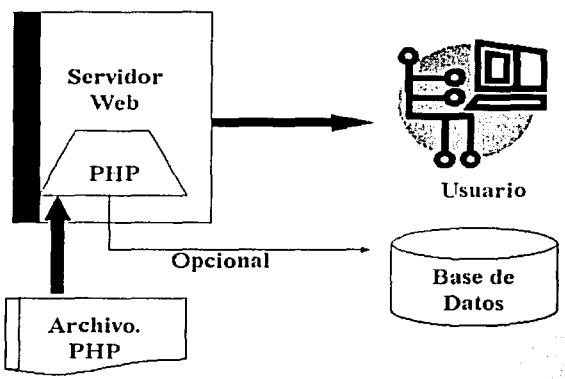


Figura 1.9 Diagrama de ejecución de programa script hecho en PHP e interactuando con una base de datos.

PHP toma código dentro de las páginas HTML, lo ejecuta en el servidor y envía el resultado al cliente. El cliente no puede visualizar el código del programa, sólo su resultado. Además, por ser un lenguaje de scripting, los programas no se compilan, sino sólo se interpretan; esto significa que es más lento en ejecutarse que, por ejemplo, un programa en C, pero al mismo tiempo los cambios en el código PHP tienen efecto de inmediato.

No todos los archivos son interpretados por PHP, sino sólo aquellos que hayan sido definidos en la configuración del servidor como tales, puesto que la interpretación de código es un proceso que toma un tiempo mayor al que se emplea en enviar una página tal como está. Es común utilizar una o varias de las siguientes extensiones en los archivos para informar al servidor que debe interpretarlas: .php, .phtml, .php3 y .php4.

Existen múltiples formas de incluir código PHP:

```
<?php echo("Hello World") ?>
<? echo("Hello World") ?>
<script language="php"> echo("Hello World"); </script>
<% echo("Hello World") %>
```

Puede ser que no todas estén disponibles en el sistema, esto depende de la instalación que se haya realizado. Ante la duda, se debe utilizar la primera forma. Todas las instrucciones se separan de la instrucción siguiente con un ; (punto y coma), y se asume que el final de la inclusión de código limita instrucciones

Un ejemplo de un programa escrito en PHP:

```
Bienvenidos a un programa escrito en PHP
Contador de 1 a 3:
<?
For($i=1;$i<4;$i++)
{
 echo $i;
 echo "<br>";
}
echo( "*******Gracias*******<br>" )
?>
```

Esto daría como resultado en el navegador:

```
Bienvenidos a un programa escrito en PHP
Contador de 1 a 3:
1
2
3
*******Gracias********
```

PHP es una herramienta para desarrolladores Web de libre distribución esto permite modificar el lenguaje para satisfacer las necesidades de quien desarrolla.

1.1.5.3.3.4 SERVLETS y JSP.

Los servlets son módulos que utilizan los servidores Web compatibles con Java, de manera similar a PHP. Por ejemplo, un servlet podría ser responsable de tomar los datos de un formulario de entrada de pedidos en HTML y aplicarle la lógica de negocios utilizada para actualizar la base de datos de pedidos de la compañía.

Los servelts pueden ser incluidos en diferentes servidores porque el programa API Servlet, el que se utiliza para escribir servlets, no asume nada sobre el entorno o protocolo del servidor. Los servlets se están utilizando ampliamente dentro de los servidores Web.

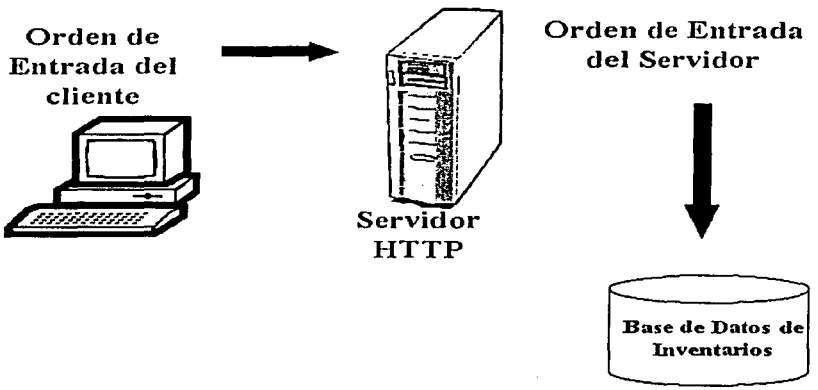


Figura 1.10 Diagrama de ejecución de un archivo servlet.

JSP (*Java Server Pages*) es una invención de la empresa SUN Microsystems Inc. provee un lenguaje de script en el lado del servidor que se comunica con clases Java, objetos RMI, CORBA, etc. La metodología de trabajo esperada es la misma que con Visual Basic con la diferencia de que se trata de una plataforma mucho más abierta (Figura 1.10).

El código JSP se puede colocar dentro de las páginas HTML, o se puede precompilar en servlets (programas en Java que también pueden ser compilados en servlets). El servidor pasa las variables hacia y desde el ambiente de ejecución del servlet.

Para programar en JSP se requiere conocer Java, ser metódico y ordenado.

1.1.5.3.4 Servidores de WWW.

Es necesario contar con un programa que responda a las peticiones http que los clientes harán. Este programa, para el desarrollo de ésta tesis, deberá soportar algún lenguaje script, así como confiabilidad y seguridad, para ello existen diferentes programas en el mercado que ofrecen diferentes opciones.

1.1.5.3.4.1 IIS.

(*Internet Information Server*) Servidor web para entornos Windows. Es uno de los servidores web más usados. Su uso es limitado a plataformas como Windows NT y Windows 2000. Se utiliza en conjunto con ASP para el desarrollo de páginas dinámicas.

1.1.5.3.4.2 Apache.

El Proyecto Apache es un esfuerzo de desarrollo de software en colaboración destinado a crear un servidor HTTP (servidor web) robusto, comparable a los comerciales y con muchas características y que permita disponer gratuitamente de su código fuente. El proyecto está gestionado conjuntamente por un grupo de voluntarios diseminados por todo el mundo, usando la red Internet para comunicarse, planear y desarrollar el servidor y la documentación correspondiente. Estos voluntarios son conocidos como el Grupo Apache (Apache Group). Además, cientos de usuarios contribuyen al proyecto con ideas, código y documentación.

1.1.5.3.5 Bases de Datos.

En rigor, una Base de Datos es el conjunto de datos almacenados con una estructura lógica. Es decir, tan importante como los datos, es la estructura conceptual con la que se relacionan entre ellos. En la práctica, podemos pensar esto como el conjunto de datos más los programas (o software) que hacen de ellos un conjunto consistente.

Si no tenemos los dos factores unidos, no podemos hablar de una base de datos, ya que ambos combinados dan la coherencia necesaria para poder trabajar con los datos de una manera sistemática.

Un Dato es una representación abstracta de algo, la información es el conjunto de datos que nos reportan algo que es de nuestro interés.

1.1.5.3.5.1 Manejadores de Base de Datos.

Un manejador de base de datos DBMS (Database Management System) es un módulo de programas que constituye la interfaz entre los datos almacenados en la base de datos y los programas de aplicaciones. El DBMS se encarga de las siguientes tareas:

- Interacción con el manejador de archivos.
- Permite almacenar, obtener y modificar los datos.
- Seguridad.
- Respaldo y recuperación.

Existen en el mercado diferentes Manejadores de Base de Datos que pueden utilizarse en una PC entre las que se encuentran:

- MySQL.
- SQL Server.
- Oracle.
- Postgresql.

De los cuales sólo MySQL, Postgresql y Oracle funcionan bajo Linux y de éstos tres sólo Postgresql es de libre distribución.

1.1.5.3.5.1.1 MySQL

En el mundo GNU (2.2.1 GNU software libre Capítulo II), una de las bases de datos que se reseña en cualquier referencia de aplicaciones de éste tipo bajo LINUX, es MySQL aunque no está incluida en ninguna distribución ya que no tiene licencia GNU como tal, para comercializarla a ella o a cualquier software que la utilice o se sirva de ésta habrá que adquirir una licencia.

Principales Características

- Escrito en C y C++.
- Clientes C, C++, JAVA, Perl, TCL.
- Multiproceso, es decir puede usar varias CPU si éstas están disponibles.
- Sistema de contraseñas y privilegios.
- Todas la palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- 16 índices por tabla, cada índice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.
- Todas las columnas pueden tener valores por defecto.
- Utilidad (Isamchk) para chequear, optimizar y reparar tablas.
- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.
- Diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.

1.1.5.3.5.1.2 SQL Server

Dentro de las ventajas de Microsoft SQL Server se encuentran

- Trasacctions (Transacciones)
- Tiggers (Disparadores)
- Constrains (Restricciones)
- Replication (Replicación)
- Backup & Recovery (Respaldo y Recuperación)
- Rules (Reglas)
- Stored Procedures/Functions (Procedimientos Almacenados/Funciones)
- Integridad Referencial
- Logging (Ingreso controlado por claves)
- Extensible
- Integridad de datos
- Funciones definidas por el usuario
- Cliente/Servidor
- Ambiente gráfico.

Desventajas

- Requiere de un costo por licencias
- Sólo funciona en Sistemas de Microsoft.

1.1.5.3.5.1.3 Oracle

Características de Oracle8i

Manejo de contenido para Internet

WebDB es una herramienta que permite la construcción de portales o sitios web dinámicos basados en información extraída de la base de datos. Todo el sitio web entonces se encuentra almacenado en la base de datos lo que le brinda mucha seguridad y gran escalabilidad. Además permite el manejo de contenido almacenado en la base de datos y búsquedas inteligentes sobre los mismos. Todas las operaciones se realizan a través de asistentes amigables y a través del web.

Internet File System (IFS) es una herramienta que permite crear un sistema de archivos almacenado dentro de la base de datos. Los PCs de la red pueden entonces mapear este sistema de archivos como si fuera otra unidad de disco, y por ende se pueden almacenar archivos de manera segura y transparente para el usuario final. Además, se pueden realizar conexiones con FTP, HTTP, POP3, etc con dicho sistema de archivos.

Intermedia es una opción que permite el almacenamiento y utilización de objetos multimedia dentro de la base de datos que además son accesibles a través del web. Incluye opciones para manejo de texto, audio, video, imagen, data espacial, etc y permite integrar toda la data de la organización cualquier aplicación.

Java

Oracle8i es la primera base de datos en soportar programas escritos en Java dentro de la base de datos. Asi, se pueden crear stored procedures y triggers utilizando el lenguaje Java. La base de datos tiene entonces un ambiente para ejecutar y compilar dichos programas.

Soporta todos los modelos de desarrollo se aplicaciones en Java como son: CORBA, Enterprise Java Beans y Java Stored Procedures.

Ademas cuenta con herramientas para construir componentes Java y publicarlos como son: JDeveloper y JPublisher.

Algunas otras características son:

- Ambiente Objeto-Relacional
- Análisis de los redo log con Log Miner
- Drop Column
- Locally Managed Tablespaces
- Creación, reconstrucción y defragmentación en línea de los índices
- Read-Only Tablespaces
- Tablas temporales
- Definición de rutinas externas

Desventajas

- Costo.

1.1.5.3.5.1.4 PostgreSQL.

PostgreSQL es un administrador de Bases de Datos Relacional, que como otros productos libres y comerciales salió de la investigación de Universidades. En el caso de PostgreSQL este surge del proyecto desarrollado en la Universidad de California en el Departamento de Ciencias de la Computación.

PostgreSQL soporta la mayor cantidad de las instrucciones de SQL que es el estándar, y por ello es uno de los manejadores de Bases de Datos más populares y de mejor funcionamiento dentro de lo que es el software libre, que para lograr una mayor eficiencia une todas las estructuras clásicas con los conceptos de la programación orientada a objetos, lo que la convierte en una base de datos objeto-relacional.

PostgreSQL es un administrador de Base de Datos con todas las características y ventajas de un administrador comercial. Soporta: sintaxis SQL92, transacciones, disparadores, restricciones, llaves foráneas, replicación, etc. Además de la conectividad ODBC para ser utilizado con aplicaciones Windows permitiendo de esta manera generar nuevas aplicaciones, mantener las que ya existan y hacer emigraciones desde Access, Visual Basic, Foxpro, C/C++, Delphi, etc., y JDBC 2.0 para Java.

Un concepto que aparece en PostgreSQL y los distingue de otros modelos es la herencia, este concepto surge de la programación orientada a objetos. Así pues, cuando se crea una nueva clase heredada de otra, la clase creada adquiere toda las características de la clase de la que proviene, más las características que se definan en la nueva clase. De la misma forma cuando se hereda una clase, se esta creando una tabla igual a la origen de la herencia.

Para su buen funcionamiento utiliza el modelo cliente/servidor, que nos da como resultado un proceso por usuario. Este modelo consta de un programa supervisor, uno o varios programas de aplicación, un servidor de base de datos por cada programa de aplicación. El programa supervisor se ejecuta en el sistema como un demonio y gestiona una colección de base de datos. La conexión entre el programa de aplicación y el servidor de base de datos se realiza por medio de un socket TCP/IP. El mecanismo de conexión (Figura 2.4) es el siguiente:

El programa supervisor, que en adelante llamaremos postmaster, permanece a la escucha de un puerto TCP/IP. Cuando un cliente desea acceder a la base de datos lanza un connect dirigido al puerto del postmaster, el cual al recibir la petición realiza una copia (fork) de si mismo. El nuevo proceso creado con la función fork permanece en sesión con la aplicación cliente, mientras que el postmaster vuelve a la escucha para atender otras peticiones de conexión, este modelo de arquitectura, al estar apoyado en TCP/IP, permite que los clientes puedan estar tanto en forma local como remota.

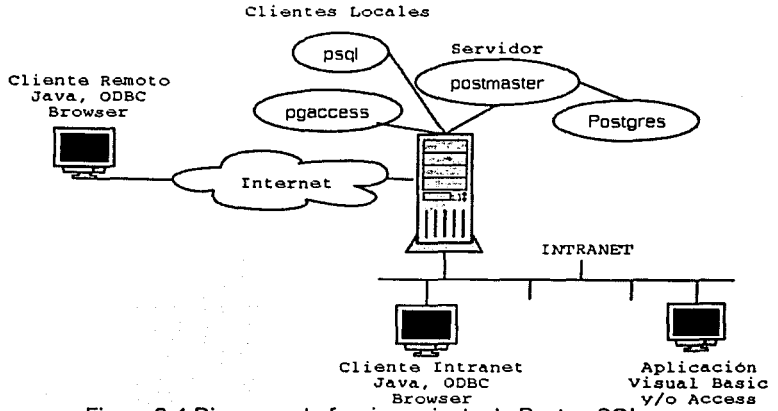


Figura 2.4 Diagrama de funcionamiento de PostgreSQL

1.1.5.3.5.1.4.1 Características Operacionales.

- Trasacctions (Transacciones)
- Tiggers (Disparadores)
- Constrains (Restricciones)
- Replication (Replicación)
- Backup & Recovery (Respaldo y Recuperación)
- Rules (Reglas)
- Stored Procedures/Functions (Procedimientos Almacenados/Funciones)
- Integridad Referencial
- Sintaxis ANSI SQL 89, 92 y 98
- Logging (Ingreso controlado por claves)
- Extensible
- Orientación de Objetos
- Integridad de datos
- Funciones definidas por el usuario
- Cliente/Servidor

1.1.5.3.5.1.4.2 Requerimientos mínimos para su instalación.

- Memoria principal 8 MB
- Espacio libre en disco duro 100 MB
- Sistema Operativo Windows 95/98/NT/ME/2000 ó Linux
- Protocolo TCP/IP

1.1.5.3.5.1.4.3 Alcances de PostgreSQL.

Descripción	Valor
El tamaño de la base de datos	Limitado por la capacidad de almacenamiento del hardware.
El tamaño máximo de una tabla	64 Terabytes.
Máximo tamaño de un campo	1 Gb.
Número de tuplas o registros	Ilimitados.
Máxima cantidad de columnas en una tabla	1600
Cantidad de índices por tabla	Ilimitado

Tabla 2.5 Alcances de PostgreSQL

Además de todas las características antes mencionadas, mencionaremos aquellas que hacen de PostgreSQL una de las mejores alternativas:

- Confiable.
- Rápida.
- Robusta.
- De libre distribución.
- Puede ser utilizada en varias plataformas entre ellas Windows y Linux.

1.2 ROBOTS UTILIZADOS EN OPERACIONES REMOTAS A TRAVÉS DE LA INTERNET.

1.2.1 Robot.

Manipulador automático servo-controlado, reprogramable, polivalente, capaz de posicionar y orientar piezas útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables para la ejecución de tareas variadas. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Su principal característica es la de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.

1.2.2 Clasificación de los robots.

Existen varias clasificaciones de robots, para esta tesis se clasificó a los robots de acuerdo a su funcionalidad en dos grupos:

- Robots manipuladores.
- Robots móviles.

1.2.2.1 Robots manipuladores.

Son sistemas mecánicos multifuncionales, con un sencillo sistema de control, que permite gobernar el movimiento de sus elementos, de los siguientes modos:

a) ***Manual***: Cuando el operario controla directamente la tarea del manipulador.

b) ***De secuencia fija***: cuando se repite, de forma invariable, el proceso de trabajo preparado previamente.

c) ***De secuencia variable***: Se pueden alterar algunas características de los ciclos de trabajo.

Existen muchas operaciones básicas que pueden ser realizadas óptimamente mediante manipuladores, se considera seriamente el empleo de estos dispositivos cuando las funciones de trabajo son sencillas y repetitivas.

Algunos ejemplos de robots manipuladores controlados a través de la Internet son:

1.2.2.1.1 El proyecto Mercury.

El primer robot de la Internet (Septiembre 1, 1994- Marzo 31, 1995). Permitió a los usuarios de la Internet controlar remotamente un robot de IBM para ver y excavar objetos enterrados por medio de turbinas de aire comprimido. Fue desarrollado por la Universidad del Sur de California para el Navegador Mosaic. El archivo del sitio incluye antecedentes, trayectoria y bitácora de operación (Figura 1.11).

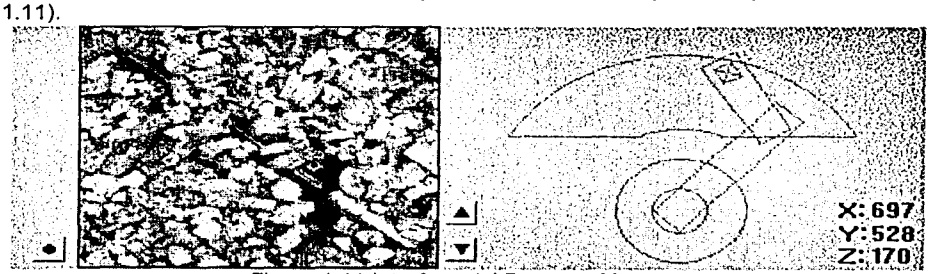


Figura 1.11 Interfase del Proyecto Mercury

1.2.2.1.2 Telerobot australiano.

El telerobot de la Universidad del Oeste de Australia (University of Western Australia's) se puso en línea a finales de Septiembre de 1994. Ken Taylor y Barney Dalton; el telerobot cambia, construye y destruye bloques en una mesa cuadriculada. El brazo es movido a través de coordenadas numéricas y está diseñado de tal manera que incluye giro e inclinación (además del plano de coordenadas x,y). Los movimientos del robot se ven a través de cuatro cámaras con diferentes ventajas (con acercamiento, calidad de imagen y tamaño bajo el control del usuario), los usuarios pueden elegir entre diferentes interfaces e incluso diseñar su propia interfase. Los usuarios registrados tienen prioridad de acceso al robot. El sitio ofrece un valioso tutorial, un cuarto de chat telerobótico, una cámara web del laboratorio (Figura 1.12).

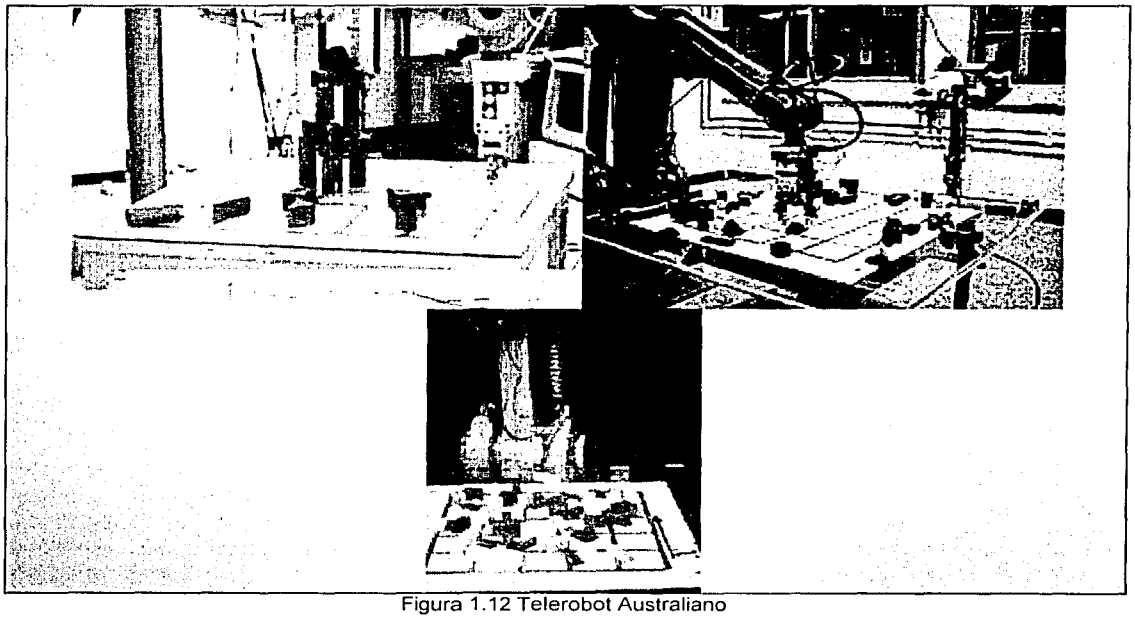


Figura 1.12 Telerobot Australiano

1.2.2.1.3 Telescopio robótico Bradford.

El telescopio robótico de la Universidad de Bradford está localizado en las afueras de Yorkshire, Inglaterra. Los usuarios pueden mandar observaciones al telescopio, completamente automatizado el cuál decide cuándo las condiciones son buenas para verlas (Figura 1.13).

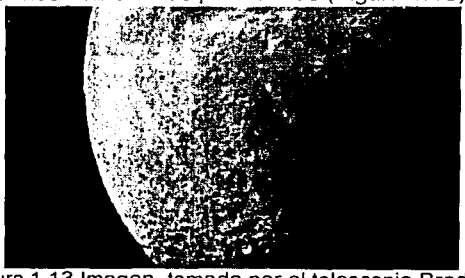


Figura 1.13 Imagen tomada por el telescopio Bradfort

1.2.2.1.4 El telejardín.

En línea desde Junio de 1995; se encuentra actualmente en el Centro de Artes y Electrónica en Austria. Permite a los usuarios cuidar y atender un jardín viviente. Los invitados solo pueden ver el jardín, pero los miembros pueden participar activamente plantando y regando las plantas. (Figura 1.14)

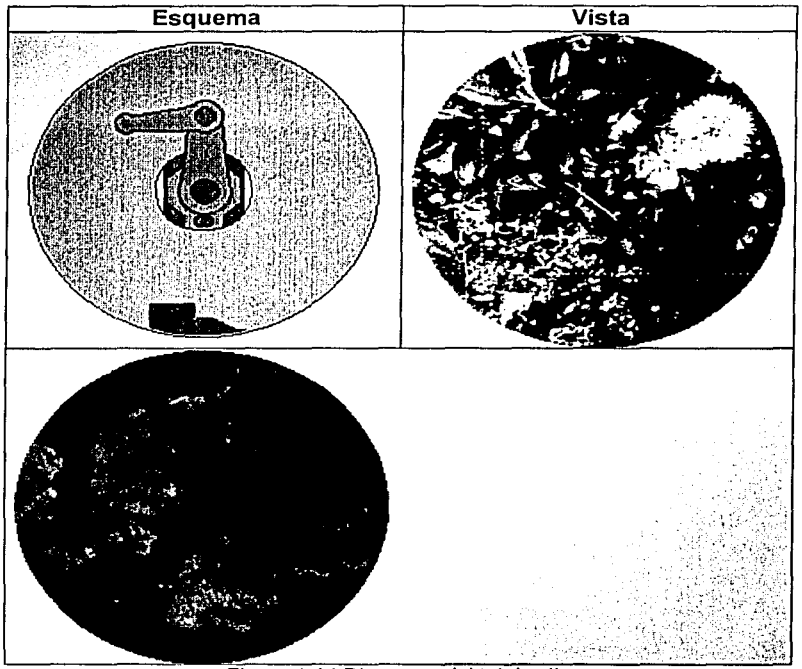


Figura 1.14 Diagrama del telejardín

1.2.2.1.5 Robotoy.

Robotoy fue creado en la Universidad de Wollolong, Australia, en conjunto con la empresa Vodafone, en el Reino Unido. El reto es tomar los bloques blancos esparcidos en una mesa. Los usuarios tienen el control de cada una de las uniones del robot, así como de dos cámaras para ver sus movimientos. (Figura 1.15)

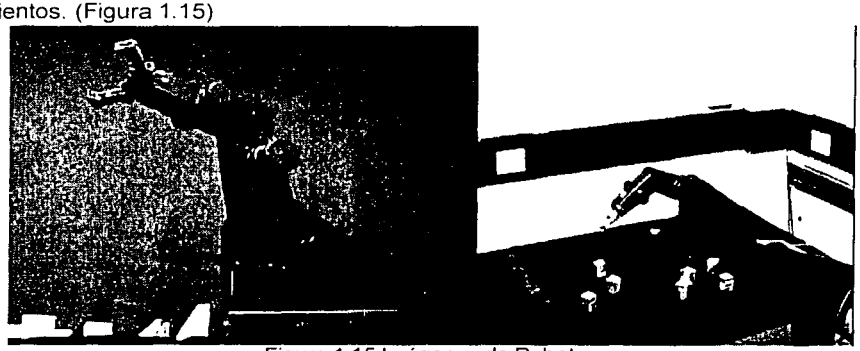


Figura 1.15 Imágenes de Robotoy

1.2.2.1.6 Telerobot del Centro de Ciencia Carnegie.

El robot de Pittsburgh ofrece muchas de las características de control como su predecesor Australiano, no obstante ofrece no solo dos cámaras, sino cuatro cámaras. Los usuarios pueden aprender a usar el sistema de coordenadas del robot por medio de un tutorial. (Figura 1.16)

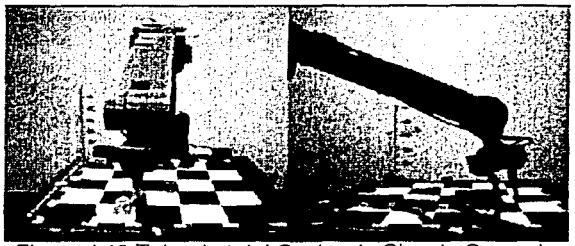


Figura 1.16 Telerobot del Centro de Ciencia Carnegie

1.2.2.1.7 Puma Paint.

El Puma Paint de Matthew Stein, en la Universidad de Wilkes. Permite a los usuarios pintar con brochas usando un brazo robótico industrial. La interfase tiene dos ventanas mostrando en vivo el sitio de trabajo y permite a los usuarios manipular la saturación de la brocha, el color y la presión en el lienzo. Además, a petición, las pinturas completas son enviadas (vía servicio postal de los Estados Unidos) por correo.

1.2.2.2 Robots Móviles.

Los robots móviles están provistos de patas, ruedas u orugas que los capacitan para desplazarse de acuerdo a su programación. Elaboran la información que reciben de su entorno a través de sus propios sistemas de sensores y se emplean en determinado tipo de instalaciones industriales, sobre todo para el transporte de mercancías en cadenas de producción y almacenes. También se utilizan robots de este tipo para la investigación en lugares de difícil acceso o muy distantes, como es el caso de la exploración espacial y las investigaciones o rescates submarinos

1.2.2.2.1 Xavier.

Xavier es un robot producido por la Universidad de Carnegie Mellon junto con la Universidad de Bonn. Se traslada a través de los laboratorios de la universidad y puede tomar fotos de los lugares a los que los visita, así como realizar tareas de oficina como trasladar papeles y otros documentos de importancia. (Figura 1.17)

Figura 1.17 Robot Xavier

1.2.2.2.2 Modelo Interactivo de tren.

En la Universidad de Ulm han colocado un set con trenes. Una segunda cámara ha sido colocada para dar una vista más cercana los usuarios pueden elegir que tren y que ruta utilizar. El sitio contiene un archivo de fotos de choques de trenes. (Figura 1.18)

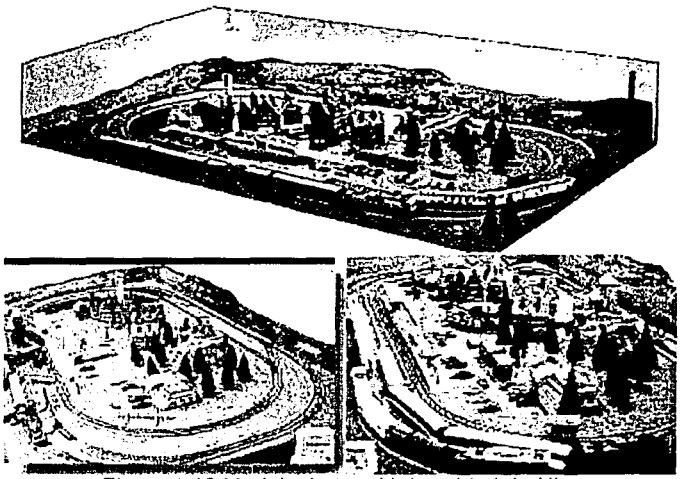


Figura 1.18 Modelo de tren Universidad de Ulm

1.2.2.2.3 Rhino.

Rhino, eso un robot guía de turistas, es un proyecto entre el instituto de informática de la Universidad de Bonn y la Universidad de Carnegie Melon. El museo alemán de Bonn es el primer museo tecnológico de Alemania. Rhino conduce tours por medio de la Internet. Este robot es de características similares al robot Xavier. (Figura 1.19)

Figura 1.19 Fotografía del robot RHINO

1.2.2.2.4 Alicia (Alice).

Alicia es el robot móvil inteligente más pequeño y barato, desarrollado por laboratorios de sistemas autónomos. Desarrollos para Alicia incluyen cooperación y comunicación multi robot, así como control por la Internet. (Figura 1.20)

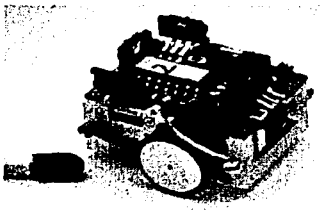


Figura 1.20 Fotografía del robot móvil más pequeño, Alice.

1.2.2.2.5 Red Rover.

Esta es una demostración del proyecto realizado por la compañía IBM en su almacén de reutilización en Endicott, NY. El proyecto promete permitir a los clientes interesados a ver los productos del almacén antes de hacer su pedido. (Figura 1.21)

Figura 1.21 Fotografía del robot Red Rover, IBM.

1.2.2.2.6 Rita.

En la Universidad de Birmingham un robot de entrenamiento, a través de algoritmos de aprendizaje desarrollados por Ricardo Poli y Axel Grobmann. Conocimientos previos sirven como base para nuevos conocimientos, a través de la ejecución de tareas relacionadas, que ella aprende a navegar por un ambiente controlado y a usar su efector final tipo pinza. El equipo se encuentra actualmente investigando la construcción de redes neuronales las cuales la adaptarán muchas tareas a las que se enfrenta. La interfase el Web fue implementada para el aprendizaje del robot. (Figura 1.22)

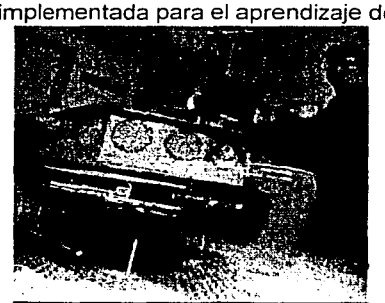


Figura 1.22 Fotografía del robot Rita Universidad de Birmingham

CAPÍTULO II
SISTEMAS DE LIBRE DISTRIBUCIÓN COMO ALTERNATIVA.

A lo largo del Capítulo 1 se vieron los conceptos básicos requeridos para entender los alcances de ésta tesis y que existen múltiples sistemas para el desarrollo de aplicaciones a través de la Internet.

Es importante dejar de pensar que el asunto de desarrollar aplicaciones Web es: leer de una base de datos, mostrar un formulario, leer los datos del formulario, escribir a la base de datos. No se puede seguir en esa línea mucho tiempo. Se debe apuntar más alto, a formas de desarrollar aplicaciones de más alto nivel.

En el Capítulo 2 se verá las ventajas de utilizar equipo de cómputo PC así como el uso de software libre para el desarrollo de programas y proyectos de robótica.

2.1 LA PC COMO MEDIO DE INTERACCIÓN ENTRE EL USUARIO, LA INTERNET Y EL ROBOT.

Han pasado un poco más de 20 años desde el nacimiento de la primera computadora de escritorio. En todos estos años su evolución ha sido impresionante, a tal punto que ha cambiado la forma de trabajo en la mayoría de las empresas. Actualmente la PC tiene funciones que van más allá del procesamiento de palabras, manejo de base de datos o navegar por la Internet.

La PC se ha convertido en una herramienta básica, pero su potencial radica no sólo en su estructura física (hardware) que resulta más barato que el de una Estación de Trabajo, sino también existe mayor cantidad de software para la PC como Sistemas Operativos y programas para uso general, que en el caso de este trabajo se requerirán: Sistema Operativo, Servidor Web, Lenguaje Script, soporte para Cámara Digital.

Muchos profesionales de la tecnología de la información no están dispuestos a invertir en hardware más costoso, como es el empleado en la mayoría de los sistemas operativos UNIX comerciales, así que veremos alternativas de sistemas operativos que existen para la PC.

2.1.1 El Sistema Operativo.

El sistema operativo es un conjunto de programas que tienen como función principal administrar todos los recursos de la computadora como:

- Administración de la memoria, para todos los programas en ejecución.
- Administración del tiempo de procesador, que estos programas en ejecución utilizan.
- Es el encargado de que se pueda acceder a los elementos periféricos de la computadora.

El sistema operativo es la base del funcionamiento de una computadora en general. Su función consiste en crear la infraestructura lógica que permita grabar datos en los discos y disquetes, así como generar, abrir, eliminar y mover archivos, organizar todo lo que sucede en la computadora, coordinando las actividades de los distintos programas. Además, se encarga de administrar la memoria, cortarla en porciones y administrarla entre los programas.

La oferta de sistemas operativos disponible para PC actualmente es limitada. La mayoría de las PCs hogareñas cuentan con el sistema operativo Windows en todas sus versiones: 95, 98, 2000, NT, ME, SE, XP. Por otra parte, hace tiempo surgió un sistema que está ganando adeptos entre los usuarios: se trata de Linux. Aunque es más complicado de utilizar que las versiones de Windows, ofrece las ventajas de ser gratuito y estar dotado por recursos mas poderosos.

2.2 SISTEMAS DE LIBRE DISTRIBUCIÓN Y SISTEMAS PROPIETARIO.

El software propietario es aquel por el que, una empresa o persona que lo desarrolla, cobra una cantidad de dinero por su comercialización o su uso. En cambio el software libre tiene otras características fuera de lo común que lo convierte en una alternativa atractiva y económica de solucionar los mismos problemas que se solucionan con software propietario pero a un menor costo. Para entender qué es el software libre debemos conocer el proyecto que inició todo lo relacionado con él.

2.2.1 GNU (software libre).

GNU acrónimo recursivo de *GNU's Not Unix*. Proyecto iniciado en 1984 por Richard Stallman para obtener un sistema operativo completo, totalmente libre, basado en el sistema operativo Unix. Para fomentar la libertad y cooperación en la creación de software, Stallman fundó también la *Free Software Foundation* y redactó la GPL o *GNU General Public License* que establece las características del *Free Software* (software libre).

2.2.2 Características del software libre.

No propietario. No hay que tener autorización de nadie ni firmar ningún contrato para poder adquirirlo o usarlo. El software no es una víctima de la mercadotecnia. No se depende de ninguna compañía.

Distribuible. Se pueden hacer tantas copias como se desee, incluso venderlas. Que sea no propietario no implica que sea no comercial; una compañía puede vender un CD con software libre para facilitar el trabajo de obtenerlo, pero no puede impedir que se copie y quien esté dispuesto a obtener las fuentes directamente, las podrá adquirir gratuitamente.

Accesible. El código fuente está disponible. El movimiento *Open Source* es muy similar: predica la conveniencia de distribuir el código fuente para el desarrollo de software, pero el acceso al código fuente solo implica algunas de las libertades implícitas en el software libre.

Modificable. Se puede mejorar el programa, incluso redistribuir la modificación. La depuración se paraleliza: rapidez en el desarrollo y calidad del resultado son compatibles. Se diluye la diferencia entre usuario y desarrollador.

Reusable. Se puede aprovechar código ya escrito, siempre y cuando el nuevo código mantenga las mismas libertades del software libre.

Sin garantías. Nadie asegura que funcionará, nadie se hace cargo de ningún daño. No se está desamparado: el soporte sigue otros cauces distintos a los tradicionales.

Contagioso. Cualquier trabajo derivado de un programa con licencia GPL está también bajo licencia GPL. Se "prohíbe prohibir".

También existe la *LGPL* o *GNU Lesser General Public License*, que permite desarrollar programas (propietarios) que usen librerías bajo GPL. Se ha programado todo tipo de aplicaciones bajo licencia GPL: compiladores e intérpretes para docenas de lenguajes, software científico, herramientas de oficina, navegadores y servidores *web*, emuladores, juegos, herramientas de cifrado, informática paralela, y utilidades de todo tipo.

Utilizando software propietario que requiere un pago de licencias por su uso y manteniéndose en regla, una organización puede llegar a necesitar invertir varias decenas de miles de pesos en la compra del mismo. El estado actual de nuestro país, tiene a las empresas y demás organizaciones en un jaque económico, y muy pocas pueden mantenerse en regla con respecto a las licencias tan restrictivas de la mayoría del software propietario. GNU/Linux es la mejor opción, la inversión en la compra de software es prácticamente nula, lo que permite reinvertir la totalidad o parte del dinero en capacitación, o la adquisición de mayor equipamiento.

Pero no sólo son económicas las ventajas de utilizar GNU/Linux. La misma naturaleza libre de todo el sistema, asegura la calidad y estabilidad de sus componentes, presentando de esta manera una plataforma de trabajo confiable y eficiente. Además, la gran versatilidad que posee GNU/Linux, permite que el sistema pueda utilizarse en ámbitos de trabajo totalmente dispares, desde un equipo de oficina, pasando por estaciones de trabajo para desarrolladores de software, hasta servidores de red con altos requerimientos. El hecho de que aproximadamente un 80% de los servidores de Web en la Internet estén manejados por software libre es una de las tantas pruebas que confirman lo que aquí se asegura.

2.2.3 El Proyecto Linux.

El proyecto LINUX se compone por personas que han estado apostando a este sistema desde sus comienzos, ésta es la alternativa más conveniente de solucionar las necesidades informáticas de cualquier organización en la actualidad y en el futuro.

El uso de Linux proporciona muchas ventajas. De los numerosos sistemas operativos disponibles en la actualidad, Linux es el sistema gratuito más popular y de amplio acceso. Para las PCs de IBM, Linux proporciona un sistema completo con capacidades integradas para multiusuario y multitareas que aprovechan toda la potencia de procesamiento de los sistemas de cómputo 386 y superiores. Linux viene con una adaptación del protocolo para red TCP/IP.

Linux, se puede conectar a la Internet y al vasto mundo de información que esta red abarca; se tiene acceso a un sistema de correo electrónico completo para enviar y recibir mensajes por medio de la Internet. También hay una interfaz gráfica para usuario (GUI, por sus siglas en inglés) dedicada a Linux, llamada XFree86, la cual está basada en el popular sistema X Windows. XFree86 es una adaptación del sistema X Windows que se distribuye en forma gratuita con Linux; esta interfaz proporciona los elementos comunes de una GUI que el usuario puede encontrar en otras plataformas comerciales de GUI, como Windows y OS/2.

2.2.4 Comparación de sistemas de libre distribución con sistemas propietario.

Windows NT se escoge a veces por razones presupuestarias, en la tabla 2.1 se muestran los costos de licencia por su uso.

Windows NT 4.0 Server	5 usuarios	$ 809 USD
Windows NT 4.0 Server	10 usuarios	$ 1,129 USD
Enterprise Edition	25 usuarios	$ 3,999 USD
Enterprise Edition	50 usuarios	$ 4,799 USD

Tabla 2.1 Costos de Windows NT

Lo que realmente importa es el costo global de la implantación de Windows NT; esto incluye la administración del sistema, los costos asociados con caídas del sistema, llamadas de teléfono para soporte técnico, pérdida de datos por la no-fiabilidad del sistema, etc.

Para otros profesionales, conscientes de los costos de implementación, la elección es *Linux, FreeBSD, NetBSD* u *OpenBSD*. No tienen costo por licencia de uso y sin embargo son tan estables y funcionales como los sistemas operativos UNIX comerciales (*SunOS, Solaris, AIX, HP-UX, IRIX, Digital UNIX*).

Windows NT Server	$ 4,636 USD
Red Hat Linux	$ 49.95 USD

Tabla 2.2 Inversión en dólares americanos por Windows NT Server versión básica, comparado con Linux Red Hat.

Linux es el sistema operativo UNIX más popular y corre en una extensa gama de plataformas: *Sun, Intel, DEC Alpha, PowerPC, PowerMac*, etc.

2.2.5 Comparación Windows NT *vs.* UNIX.

Hemos hecho la comparación sobre la base de 4 aspectos:

- Funcionalidad.
- Fiabilidad.
- Administración del sistema.
- Rendimiento.

2.2.5.1 Funcionalidad.

El sistema operativo UNIX puede hacer todo lo que hace Windows NT y más.

A veces se considera a Windows NT como un sistema operativo *multi usuario*, pero esto puede provocar confusión. Un Servidor NT *valida* a un usuario autorizado, y una vez que el usuario está conectado a la red NT, lo único que puede hacer es acceder archivos e impresoras. Un usuario de NT sólo puede correr aplicaciones especiales tipo cliente/servidor.

El sistema UNIX es *multi usuarios* y los usuarios en UNIX pueden correr cualquier aplicación en el servidor.

- En Windows NT se tiene que adquirir un paquete de software adicional para configurar un servidor de correo electrónico. Muchas compañías usan *Microsoft Exchange Server* (licencia para 25 usuarios $ 3,495 USD, 50 usuarios $ 4,859 USD).
- El sistema operativo Unix viene con el programa *Sendmail* (gratis).
- Seguridad (password y permisos de archivos) y cuotas:
- Windows NT utiliza NTFS para seguridad de archivos (a veces no lo usan) y aún carece de cuotas, para limitar el uso de disco a usuarios o grupos de usuarios.
- Unix contempla seguridad y cuotas.
- Facilidad de configuración y capacidad de configurar al servidor sin deshabilitar el sistema.
- Cualquier cambio a la configuración de NT requiere apagar el sistema y reinicializarlo (cambio de IP, gateway, modem, etc.).
- En UNIX se pueden activar o desactivar *drivers* o dispositivos sin necesidad de reiniciar el sistema.

2.2.5.2 Fiabilidad.

Actualmente quizás la fiabilidad es preferible a la rapidez. Aunque el rendimiento depende enormemente de la plataforma hardware, el sistema operativo influye más en la fiabilidad.

- Se dice que NT es un sistema operativo *estable*, pero esta afirmación no es muy precisa. Es cierto que NT es una gran mejora comparado con Windows 3.1 o Windows 95, pero dista mucho de alcanzar la estabilidad que ofrecen aún los sistemas operativos UNIX *freeware*.
- *Pantalla Azul de la Muerte*. La pantalla que muestra unos números hexadecimales sobre un fondo azul. La única salida es reinicializar el sistema apagándolo.
- NT es propenso a los virus para plataformas INTEL. Los sistemas operativos de *Microsoft* siguen leyendo del MBR de los discos duros. NT puede ser afectado por virus diseñados hace 10 años para MS-DOS.

- El equivalente a la *Pantalla Azul de la Muerte* en UNIX es el *pánico de kernel*. Las caídas de sistema en UNIX son eventos raros, casi siempre, se deben a fallos de hardware. En general un servidor UNIX se apaga debido a dichos fallos, actualizaciones de hardware, apagones extensos, actualizaciones del *kernel*.

2.2.5.3 Administración del Sistema.

La afirmación de que NT es más fácil de administrar debido a su GUI (siglas en inglés de Interfase de Usuario Gráfica) le falta argumentos. La ventaja de cualquier GUI sobre CLI (siglas en inglés Interfase de Línea de Comandos) es cuestionable.

- NT viene con un GUI. No se puede trabajar en el ámbito de CLI.
- UNIX puede trabajar en CLI. Tambien existe una variedad de GUI para cada sistema (OpenLook, X11, etc.)

2.2.5.4 Rendimiento.

La potencia de procesamiento depende principalmente del hardware. Decir que UNIX tiene mejor rendimiento que NT sería inapropiado, si se comparan sistemas de distinta arquitectura. En una misma plataforma, no existen formas para comparar Linux o FreeBSD contra NT. Sin embargo hay consenso en que Linux o FreeBSD tienen mejor rendimiento que NT considerando que los kernels de UNIX se confeccionan a las necesidades, y a veces sólo contienen lo necesario para el sistema, Linux o FreeBSD pueden funcionar más eficientemente que NT. Cualquier sistema operativo que requiera de menos recursos vencerá a un sistema operativo inflado como sería NT.

UNIX no requiere de una interfase gráfica como lo requiere NT. Es bien sabido que las gráficas requieren de mayor espacio de disco y memoria. Lo mismo sucede para archivos de sonidos, que parecen ser muy importantes para el sistema operativo de *Microsoft*.

2.2.5.5 Linux vs. Windows NT Server 4.0.

Debido a que la plataforma que NT utiliza más es INTEL, se escogió al sistema Linux para realizar una comparación (Tabla 2.3).

Componente	Sistema Operativo Linux	Windows NT Server 4.0
Sistema Operativo	Gratis o $49.95 US (CD-ROM)	5-Usuarios $809 US 10-Usuarios $1129 US Enterprise Ed. 25-Usuarios $3,999 US
Soporte técnico gratis en línea	Sí, Linux Online o Redhat	No
Fuentes del Kernel	Sí	No
Servidor Web	Apache Web Server	MS IIS
Servidor FTP	Sí	Sí
Servidor Telnet	Sí	No
Servidor SMTP/POP3	Sí	No
DNS	Sí	Sí
Sistema de archivos de red	NFS y SMB	SMB
Servidor de Ventanas X (Correr aplicaciones remotas basadas en GUI)	Sí	No
Herramientas de Administración Remotas	Sí, todas	Sólo "User Manager for Domains" y "Server Manager"
Servidor de Noticias	Sí	No
Compiladores C y C++	Sí	No
Perl 5.0	Sí	No
Control de Revisión	Sí	No
Número de sistemas de archivo posibles	32	3
Cuotas de Disco	Sí	No
Número de GUI's a escoger	4	1

Tabla 2.3 Comparaciones entre Linux y Windows NT Server 4.0

2.2.5.6 Algunas compañías que usan UNIX.

- Amazon.com Digital UNIX Alpha Server 2000.
- Boeing. HP-UX, IRIX, Solaris y algo de NT.
- Dallas Cowboys. IRIX y UNIX System V Rel. 4.0.
- Dow Corning. Solaris.
- Hotmail (Pertenece a Microsoft). Solaris, FreeBSD. Intentaron emigrar a NT, pero para ofrecer los servicios a más de 10,000,000 de usuarios, fué demasiado para NT.
- Servicio Postal de Estados Unidos. Cuenta con 900 sistemas Linux a lo largo de E.U. para determinar automáticamente el destino de la correspondencia. Cada sistema consiste de 5 Pentium Pro dual 200MHz o un sistema dual sencillo.
- Yahoo!. FreeBSD.

El sistema operativo Linux sería la elección para sitios con presupuesto limitado o en corporaciones enormes que demanden servidores de varios procesadores y que requieran de sistemas operativos escalables.

Una compañía pequeña o mediana, que tenga que correr procesos en tareas no críticas, que esté dispuesto a contratar administradores adicionales para operar el servicio de correo Exchange o el Servidor de Información de Internet de Microsoft (IIS), que tenga presupuesto substancial para licencias de Microsoft "por servidor" o "por usuario", entonces NT sería la elección.

CAPÍTULO III
EL DESARROLLO

A lo largo de los capítulos anteriores se ha visto lo que es la Internet, cómo funciona y se encuentra estructurada, que existen equipos que ofrecen información y toda clase de servicios a través de ésta estructura, además de que toda esa estructura se encuentra sostenida por distintos programas que pueden clasificarse como programas propietario que requieren del pago de licencias y permisos y que existen programas de libre distribución que se sostienen gracias al esfuerzo de miles de voluntarios a través del mundo y que no requieren pago alguno. En éste capítulo se verá cómo es posible implementar, a través de programas de libre distribución y desarrollando algunos programas bajo lenguajes que también son de libre distribución, todo un sistema que pueda permitir a un usuario, remotamente, mover un robot.

El desarrollo se llevó a cabo en 13 etapas:

1. Investigación del Robot a utilizar.
    a. Sistema de control.
    b. Configuración física.
    c. Lenguaje de programación.
    d. Protocolo de comunicación.
2. Fabricación de cables de comunicación entre el controlador y la PC.
3. Búsqueda y selección de un sistema operativo que además de ser de libre distribución permita conectividad al Internet, que permita el desarrollo de programas que controlen los puertos del equipo, que pueda instalarse un programa servidor de WWW.
4. Búsqueda y selección de un programa servidor de servicios WWW que sea compatible con el sistema operativo seleccionado y que además sea de libre distribución.
5. Búsqueda y selección de un lenguaje de programación script compatible con el sistema operativo, el servidor de WWW seleccionado y que además sea de libre distribución.
6. Búsqueda y selección de una base de datos de libre distribución compatible con el sistema operativo seleccionado y que además permita la interacción con el lenguaje script seleccionado.
7. Búsqueda y selección de un programa para cámara Web de libre distribución compatible con el sistema operativo seleccionado y que además permita tomar imágenes que se puedan colocar al público con el servidor de WWW.
8. Instalación y configuración de los programas de libre distribución seleccionados.
    a. Sistema Operativo
    b. Servidor de WWW
    c. Lenguaje script.
    d. Base de Datos
    e. Programa de cámara Web.
9. Desarrollo de programas que permitan controlar el envío de datos a través del puerto de la computadora (control del robot).
10. Desarrollo de programas script que permitan interactuar con los programas de control del robot y el usuario que accede al servidor WWW.
11. Diseño de tablas en la base de datos que permita el almacenamiento de información como usuarios, programas ACL para su posterior ejecución.
12. Desarrollo de pruebas

3.1 CARACTERÍSTICAS DEL ROBOT.

El robot se compone de 7 elementos básicos:

- El Controlador.
- El manipulador.
- Sensores.
- Unidad de potencia externa.
- Efector final.
- Caja de enseñanza.
- Computadora o terminal.

Se utilizó un manipulador SCORBOT-ER V del laboratorio de Manufactura Flexible de la Facultad de Ingeniería. (Figura 3.1)

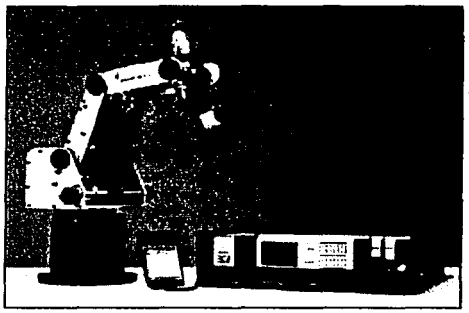


Figura 3.1 Robot SCORBOT, caja de enseñanza y controlador.

3.1.1 Controlador.

El controlador (Figura 3.2) es el centro de comando del robot. El sistema mecánico del brazo no puede funcionar sin los comandos y datos proporcionados por una computadora.

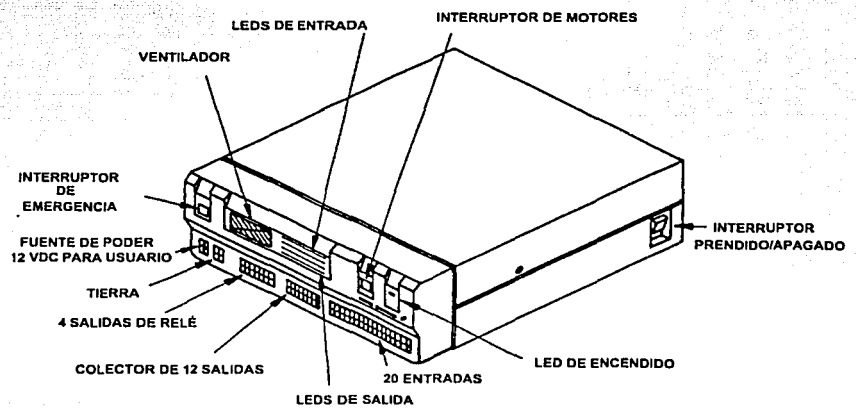


Figura 3.2 Partes del controlador

3.1.2 El manipulador.

El SCORBOT-ER V plus es un robot articulado verticalmente de 5 grados de libertad. Gracias a sus sensores (Figura 3.3) el controlador determina la posición del robot y por medio de los servo-motores realiza los movimientos de sus partes.

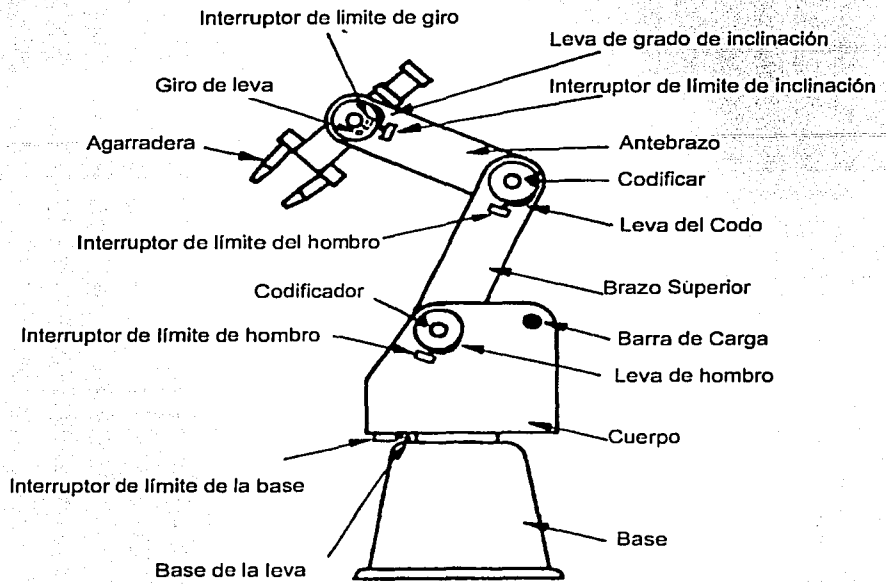


Figura 3.3 Sensores y partes del manipulador

Características		
Objeto	**Especificación**	**Observaciones**
Actuadores	Servo motores eléctricos de CD	
Capacidad del motor(ejes 1-5)	15 oz,in 70 W	Torque máximo (atazque) Potencia para torque máximo
Transmisión	Engranes, bandasdentadas, tornillos de posicionamiento	
Carga máxima de trabajo	1000 [g]	Incluyendo el efector final
Repetibilidad	0.5 [mm]	
Velocidad máxima de trayectoria	600 [mm/s]	
Peso	Aprox. 11 kg Aprox. 11.5 kg	SCORBOT-ER V SCORBOT-ER V plus
Retroalimentación	Codificadores ópticos en todos los ejes	

Tabla 3.4 Características del SCORBOT

3.1.3 Rangos de Operación.

En la tabla 3.5 y en la figura 3.6 se muestran los rango de operación y el espacio que ocupa cuando opera del robot.

Objeto	Especificación	Observaciones
Base	310°	Eje 1
Hombro	+130° -35°	Eje 2
Codo	+-130°	Eje 3
Inclinación de la muñeca	+-130°	Eje 4
Giro de muñeca	Ilimitado (mecánicamente) +-1800° (eléctricamente ER V) +-570° (eléctricamente ER V plus)	Eje 5
Abertura de la pinza	75 [mm](2.95 [in]) 65 [mm] (2.56 [in])	Excluyendo cojinetes de hule Incluyendo cojinetes de hule

Tabla 3.5 Rangos de Operación del robot SCORBOT

Radio de Operación: 610 [mm] (24 [in]).

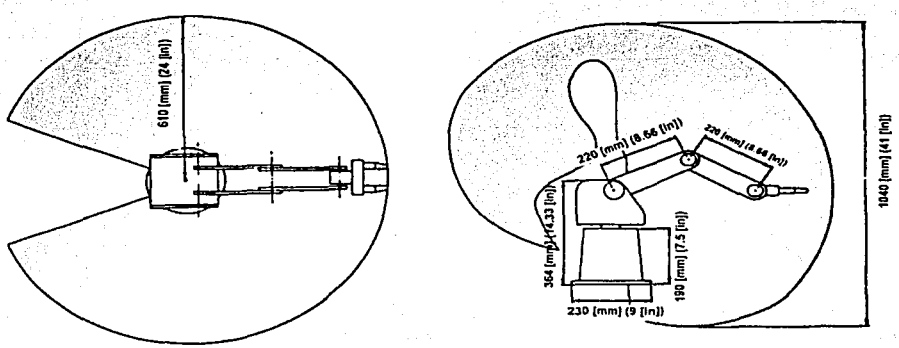


Figura 3.6 Volumen de Trabajo

3.1.4 El efector final.

El efector final (Figura 3.7) es una pinza con la cual el robot puede cargar, sostener y trasladar objetos a lo largo de su rango de operación.

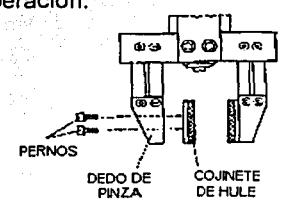


Figura 3.7 Pinza

3.1.5 Caja de enseñanza o teach pendant.

Es un teclado multifuncional (Figura 3.8) que permite, entre otras cosas, activar o desactivar el controlador, mover el robot manualmente en coordenadas XYZ o en modo UNIONES, grabar posiciones, abrir y cerrar la agarradera.

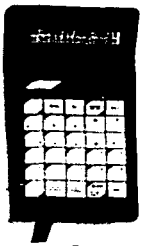


Figura 3.8 Caja de enseñanza o teach pendant.

3.2 LENGUAJE ACL.

ACL es un ambiente de programación para la operación del controlador SCORBOT multitareas.

ACL proporciona las siguientes funciones:

- Ejecuta comandos directamente al robot.
- Controla datos de entrada / salida.
- Habilita al usuario para programar el robot.
- Corre todos los programas seleccionados al mismo tiempo (soporta multitareas).
- Sincroniza la ejecución de los programas.

3.2.1 Comandos ACL.

Los comandos están divididos, en general, en dos grupos:

1. Comandos de modo directo, llamado modo **MONITOR o DIRECT,** permiten que el comando sea ejecutado inmediatamente.
2. Comandos de modo indirecto, llamado modo **EDITOR,** permite que el usuario escriba programas que serán posteriormente ejecutados por el comando RUN.

Para ver una lista de algunos comandos vea el Apéndice B.

3.3 Protocolo de comunicación con el controlador.

Se investigó la manera en que la PC se comunica con el controlador ACL y se llegó a la conclusión de que el controlador ACL recibe series de caracteres a través del cable RS-232 y éste a su vez almacena cada carácter en un buffer y envía cada uno de los caracteres enviados de regreso a la PC, al recibir un carácter de fin de línea ("\n") el controlador trata de interpretar la serie de caracteres recibidos en el buffer y los ejecuta, si la serie de caracteres es un comando ACL reconocido por el controlador lo ejecuta. No se tiene un control de errores propiamente, es decir, que el controlador no regresa ningún carácter de control para indicar si el comando enviado fue ejecutado por el robot de manera correcta.

3.3.1 Protocolo de comunicación con el controlador.

Para comunicar la PC con el controlador ACL es necesario conectarse a través del puerto de comunicaciones RS-232, para lo cual se debe contar con un puerto serial en la PC y un cable de comunicación debidamente fabricado. Para fabricar el cable (Ver Apéndice C.1.2) se debe contar con los siguientes materiales:

- 1 conector RS-232 hembra DB9 (Conexión a PC).
- 1 conector RS-232 hembra DB25 (Conexión al controlador ACL).
- Cautín.
- Soldadura.
- Cable de impresora
- Diagrama de conexiones

Se deben unir cada uno de los pines (puntas en forma de agujas) de cada conector con el cautín y la soldadura de acuerdo al diagrama ubicado en el Apéndice C.1.2.

3.4 EL PROYECTO.

El sistema propuesto es un conjunto de programas de libre distribución para lograr el movimiento remoto del SCORBOT por medio de una página Web colocada al público.

3.4.1 Descripción.

3.4.1.1 Conexiones.

El equipo PC, denominado SERVIDOR, con el sistema operativo Linux instalado, es conectado a través de una tarjeta de red a la Internet, a su vez se encuentra conectado, con el cable de conexión serial fabricado como se describe en 3.3.1 al controlador ACL del SCORBOT, los motores del ROBOT SCORBOT se encuentran conectados al controlador. La cámara Web se encuentra conectada a través de un cable USB al SERVIDOR. El usuario se conecta desde su equipo a la Internet y por medio de un navegador Web accede al SERVIDOR (Figura 3.9).

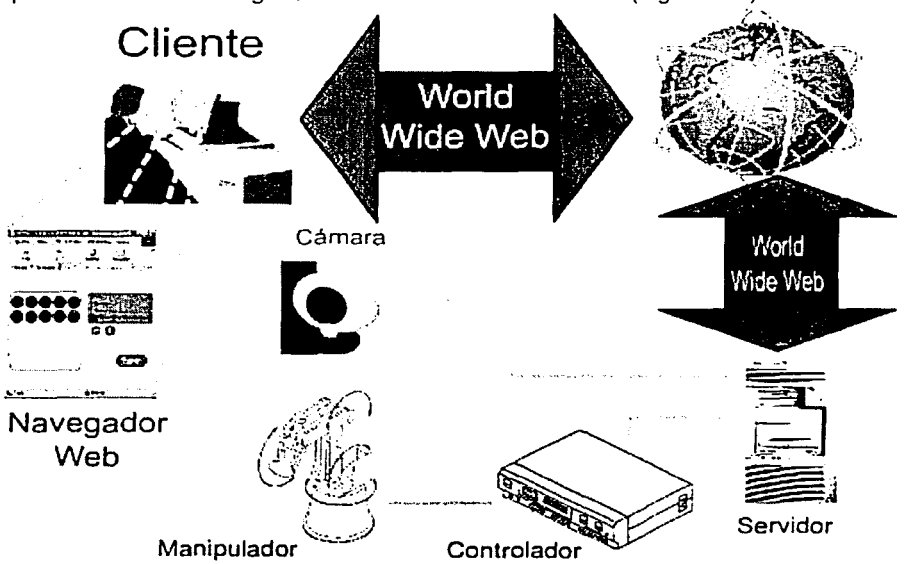


Figura 3.9 Diagrama de conexión del controlador, PC y robot

3.4.1.2 Relación entre programas.

El SERVIDOR, gracias al programa servidor Web (Apache), muestra programas script escritos en PHP, estos permiten ejecutan los programas escritos en C que a su vez mandan comandos de lenguaje ACL por medio del puerto serial del SERVIDOR a través de un cable RS-232 al CONTROLADOR ACL, el CONTROLADOR manda las señales a los servomotores del SCORBOT proporcionando así el movimiento (Figura 3.10).

El SERVIDOR permite el uso del programa sólo a ciertos usuarios gracias a que dentro de la BASE DE DATOS se encuentran almacenados aquellos usuarios que tienen derecho a usar el sistema, al mismo tiempo el SERVIDOR ejecuta un programa llamado webcam, el cual permite tomar imágenes en un intervalo de tiempo indicado y que almacena en un archivo que también es ofrecido por el SERVIDOR WEB.

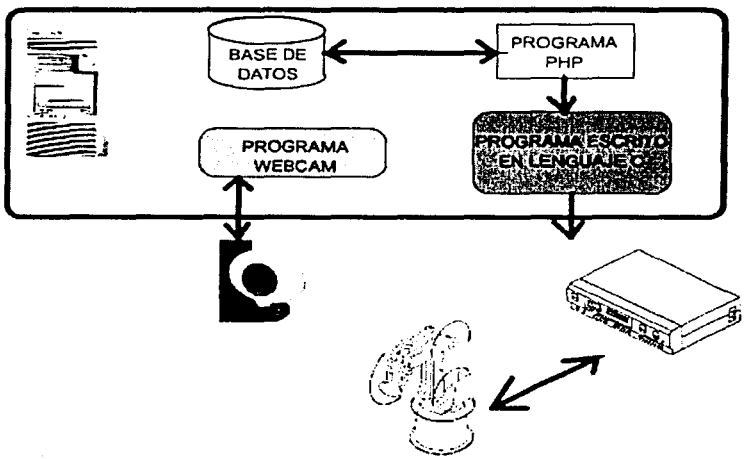


Figura 3.10 Diagrama de programas y su relación.

3.4.2 Hardware y software utilizado.

Para el desarrollo se utilizaron las siguientes partes de hardware que aparecen en la tabla 3.11 y en la tabla 3.12 el software de libre distribución que fue seleccionado para la elaboración del proyecto.

Hardware
PC Pentium II 600MHZ
132 MB RAM
3.4 GB en Disco Duro
Cámara Digital Life View.
Controlador ACL
Manipulador SCORBOT-ER V
Cable de comunicaciones Serial

Tabla 3.11 Hardware utilizado para el proyecto.

Software	
Sistema Operativo	Linux RedHat versión 7.1 kernel 2.2.4
Servidor Web	Apache ver. 1.3.4
Lenguaje Script	PHP ver. 4
Base de Datos	PostgreSQL 7.0
Programa para cámara web.	Programa webcam.
Navegador de WWW.	Netscape 6.2

Tabla 3.12 Software utilizado para el proyecto.

3.4.2.1 El Sistema Operativo Linux.

El sistema *Linux* o *FreeBSD* pueden fácilmente rebasar el rendimiento y funcionalidad de una solución NT, realizarlo sobre una plataforma INTEL relativamente económica y cuyo costo sería de un precio que difícilmente se puede mejorar con un sistema NT.

El sistema Linux gana en definitiva.

- Ofrece una variedad de proveedores (no hay monopolio).
- Es escalable.
- Hace uso de los recursos más eficientemente.
- Permite la administración remota del sistema.
- Ofrece la capacidad de realizar cómputo remotamente.
- Posee capacidad de multiusuarios.
- Existe una gran variedad de software.
- Los estándares son independientes de los proveedores (POSIX).
- Tiene control sobre el espacio de disco de los usuarios.
- No le afectan los virus diseñados para MS-DOS.

En síntesis:

- Windows NT sólo corre en plataformas INTEL o Alpha, no tiene CLI, tiene un sólo GUI, sólo hay agentes de transferencia de correo comerciales, etc.
- Windows NT es *estático*, es decir no es posible construir un *kernel* apropiado.
- Microsoft se está convirtiendo en un monopolio, no en un estándard.
- UNIX ofrece opciones. Corre en cualquier tipo de plataforma, trabaja con CLI o GUI, opciones de software comercial y *freeware*, diversos proveedores a escoger.
- UNIX es *dinámico*, se puede configurar un *kernel* apropiado a las necesidades de cualquier persona o proyecto.

Para su instalación basta con conseguir los CD´s de instalación que se encuentran disponibles en la Internet. Existen diferentes distribuciones de Linux dentro de las que destacan:

- RedHat www.redhat.com
- SuSe www.suse.com
- Debian www.debian.org
- Slakeware www.linux.org

Se eligió la distribución de RedHat ya que es una de las distribuciones más estables y de mayor facilidad en la instalación.

3.4.2.2 El Servidor de WWW Apache.

Para el sistema operativo Linux existe sólo un programa servidor Web de libre distribución, el Servidor Web Apache, el cual es además uno de los más rápidos que existen.

Para su instalación basta con conseguir los archivos de instalación o códigos fuente que se encuentran disponibles en la Internet en el sitio www.apache.org. Se desarrolló un programa escrito en bash para Linux que puede ejecutarse en la línea de comandos de Linux que sirve como ayuda en la instalación del Servidor de WWW Apache, PHP y PostgreSQL en Linux (Ver Apéndice A.1).

3.4.2.3 El lenguaje script PHP.

Para el servidor Apache, como ya se mencionó en el Capitulo I (1.1.5.3.3.3) existe un proyecto de lenguaje script denominado PHP y que a últimas fechas se ha convertido en uno de los lenguajes más utilizados dentro del desarrollo de portales en la Internet.

Para su instalación basta con conseguir los archivos de instalación o códigos fuente que se encuentran disponibles en la Internet en el sitio www.php.net. (Ver Apéndice A.1).

3.4.2.4 La base de datos PostgreSQL.

Dentro de las bases de datos que existen para Linux, sólo una de ellas es de libre distribución: Postgresql y además tiene características que nos permitirán alcanzar los objetivos de la tesis como es el almacenamiento de archivos, manejo de SQL.

Para su instalación basta con conseguir los archivos de instalación o códigos fuente que se encuentran disponibles en la Internet en el sitio www.postgresql.org. (Ver Apéndice A.2).

3.4.3 Diseño de la Base de Datos.

La base de datos funciona como almacenamiento de las posiciones que se graban en el controlador para el robot SCORBOT, así como de los usuarios que tienen derecho a acceder al sistema.

Para diseñar la base de datos debemos conocer un poco de SQL y Postgresql.

3.4.3.1 SQL.

SQL es un método basado en un potente lenguaje, para organizar, administrar y consultar datos almacenados en una computadora. SQL es una sigla que deviene de su nombre en inglés "Structured Query Language" (Lenguaje de Consulta Estructurado). Más específicamente SQL esta definido en torno al modelo de bases de datos relacionales, basado en el álgebra relacional, esto le da a SQL las ventajas que lo imponen como el sistema de mayor aceptación. Algunas de las ventajas son:

- Marco teórico sólido, fundamentado en el álgebra relacional
- Simplicidad de conceptos (modelo de base de datos: tablas = líneas x columnas)
- Definición de vínculos en la consulta, esto le da a SQL una gran flexibilidad
- Fácil y rápido aprendizaje
- Arquitectura cliente-servidor
- Integración con cualquier lenguaje de programación
- Estandarización

IBM empezó a comercializar en 1981 el SQL y desde entonces este producto ha tenido un papel importante en el desarrollo de la bases de datos relacionales. IBM propuso y fue aceptada , una versión de SQL al Instituto de Estándares Nacional Americano(ANSI) y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

3.4.3.1.1 Comandos SQL.

Existen dos tipos de comandos SQL:

- Los CYA (Creación y Alteración) que permiten crear y definir nuevas bases de datos, campos e índices. (Tabla 3.13)
- Los BYM (Búsqueda y Modificación) que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos. (Tabla 3.14)

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Tabla 3.13 Comandos de creación y alteración. (CYA)

Comando	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

Tabla 3.14 Comandos de Búsqueda y Modificación (BYM)

3.4.3.1.1.1 Cláusulas.

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular. (Tabla 3.15)

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

Tabla 3.15 Cláusulas SQL.

3.4.3.1.1.2 Operadores lógicos.

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Tabla 3.16 Operadores lógicos.

3.4.3.1.1.3 Operadores de comparación.

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo

Tabla 3.17 Operadores de comparación.

3.4.3.1.1.4 Funciones de agregado.

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros (Tabla 3.18).

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de la selección.
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado.
MAX	Utilizada para devolver el valor más alto de un campo especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado.

Tabla 3.18 Funciones de agregado.

3.4.3.1.2 Consultas de selección.

Las consultas de selección se utilizan para indicar a PostgreSQL que devuelva información de la base de datos, esta información es devuelta en forma de conjunto de registros.

3.4.3.1.2.1 Consultas básicas.

La sintaxis básica de una consulta de selección es la siguiente:

```
SELECT Campos FROM Tabla;
```

En donde campos es la lista de campos que se deseen recuperar y tabla es el origen de los mismos, por ejemplo:

```
SELECT Nombre, Telefono FROM Clientes;
```

Esta consulta devuelve los campos nombre y teléfono de la tabla clientes.

3.4.3.1.2.2 Ordenar los registros.

Adicionalmente se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la cláusula ORDER BY Lista de Campos. En donde Lista de campos representa los campos a ordenar. Ejemplo:

```
SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY Nombre;
```

Esta consulta devuelve los campos CodigoPostal, Nombre, Telefono de la tabla Clientes ordenados por el campo Nombre.
Se pueden ordenar los registros por mas de un campo, como por ejemplo:

```
SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY
    CodigoPostal, Nombre;
```

Incluso se puede especificar el orden de los registros: ascendente mediante la cláusula (ASC -se toma este valor por defecto) ó descendente (DESC)

```
SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY
CodigoPostal DESC , Nombre ASC;
```

3.4.3.1.2.3 Alias.

En determinadas circunstancias es necesario asignar un nombre a alguna columna determinada de un conjunto devuelto, otras veces por simple capricho o por otras circunstancias. Para resolver todas ellas tenemos la palabra reservada AS que se encarga de asignar el nombre que deseamos a la columna deseada. Tomado como referencia el ejemplo anterior podemos hacer que la columna devuelta por la consulta, en lugar de llamarse apellido (igual que el campo devuelto) se llame Empleado. En este caso procederíamos de la siguiente forma:

```
SELECT Apellido AS Empleado FROM Empleados;
```

3.4.3.1.3 Criterios de selección.

Se ha visto la forma de recuperar los registros de las tablas, las formas empleadas devolvían todos los registros de la mencionada tabla. A lo largo de este capítulo se estudiarán las posibilidades de filtrar los registros con el fin de recuperar solamente aquellos que cumplan las condiciones preestablecidas.

3.4.3.1.3.1 Operadores lógicos.

Los operadores lógicos (Tabla 3.16) soportados por SQL son: AND, OR, XOR, IS y NOT. A excepción de los dos últimos todos poseen la siguiente sintaxis:

```
<expresión1>  operador  <expresión2>
```

En donde expresión1 y expresión2 son las condiciones a evaluar, el resultado de la operación varía en función del operador lógico. La tabla 3.19 muestra los diferentes posibles resultados.

<expresión1>	Operador	<expresión2>	Resultado
Verdad	AND	Falso	Falso
Verdad	AND	Verdad	Verdad
Falso	AND	Verdad	Falso
Falso	AND	Falso	Falso
Verdad	OR	Falso	Verdad
Verdad	OR	Verdad	Verdad
Falso	OR	Verdad	Verdad
Falso	OR	Falso	Falso
Verdad	XOR	Verdad	Falso
Verdad	XOR	Falso	Verdad
Falso	XOR	Verdad	Verdad
Falso	XOR	Falso	Falso

Tabla 3.19 Posibles resultados de los operadores lógicos.

Si a cualquiera de las anteriores condiciones se le antepone el operador NOT el resultado de la operación será el contrario al devuelto sin el operador NOT.

```
SELECT * FROM Empleados WHERE Edad > 25 AND Edad < 50;
SELECT * FROM Empleados WHERE (Edad > 25 AND Edad < 50) OR
Sueldo = 100;
SELECT * FROM Empleados WHERE NOT Estado = 'Soltero';
SELECT * FROM Empleados WHERE (Sueldo > 100 AND Sueldo < 500)
OR  (Provincia = 'Madrid' AND Estado = 'Casado');
```

3.4.3.1.3.2 Intervalos de valores.

Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador Between cuya sintaxis es:

campo [Not] Between valor1 And valor2 (la condición Not es opcional)

En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si anteponemos la condición Not devolverá aquellos valores no incluidos en el intervalo.

```
SELECT * FROM Pedidos WHERE CodPostal Between 28000 And 28999;
(Devuelve los pedidos realizados en la provincia de Madrid)

SELECT IIf(CodPostal Between 28000 And 28999, 'Provincial', 'Nacional')
FROM Editores;
(Devuelve el valor 'Provincial' si el código postal se encuentra en el intervalo,
'Nacional' en caso contrario)
```

3.4.3.1.3.3 Operador LIKE.

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

expresión LIKE modelo

En donde expresión es una cadena modelo o campo contra el que se compara expresión. Se puede utilizar el operador LIKE para encontrar valores en los campos que coincidan con el modelo especificado. Por modelo puede especificar un valor completo (Ana María), o se pueden utilizar caracteres comodín como los reconocidos por el sistema operativo para encontrar un rango de valores (Like An*).

El operador Like se puede utilizar en una expresión para comparar un valor de un campo con una expresión de cadena. Por ejemplo, si introduce Like C* en una consulta SQL, la consulta devuelve todos los valores de campo que comiencen por la letra C. En una consulta con parámetros, puede hacer que el usuario escriba el modelo que se va a utilizar.

El ejemplo siguiente devuelve los datos que comienzan con la letra P seguido de cualquier letra entre A y F y de tres dígitos:

```
LIKE 'P[A-F]###'
```

Este ejemplo devuelve los campos cuyo contenido empiece con una letra de la A a la D seguidas de cualquier cadena.

```
LIKE '[A-D]*'
```

En la tabla 3.20 se muestra cómo utilizar el operador LIKE para comprobar expresiones con diferentes modelos.

Tipo de coincidencia	Modelo Planteado	Coincide	No coincide
Varios caracteres	'a*a'	'aa', 'aBa', 'aBBBa'	'aBC'
Carácter especial	'a[*]a'	'a*a'	'aaa'
Varios caracteres	'ab*'	'abcdefg', 'abc'	'cab', 'aab'
Un solo carácter	'a?a'	'aaa', 'a3a', 'aBa'	'aBBBa'
Un solo dígito	'a#a'	'a0a', 'a1a', 'a2a'	'aaa', 'a10a'
Rango de caracteres	'[a-z]'	'f', 'p', 'j'	'2', '&'
Fuera de un rango	'[!a-z]'	'9', '&', '%'	'b', 'a'
Distinto de un dígito	'[!0-9]'	'A', 'a', '&', '~'	'0', '1', '9'
Combinada	'a[!b-m]#'	'An9', 'az0', 'a99'	'abc', 'aj0'

Tabla 3.20 Ejemplos de expresiones para usarse con el operador LIKE.

3.4.3.1.3.4 Operador IN.

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los en una lista. Su sintaxis es:

expresión [Not] In(valor1, valor2, . . .)

```
SELECT * FROM Pedidos WHERE Provincia In ('Madrid', 'Barcelona', 'Sevilla');
```

3.4.3.1.3.5 Cláusula WHERE.

La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. Después de escribir esta cláusula se deben especificar las condiciones expuestas en los apartados 3.4.3.1.3.1 y 3.4.3.1.3.2. Si no se emplea esta cláusula, la consulta devolverá todas las filas de la tabla. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM (Figura 3.21).

```
SELECT Apellidos, Salario FROM Empleados WHERE Salario > 21000;
SELECT Id_Producto, Existencias FROM Productos
WHERE Existencias <= Nuevo_Pedido;
SELECT * FROM Pedidos WHERE Fecha_Envio = #5/10/94#;
SELECT Apellidos, Nombre FROM Empleados WHERE Apellidos = 'King';
SELECT Apellidos, Nombre FROM Empleados WHERE Apellidos Like 'S*';
SELECT Apellidos, Salario FROM Empleados WHERE Salario Between 200 And 300;
SELECT Apellidos, Salario FROM Empl WHERE Apellidos Between 'Lon' And 'Tol';
SELECT Id_Pedido, Fecha_Pedido FROM Pedidos WHERE Fecha_Pedido
Between #1-1-94# And  #30-6-94#;
SELECT Apellidos, Nombre, Ciudad FROM Empleados WHERE Ciudad
In ('Sevilla', 'Los Angeles', 'Barcelona');
```

Figura 3.21 Ejemplos del uso de la cláusula WHERE.

3.4.3.1.4 Agrupamiento de registros.

Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro. Para cada registro se crea un valor sumario si se incluye una función SQL agregada, como por ejemplo Sum o Count, en la instrucción SELECT. Su sintaxis es:

```
SELECT campos FROM tabla WHERE criterio GROUP BY campos del grupo
```

GROUP BY es opcional. Los valores de resumen se omiten si no existe una función SQL agregada en la instrucción SELECT. Los valores Null en los campos GROUP BY se agrupan y no se omiten.

Se utiliza la cláusula WHERE para excluir aquellas filas que no desea agrupar, y la cláusula HAVING para filtrar los registros una vez agrupados.

A menos que contenga un dato text u Objeto, un campo de la lista de campos GROUP BY puede referirse a cualquier campo de las tablas que aparecen en la cláusula FROM, incluso si el campo no esta incluido en la instrucción SELECT, siempre y cuando la instrucción SELECT incluya al menos una función SQL agregada.

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función SQL agregada.

```
SELECT Id_Familia, Sum(Stock) FROM Productos GROUP BY Id_Familia;
```

Una vez que GROUP BY ha combinado los registros, HAVING muestra cualquier registro agrupado por la cláusula GROUP BY que satisfaga las condiciones de la cláusula HAVING.

HAVING es similar a WHERE, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuales de ellos se van a mostrar.

```
SELECT Id_Familia Sum(Stock) FROM Productos GROUP BY Id_Familia
  HAVING Sum(Stock) > 100 AND NombreProducto Like BOS*;
```

3.4.3.1.5 Funciones de agregado.

3.4.3.1.5.1 AVG.

Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta. Su sintaxis es la siguiente:

```
Avg(expr)
```

En donde expr representa el campo que contiene los datos **numéricos** para los que se desea calcular la media o una expresión que realiza un cálculo utilizando los datos de dicho campo. La media calculada por Avg es la media aritmética (la suma de los valores dividido por el número de valores). La función Avg no incluye ningún campo Null en el cálculo.

```
SELECT Avg(Gastos) AS Promedio FROM Pedidos WHERE Gastos > 100;
```

3.4.3.1.5.2 Count.

Calcula el número de registros devueltos por una consulta. Su sintaxis es la siguiente:

```
Count(expr)
```

En donde expr contiene el nombre del campo que desea contar. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL). Puede contar cualquier tipo de datos incluso texto.

Aunque expr puede realizar un cálculo sobre un campo, Count simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan en los registros.

```
SELECT Count(*) AS Total FROM Pedidos;
```

Si expr identifica a múltiples campos, la función Count cuenta un registro sólo si al menos uno de los campos no es Null. Si todos los campos especificados son Null, no se cuenta el registro.

```
SELECT Count(FechaEnvío & Transporte) AS Total FROM Pedidos;
```

3.4.3.1.5.3 Max, Min.

Devuelven el mínimo o el máximo de un conjunto de valores contenidos en un campo especifico de una consulta. Su sintaxis es:

```
Min(expr)
Max(expr)
```

En donde expr es el campo sobre el que se desea realizar el cálculo. Expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

```
SELECT Min(Gastos) AS ElMin FROM Pedidos WHERE País = 'España';
SELECT Max(Gastos) AS ElMax FROM Pedidos WHERE País = 'España';
```

3.4.3.1.5.4 Sum.

Devuelve la suma del conjunto de valores contenido en un campo especifico de una consulta. Su sintaxis es:

```
Sum(expr)
```

En donde expr respresenta el nombre del campo que contiene los datos que desean sumarse o una expresión que realiza un cálculo utilizando los datos de dichos campos. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

```
SELECT Sum(PrecioUnidad * Cantidad) AS Total FROM DetallePedido;
```

3.4.3.1.6 Consultas de acción.

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir y borrar y modificar registros.

3.4.3.1.6.1 DELETE.

Crea una consulta de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es:

```
DELETE FROM Tabla WHERE criterio
```

DELETE es especialmente útil cuando se desea eliminar varios registros.

Una vez que se han eliminado los registros utilizando una consulta de borrado, no puede deshacer la operación. Si desea saber qué registros se eliminarán, primero examine los resultados de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de borrado. Mantenga copias de seguridad de sus datos en todo momento. Si elimina los registros equivocados podrá recuperarlos desde las copias de seguridad.

```
DELETE FROM Empleados WHERE Cargo = 'Vendedor';
```

3.4.3.1.6.2 INSERT INTO.

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipo: Insertar un único registro ó Insertar en una tabla los registros contenidos en otra tabla.

3.4.3.1.6.2.1 Insertar un registro único.

En este caso la sintaxis es la siguiente:

```
INSERT INTO Tabla (campo1, campo2, .., campoN)
  VALUES (valor1, valor2, ..., valorN)
```

Esta consulta graba en el campo1 el valor1, en el campo2 y valor2 y así sucesivamente. Hay que prestar especial atención a acotar entre comillas simples (') los valores literales (cadenas de caracteres).

3.4.3.1.6.2.2 Insertar registros de otra tabla.

En este caso la sintaxis es:

```
INSERT INTO Tabla [IN base_externa] (campo1, campo2, ..., campoN)
SELECT TablaOrigen.campo1, TablaOrigen.campo2, ...,
TablaOrigen.campoN FROM TablaOrigen
```

En este caso se seleccionarán los campos 1,2, ..., n de la tabla origen y se grabarán en los campos 1,2,.., n de la Tabla. La condición SELECT puede incluir la cláusula WHERE para filtrar los registros a copiar. Si Tabla y TablaOrigen poseen la misma estructura podemos simplificar la sintaxis a:

```
INSERT INTO Tabla  SELECT TablaOrigen.* FROM TablaOrigen
```

De esta forma los campos de TablaOrigen se grabarán en Tabla, para realizar esta operación es necesario que todos los campos de TablaOrigen estén contenidos con igual nombre en Tabla. Con otras palabras que Tabla posea todos los campos de TablaOrigen (igual nombre e igual tipo).

En este tipo de consulta hay que tener especial atención con los campos contadores o autonuméricos puesto que al insertar un valor en un campo de este tipo se escribe el valor que contenga su campo homólogo en la tabla origen, no incrementandose como le corresponde.

Se puede utilizar la instrucción INSERT INTO para agregar un registro único a una tabla, utilizando la sintaxis de la consulta de adición de registro único tal y como se mostró anteriormente. En este caso, su código especifica el nombre y el valor de cada campo del registro. Debe especificar cada uno de los campos del registro al que se le va a asignar un valor así como el valor para dicho campo. Cuando no se especifica dicho campo, se inserta el valor predeterminado o Null. Los registros se agregan al final de la tabla.

También se puede utilizar INSERT INTO para agregar un conjunto de registros pertenecientes a otra tabla o consulta utilizando la cláusula SELECT ... FROM como se mostró anteriormente en la sintaxis de la consulta de adición de múltiples registros. En este caso la cláusula SELECT especifica los campos que se van a agregar en la tabla destino especificada. La tabla destino u origen puede especificar una tabla o una consulta. Si la tabla destino contiene una clave principal, hay que asegurarse que es única, y con valores no-Null ; si no es así, no se agregarán los registros. Si se agregan registros a una tabla con un campo Contador , no se debe incluir el campo Contador en la consulta. Se puede emplear la cláusula IN para agregar registros a una tabla en otra base de datos.

Se pueden averiguar los registros que se agregarán en la consulta ejecutando primero una consulta de selección que utilice el mismo criterio de selección y ver el resultado. Una consulta de adición copia los registros de una o más tablas en otra. Las tablas que contienen los registros que se van a agregar no se verán afectadas por la consulta de adición. En lugar de agregar registros existentes en otra tabla, se puede especificar los valores de cada campo en un nuevo registro utilizando la cláusula VALUES. Si se omite la lista de campos, la cláusula VALUES debe incluir un valor para cada campo de la tabla, de otra forma fallará INSERT.

```
INSERT INTO Clientes SELECT Clientes_Viejos.* FROM Clientes_Nuevos;
INSERT INTO Empleados (Nombre, Apellido, Cargo)
VALUES ('Luis', 'Sánchez', 'Becario');
INSERT INTO Empleados SELECT Vendedores.* FROM Vendedores
WHERE Fecha_Contratacion < Now() - 30;
```

3.4.3.1.6.3 UPDATE.

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

```
UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2, ... CampoN=ValorN WHERE Criterio;
```

UPDATE es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez. El ejemplo siguiente incrementa los valores Cantidad pedidos en un 10 por ciento y los valores Transporte en un 3 por ciento para aquellos que se hayan enviado al Reino Unido.:

```
UPDATE Pedidos SET Pedido = Pedidos * 1.1, Transporte = Transporte * 1.03
WHERE PaisEnvío = 'ES';
```

UPDATE no genera ningún resultado. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización.

```
UPDATE Empleados SET Grado = 5 WHERE Grado = 2;
UPDATE Productos SET Precio = Precio  * 1.1 WHERE Proveedor = 8 AND Familia = 3;
```

Si en una consulta de actualización suprimimos la cláusula WHERE todos los registros de la tabla señalada serán actualizados.

```
UPDATE Empleados SET Salario = Salario * 1.1
```

3.4.3.1.7 Subconsultas.

Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, SELECT...INTO, INSERT...INTO, DELETE, o UPDATE o dentro de otra subconsulta. Puede utilizar tres formas de sintaxis para crear una subconsulta:

comparación [ANY | ALL | SOME] (instrucción SQL)
expresión [NOT] IN (instrucción SQL)
[NOT] EXISTS (instrucción SQL)

En donde:

comparación
Es una expresión y un operador de comparación que compara la expresión con el resultado de la subconsulta.

expresión
Es una expresión por la que se busca el conjunto resultante de la subconsulta.

instrucción sql
Es una instrucción SELECT, que sigue el mismo formato y reglas que cualquier otra instrucción SELECT. Debe ir entre paréntesis.

Se puede utilizar una subconsulta en lugar de una expresión en la lista de campos de una instrucción SELECT o en una cláusula WHERE o HAVING. En una subconsulta, se utiliza una instrucción SELECT para proporcionar un conjunto de uno o más valores especificados para evaluar en la expresión de la cláusula WHERE o HAVING.

Se puede utilizar el predicado ANY o SOME, los cuales son sinónimos, para recuperar registros de la consulta principal, que satisfagan la comparación con cualquier otro registro recuperado en la subconsulta. El ejemplo siguiente devuelve todos los productos cuyo precio unitario es mayor que el de cualquier producto vendido con un descuento igual o mayor al 25 por ciento:

```
SELECT * FROM Productos WHERE PrecioUnidad > ANY
(SELECT PrecioUnidad FROM DetallePedido WHERE
Descuento >= 0 .25);
```

El predicado ALL se utiliza para recuperar únicamente aquellos registros de la consulta principal que satisfacen la comparación con todos los registros recuperados en la subconsulta. Si se cambia ANY por ALL en el ejemplo anterior, la consulta devolverá únicamente aquellos productos cuyo precio unitario sea mayor que el de todos los productos vendidos con un descuento igual o mayor al 25 por ciento. Esto es mucho más restrictivo.

El predicado IN se emplea para recuperar únicamente aquellos registros de la consulta principal para los que algunos registros de la subconsulta contienen un valor igual. El ejemplo siguiente devuelve todos los productos vendidos con un descuento igual o mayor al 25 por ciento:

```
SELECT * FROM Productos WHERE IDProducto IN
(SELECT IDProducto FROM DetallePedido WHERE Descuento >= 0.25);
```

Inversamente se puede utilizar NOT IN para recuperar únicamente aquellos registros de la consulta principal para los que no hay ningún registro de la subconsulta que contenga un valor igual.

El predicado EXISTS (con la palabra reservada NOT opcional) se utiliza en comparaciones de verdad/falso para determinar si la subconsulta devuelve algún registro.

Se puede utilizar también alias del nombre de la tabla en una subconsulta para referirse a tablas listadas en la cláusula FROM fuera de la subconsulta. El ejemplo siguiente devuelve los nombres de los empleados cuyo salario es igual o mayor que el salario medio de todos los empleados con el mismo título. A la tabla Empleados se le ha dado el alias T1:

```
SELECT Apellido, Nombre, Titulo, Salario FROM Empleados AS T1
WHERE Salario >= (SELECT Avg(Salario) FROM Empleados
WHERE T1.Titulo = Empleados.Titulo) ORDER BY Titulo;
```

En el ejemplo anterior , la palabra reservada AS es opcional.

```
SELECT Apellidos, Nombre, Cargo, Salario FROM Empleados
WHERE Cargo LIKE "Agente Ven*" AND Salario > ALL (SELECT Salario FROM
Empleados WHERE (Cargo LIKE "*Jefe*") OR (Cargo LIKE "*Director*"));
```

Obtiene una lista con el nombre, cargo y salario de todos los agentes de ventas cuyo salario es mayor que el de todos los jefes y directores.

```
SELECT DISTINCTROW NombreProducto, Precio_Unidad FROM Productos
WHERE (Precio_Unidad = (SELECT Precio_Unidad FROM Productos WHERE
Nombre_Producto = "Almíbar anisado");
```

Obtiene una lista con el nombre y el precio unitario de todos los productos con el mismo precio que el almíbar anisado.

```
SELECT DISTINCTROW Nombre_Contacto, Nombre_Compañia, Cargo_Contacto,
Telefono FROM Clientes WHERE (ID_Cliente IN (SELECT DISTINCTROW
ID_Cliente FROM Pedidos WHERE Fecha_Pedido >= #04/1/93# <#07/1/93#);
```

Obtiene una lista de las compañías y los contactos de todos los clientes que han realizado un pedido en el segundo trimestre de 1993

.

```
SELECT Nombre, Apellidos FROM Empleados AS E WHERE EXISTS
(SELECT * FROM Pedidos AS O WHERE O.ID_Empleado = E.ID_Empleado);
```

Selecciona el nombre de todos los empleados que han reservado al menos un pedido.

```
SELECT DISTINCTROW Pedidos.Id_Producto, Pedidos.Cantidad,
(SELECT DISTINCTROW Productos.Nombre FROM Productos WHERE
Productos.Id_Producto = Pedidos.Id_Producto) AS ElProducto FROM
Pedidos WHERE Pedidos.Cantidad > 150 ORDER BY Pedidos.Id_Producto;
```

Recupera el Código del Producto y la Cantidad pedida de la tabla pedidos, extrayendo el nombre del producto de la tabla de productos.

3.4.3.1.8 Consultas de unión internas.

Las vinculaciones entre tablas se realiza mediante la cláusula INNER que combina registros de dos tablas siempre que haya concordancia de valores en un campo común. Su sintaxis es:

```
SELECT campos FROM tb1 INNER JOIN tb2 ON tb1.campo1 comp tb2.campo2;
```

En donde:

tb1, tb2
Son los nombres de las tablas desde las que se combinan los registros.

campo1, campo2
Son los nombres de los campos que se combinan. Si no son numéricos, los campos deben ser del mismo tipo de datos y contener el mismo tipo de datos, pero no tienen que tener el mismo nombre.

comp
Es cualquier operador de comparación relacional : =, <, >, <=, >=, o <>.

Se puede utilizar una operación INNER JOIN en cualquier cláusula FROM. Esto crea una combinación por equivalencia, conocida también como unión interna. Las combinaciones Equi son las más comunes; éstas combinan los registros de dos tablas siempre que haya concordancia de valores en un campo común a ambas tablas. Se puede utilizar INNER JOIN con las tablas Departamentos y Empleados para seleccionar todos los empleados de cada departamento.

Si se intenta combinar campos que contengan datos text u Objeto, se produce un error. Se pueden combinar dos campos numéricos cualesquiera, incluso si son de diferente tipo de datos. Por ejemplo, puede combinar un campo Numérico para el que la propiedad Size de su objeto Field está establecida como Entero, y un campo Contador.

El ejemplo siguiente muestra cómo podría combinar las tablas Categorías y Productos basándose en el campo IDCategoria:

```
SELECT Nombre_Categoría, NombreProducto
FROM Categorias INNER JOIN Productos
ON Categorias.IDCategoria = Productos.IDCategoria;
```

En el ejemplo anterior, IDCategoria es el campo combinado, pero no está incluido en la salida de la consulta ya que no está incluido en la instrucción SELECT. Para incluir el campo combinado, incluir el nombre del campo en la instrucción SELECT, en este caso, Categorias.IDCategoria.
También se pueden enlazar varias cláusulas ON en una instrucción JOIN, utilizando la sintaxis siguiente:

```
SELECT campos
FROM tabla1 tb1 INNER JOIN tabla2 tb2
ON tb1.campo1 comp tb2.campo1 AND
ON tb1.campo2 comp tb2.campo2) OR
ON tb1.campo3 comp tb2.campo3)];
```

También puede anidar instrucciones JOIN utilizando la siguiente sintaxis:

```
SELECT campos FROM tb1 INNER JOIN
(tb2 INNER JOIN [( ]tb3
[INNER JOIN [( ]tablax [INNER JOIN ...)]
ON tb3.campo3 comp tbx.campox)]
ON tb2.campo2 comp tb3.campo3)
ON tb1.campo1 comp tb2.campo2;
```

Si empleamos la cláusula INNER en la consulta se seleccionarán sólo aquellos registros de la tabla de la que hayamos escrito a la izquierda de INNER JOIN que contengan al menos un registro de la tabla que hayamos escrito a la derecha.

3.4.3.1.9 Consultas de unión externas.

Se utiliza la operación UNION para crear una consulta de unión, combinando los resultados de dos o más consultas o tablas independientes. Su sintaxis es:

```
[TABLE] consulta1 UNION [ALL] [TABLE]
consulta2 [UNION [ALL] [TABLE] consultan [ ... ]]
```

En donde:

consulta1, consulta2, consultan
Son instrucciones SELECT, el nombre de una consulta almacenada o el nombre de una tabla almacenada precedido por la palabra clave TABLE.

Puede combinar los resultados de dos o más consultas, tablas e instrucciones SELECT, en cualquier orden, en una única operación UNION. El ejemplo siguiente combina una tabla existente llamada Nuevas Cuentas y una instrucción SELECT:

```
TABLE [Nuevas Cuentas] UNION ALL SELECT * FROM Clientes
WHERE [Cantidad pedidos] > 1000;
```

Si no se indica lo contrario, no se devuelven registros duplicados cuando se utiliza la operación UNION, no obstante puede incluir el predicado ALL para asegurar que se devuelven todos los registros. Esto hace que la consulta se ejecute más rápidamente. Todas las consultas en una operación UNION deben pedir el mismo número de campos, no obstante los campos no tienen porqué tener el mismo tamaño o el mismo tipo de datos.

Se puede utilizar una cláusula GROUP BY y/o HAVING en cada argumento consulta para agrupar los datos devueltos. Puede utilizar una cláusula ORDER BY al final del último argumento consulta para visualizar los datos devueltos en un orden específico.

```
SELECT [Nombre de compañía], Ciudad FROM Proveedores WHERE
País = 'Brasil' UNION SELECT [Nombre de compañía], Ciudad FROM Clientes
WHERE País = "Brasil"
```
Recupera los nombres y las ciudades de todos proveedores y clientes de Brasil

```
SELECT [Nombre de compañía], Ciudad FROM Proveedores WHERE País = 'Brasil'
UNION SELECT [Nombre de compañía], Ciudad FROM Clientes WHERE País =
'Brasil' ORDER BY Ciudad
```
Recupera los nombres y las ciudades de todos proveedores y clientes radicados en Brasil, ordenados por el nombre de la ciudad

```
SELECT [Nombre de compañía], Ciudad FROM Proveedores WHERE País = 'Brasil'
UNION SELECT [Nombre de compañía], Ciudad FROM Clientes WHERE País =
'Brasil' UNION SELECT [Apellidos], Ciudad FROM Empleados WHERE Región =
'América del Sur'
```
Recupera los nombres y las ciudades de todos los proveedores y clientes de brasil y los apellidos y las ciudades de todos los empleados de América del Sur

```
TABLE [Lista de clientes] UNION TABLE [Lista de proveedores]
```
Recupera los nombres y códigos de todos los proveedores y clientes

3.4.3.1.10 Estructuras de tablas.

3.4.3.1.10.1 Creación de tablas nuevas.

Su sintaxis es:

```
CREATE TABLE tabla (campo1 tipo (tamaño) índice1 ,
  campo2 tipo (tamaño) índice2 , ...,
  índice multicampo , ... )
```

En donde:

Parte	Descripción
Tabla	Es el nombre de la tabla que se va a crear.
campo1 campo2	Es el nombre del campo o de los campos que se van a crear en la nueva tabla. La nueva tabla debe contener, al menos, un campo.
Tipo	Es el tipo de datos de campo en la nueva tabla. (Ver Tipos de Datos)
Tamaño	Es el tamaño del campo sólo se aplica para campos de tipo texto.
índice1 índice2	Es una cláusula CONSTRAINT que define el tipo de índice a crear. Esta cláusula es opcional.
índice multicampos	Es una cláusula CONSTRAINT que define el tipo de índice multicampos a crear. Un índice multi campo es aquel que está indexado por el contenido de varios campos. Esta cláusula es opcional.

Tabla 3.22 Parámetros para crear una tabla en SQL.

```
CREATE TABLE Empleados (Nombre TEXT (25) , Apellidos TEXT (50));
```

Crea una nueva tabla llamada Empleados con dos campos, uno llamado Nombre de tipo texto y longutid 25 y otro llamado apellidos con longitud 50.

```
CREATE TABLE Empleados (Nombre TEXT (10), Apellidos TEXT,
Fecha_Nacimiento DATETIME) CONSTRAINT IndiceGeneral UNIQUE
([Nombre], [Apellidos], [Fecha_Nacimiento]);
```

Crea una nueva tabla llamada Empleados con un campo Nombre de tipo texto y longitud 10, otro con llamado Apellidos de tipo texto y longitud predeterminada (50) y uno más llamado Fecha_Nacimiento de tipo Fecha/Hora. También crea un índice único (no permite valores repetidos) formado por los tres campos.

```
CREATE TABLE Empleados (ID INTEGER CONSTRAINT IndicePrimario PRIMARY,
Nombre TEXT, Apellidos TEXT, Fecha_Nacimiento DATETIME);
```

Crea una tabla llamada Empleados con un campo Texto de longitud predeterminada (50) llamado Nombre y otro igual llamado Apellidos, crea otro campo llamado Fecha_Nacimiento de tipo Fecha/Hora y el campo ID de tipo entero el que establece como clave principal.

3.4.3.1.10.1.1 Cláusula CONSTRAIN.

Se utiliza la cláusula CONSTRAINT en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar índices. Existen dos sintaxis para esta cláusula dependiendo si desea Crear ó Eliminar un índice de un único campo o si se trata de un campo multiíndice.

Para los índices de campos únicos:

```
CONSTRAINT nombre {PRIMARY KEY | UNIQUE | REFERENCES tabla externa
[(campo externo1, campo externo2)]}
```

Para los índices de campos múltiples:

```
CONSTRAINT nombre {PRIMARY KEY (primario1[, primario2 [, ...]]) |
UNIQUE (único1[, único2 [, ...]]) |
FOREIGN KEY (ref1[, ref2 [, ...]]) REFERENCES tabla externa [(campo
externo1
[,campo externo2 [, ...]])]}
```

Donde:

Parte	Descripción
nombre	Es el nombre del índice que se va a crear.
PrimarioN	Es el nombre del campo o de los campos que forman el índice primario.
UnicoN	Es el nombre del campo o de los campos que forman el índice de clave única.
RefN	Es el nombre del campo o de los campos que forman el índice externo (hacen referencia a campos de otra tabla).
tabla externa	Es el nombre de la tabla que contiene el campo o los campos referenciados en refN
Campos externos	Es el nombre del campo o de los campos de la tabla externa especificados por ref1, ref2, ..., refN

Tabla 3.23 Parámetros para crear restricciones de campos múltiples.

Si se desea crear un índice para un campo cuando se esta utilizando las instrucciones ALTER TABLE o CREATE TABLE la cláusula CONTRAINT debe aparecer inmediatamente después de la especificación del campo indexado.

Si se desea crear un índice con múltiples campos cuando se está utilizando las instrucciones ALTER TABLE o CREATE TABLE la cláusula CONSTRAINT debe aparecer fuera de la cláusula de creación de tabla.

Donde:

Tipo de Índice	Descripción
UNIQUE	Genera un índice de clave única. Lo que implica que los registros de la tabla no pueden contener el mismo valor en los campos indexados.
PRIMARY KEY	Genera un índice primario el campo o los campos especificados. Todos los campos de la clave principal deben ser únicos y no nulos, cada tabla sólo puede contener una única clave principal.
FOREIGN KEY	Genera un índice externo (toma como valor del índice campos contenidos en otras tablas). Si la clave principal de la tabla externa consta de más de un campo, se debe utilizar una definición de índice de múltiples campos, listando todos los campos de referencia, el nombre de la tabla externa, y los nombres de los campos referenciados en la tabla externa en el mismo orden que los campos de referencia listados. Si los campos referenciados son la clave principal de la tabla externa, no tiene que especificar los campos referenciados, predeterminado por valor, el motor Jet se comporta como si la clave principal de la tabla externa fueran los campos referenciados .

Tabla 3.24 Parámetros para crear índices a los campos de la tabla.

3.4.3.1.10.2 Creación de índices.

La sintaxis para crear un índice en una tabla ya definida en la siguiente:

```
CREATE [ UNIQUE ] INDEX índice
ON tabla (campo [ASC|DESC][, campo [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]
```

En donde:

Parte	Descripción
índice	Es el nombre del índice a crear.
tabla	Es el nombre de una tabla existentes en la que se creará el índice.
campo	Es el nombre del campo o lista de campos que constituyen el índice.
ASC|DESC	Indica el orden de los valores de los campos ASC indica un orden ascendente (valor predeterminado) y DESC un orden descendente.
UNIQUE	Indica que el índice no puede contener valores duplicados.
DISALLOW NULL	Prohíbe valores nulos en el índice
IGNORE NULL	Excluye del índice los valores nulos incluidos en los campos que lo componen.
PRIMARY	Asigna al índice la categoría de clave principal, en cada tabla sólo puede existir un único indice que sea "Clave Principal". Si un índice es clave principal implica que que no puede contener valores nulos ni duplicados.

Tabla 3.25 Parámetros para crear índices con el comando CREATE INDEX.

Se puede utilizar CREATE INDEX para crear un seudo índice sobre una tabla adjunta en una fuente de datos ODBC tal como SQL Server que no tenga todavía un índice. No necesita permiso o tener acceso a un servidor remoto para crear un seudo índice, además la base de datos remota no es consciente y no es afectada por el seudo índice. Se utiliza la misma sintaxis para las tabla adjunta que para las originales. Esto es especialmente útil para crear un índice en una tabla que sería de sólo lectura debido a la falta de un índice.

```
CREATE INDEX MiIndice ON Empleados (Prefijo, Telefono);
Crea un índice llamado MiIndice en la tabla empleados con los campos Prefijo y Telefono.

CREATE UNIQUE INDEX MiIndice ON Empleados (ID) WITH DISALLOW NULL;
Crea un índice en la tabla Empleados utilizando el campo ID, obligando que el campo ID
no contenga valores nulos ni repetidos.
```

3.4.3.1.10.3 Modificar el diseño de una tabla.

Modifica el diseño de una tabla ya existente, se pueden modificar los campos o los índices existentes. Su sintaxis es:

```
ALTER TABLE tabla {ADD {COLUMN tipo de campo[(tamaño)] [CONSTRAINT índice]
   CONSTRAINT índice multicampo} |
   DROP {COLUMN campo I CONSTRAINT nombre del índice} }
```

En donde:

Parte	Descripción
Tabla	Es el nombre de la tabla que se desea modificar.
Campo	Es el nombre del campo que se va a añadir o eliminar.
Tipo	Es el tipo de campo que se va a añadir.
Tamaño	El tamaño del campo que se va a añadir (sólo para campos de texto).
Índice	Es el nombre del índice del campo (cuando se crean campos) o el nombre del índice de la tabla que se desea eliminar.
Índice multicampo	Es el nombre del índice del campo multicampo (cuando se crean campos) o el nombre del índice de la tabla que se desea eliminar.

Tabla 3.26 Parámetros para modificar tablas a través de ALTER TABLE.

Operación	Descripción
ADD COLUMN	Se utiliza para añadir un nuevo campo a la tabla, indicando el nombre, el tipo de campo y opcionalmente el tamaño (para campos de tipo texto).
ADD	Se utiliza para agregar un índice de multicampos o de un único campo.
DROP COLUMN	Se utiliza para borrar un campo. Se especifica únicamente el nombre del campo.
DROP	Se utiliza para eliminar un índice. Se especifica únicamente el nombre del índice a continuación de la palabra reservada CONSTRAINT.

Tabla 3.27 Parámetros para agregar o eliminar columnas a las tablas a través de ALTER TABLE

```
ALTER TABLE Empleados ADD COLUMN Salario CURRENCY;
Agrega un campo Salario de tipo Moneda a la tabla Empleados.

ALTER TABLE Empleados DROP COLUMN Salario;
Elimina el campo Salario de la tabla Empleados.

ALTER TABLE Pedidos ADD CONSTRAINT RelacionPedidos FOREIGN KEY
(ID_Empleado) REFERENCES Empleados (ID_Empleado);
Agrega un indice externo a la tabla Pedidos. El índice externo se basa en el campo ID_Empleado y
se refiere al campo ID_Empleado de la tabla Empleados. En este ejemplo no es necesario indicar
el campo junto al nombre de la tabla en la cláusula REFERENCES, pues ID_Empleado es la clave
principal de la tabla Empleados.

ALTER TABLE Pedidos DROP CONSTRAINT RelacionPedidos;
Elimina el Índice de la tabla Pedidos.
```

3.4.3.2 Uso de PostgreSQL.

Postgresql trae consigo una interfase de comandos donde se pueden ejecutar todas las instrucciones SQL vistas en la sección anterior (3.4.3.1). Para acceder a ésta interfase de comandos basta ejecutar desde un shell de Linux:

```
#> psql nombre_base_datos
```

Existen numerosos comandos de PostgreSQL que no describiremos aquí debido a que se encuentra fuera del alcance de esta Tesis pero el lector puede ver en la bibliografía al final para mayor información sobre el uso de PostgreSQL y sus comandos, basta decir que para el diseño de la base de datos del sistema propuesto para el proyecto se muestra a continuación (Figura 3.28) la lista de comandos SQL que se ejecutaron para crear las tablas del sistema.

En la Figura 3.29 se muestra un esquema más representativo del diseño de la tablas que se generaron con lenguaje SQL en PostgreSQL así como una descripción de los datos que se almacena en cada una de las columnas de las tablas.

```
\connect - nobody
CREATE SEQUENCE "Clave_Punto_n_seq" start 1 increment 1 maxvalue 2147483647 minvalue 1  cache 1 ;
\connect - postgres
CREATE SEQUENCE "Clave_Archivo_n_seq" start 25 increment 1 maxvalue 2147483647 minvalue 1  cache
1 ;
SELECT nextval ('"Clave_Archivo_n_seq"');
REVOKE ALL on "Clave_Archivo_n_seq" from PUBLIC;
GRANT ALL on "Clave_Archivo_n_seq" to "nobody";
\connect - nobody
CREATE TABLE "puntos" (
        "Clave_Punto_n" int4 DEFAULT text(nextval('Clave_Punto_n_seq'::text)) NOT NULL,
        "Nombre_Punto_c" character varying(5) NOT NULL,
        "X_n" int4 NOT NULL,
        "Y_n" int4 NOT NULL,
        "Z_n" int4 NOT NULL,
        "P_n" int4 NOT NULL,
        "R_n" int4 NOT NULL,
        "Ac_n" character varying(1) DEFAULT 'c'::"varchar",
        PRIMARY KEY ("Clave_Punto_n")
);
\connect - postgres
CREATE TABLE "Usuarios" (
        "Clave_Usuario_n" int4 DEFAULT nextval('Clave_Usuario_n_seq'::text) NOT NULL,
        "Login_Usuario_c" character varying(50) NOT NULL,
        "Nombre_Usuario_c" character varying(80),
        "Password" character varying(32),
        "Clave_Grupo_n" int4
);
REVOKE ALL on "Usuarios" from PUBLIC;
GRANT INSERT,UPDATE,DELETE,SELECT on "Usuarios" to "nobody";
CREATE TABLE "archivos" (
        "Clave_Archivo_n" int4 DEFAULT text(nextval('Clave_Archivo_n_seq'::text)) NOT NULL,
        "Nombre_Archivo_c" character varying(50) NOT NULL,
        "Archivo_o" oid,
        "Mime_c" text,
        PRIMARY KEY ("Clave_Archivo_n")
);
REVOKE ALL on "archivos" from PUBLIC;
GRANT ALL on "archivos" to "nobody";
\connect - nobody
\connect - postgres
INSERT INTO "Usuarios"
("Clave_Usuario_n","Login_Usuario_c","Nombre_Usuario_c","Password","Clave_Grupo_n") VALUES
(1,'prueba','Usuario de Prueba','tesis',1);
INSERT INTO "archivos" ("Clave_Archivo_n","Nombre_Archivo_c","Archivo_o","Mime_c") VALUES
(23,'CARLOS1',45281,'text/plain');
INSERT INTO "archivos" ("Clave_Archivo_n","Nombre_Archivo_c","Archivo_o","Mime_c") VALUES
(25,'carlos',45313,'text/plain');
\connect - nobody
CREATE UNIQUE INDEX "puntos_Nombre_Punto_c_key" on "puntos" using btree ( "Nombre_Punto_c"
"varchar_ops" );
\connect - postgres
CREATE UNIQUE INDEX "Usuarios_Login_Usuario_c_key" on "Usuarios" using btree ( "Login_Usuario_c"
"varchar_ops" );
CREATE UNIQUE INDEX "archivos_Nombre_Archivo_c_key" on "archivos" using btree ( "Nombre_Archivo_c"
"varchar_ops" );
\connect - nobody
CREATE UNIQUE INDEX "comandos_Nombre_Punto_c_key" on "puntos" using btree (
"Nombre_Punto_c" "varchar_ops" );
```

Figura 3.28 Comandos ejecutados en PostgreSQL para la creación de las tablas del proyecto.

Tabla "Usuarios"		
Atributo	**Tipo**	**Descripción**
Clave_Usuario_n	Integer	Clave Controlada por PostgreSQL
Login_Usuario_c	varchar(50)	Nombre de Usuario
Nombre_Usuario_c	varchar(80)	Nombre Completo
Password	varchar(32)	Clave de Acceso

Tabla "archivos"		
Atributo	**Tipo**	**Descripción**
Clave_Archivo_n	integer	Clave Controlada por PostgreSQL
Nombre_Archivo_c	varchar(50)	Nombre del archivo
Archivo_o	Oid	Archivo de texto con comandos ACL (Objeto)

Tabla "Puntos_Archivos"		
Atributo	**Tipo**	**Descripción**
Clave_P_A_n	integer	Clave Controlada por PostgreSQL
Clave_Archivo_n	integer	Clave del archivo
Clave_Punto_n	Integer	Clave del Punto

Tabla "Puntos"		
Atributo	**Tipo**	**Descripción**
Clave_Punto_n	integer	Clave Controlada por PostgreSQL
Punto_c	varchar(6)	Nombre del Punto
X_f	float8	Posición X
Y_f	float8	Posición Y
Z_f	float8	Posición Z
P_f	Float8	Inclinación del antebrazo en grados
R_f	Float8	Posición Giro de la agarradera en grados
Gripper_c	varchar(1)	Posición del Gripper 1 – abierto 0 - cerrado

Figura 3.29 Esquema de las tablas creadas para conformar la base de datos para el sistema.

En PostgreSQL existen diferentes tipos de datos que pueden ser almacenados; para efectos de simplificación, en la Tabla 3.30, se muestran sólo algunos de ellos.

Tipo	Descripción
Varchar(<tam>)	Almacena <tam> número de caracteres
Text	Almacena un número ilimitado de caracteres
Float8	Almacena números de punto flotante
Oid	Almacena un objeto, pude ser cualquier tipo de información.
Integer	Almacena números enteros

Tabla 3.30 Algunos tipos de valores que se puede almacenar en PostgreSQL.

3.4.4 Desarrollo de programas.

3.4.4.1 Programación de puerto serial en Linux.

La mayor parte de los sistemas UNIX soportan la interfase de terminal serial POSIX para cambiar parámetros como velocidad de transferencia de datos, tamaño de carácter, etcétera. La primera cosa que se tiene que hacer al programar para el puerto serial es incluir, como parte del programa en C, el archivo **<termios.h>**; el cual define la estructura de control de la terminal, así como también las funciones de control POSIX.

Las funciones principales de POSIX son *tcgetattr* y *tcsetattr*. Estos ajustan y toman las propiedades de la terminal, respectivamente, se debe proporcionar un apuntador a la estructura termios que contiene todas las opciones disponibles. (Ver Apéndice D.1)

Los dispositivos /dev/ttyS* tienen como misión conectar terminales a linux, y están configurados para este uso al arrancar el sistema. Se debe tener presente esto cuando se programen comunicaciones con un dispositivo. Por ejemplo, los puertos están configurados para escribir en pantalla cada carácter enviado desde el dispositivo, que normalmente tiene que ser cambiado para la transmisión de datos.

Todos los parámetros se pueden configurar con un programa. La configuración se guarda en una estructura struct termios (Figura 3.31), que está definida en <asm/termbits.h>:

```
#define NCCS 19
struct termios {
    tcflag_t c_iflag;       /* parametros de modo entrada */
    tcflag_t c_oflag;       /* parametros de modo salida */
    tcflag_t c_cflag;       /* parametros de modo control */
    tcflag_t c_lflag;       /* parametros de modo local */
    cc_t c_line;            /* disciplina de la linea */
    cc_t c_cc[NCCS];        /* caracteres de control */
};
```

Figura 3.31 Descripción de la estructura termios

Este archivo también incluye todas las definiciones de parámetros. Los parámetros de modo entrada de c_iflag manejan todos los procesos de entrada, lo cual significa que los caracteres enviados desde el dispositivo pueden ser procesados antes de ser leídos con la función read.

De forma similar c_oflag maneja los procesos de salida. c_cflag contiene la configuración del puerto, como la velocidad en baudios, bits por carácter, bits de parada, etc... Los parámetros de modo local se guardan en c_lflag. Determinan si el carácter tiene eco, señales enviadas al programa, etc.

Finalmente la tabla c_cc define el carácter de control para el fin de archivo, parada, etc... Los valores por defecto de los caracteres de control están definidos en <asm/termios.h>. Los parámetros están descritos en el Apéndice D.1.

3.4.4.1.1 Desarrollo de programas de puerto serial para comunicación con el controlador ACL.

Con el uso de la estructura termios para lenguaje C se desarrollaron 2 programas en lenguaje C para la transmisión de datos al controlador ACL.

El archivo scorbot.c (Apéndice D.2.1) manda cadenas de texto proporcionadas como parámetros al momento de ser ejecutado. El archivo scorbot_archivo.c (Apéndice D.2.2) recibe como parámetro el nombre de un archivo el cual debe contener cadenas de texto ascii que el programa manda al controlador. En los apéndices se deja muestra el código fuente desarrollado, de cada uno de éstos dos programas, para llevar a cabo las tareas de comunicarse con el controlador ACL.

3.4.4.2 Desarrollo de programas script en PHP (Servidor-Cliente).

Los programas escritos en PHP sirven como interfase entre el servidor Web y los programas escritos en C relacionados al puerto serial y el controlador. El sistema consta de varios archivos:

1. Página principal con control de acceso por medio de base de datos.

2. Un archivo para el envío directo de comandos ACL al controlador.
3. Un archivo para realizar movimientos del robot de manera análoga al teach pendant.
4. Un archivo para el almacenamiento de puntos en la base de datos.
5. Un archivo para el almacenamiento de programas para ser enviados y ejecutados en el servidor posteriormente.

3.4.4.3 Descripción de los archivos del sistema.

3.4.4.3.1 Archivo index.php.

Se utiliza un programa principal donde se identifica al usuario de un catálogo almacenado en la base de datos a este archivo se le dio el nombre de index.php (Figura 3.32).

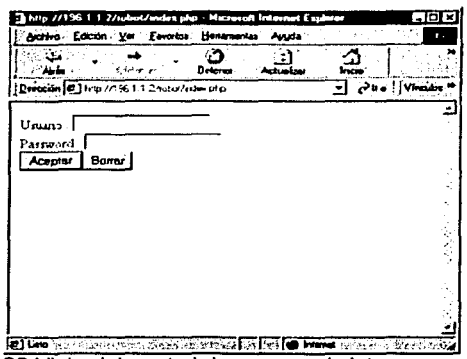


Figura 3.32 Vista del control de acceso al sistema en ejecución

3.4.4.3.2 Archivo setup.php.

Se creó el archivo setup.php (Figura 3.33) para que a través de variables globales se permita una personalización del funcionamiento del sistema. Basta simplemente con descomentar las partes del archivo setup.php y comentar otras para que el sistema corra en una plataforma u otra o se utilice una base de datos diferente a la propuesta en esta Tesis.

```
<?
session_start();
if ( !isset($SESSION)){
    session_register("SESSION");
}
class object {};

/* setup the configuration object */
$CFG = new object;
$CFG->basedatos = "robot";

// Configuración para correr dentro de Linux con PostgreSQL como
//   base de datos
 $CFG->dbtipo = "postgres";
 $CFG->wwwroot      = "/robot";
 $CFG->dirroot      = "/usr/local/apache/htdocs/robot";

// configuración Si el sistema correrá dentro de Windows con MySQL como
//base de datos
```

```
// $CFG->dbtipo = "mysql";
// $CFG->wwwroot     = "/sistema";
// $CFG->dirroot     = "c:/htdocs/sistema";
// $CFG->dbhost = "localhost";
// $CFG->dbname = "robot";
// $CFG->dbuser = "";
// $CFG->dbpass = "";

// Tipo de graficos que se usarán según la versión de librería de
gráficos
// instalada en el servidor Web (librería gd)

//$CFG->tipoimg      ="gif";
$CFG->tipoimg      ="png";

// Directorios donde residen las imagenes, las librerias o cualquier
directorio
// donde se coloquen archivos

$CFG->templatedir = "$CFG->dirroot/templates";
$CFG->libdir      = "$CFG->dirroot/lib";
$CFG->imagedir    = "$CFG->wwwroot/imagenes";

// Indica los módulos o librerías que se van a utilizar a lo largo del
sistema

include "$CFG->libdir/dblib.inc";
include "$CFG->libdir/stdlib.inc";
include "$CFG->libdir/recordset.inc";

// Se requiere del módulo de control de acceso llamado login.php
require "$CFG->dirroot/login.php";
?>
```

Figura 3.33 Código fuente archivo setup.php

3.4.4.3.3 Archivo login.php.

El módulo de control de acceso consta de una función que permite saber si el usuario que utilizará el sistema es un usuario conocido y que para ello se le asignó una cuenta de usuario almacenada en la base de datos en la tabla Usuarios (Figura 3.34).

```
<?
session_start();
if (isset($usuario)){
      $SESSION["usuario"]=$usuario;
      $SESSION["password"]=$password;
       if (verifica_usuario($SESSION["usuario"],$SESSION["password"])){
         $SESSION["nombre"]=$SESSION["usuario"];
         return 1;
     } else {
           echo "<h5>Usuario incorrecto</h5>";
           unset($SESSION["usuario"]);
           unset($SESSION["password"]);
           echo "<form action=index.php method=post target=_top>";
           echo "Usuario : <input type=text name=usuario size=20 length=40
value=><BR>";
           echo "Password : <input type=password name=password size=20
length=40 value=><BR>";
```

```
            echo "<input type=submit name=login value=Aceptar>";
            echo "<input type=reset name=reset value=Borrar>";
            echo "</form>";
            die;
      }
} else if(isset($SESSION["usuario"])) {
   if(!verifica_usuario($SESSION["usuario"],$SESSION["password"])){
      require "logout.php";
      die;
   }
} else {
      echo "<form action=index.php method=post target=_top>";
      echo "Usuario : <input type=text name=usuario size=20 length=40
value=><BR>";
      echo "Password : <input type=password name=password size=20 length=40
value=><BR>";
      echo "<input type=submit name=login value=Aceptar>";
      echo "<input type=reset name=reset value=Borrar>";
      echo "</form>";
      die;
}

function verifica_usuario($usuario,$password) {
global $CFG;
/*  esta función accesa a la base de datos y verifica que el usuario y
    el password que se dieron en login.php existan */
if(db_busca("Login_Usuario_c","[Usuarios]","[Login_Usuario_c]='".$usuario."'
and [Password]='".$password."'",$CFG->basedatos)){
      return 1;
    } else {
      return 0;
    }
}
?>
```

Figura 3.34 Código fuente archivo login.php

3.4.4.3.4 Archivo logout.php.

Este archivo termina la sesión del usuario y muestra un mensaje de que el usuario salió del sistema (Figura 3.35).

```
<?
include_once("setup.php");
session_start();
session_destroy();
echo "<p><h5>Acaba de salir del Sistema</h5></p>";
echo "<a href=$CFG->wwwroot/index.php target=_top>Regresar al menu
principal </a>";
?>
```

Figura 3.35 Código fuente archivo logout.php

3.4.4.3.5 Archivo comandos.php.

Archivo para el envío directo de comandos ACL al controlador.

Con este archivo (Figura 3.36) el usuario coloca un comando ACL en la caja de texto y al presionar el botón de envío, el programa comandos.php ejecuta el programa scorbot mandándole como

parámetros el comando ACL escrito en el cuadro de texto. En la Figura 3.37 se puede observar un ejemplo de la vista en ejecución del archivo comandos.php.

```
<?
session_start();
include_once("setup.php");
IF (lisset($comando)){
   printf("<FORM METHOD=post ACTION=comandos.php>");
   printf("Comando:<INPUT NAME=comando TYPE=TEXT VALUE= ><BR>");
   printf("<INPUT NAME=Enviar TYPE=submit VALUE=Enviar >");
   printf("</FORM>");
} else {
   printf("<FORM METHOD=post ACTION=comandos.php>");
   printf("Comando:<INPUT NAME=comando TYPE=TEXT VALUE= ><BR>");
   printf("<INPUT NAME=Enviar TYPE=submit VALUE=Enviar >");
   printf("</FORM>");
   printf("Mandando...%s\n<BR>",$comando);
   $rp = exec("/usr/local/apache/htdocs/robot/scorbot ".$comando,$i);
   echo $rp."<br>".$i;
}
?>
```

Figura 3.36 Código fuente archivo comandos.php

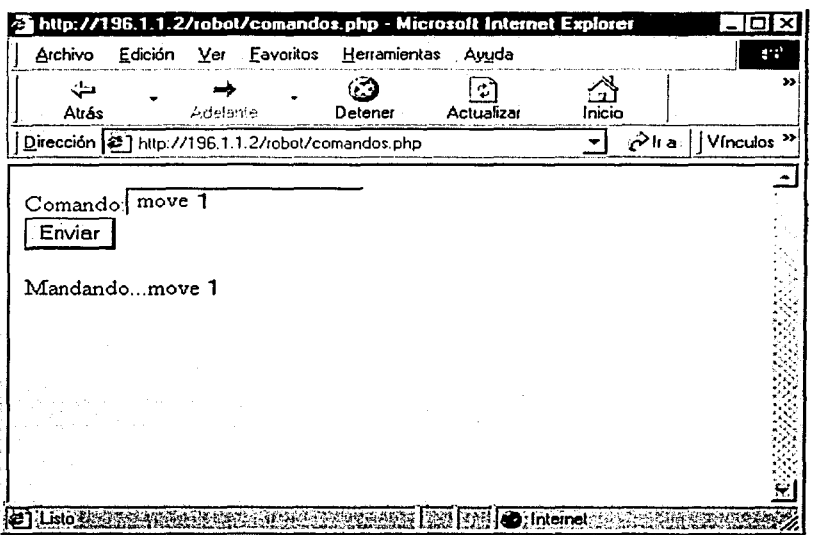


Figura 3.37 Vista del archivo comandos.php en ejecución

3.4.4.3.6 Archivo teach_pendant.php.

Se desarrolló una interfase gráfica (Figura 3.38)donde se colocaron botones similares a los que se encuentran en el Teach Pendant, con esto al momento de presionar los botones gráficos el usuario puede mover el robot como si hubiera presionado un botón en el Teach Pendant. Se desarrolló un programa script que permite al usuario mover el SCORBOT, de manera similar a usar el Teach

Pendant, para ello el usuario puede indicar que tan preciso puede ser el movimiento o que tan rápido puede ser, al mismo tiempo puede ver las acciones del robot a través de una imagen que es actualizada cada cierto tiempo por medio de una cámara web.

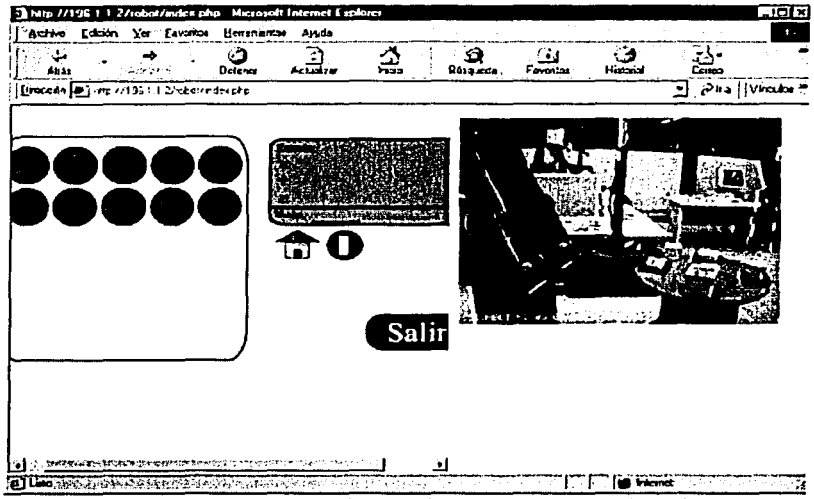


Figura 3.38 Vista del archivo teach_pendant.php en ejecución.

Como ya se había mencionado en el Capítulo 3.2.1 existen 2 estados en el controlador:

- Modo EDITOR.
- Modo MONITOR o DIRECT.

A través del puerto serial conectado al controlador es posible que el controlador pase de un estado a otro y ejecutar comandos ACL con los que podemos hacer que realice varias funciones (ver Apéndice B).

Una de las desventajas del controlador ACL es que no es posible determinar en qué estado se encuentra el controlador ya que sólo recibe las instrucciones y las realiza si fueron escritas de manera correcta, es decir que no es posible preguntarle al controlador en qué modo está o si las acciones enviadas son correctas.

Para mover el robot manualmente se manda al controlador un carácter '~' para indicarle que se quiere trabajar en modo manual, este carácter sólo cambia de un modo a otro, es decir que si se encuentra en modo Editor y se le envía un carácter '~' el controlador pasará a modo Manual y viceversa.

Dentro del modo manual es posible controlar el robot por medio de movimientos por sus UNIONES (Figura 3.39) o por medio de sus ejes coordenados XYZ (Figura 3.40)

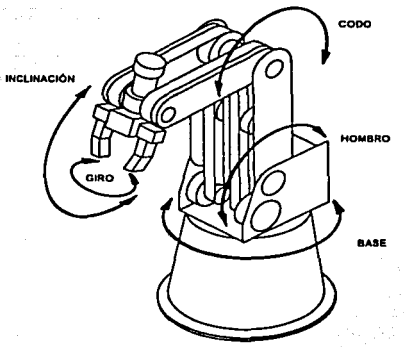


Figura 3.39 Movimientos del Robot SCORBOT por sus uniones

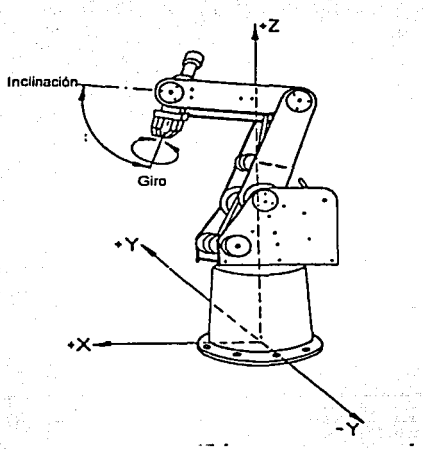


Figura 3.40 Movimientos por Ejes Coordenados del SCORBOT

Dentro del modo MANUAL	
Carácter	FUNCION
_ (subguión)	MODO UNIONES
X	PASA DE UNIONES A XYZ

Tabla 3.41 Paso a modo XYZ y modo UNIONES

MODO UNIONES	
Caracteres	Movimiento
1, Q	Eje 1 (base)
2, W	Eje 2 (hombro)
3, E	Eje 3 (codo)
4, R	Eje 4 (inclinación)
5, T	Eje 5 (giro)
6, T	Abre/Cierra (agarradera)

Tabla 3.42 Movimientos por uniones del SCORBOT

MODO XYZ	
Caracteres	Movimiento
1, Q	X +, X-
2, W	Y+, Y-
3, E	Z+, Z-
4, R	Inclinación arriba/abajo mientras que la parte final de la agarradera mantiene su posición

Tabla 3.43 Movimientos por Ejes Coordenados del SCORBOT

Cada vez que se manda un carácter ASCII mostrado en las tablas 3.41 y 3.42 y 3.43 respectivamente, por el puerto serial y al estar en modo monitor, el controlador realiza el movimiento en el SCORBOT y es posible visualizar el movimiento mediante una imagen que se refresca cada segundo en la pantalla del navegador.

Cuando se presiona una de las imágenes (Figura 3.44) que representan botones dentro de un panel de control, el archivo ejecuta el archivo scorbot.c (Apéndice D.2.1) con el carácter correcto para el movimiento requerido, dentro del panel de control se encuentra un cuadro de texto que indica la precisión del movimiento. La precisión no es mas que la repetición del carácter que realizará el movimiento, es decir que si el número es 0 o 1 el programa sólo manda 1 carácter, en caso de ser 2 el programa mandará 2 caracteres y así sucesivamente, lo que hará que a cada presión del botón la reacción del robot se verá incrementada ese número de veces.

3.4.4.3.7 Archivo archivo.php.

Con este archivo (Figura 3.44) es posible agregar, a la base de datos, archivos escritos en una computadora local, que contengan comandos ACL. Posteriormente se emplean los archivos: carga.php y ejecuta.php para cargarlos al controlador y que el SCORBOT los ejecute.

```
<?
$Gdbhost="localhost";
$Gdbname="robot";
   function pg_open() {
      global $SERVER_NAME, $REQUEST_URI;
      global $Gdbhost, $Gdbname;
      $conn = pg_pconnect($Gdbhost,"","",$Gdbname);
      if (!$conn) {
            error_log("Unable to open $Gdbname while processing
$SERVER_NAME".$REQUEST_URI);
      }
          return $conn;
}

function pg_send($sql) {
          global $sql_debug;
          $conn = pg_open();
          if ($sql_debug) { print "Sending on $conn: $sql<br>"; }
```

```
        return pg_exec($conn, $sql);
}

if(!isset($Nombre_Archivo_c)){
   print "<FORM ENCTYPE=\"multipart/form-data\" ACTION=\"$PHP_SELF\" METHOD=POST>";
   print "<input type=\"hidden\" name=\"MAX_FILE_SIZE\" value=\"300000\">";
   print "<input type=\"hidden\" name=\"Archivo_n\" value=\"$person_id\">";
   print "Nombre del archivo: <input type=\"textbox\" name=\"Nombre_Archivo_c\" value=\"\"><BR>";
   print "Presiona aquí para obtener el archivo de su disco duro: <input name=\"userfile\"
type=\"file\">";
   print "<input type=\"submit\" name=\"submit\" value=\"Enviarlo\"><br>";
   print "Máximo 300,000 bytes.";
   print "</form>";
} elseif($Nombre_Archivo_c!="" and $submit=="Enviarlo"){
   $result = pg_send("SELECT nextval('\"Clave_Archivo_n_seq\"') AS \"numero\"");
   $numero = pg_result($result, 0, 'numero');
   $image = fopen($userfile, "r");
   $buf = fread($image, 999999);
   $conn = pg_open();
   pg_exec($conn,"begin");
   $oid = pg_locreate($conn);
   $fd = pg_loopen($conn, $oid, "w");
   $bytes = pg_lowrite($fd, $buf);
   pg_loclose($fd);
   printf("Tipo de archivo: %s<BR>",$userfile_type);
   pg_exec($conn, "INSERT INTO archivos
(\"Clave_Archivo_n\",\"Nombre_Archivo_c\",\"Archivo_o\",\"Mime_c\") VALUES
($numero,'$Nombre_Archivo_c',$oid,'$userfile_type')");
   pg_exec($conn, "commit");
   pg_exec($conn, "end");
} else {
   print "Falta nombre de archivo<br>";
   print "<FORM ENCTYPE=\"multipart/form-data\" ACTION=\"$PHP_SELF\" METHOD=POST>";
   print "<input type=\"hidden\" name=\"MAX_FILE_SIZE\" value=\"300000\">";
   print "<input type=\"hidden\" name=\"Archivo_n\" value=\"$person_id\">";
   print "Nombre del archivo: <input type=\"textbox\" name=\"Nombre_Archivo_c\"
value=\"$Nombre_Archivo_c\"><BR>";
   print "Presiona aquí para obtener el archivo de su disco duro: <input name=\"userfile\"
type=\"file\">";
   print "<input type=\"submit\" name=\"submit\" value=\"Enviarlo\"><br>";
   print "Máximo 300,000 bytes.";
   print "</form>";
 }
?>
```

Figura 3.44 Codigo fuente archivo archivo.php.

El programa archivo.php en ejecución (Figura 3.45), de derecha a izquierda: 1.- Se ingresa al programa archivo.php desde la máquina local con un navegador web, 2.- Se da un nombre al programa que se va a agregar a la base de datos, y finalmente se busca en la computadora local el archivo a guardar y finalmente se envía al presionar el botón de enviar.

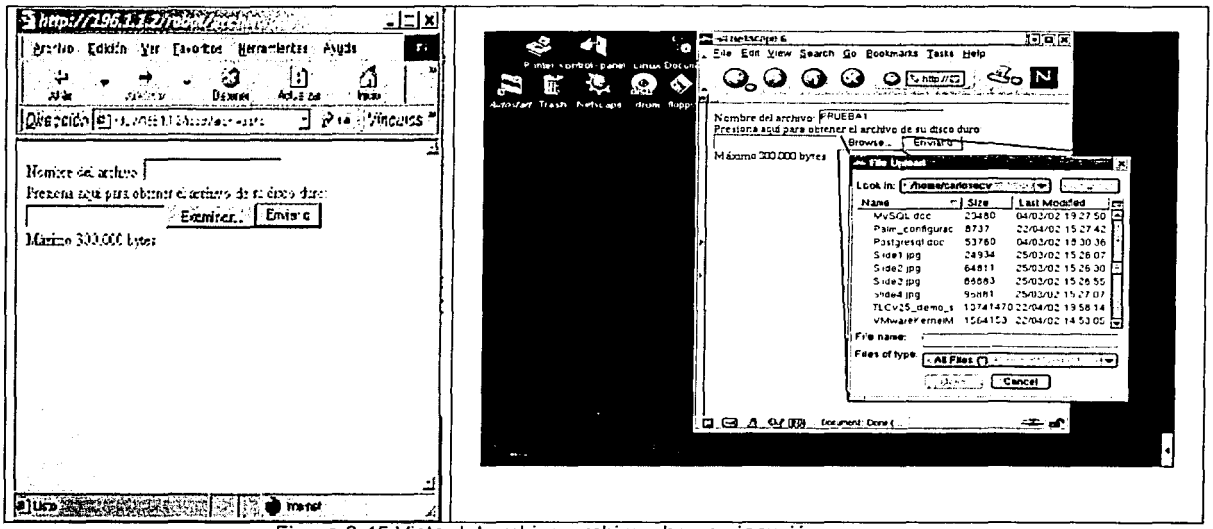


Figura 3.45 Vista del archivo archivo.php en ejecución.

3.4.4.4 Programa de la cámara Web.

En el servidor se ejecuta un programa denominado webcam, cuya función es la de tomar una imagen la cual provee una cámara digital por medio de un puerto USB, esta imagen es tomada en un tiempo determinado. El programa webcam permite actualizar un archivo con la información de la imagen tomada, además de colocar la fecha de exposición sobre la imagen.

En la página web el usuario puede ver el cambio en la imagen actualizada por el programa webcam, esto es posible al programar una página HTML que indica cada cuando el navegador refrescará la ventana que el usuario está viendo y así vea la nueva imagen (Figura 3.38).

```
<HEAD>
<META HTTP-EQUIV="refresh" content="2;URL=camara.html">
</HEAD>

<IMG SRC=camara/webcam.jpg >
```

Figura 3.46 Código fuente del archivo camara.html

La opción content=2 que aparece en el archivo camara.html indica en cuantos segundos el navegador refrescará el contenido del archivo camara.html al usuario; el archivo camara.html contiene una liga a la imagen la imagen webcam.jpg (Figura 3.46) que es actualizada constantemente por el programa webcam.

CAPÍTULO IV
PRUEBAS

Para la demostración de la tesis se realizaron 3 pruebas diferentes:

1. Movimiento manual remoto.
2. Ejecutar comandos vía remota.
3. Ejecutar programas vía remota guardados previamente en una base de datos.
4. Mover el robot a posiciones definidas previamente en la base de datos y poder hacer secuencias de esos puntos.

Se llenaron las tablas en la base de datos con los siguientes registros (Figura 4.1):

Usuarios

Clave_Usuario_n	Login_Usuario_c	Nombre_Usuario_c	Password	Clave_Grupo_n
1	prueba	Usuario de Prueba	tesis	1

Archivos

Clave_Archivo_n	Nombre_Archivo_c	Archivo_o
1	Prueba1	
2	Prueba2	

Puntos

Clave_Punto_n	Punto_c	X_f	Y_f	Z_f	P_f	R_f	Griper_c
1	P1	50	40	30	25	30	1
2	P2	60	40	30	25	30	1
3	P3	70	40	30	25	30	1
4	P4	80	40	30	25	30	0

Puntos_Archivos

Clave_P_A_n	Clave_Archivo_n	Clave_Punto_n
1	1	1
2	1	2
3	1	3
4	1	4
5	2	3
6	2	1

Figura 4.1 Registros agregados para las pruebas en la base de datos.

Al encender el sistema es necesario tomar las siguientes consideraciones y limitaciones:

1. El encendido
   a. Debe hacerse físicamente en la ubicación del controlador y el robot.
   b. Al encender el controlador debe encenderse la fuente de energía de los motores.

2. Prender, de manera lógica, el controlador mediante el envío del comando CON. Sin este paso los comandos que sean enviados al controlador no podrán ejecutarse, para ello existe un botón que envía, al presionarlo, el respectivo comando.

4.1 MOVIMIENTO MANUAL REMOTO.

Objetivo.
Mover el robot, de manera análoga al dispositivo Teach Pendant por medio de la Internet y visualizar los movimientos mediante una imagen proporcionada por la cámara Web.

4.1.1 Resultados.

Fue posible el movimiento del SCORBOT de manera análoga al Teach Pendant. Por medio de los botones proporcionados en la página web.

Dentro de las limitaciones del programa se encuentran:

- El tiempo en que la imagen, de la cámara web, se actualiza.

La actualización de la imagen ocurre de manera que se podría mover el robot a una posición donde golpee algún objeto y se corra el riesgo de dañar los mecanismos del robot, además de que para personas que estén conectadas a velocidades de transmisión muy bajas, es posible que no puedan ver la imagen ya que cuando apenas reciben los datos de la imagen, el programa webcam ya modificó los datos de la imagen y esto provoca que la imagen salga distorcionada o ni siquiera aparezca en su navegador.

- El modo en que se envían los comandos.

No es posible en que modo se encuentra el controlador, es decir si se encuentra en modo editor o modo manual, lo que puede causar confusión si no se tiene cuidado de tener el controlador en el modo manual. En caso de no estar en modo manual el robot no podrá realizar los movimientos que se le pidan ya que no reconocerá los caracteres que el programa le envíe provocando errores que quizá dejen fuera de funcionamiento al controlador.

- Determinar si el movimiento se ejecutó o hubo un error.

No hay una función que permita determinar que el comando enviado al controlador haya sido ejecutado con éxito debido a que el controlador no devuelve ningún carácter que indique esto.

Logro.

- Es posible manipular objetos desde un lugar remoto.

4.2 EJECUTAR COMANDOS VÍA REMOTA.

Objetivo.
Mover el robot al mandar comandos ACL por medio de la Internet para su ejecución inmediata del controlador y visualización de las reacciones por medio de una cámara Web.

En esta prueba es posible mandar comandos directamente al controlador por medio de un script que pide el comando a enviar, el comando es mandado al controlador inmediatamente por medio del programa escrito en C scorbot.c (Ver Apéndice D.2.1).

Al encender el sistema es necesario tomar en cuenta las siguientes limitaciones:

1. El encendido
    a. Debe hacerse físicamente en la ubicación del controlador y el robot.
    b. Al encender el controlador debe encenderse la fuente de energía de los motores

2. Prender logicamente el controlador mediante el envio del comando CON. Sin este paso los comandos que sean enviados al controlador no podran ejecutarse

3. Pasar a modo edición con el envío del carácter '~', en caso de que el controlador se encuentre en modo manual. No es posible identificar, de manera precisa, en que modo se encuentra el controlador, el envío del carácter '~', solamente cambia de un modo a otro .

4. Enviar comandos por medio de la página web comandos.php.

4.2.1 Resultados.

Las pruebas fueron satisfactorias, se escriben comandos en el cuadro de texto y, al presionar el botón de enviar, el programa script pasa la cadena de caracteres al programa del puerto serie para que éste lo envíe al controlador ACL y realice la acción correspondiente.

Dentro de los problemas que se presentaron están:

- Seguridad sobre lo que escribe el usuario.

No se tiene un control sobre lo que el usuario puede colocar en el cuadro de texto, El usuario podría encender o apagar accidentalmente los motores del robot, mandar algún comando que pudiera dañar los mecanismos del robot como un movimiento que provoque un impacto o hacer que el controlador se quede en modo editor de programa al tratar de modificar un programa y no terminar de guardarlo.

Logro.

- Es posible mandar todo tipo de instrucciones al controlador por medio de esta interfase, lo que hace que el manejo remoto sea flexible, como si el usuario estuviera directamente conectado al puerto serial del controlador.

4.3 EJECUTAR PROGRAMAS VÍA REMOTA GUARDADOS PREVIAMENTE EN UNA BASE DE DATOS.

Objetivo.
Escribir programas ACL en la computadora Cliente y poder almacenarlos en la base de datos para su ejecución posterior por parte del controlador.

4.3.1 Resultados.

En esta prueba el usuario puede hacer un programa ACL desde un editor de texto, almacenarlo en la base de datos, grabarlo en el controlador y ejecutarlo cuando se desee.

Entre los inconvenientes están:

- El formato de los archivos enviados.

El archivo debe tener una extensión txt o si no el programa no puede determinar si el archivo que está enviando el usuario es una imagen o verdaderamente es texto.

- Datos incorrectos.

El programa no puede determinar si los comandos que se encuentran escritos dentro del documento enviado están correctamente escritos con la sintaxis ACL o si los puntos almacenados en el controlador y que son llamados por los programas existen y son los correctos.

- El tiempo de ejecución.

El tiempo entre una y otra instrucción del documento es de 1 segundo lo que podría provocar que el robot no alcanzara a terminar su trayectoria a la posición indicada si se le pide en la siguiente instrucción que realice un movimiento a otra posición.

- Limitación del tamaño del programa.

El tamaño máximo del programa almacenado en la base de datos está limitada a 3 MB que es una limitante del servidor Web.

Logro.

- El número de programas que es posible manejar es prácticamente ilimitado (está dado por el límite de espacio en disco duro de la base de datos).
- Para esta prueba, el número de líneas de código de programa no está limitado por la capacidad del controlador.
- La ejecución del programa se puede dar de manera remota.

4.4 MOVER EL ROBOT A POSICIONES DEFINIDAS PREVIAMENTE EN LA BASE DE DATOS Y PODER HACER SECUENCIAS DE ESOS PUNTOS.

Objetivo.
Grabar posiciones en una base de datos y pedir el movimiento del robot hacia esos puntos almacenados.

4.4.1 Resultados.

En esta prueba el usuario puede guardar posiciones en la base de datos, para posteriormente indicarle que grabe las posiciones en el controlador y por medio de un programa pedir que el robot realice los movimientos a esas posiciones.

Entre los inconvenientes están:

- El tiempo de ejecución de los movimientos está limitada por la respuesta del controlador.
- No hay un control sobre el punto almacenado, es decir que si el usuario almacena un punto que no es válido para el controlador, éste no podrá ejecutarse.

Logro.

- El número de puntos que es posible manejar es prácticamente ilimitado (está dado por el límite de espacio en disco duro de la base de datos).
- Es posible guardar los puntos que se saben correctos para el posicionamiento del robot y utilizarlos en diferentes programas sin necesidad de volver a guardarlos, además de que es posible guardar una descripción del punto indicado.

CONCLUSIONES

Al inicio de esta tesis se planteó la posibilidad controlar un robot a través de la red Internet. Al final las siguientes metas fueron alcanzadas.

Utilizar plataformas libres para el desarrollo de los sistemas de control.

En un 100% fue posible el uso de software libre gracias a que:

- Todos los programas seleccionados para el desarrollo de la tesis son de libre distribución. (En la Figura C.1 se muestra la lista de programas seleccionados para el proyecto)
- Los programas desarrollados para la comunicación con el controlador ACL fueron desarrollados en Lenguaje C y compilados con gcc que es un compilador de libre distribución.
- El lenguaje script PHP es de libre distribución por lo que todos los programas desarrollados para el proyecto fueron creados en un lenguaje de libre distribución.

Sistema Operativo	RedHat Linux 7.1
Base de Datos	PostgreSQL 7.0
Servidor Web	Apache 1.13.4
Lenguaje script para archivos del servidor Web	PHP ver 4.1
Lenguaje C de desarrollo para programas del puerto serie	Gcc GNU
Programa cámara Web	Webcam

Figura C.1 Lista de programas y lenguajes de libre distribución utilizados para el desarrollo del proyecto.

Utilizar un robot SCORBOT existente en el laboratorio de Sistemas de Manufactura Flexible de la Facultad de Ingeniería de la UNAM para el movimiento remoto por medio de conexiones Internet.

Fue posible mover el robot desde un lugar remoto, lo que abre nuevas posibilidades de uso a los equipos existentes en el laboratorio, esto permitirá desarrollar tecnologías innovadoras.

Lo anterior demostró lo siguiente:

1. Es posible, mediante el uso de una computadora, conectada a la Internet y auxiliada por los controladores proporcionados por el fabricante, ordenar las acciones que realizará el robot desde otra computadora remota.
2. Se demostró que es posible integrar un conjunto de programas de software libre para operar, desde la computadora remota por la Internet, el robot.
3. Con un programa escrito en C es posible mandar comandos al controlador del fabricante por medio del puerto serie de la computadora.
4. El usuario es capaz de mover las partes del robot desde su computadora por la Internet como si estuviera presente a través de un teclado multifuncional colocado en una página en un servidor Web.
5. Es factible utilizar una base de datos de libre distribución para controlar el acceso a la operación remota del robot.
6. Se pueden almacenar posiciones del robot en la base de datos para posteriormente pedir al controlador que mueva el robot a esas posiciones.
7. La interacción entre el usuario en la computadora remota y el robot puede llevarse a cabo a través de una página Web desarrollada a base de instrucciones escritas en un lenguaje script en la computadora, ese lenguaje script es de libre distribución.
8. El usuario podrá mandar comandos en lenguaje del controlador directamente al robot por medio de la página Web.
9. El usuario podrá escribir programas en lenguaje del controlador y almacenarlas en la base de datos para ejecutarlos posteriormente.
10. Es posible visualizar las posiciones del robot a través de las imágenes tomadas por una cámara de video y mostrar el cambiode esas posiciones a través de una página Web.

Sistema actual vs sistema controlado por la Internet

El sistema que actualmente se encuentra en uso por los alumnos de la Facultad de Ingeniería en el laboratorio de Manufactura Flexible resulta limitado por tener características que hoy en día resultan obsoletas.

Ventajas		
Ventajas	Sistema actual	Sistema controlado por la Internet
Manejo remoto por la Internet.	NO	SI
Envío de comandos al controlador.	SI	SI
Envío remoto de programas para ejecución.	NO	SI
Es posible extender o enriquecer el programa.	NO	SI
El programa es amigable y en ambiente de ventanas.	NO	SI
Manejo manual del robot	SI	SI
Control de errores	NO	NO

Beneficios del sistema controlado por la Internet.

- Manejo remoto del robot SCORBOT por métodos manuales.
- Ejecución de comandos que permiten una mayor flexibilidad en el manejo del robot.
- El sistema es accesible ya que utiliza programas de libre distribución que permiten una baja inversión respecto de otros sistemas que requieren de la compra de licencias por su uso.
- El sistema es fácil de aprender ya que sigue el tipo de diseño de las páginas Web que ha demostrado ser un modelo sencillo de entender y utilizar.
- Es posible extender y/o modificar los programas para que el robot realice operaciones que de otra manera serían imposibles con el sistema actual.

Deficiencias.

- A grandes distancias la imagen de la cámara web puede que no funcione debido al retardo de la carga de la información.
- El encendido del controlador y de la energía para los motores del robot debe ser de manera manual y en el sitio donde se localiza el robot.
- La falta de un control de errores entre el controlador y los programas del puerto serie pueden provocar que el controlador se bloquee y se requiera apagarlo y prenderlo nuevamente.

Para futuros proyectos.

- Se requiere aumentar la seguridad en cuanto al envío de comandos al controlador.
- Es importante tener un control de errores para evitar que el sistema deje de funcionar.
- Personalizar los programas para que cada usuario tenga acceso a sus propios programas almacenados en la base de datos.
- Permitir que se puedan ejecutar programas simultáneos en el controlador.
- Es posible eliminar el encendido manual del controlador y de los motores colocando un encendido electrónico que sería posible controlar a través del puerto paralelo de la computadora y conectándolo a las líneas de alimentación de cada uno de los elementos respectivamente.
- Crear una interfase que permita al usuario guardar programas completos en el controlador y ejecutarlos con el comando RUN para que la ejecución de los programas pueda ser sin supervisión del usuario.

Aplicaciones y proyectos a futuro.

Ésta Tesis pretende ser una herramienta que abra la posibilidad de obtener nuevas formas de aprovechar el concepto de mover dispositivos electromecánicos remotamente, aprovechando la tecnología de la Internet, que actualmente es un paradigma que ha llegado y que definitivamente continuamente cambia la manera en que la humanidad se relaciona entre sí y que en el futuro podrá ser algo cotidiano el estar telepresente mediante un robot, de forma humana, que imite los movimientos de una persona y que ésta persona se encuentre, a miles de km de distancia del robot y, quizá, gracias a la infraestructura de lo que hoy conocemos como la Internet.

Seguridad

Monitoreo con cámara, encendido y apagado de luces del hogar o de la oficina, registro de apertura y cierre de puertas y ventanas con registros en base de datos y visualización de reportes en línea.

Medicina

Manipulación de virus o bacterias peligrosas. Estudios de microscopía remota, sería posible visualizar una muestra de tejido por medio de una cámara y usar un robot que coloque las muestras, las almacene y que el sistema registre en bases de datos los ingresos al almacén para un análisis posterior y sin necesidad de estar en el lugar de trabajo.

Exploración

Manejo de robots móviles por control remoto para la exploración de terrenos peligrosos o de difícil acceso para el hombre.

Manufactura

Iniciar y detener producciones, monitorear fabricaciones y llevar un estado de las piezas fabricadas así como del control de calidad, todo ello desde un lugar remoto.

Investigación y Docencia

Entre los proyectos para investigación y docencia está la implementación de procesos de monitoreo remoto, como la revisión de un cultivo de bacterias desde un sitio web, medición de la temperatura, humedad de un lugar determinado.

Mantenimiento de un estanque de peces como el control de PH, filtración, nivel del agua, densidad de población ya que este modelo se podría aplicar a otros dispositivos sensores y actuadores como medidores de PH, viscocímetros, vólmetros, termómetros, etc.

Sistema de Manufactura Flexible controlado por la Internet

Es posible usar como base este trabajo para producir un sistema más completo que incluya el manejo del Centro de Manufactura Flexible de la Facultad. Utilizando los controladores y equipos del laboratorio sería posible controlar la producción a través de un sitio web con la posibilidad de registrar el estado de los procesos en línea.

APÉNDICE A

A.1 Instalación Servidor Web Apache con lenguaje script PHP.

1. Se consiguen los archivos de la Internet de los sitios:

http://www.php.net php-4.0.3pl1.tar.gz
http://www.apache.org apache_1.3.14.tar.gz

2. Descompactar los archivos

tar zxvf php-4.0.3pl1.tar.gz
tar zxvf apache_1.3.14.tar.gz

3. Ejecutar, como usuario root, este script llamado instalar_php_apache:

```
#########################################################################
#  18 Octubre 2001
#  Diseñado por: Ing. Carlos Enrique Contreras Vara
#  Nombre del archivo: instalar_php_apache
#  Notas Importantes:
#
#  Archivo para instalar y preparar lo necesario para
#  instalar el servidor web Apache y PHP como módulo de Apache
#
#  Verificar previamente la creación del usuario postgres
#
#  Ejecutar como root
#
#  Al finalizar la ejecución de este script
#
#  ejecutar apache
#  (ver documentación que viene en el archivo de internet)
#
#########################################################################

# Modificar colocando el directorio donde
# creó este archivo llamado instalar_php_apache

INSTALADOR=/usr/src/instalar

# Modificar colocando el directorio donde
# descompactó los archivos de php y apache

APACHE=/apps/swtmp/apache_1.3.14
PHP=/apps/swtmp/php-4.0.3pl1

export APACHE PHP INSTALADOR

echo CONFIGURANDO INICIALMENTE APACHE

cd $APACHE
./configure
```

```
echo CONFIGURANDO PHP

cd $PHP
./configure --with-pgsql --with-apache=$APACHE --enable-track-vars
make
make install

echo INSTALANDO APACHE

cd $APACHE
./configure --activate-module=src/modules/php4/libphp4.a
make
make install

cat $INSTALADOR/config_srm.conf >>/usr/local/apache/conf/srm.conf
cp $INSTALADOR/apache /etc/rc.d/init.d/
```

Archivo config_srm.conf :

```
Addtype application/x-httpd-php .php .phtml .php3 .php4
```

A.2 Instalación Base de Datos PostgreSQL.

1. Se consigue el archivo de la Internet en el sitio de:

   http://www.postgres.org postgresql-7.0.2.tar.gz

2. Descompactar el archivo

   tar zxvf postgresql-7.0.2.tar.gz

3. Ejecutar este script llamado instalar_postgres:

```
#!/sbin/bash
#############################################################################
#   18 Octubre 2001
#   Diseñado por: Ing. Carlos Enrique Contreras Vara
#   Nombre del archivo: instalar_postgres
#   Notas Importantes:
#
#   Archivo para instalar y preparar lo necesario para
#   instalar base de datos postgresql
#
#   Verificar previamente la creación del usuario postgres
#
#   adduser -d /home/postgres -G daemon postgres
#
#   Ejecutar como root
#
#   Al finalizar la ejecución de este script
#
#   Entrar como usuario postgres e inicializar la base
#   de datos (ver documentación que viene en el archivo de internet)
#
```

```
echo CONFIGURANDO PHP

cd $PHP
./configure --with-pgsql --with-apache=$APACHE --enable-track-vars
make
make install

echo INSTALANDO APACHE

cd $APACHE
./configure --activate-module=src/modules/php4/libphp4.a
make
make install

cat $INSTALADOR/config_srm.conf >>/usr/local/apache/conf/srm.conf
cp $INSTALADOR/apache /etc/rc.d/init.d/
```

Archivo config_srm.conf :

```
Addtype application/x-httpd-php .php .phtml .php3 .php4
```

A.2 Instalación Base de Datos PostgreSQL.

1. Se consigue el archivo de la Internet en el sitio de:

   http://www.postgres.org postgresql-7.0.2.tar.gz

2. Descompactar el archivo

   tar zxvf postgresql-7.0.2.tar.gz

3. Ejecutar este script llamado instalar_postgres:

```
#!/sbin/bash
################################################################################
#   18 Octubre 2001
#   Diseñado por: Ing. Carlos Enrique Contreras Vara
#   Nombre del archivo: instalar_postgres
#   Notas Importantes:
#
#   Archivo para instalar y preparar lo necesario para
#   instalar base de datos postgresql
#
#   Verificar previamente la creación del usuario postgres
#
#   adduser -d /home/postgres -G daemon postgres
#
#   Ejecutar como root
#
#   Al finalizar la ejecución de este script
#
#   Entrar como usuario postgres e inicializar la base
#   de datos (ver documentación que viene en el archivo de internet)
#
```

```
#  Crear usuario nobody en la base de datos
#
#############################################################################

# Modificar colocando el directorio donde
# creó este archivo llamado instalar_postgres

INSTALADOR=/apps/swtmp/instalar

# Modificar colocando el directorio donde
# descompactó el archivo de postgresql

POSTGRES=/apps/swtmp/postgresql-7.0.2

export POSTGRES INSTALADOR

echo Configurando POSTGRES

cd $POSTGRES/src/
./configure
make
make install

echo CAMBIANDO PERMISOS PARA POSTGRES

# Agrega la librería de postgres a /etc/ld.so.conf

cat $INSTALADOR/config_ld.so.conf >>/etc/ld.so.conf

# Refresca las librerias
/sbin/ldconfig

mkdir /usr/local/pgsql/logs/
touch /usr/local/pgsql/logs/server.log
chown -R postgres:daemon /usr/local/pgsql

#
#  Instala los manuales en línea de postgres
#
cd $POSTGRES/doc
tar zxvf man.tar.gz -C /usr/man/

cp $INSTALADOR/runpostgres ~postgres
chown postgres:daemon ~postgres/runpostgres

cat $INSTALADOR/config_rc.local >>/etc/rc.d/rc.local
```

Archivo config_ld.so.conf:

```
/usr/local/pgsql/lib
```

Archivo config_ rc.local:

```
su -c '-postgres/runpostgres' postgres
```

Archivo runpostgres:

```
#!/bin/bash
export PGDATESTYLE=European
/usr/local/pgsql/bin/postmaster -i -D /apps/bases >>
/usr/local/pgsql/logs/server.log 2>&1 &
```

Nota: /apps/bases es el directorio donde se encuentra la base de datos creada para PostgreSQL (ver documentación www.postgres.org)

APÉNDICE B

B.1 Referencia de algunos comandos de lenguaje ACL.

B.1 .1 Camandos de control del Robot.

MOVE	CLRBUF	OPEN	HOME
MOVED	CLOSE	CLR	MPROFILE
MOVEL	JAW	INT	
MOVELD	CON	SETANOUT	
MOVEC	SPEED	COFF	
MOVECD	SHOW	SPEED	
MOVES	TON	SHOW DAC	
MOVESD	EXACT	TOFF	

COMANDO | FORMATO	DESCRIPCION	MODO	NOTAS
MOVE MOVE <*pos*>	Mueve el robot a la posición especificada a la velocidad actual.	**DIRECT, EDIT**	
MOVE <*pos*> <*time*>	Mueve el robot a la posición especificada a la velocidad actual en el tiempo especificado.	**DIRECT, EDIT**	La ejecución de este comando es afectada por el comando EXACT.
MOVED <*pos*> { <*time*> }	Similar a MOVE excepto que el programa continúa al siguiente comando sólo hasta haber alcanzado exactamente la posición.	**EDIT**	
MOVEC MOVEC <*pos1*> <*pos2*>	Mueve el robot a la posición 1, con una trayectoria circular pasando por la posición 2.	**DIRECT, EDIT**	
MOVECD <*pos1*> <*pos2*>	Similar a MOVEC, excepto que el programa continúa al siguiente comando sólo hasta haber alcanzado exactamente la posición.	**EDIT**	Ver comando EXACT.
MOVEL MOVEL <*pos*>	Mueve el robot en línea recta a la posición a la velocidad actual.	**DIRECT, EDIT**	
MOVEL <pos> <time>	Mueve el robot en línea recta a la posición en el tiempo especificado.	**DIRECT, EDIT**	
MOVELD <pos> { <time> }	Similar a MOVEL, excepto que el programa continúa al siguiente comando sólo hasta haber alcanzado exactamente la posición.	**EDIT**	Ver comando EXACT.

COMANDO | FORMATO	DESCRIPCION	MODO	NOTAS
MOVES MOVES *<pvect>* *<start>***<end>**	Mueve el robot suavemente a través de todas las posiciones indicadas en el vector, desde el inicio hasta el fin de las posiciones. Sin aceleraciones ni desaceleraciones o pausas entre cada posición. Mueve a la velocidad actual.	**DIRECT, EDIT**	
MOVES *<pvect>* *<start>***<end> <time>**	Similar a MOVES, pero sujeto a un tiempo definido.	**DIRECT, EDIT**	
MOVESD *<pvect>* *<start>*<end>	Similar a MOVES, excepto que el programa continúa al siguiente comando sólo hasta haber alcanzado exactamente la posición.	**EDIT**	Refer to the EXACT
OPEN OPEN	Abre la pinza totalmente.	**DIRECT, EDIT**	El comando estándar para la apertura de la pinza.
OPEN { *<var>* }	Abre la pinza con fuerza adicional.		Inicializa el DAC de la pinza a *<var>*.
CLOSE CLOSE	Cierra la pinza totalmente.	**DIRECT, EDIT**	El comando estándar para el cierre de la pinza.
CLOSE <var>	Cierra la pinza con fuerza adicional. Inicializa el DAC de la pinza a *<var>*.	**EDIT**	Usar con precaución. Puede dañar la pinza. 0≤<var>≤5000
SPEED SPEED *<val>*	Inicializa el valor de la velocidad actual para todos los ejes	**DIRECT, EDIT**	1 ≤ *<val>≤ 100.* *Default es 50.*
SPEED{A/B} <val>	Inicializa la velocidad para el grupo A o B.		
SPEEDC *<val>* *<axis>*	Inicializa la velocidad para los ejes en el grupo C.		
EXACT EXACT A/B/C	Inizializa los movimientos en modo preciso.	**DIRECT, EDIT**	Sólo afecta comandos co el sufijo 'D': MOVED, MOVELD, MOVECD, MOVESD
EXACT OFF A,B/C	on/off en grupo A, B o C.	**DIRECT, EDIT**	

COMANDO | FORMATO	DESCRIPCION	MODO	NOTAS
HOME HOME { <n> }	Envía todos los ejes del robot, o un eje específico, a la posición home.	**DIRECT, EDIT**	Para correr HOME desde el teach pendant; teclear: RUN O
CLR CLR <n>	Borra (zeros) el valor del encoder específico	**DIRECT**	1≤ n≤ 11
LR *	Borra todos los encoders.		
COFF COFF	Apaga todos los servo controles de todos los ejes.	**DIRECT**	
COFFAIB	Apaga los servocontroles de los grupos A o B.		
COFF <axis>	Apaga todos los servo controles del eje especificado.		
CON CON	Prende todos los servo controles de todos los ejes.	**DIRECT**	
CON A/B	Prende los servocontroles de los grupos A o B.		
CON <axis>	Prende todos los servo controles del eje especificado.		

B.1 .2 Comandos de Control y Programación de Tiempo Real.

RUN	POST
PRIORITY	A (ABORT)
SET TME	STOP
PEND	SUSPEND
QPOST	CONTINUE
QPEND	DELAY
TRIGGER	WAIT

COMANDO | FORMATO	DESCRIPCION	MODO	NOTAS
RUN RUN *<prog >*	Corre el programa de usuario.	**DIRECT, EDIT**	
RUN *<prog> <priority>*	Corre el programa de usuario, sujeto a una prioridad.		
ABORT A	Aborta, inmediatamente todos los programas ejecutándose y detiene el movimiento de los ejes.	**DIRECT**	
A *<prog>*	Aborta el programa especificado.		
STOP STOP	Aborta todos los programas en ejecución.	**EDIT**	
STOP *<prog>*	Aborta la ejecución del programa especificado.		
SUSPEND SUSPEND *<prog>*	Suspende la eecución de un programa.	**DIRECT, EDIT**	
CONTINUE CONTINUE *<prog>*	Continúa la ejecución de un programa previamente detendido por el comando SUSPEND.	**DIRECT, EDIT**	
PRIORITY PRIORITY *<prog> <var>*	Inicializa la prioridad de *<prog>* a <var>. Si el CPU está cargado, primero maneja las tareas y programas con mayor prioridad.	**EDIT**	**1 ≤ *<var>* ≤ 10. Default 5.**
DELAY DELAY *<var>*	Suspende la ejecución del programa por el tiempo especificado.	**EDIT**	*<var> está especificado en unidades de* 10 millisegundos.
WAIT **WAIT** *<varl> <cond>*<var2>	Suspende la ejecución del programa, hasta que la condición se satisfaga (verdadera)	**EDIT**	Condition can be: <,>,= , < =,> = , < >

B.1.3 Comandos de Manipulación y Definición de Posiciones.

DEFP	TEACHR	SHIFTC
DIMIP	SETPV	SETP
HERE	SETPVC	UNDEF
HERER	SHIFT	DELP
TEACH		

COMANDO / FORMATO	DESCRIPCION	MODO	NOTAS
DEFP DEFF{A/B} *<pos>*	Crea (define) una posición en el grupo A o B o eje en el grupo C	**DIRECT**	*<pos>* es un nombre de hasta 5 caracteres.
DEFPC <pos> <axis>		**EDIT**	
DIMP DIMP{A/B} *<vect[n]>*	Crea un vector de n posiciones en el grupo A o B o eje en C	**DIRECT**	*<vect> es un nombre de hasta* 5 caracteres.
DIMPC <vect [n]> <axis>		**EDIT**	
HERE HERE *<pos>*	Graba en coordenadas de uniones la localización de una posición absoluta del robot.	**DIRECT,EDIT**	Unidades de uniones del encoder. <pos> debe ser creada usando los comandos DEFP o DIMP.
HERER HERER *<pos>*	Inicializa, en coordenadas de uniones, la localización de una posición relativa a la posición actual del robot.	**DIRECT**	
HERER *<pos2> <pos1>*	Graba en coordenadas de uniones, la localización de <pos2> relativa a <pos1>.	**DIRECT EDIT**	
TEACH TEACH *<pos>*	Inicializa en coordenadas cartecianas la localización de una posición absoluta del robot.		Cartecianas: XYZ en enths of millimetros los valores de pitch/roll en décimas de grado. tenths of degree
TEACHR TEACHR *<pos>*	Inicializa en coordenadas cartecianas la localización de una posición relativa a la posición actual del robot.	**DIRECT.**	
TEACHR *<pos2> <pos1>*	Inicializa en coordenadas cartecianas <pos2> relativa a <pos1>.		

COMANDO | FORMATO	DESCRIPCION	MODO	NOTAS
SETPV SETPV <*pos*>	Inicializa en coordenadas de uniones la localización a tope de la posición del robot.	**DIRECT**	
SETPV <*pos*> <axis><val>	Inicializa en coordenadas de uniones un valor de un eje a la posición previamente guardada.	**DIRECT, EDIT**	1 ≤ <*axis*> ≤ 11
SETPVC SETPVC <*pos*> <*coord*><val>	Inicializa en coordenadas cartecianas un valor de un eje a la posición previamente guardada.	**DIRECT, EDIT**	
SHIFT SHIFT <*pos*> BY <*axis*><val>	Cambia la localización de la posición por el valor en coordenadas de uniones.	**DIRECT EDIT**	**1** ≤ <*axis*> ≤11
SHIFTC <*pos*> BY <val> <coord>	Cambia la localización de la posición por el valor en coordenadas cartecianas.	**DIRECT, EDIT**	
SETP SETP <*pos2*> = <*pos1*>	Copia el valor de <*pos1*> a<pos2>.	**DIRECT, EDIT**	
UNDEF UNDEF <*pos*>	Inicializa los valores de las posiciones. Los valores de las posiciones son eliminados, pero la posición continúa definida.	**DIRECT**	
DELP DELP <*pos*>	Borra posiciones y vectores de posiciones de la memoria RAM de usuario.	**DIRECT, EDIT**	
DELP <*pvect*>			

B.1.4 Comandos de Reporte.

ATTACH ?	DIR	SHOW
CONFIG ?	LIST	VER
DISABLE ?	STAT	FREE

COMANDO | FORMATO	DESCRIPCION	MODO	NOTAS
ATTACH ?	Muestra el estado actual ATTACH status.	**DIRECT**	
CONFIG ?	Muestra la configuración actual.	**DIRECT**	
DISABLE ?	Muestra la lista de todas las entradas y salidas apagadas.	**DIRECT**	
DIR	Muestra la lista de todos los programas de usuario	**DIRECT**	
LIST LIST { *<prog>* }	Muestra todas las líneas de todos los programas de usuario o del programa especificado.	**DIRECT**	
LISTP	Muestra una lista de rodas las posiciones definidas.	**DIRECT**	
LISTPV LISTPV *<pos>*	Muestra las coordenadas de las uniones de una posición especifica.	**DIRECT**	La coordenadas cartesianas también son desplegadas para las posiciones del robot.
POSITION	Muestra la localización actual del brazo del robot.		
LISTVAR	Muestra una lista de todas las variables de usuario y de sistema.	**DIRECT**	
STAT STAT	Muestra una lista de los programas de usuario activos: nombre, prioridad y estado.	**DIRECT**	
SHOW SHOW DIN	Muestra el estado de las 16 entradas. Si LSON está activado, muestra el estado de todfos los switches de límite.	**DIRECT**	Input ON = 1 Input 0FF = O
SHOW DOUT	Muestra el estado de las 16 salidas.		Output 0FF = O
SHOW SPEED	Muestra la velocidad actual.		

APÉNDICE C

C.1 Conectores Seriales D25 y D9.

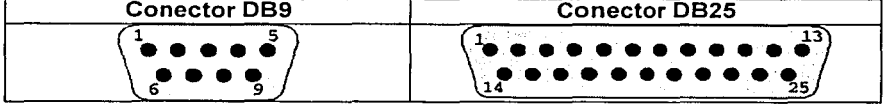
D-Tipo-25 No. Pin	D-Tipo-9 No. Pin	Abreviación	Nombre Completo
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

C.1.1 Función de los Conectores Seriales

Valor	Nombre	Función
TD	Transmit Data	Salida de datos (TXD)
RD	Receive Data	Entrada de Datos (RXD)
CTS	Clear to Send	Indica que el Modem está listo para intercambiar datos.
DCD	Data Carrier Detect	Cuando el modem detecta un "Carrier" al final de la línea, la línea se activa.
DSR	Data Set Ready	Esto le indica al UART que el modem está listo para establecer una conexión.
DTR	Data Terminal Ready	Es el lado opuesto a DSR. This tells the Modem that the UART is ready to link.
RTS	Request To Send	Informa que el UART del Modem está listo para intercambiar datos.
RI	Ring Indicator	Se activa cuando el modem detecta una señal de "ring" desde el PSTN.

C.1.2 Conexiones del puerto serial Cable DB25-DB-25, DB9-DB9, DB9-DB25, DB25-DB9.

DB9	DB25		CONEXIÓN		DB25	DB9
3	2	TD	----------→	RD	3	2
2	3	RD	←----------	TD	2	3
5	7	SG	←----------→	SG	7	5
4	20	DTR		DTR	20	4
6	6	DSR	←	DSR	6	6
1	8	CD	←	CD	8	1
7	4	RTS	←	RTS	4	7
8	5	CTS	←	CTS	5	8



C.2 Señales de comunicación del puerto serial.

Cada palabra es sincronizada usando un bit de inicio y un reloj interno en cada lado, mantiene la cuenta del tiempo.

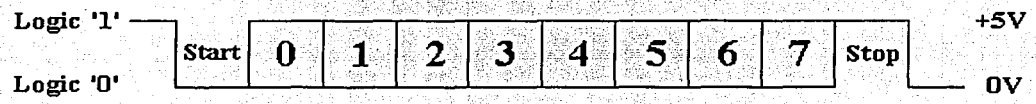


Figura FA1.1 : TTL/CMOS Onda Logica Serial

El diagrama FA1.1 muestra la onda del UART usando el formato común 8N1 que significa 8 bits de datos, Sin paridad y un bit de parada. La línea RS-232, cuando está en modo inactivo está en marca de estado (1 lógico). La transmisión comienza con un bit de inicio el cuál es (0 lógico). En ese momento cada bit es enviado por la línea, uno por uno. El LSB (Bit Menos Significativo) es enviado primero. Un Bit de parada (1 lógico) es agregado a la señal para completar la transmisión. El diagrama muestra que el bit posterior o Bit de parada debe ser 0 lógico. Esto significa que existe otra palabra y es su bit de inicio. Si no hay otra palabra más por enviar la línea de recepción permanecerá en el estado lógico 1.

APÉNDICE D

D.1 Terminal POSIX.

Entrada canónica

Los caracteres de entrada son colocados en un buffer del cual pueden ser modificados interactivamente por el usuario hasta que se reciba un carácter CR (carácter de retorno) o LF (avance de línea)

Entrada RAW o en Bruto

Los caracteres de entrada son pasados exactamente cuando se van recibiendo son enviados.

D.1.1 Estructura POSIX Termios.

Miembro	Descripción
C_cflag	Opciones de Control
C_lflag	Opciones de Linea
C_iflag	Opciones de Entrada
C_oflag	Opciones de salida
C_cc	Caracteres de Control

D.1.2 Modo de Control POSIX.

Constante	Descripción
CBAUD	Bit mask for baud rate
B0	0 baud (drop DTR)
B50	50 baud
B75	75 baud
B110	110 baud
B134	134.5 baud
B150	150 baud
B200	200 baud
B300	300 baud
B600	600 baud
B1200	1200 baud
B1800	1800 baud
B2400	2400 baud
B4800	4800 baud
B9600	9600 baud
B19200	19200 baud
B38400	38400 baud
EXTA	External rate clock
EXTB	External rate clock
CSIZE	Bit mask for data bits
CS5	5 data bits
CS6	6 data bits
CS7	7 data bits
CS8	8 data bits

CSTOPB	2 stop bits (1 otherwise)
CREAD	Enable receiver
PARENB	Enable parity bit
PARODD	Use odd parity instead of even
HUPCL	Hangup (drop DTR) on last close
CLOCAL	Local line - do not change "owner" of port
LOBLK	Block job control output
CNEW_RTSCTS	Enable hardware flow control (not supported on all platforms)

D.1.3 Constantes tcsetattr POSIX.

Constante	Descripción
TCSANOW	Realizar cambios ahora sin esperar a que los datos se completen.
TCSADRAIN	Esperar hasta que todo haya sido transmitido
TCSAFLUSH	Limpiar los buffers de entrada y salida y realizar el cambio.

D.1.4 Constantes de Modo Local POSIX (c_lflag).

Constante	Descripción
ISIG	Habilitar SIGINTR, SIGSUSP, SIGDSUSP, and SIGQUIT signals
ICANON	Habilita entrada canonica (sino en bruto)
XCASE	Pasar mayúsculas\minúsculas (obseleto)
ECHO	Habilita la repetición de los caracteres de entrada.
ECHOE	Carácter de repetición de borrado como BS-SP-BS
ECHOK	Repetir carácter NL (nueva línea) despues del caracter terminar (kill)
ECHONL	Repetir NL
NOFLSH	Deshabilita limpieza de los buffers de entrada después del carácter de interrumpir o quitar
IEXTEN	Habilitar funciones extendidas
ECHOCTL	Habilitar el caracter control como ^ y borrado como ~?
ECHOPRT	Repetir caracter borrado como caracter borrado
ECHOKE	BS-SP-BS entire line on line kill
FLUSHO	Limpieza de salida
PENDIN	Reescribe la entrada pendiente en la próxima lectura o carácter de entrada
TOSTOP	Mandar SIGTTOU para salida en segundo plano

D.1.5 Constantes de Modo de Entrada POSIX (c_iflag).

Constante	Descripción
INPCK	Habilitar chequeo de paridad
IGNPAR	Ignorar errores de paridad
PARMRK	Marca errores de paridad
ISTRIP	Tirar bits de paridad
IXON	Habilitar control de flujo por software (de salida)
IXOFF	Habilitar control de flujo por software (de entrada)
IXANY	Permitir cualquier caracter para iniciar flujo de nuevo
IGNBRK	Ignorar condición de interrupción
BRKINT	Mandar SIGINT cuando se detecte una condición de interrupción
INLCR	Mapea NL a CR
IGNCR	Ignora CR
ICRNL	Mapea CR a NL
IUCLC	Mapear mayúsculas a minúsculas
IMAXBEL	Repite BEL en una línea de entrada muy larga

D.1.6 Constantes de Modo de Salida POSIX (c_oflag).

Constante	Descripción
OPOST	Postprocess output (not set = raw output)
OLCUC	Map lowercase to uppercase
ONLCR	Map NL to CR-NL
OCRNL	Map CR to NL
NOCR	No CR output at column 0
ONLRET	NL performs CR function
OFILL	Use fill characters for delay
OFDEL	Fill character is DEL
NLDLY	Mask for delay time needed between lines
NL0	No delay for NLs
NL1	Delay further output after newline for 100 milliseconds
CRDLY	Mask for delay time needed to return carriage to left column
CR0	No delay for CRs
CR1	Delay after CRs depending on current column position
CR2	Delay 100 milliseconds after sending CRs
CR3	Delay 150 milliseconds after sending CRs
TABDLY	Mask for delay time needed after TABs
TAB0	No delay for TABs
TAB1	Delay after TABs depending on current column position
TAB2	Delay 100 milliseconds after sending TABs
TAB3	Expand TAB characters to spaces
BSDLY	Mask for delay time needed after BSs
BS0	No delay for BSs

BS1	Delay 50 milliseconds after sending BSs
VTDLY	Mask for delay time needed after VTs
VT0	No delay for VTs
VT1	Delay 2 seconds after sending VTs
FFDLY	Mask for delay time needed after FFs
FF0	No delay for FFs
FF1	Delay 2 seconds after sending FFs

D.1.7 Constantes de Caracteres de Control POSIX (c_cflag).

Constante	Descripción	Key
VINTR	Interrupt	CTRL-C
VQUIT	Quit	CTRL-Z
VERASE	Erase	Backspace (BS)
VKILL	Kill-line	CTRL-U
VEOF	End-of-file	CTRL-D
VEOL	End-of-line	Carriage return (CR)
VEOL2	Second end-of-line	Line feed (LF)
VMIN	Minimum number of characters to read	
VTIME	Time to wait for data (tenths of seconds)	

D.2 Programas escritos en lenguaje C para mandar instrucciones al controlador.

D.2.1 Archivo scorbot.c.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <stdarg.h>
int main(int argc,char **argv)
 {
       char *value;
       char *c;
       struct termios port_options;
       int serial_port;
       int n,i,t,m, tamano = 100;
       char buffer[1];
       char *character;
      /* El programa recibe por lo menos 1 argumento
      *  que debe ser el comando que se le envía al
      *  controlador ACL, en caso de no contener argumentos
      *  el programa termina
      */
       if (argc<2)
          fprintf(stderr, "Error: El numero de argumentos es incorrecto.\n
                           Este programa recibe una cadena.\n"), exit(1);
```

```
        serial_port = open("/dev/ttyS1", O_RDWR | O_NOCTTY | O_NDELAY);
        if (serial_port == -1)
            perror("open_port : Enable to open /dev/ttyS1");
        else
        {
            fcntl(serial_port, F_SETFL, FNDELAY);
            tcgetattr(serial_port, &port_options);
            cfsetispeed(&port_options, B9600);
            cfsetospeed(&port_options, B9600);
            port_options.c_cflag |= (CLOCAL | CREAD);
            port_options.c_cflag &= ~PARENB;
            port_options.c_cflag &= ~CSIZE;
            port_options.c_cflag |= CS8;
            port_options.c_lflag &= ~(ICANON | XCASE |ECHO | ECHOE |
ISIG);
            port_options.c_iflag &= ~INPCK;
            port_options.c_iflag |= (IGNPAR | ISTRIP);
            port_options.c_iflag &= ~(IXON | IXOFF | IXANY);
            port_options.c_oflag &= ~(OPOST | ONLCR);
            tcsetattr(serial_port, TCSANOW, &port_options);
            if ((value = malloc (tamano)) == NULL)
                    return 0;
            for(i=1;i<argc;i++){
               m= sscanf(argv[i],"%s",&value);
               t=strlen(argv[i]);
               n = write(serial_port, &value,t);
               n = write(serial_port, " ",1);
            }
               n = write(serial_port, "\r",1);
       close(serial_port);
}
}
```

D.2.2 Archivo scorbot_archivo.c.

```
   #include <stdio.h>
   #include <string.h>
   #include <unistd.h>
   #include <fcntl.h>
   #include <errno.h>
   #include <termios.h>
   #include <stdarg.h>

int main(int argc,char **argv)
{
    char *value;
    char *c;
    int t,m,n,size = 100;
    struct termios port_options;
    int serial_port;
    char *s;
    FILE *stream;
    if (argc<2)
      fprintf(stderr, "Error: El numero de argumentos es incorrecto.\n
Este programa recibe el nombre de un archivo.\n"), exit(1);
     serial_port = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
       if (serial_port == -1)
```

```
            perror("open_port : Enable to open /dev/ttyS1");
        else
        {
            fcntl(serial_port, F_SETFL, FNDELAY);
            tcgetattr(serial_port, &port_options);
            cfsetispeed(&port_options, B9600);
            cfsetospeed(&port_options, B9600);
            port_options.c_cflag |= (CLOCAL | CREAD);
            port_options.c_cflag &= ~PARENB;
            port_options.c_cflag &= ~CSIZE;
            port_options.c_cflag |= CS8;
            port_options.c_lflag &= ~(ICANON | XCASE |ECHO | ECHOE |
ISIG);
            port_options.c_iflag &= ~INPCK;
            port_options.c_iflag |= (IGNPAR | ISTRIP);
            port_options.c_iflag &= ~(IXON | IXOFF | IXANY);
            port_options.c_oflag &= ~(OPOST | ONLCR);
            tcsetattr(serial_port, TCSANOW, &port_options);
            if ((s = malloc (1024)) == NULL)
                        return 0;
            if ((value = malloc (1024)) == NULL)
                        return 0;
            if((stream = fopen (argv[1], "r")) != (FILE *)0) {
               while((fgets(s, 1023, stream)) != (char *)0 ) {
                     printf("%s",s);
                     t=strlen(s);
                     m=sscanf(s,"%1023c",&value);
                     t=t-1;
                     printf("Texto: %s Tamano: %d\n",&value,t);
                     n = write(serial_port, &value,t);
                     n = write(serial_port, "\r",1);
                     sleep(1);
              }
               n = write(serial_port, "\r",1);
            } else {
               printf("Error al abrir el archivo!!");
            }
        }
}
```

GLOSARIO

ACL
Advanced Control Languaje. Lenguaje de Control Avanzado. Ofrece una gran gama de capacidades para programar el robot SCORBOT-ERV. ACL es "quemado" en la memoria EPROM del controlador y es accesado vía puerto RS-232C usando casi cualquier terminal ASCII.

ActiveX
Lenguaje de programación al estilo de Java propuesto por Microsoft.

Ancho de banda / Bandwidth
Capacidad máxima de transmisión de enlace. Usualmente se mide en bits por segundo (bps). Es uno de los recursos más caros de toda la red y es una de las principales limitantes para el desarrollo de aplicaciones que requieren transferir grandes cantidades de información.

Applet
Programa escrito en Java y que se ejecuta en un navegador compatible con Java como por ejemplo HotJava(TM) o Netscape Navigator(TM).

Archie
Herramienta de software utilizada para localizar archivos en servidores FTP. A partir de 1994 empezó a caer en desuso por el auge del WWW.

ASCII
American Standard Code Information Interchange. Estándar que define como repesentar dígitos, letras, signos y signos de puntuación en computadoras.

ATS
Programa para interactuar con el controlador de ACL,corre en en cualquier computadora PC IBM compatible.

Attachment
Archivo pegado. El documento, texto, archivo o imagen anexado a un mensaje de correo electrónico.

Autenticación
Proceso mediante el cual se comprueba la identidad de un usuario en la red.

Backbone
Es la infraestrucctura de conexión principal de una red y está constituída por los enlaces de mayor velocidad dentro de dicha red.

Bps
Bits por segundo. Unidad de medida que indica los bits por segundo transmitidos por un equipo.

CGI
Abreviación de Common Gateway Interface, el CGI es un programa de interfaz que permite al servidor Web utilizar programas externos para realizar una función específica. También denominado pasarelas o CGI "scripts", estos programas consisten generalmente de de una serie de instrucciones escritas en un lenguaje de programación como C o PERL que procesan la petición de un navegador, ejecutan un programa y formatean los resultados en HTML de manera que puedan ser presentados en el navegador. Se utilizan para añadir interactividad a una página web al permitir a los usuarios rellenar y enviar formularios que podrán ser procesados (como un catálogo en línea), acceder a bases de datos por medio de una búsqueda, y obtener acceso a un sitio protegido escribiendo una contraseña. Los scripts CGI también se usan para introducir una variedad de sistemas de análisis y de tráfico de medida de audiencia de un sitio

CORBA
CORBA es el acrónimo de *Common Object Request Broker Arquitecture*. CORBA es una especificación, abierta y no dependiente de ninguna empresa, para una arquitectura e infraestructura orientada a que las aplicaciones informáticas puedan trabajar juntas a través de las redes de comunicaciones. Los programas basados en CORBA pueden interactuar unos con otros, independientemente de la compañía que los haya creado, de la computadora y sistema operativo sobre el que corran, del tipo de red y del lenguaje de programación en que hayan sido escritos.

Cliente / Client
Programa que interactúa y hace peticiones a un programa servidor que que usa el mismo protocolo.

Cookie
Mecanismo utilizado para que un servidor Web pueda guardar y leer la información en la computadora que corre el software cliente. Se utiliza para conocer las preferencias de los usuarios, para dar acceso a servidores que requieren de autenticación, etc.

Cracker
Programador que utiliza sus conocimientos para crear virus, romper la seguridad de sistemas o protección de programas con fines dañinos.

Chat
Plática. Servicio o programa que permite platicar en vivo con otros usuarios por medio de mensajes escritos en la pantalla.

Dial-up
Cuenta de Internet que permite la conexión vía módem a la red. Normalmente requiere de la contratación con un ISP (Internet Service Provider, Proveedor de servicios de Internet) quien cuenta con una conexión dedicada a la red y revende el acceso a través de bancos de módems.

Dirección IP
Internet Protocol; Protocolo de Internet. Dirección única de un dispositivo en una red TCP/IP. Consiste en cuatro números entre 0 y 255 separados por puntos (ej. 255.255.240.70)

DNS
Domain Name System; Sistemas de nombres de dominios. Sistema para facilitar la administración y localización de direcciones IP que funciona asignando uno o más alias a cada dirección IP. También suele llamarse así a las computadoras encargadas de administrar la base de datos del sistema de nombres de dominio. Una aplicación del **DNS** es la creación de nombres de dominio para correo.

Download
Bajar un archivo, es decir, transferirlo de una máquina remota a tu máquina local.

e-mail address
Dirección de correo electrónico con el siguiente formato: usuario@proveedor.**tipo.país**

Encriptación
Procedimientos para codificar información de manera que pueda transmitirse sin peligro de ser interceptada o alterada antes de que llegue al destinatario.

FAQ
Frecuently Asked Questions; Preguntas más frecuentes. Es un archivo con la respuesta a las preguntas más comunes sobre algún tema.

Firewall
Pared de Fuego. Mecanismo utilizado para proteger una red o computadora conectada a la Internet de accesos no autorizados.

Freeware
Software que ha sido puesto a disposición de la comunidad por sus autores. Estos programas pueden ser utilizados y distribuídos sin necesidad de paga alguno.

FTP Anonimo
Transferencia de archivos efectuada con la clave especial de usuario **anonymus** y con la dirección de e-mail como password. Esta es la clave de cortesía o huésped (guest) para acceder a archivos públicos (en el directorio /pub)

FTP
Siglas de File Transfer Protocol. Método muy común para transferir uno o más archivos de una computadora a otra. FTP es un medio específico de conexión de un sitio Internet para cargar y descargar archivos. FTP fue desarrollado durante los comienzos de la Internet para copiar archivos de una computadora a otra. Con la llegada del World Wide Web, y de los navegadores , ya no se necesitan conocer sus complejos comandos; se puede utilizar FTP escribiendo el URL en la barra de localización que se encuentra en la parte superior de la pantalla del navegador. Por ejemplo, al escribir *ftp://nombre.del.sitio/arpeta/nombredelarchivo.zip* se transfiere el archivo nombredelarchivo.zip al disco duro de la computadora. Al escribir *ftp://nombre.del.sitio/carpeta/* da una lista con todos los archivos disponibles en esa carpeta.
Cuando un navegador no está equipado con la función FTP, o si se quiere cargar archivos en una computadora remota, se necesitará utilizar un programa cliente FTP. Para utilizar el FTP, se necesita conocer el nombre del archivo, la computadora en que reside y la carpeta en la que se encuentra. La mayoría de los archivos están disponibles a través de "anonymous FTP", lo que significa que se puede entrar en la computadora con el nombre de usuario "anónimo" y utilizar la dirección de correo electrónico propia como contraseña.

Gateway
Conexión compartida entre una red de área local (LAN) y un sistema grande, como un computador mainframe o una *red de paquetes conmutados*, cuyos *protocolos de comunicación* son diferentes. Por lo general, además de ser más lento que un *puente* o un *enrutador*, un *gateway* es una combinación de equipo y programas con su propio procesador y memoria que se utilizan para ejecutar conversiones de protocolo.

GIF
Graphics Interchange Format; Formato de Intercambio de Gráficos. Formato estándar para imágenes gráficas en la Internet.

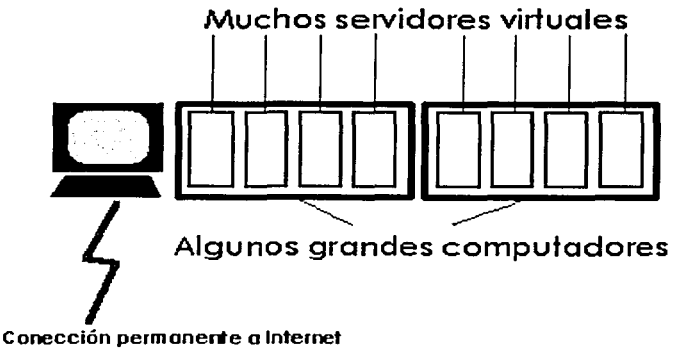
Gopher
Herramienta para organización de información en la Internet. Puede verse como un precursor del Web y, aunque estan desapareciendo, aún quedan miles de servidores Gopher en servicio y muchos navegadores, como el Netscape, tienen un cliente Gopher.

Hacker
Programador que disfruta buscando conocer más a los sistemas que emplea, pero que hace un mal intencionado uso de ellos.

Hipertexto / Hipertext
Un texto especial que contiene la dirección de otro texto, con lo que se convierte en algo más que un texto, un hipertexto.

Hosting
El Hosting se refiere a un espacio en un servidor que se renta para la colocación de páginas en la Internet. El pago por hosting incluye como base una cantidad de espacio en disco duro. Usualmente el límite más importante que impone el proveedor de hosting es un límite de transferencia en bytes mensual o similar, como un límite de número de usuarios mes.

El negocio de los servidores virtuales es el siguiente: ellos disponen de algunos grandes servidores con una línea dedicada (conección permanente). Cada uno de sus grandes servidores atiende múltiples servidores virtuales.



http
HiperText Transfer Protocol, Protocolo de transmisión de hipertexto.

Abreviatura de *Hypertext Markup Langua*ge. Lenguaje estándar de *Hipertex*to que se utiliza para crear páginas *World Wide We*b y otros documentos hipertexto. Cuando se accede a un documento HTML se observa una mezcla de texto, gráficos y encadenamientos a otros documentos. Si se selecciona un encadenamiento, el documento relacionado se abrirá automáticamente, sin importar su localización.

IIS
(*Internet Information Server*) Servidor web para entornos Windows. Es uno de los servidores web más extendidos junto al servidor Apache.

Internet
Es la red de redes. La super carretera de información, la interconexión de redes a nivel mundial.

Intranet
Red de uso privado que emplea los mismos estándares y herramientas de la Internet. Es uno de los segementos del mercado de computación que más auge está cobrando.

ISP
Internet Service Provider; Proveedor de Servicios de Internet. Compañía dedicada a revender el acceso a Internet. Puede proveer desde enlaces dial-up hasta enlaces dedicados de muy alta velocidad. También puede ofrecer servicios adicionales como desarrollo y mantenimiento de web sites, de servidores de correo electrónico, etc.

Java
Lenguaje de programación independiente de la plataforma, creado por Sun Microsystems. Está pensado para una arquitectura cliente/servidor en la que sólo es necesario intercambiar pequeñas porciones de código (llamadas Applets) que son ejecutadas por el cliente.

JPEG
Joint Photographic Experts Group; Grupo de Expertos de ensamble fotográfico. Nombre del Comité que diseño el estándar de compresión de las imágenes fotográficas conocido como JPEG/JPG. Está pensado especialmente para imágenes fotográficas con muchos colores.

Lenguaje de programación
Lenguaje que los programadores usan para comunicar instrucciones a una computadora y poder ejecutar un programa.

Liga / Link
Palabra o frase subrayada que sirve de conexión a millones de documentos hipertexto interrelacionados que forman la red a lo ancho del mundo (World Wide Web)

LINUX
Nombre con el que se denomina al kernel de un sistema operativo, que esta bajo la licencia GNU. Su creador es Linus Torvalds, y del nombre de este se genera un acrónimo "Linus uniX".

Login
Entrada, Acceso, Conectado. Clave de acceso. Proceso de entrar a un sistema dando una clave y una contraseña.

Multimedia (Multimedios)
Sistemas informáticos que integran audio, vídeo y datos. Multimedia hace referencia a la utilización simultánea de más de un tipo de medio como por ejemplo texto con sonido, imágenes estáticas o dinámicas con música, etc.

Multiusuario
El concepto de que numerosos ususrios pudieran accesar aplicaciones o el potencial de procesamiento de una sola PC era un mero sueño hace unos cuantos años. UNIX y Windows NT ayudaron a que este sueño se convirtiera en realidad. La capacidad de Linux para asignar tiempo del microprocesador a numerosas aplicaciones Simultáneas se prestó como consecuancia a servir a

numerosas personas al mismo tiempo, cada una ejecutando una o mas aplicaciones.

Navegador / Browser
Programa cliente que permite navegar o recorrer el WWW manejando cualquiera de los siguientes protocolos y servicios: http, FTP, Gopher, e-mail y News.

Nombre de dominio
Domain Name. Nombre base de un lugar en la Internet, por ejemplo www.ienlaces.com.mx.

Página Hogar / Home Page
Documento en el WWW que es la página de entrada o principal de una organización, empresa o persona.

PHP
PHP (acronimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Una respuesta corta y concisa, pero que significa realmente? Un ejemplo nos aclarará las cosas:

Ejemplo 1-1. Un ejemplo introductorio

```
<html>
<head>
<title>Ejemplo PHP</title>
</head>
<body>
<?php echo "Hola, este es un ejemplo con PHP!"; ?>
</body>
</html>
```

Podemos ver que no es lo mismo que un script CGI escrito en otro lenguaje de programación como Perl o C. En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (introducido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producir un texto). El código PHP se incluye entre *etiquetas especiales de comienzo y final* que nos permitirán entrar y salir del modo PHP. Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la má;quina cliente, es que el código PHP es ejecutado en el servidor. Si tuviesemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente sólamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

Plug-In
Módulo que se añade al programa navegador para realizar funciones adicionales a la navegación, como pueden ser el despliegue de animaciones, sonido en tiempo real, etc.

POP
Post Oficce Protocol; Protocolo de Oficina Postal. Protocolo empleado por el software cliente para extraer mensajes de los servidores de correo.

PPP
Point to Point Protocol. Protocolo punto a punto que maneja TCP/IP por líneas telefónicas y que permite correr un programa cliente (navegador, correo electrónico u otro) en la máquina del usuario.

Protocolos de comunicaciones
Medios de comunicación establecidas para regir la manera en que se transmiten los datos en una red de computadoras (TCP/IP, IPX, etc.)

RMI
Es un mecanismo RPC exclusivo de Java y que permite que objetos creados con este lenguaje y que se ejecutan en computadoras distintas y puedan comunicarse entre sí.

Script
Es un pequeño programa con una serie de instrucciones, las cuales ejecutan automáticamente una tarea específica. Los script son a menudo ejecutados cuando un programa es instalado o cuando una página web se descarga. Ellos pueden ser escritos en diferentes lenguajes de programación.

Search Engine
Máquina de búsqueda. Programa que permite a los usuarios buscar información a través de la Internet. En el WWW algunas de las más famosas son YAHOO, LYCOS, WEBCRAWLER.

Servidor /Server
Sistema que proporciona recursos (por ejemplo, servidores de archivos, servidores de nombres). En la Internet este término se utiliza a menudo para designar a aquellos sistemas que suministran información a los usuarios de la red.

Shareware /programas compartidos
Programas informáticos que se distribuyen para hacer las pruebas, con el compromiso de pagar al autor su precio, que es usualmente bajo, una vez probado el programa y haya pasado cierto tiempo de uso.

Sistema Operativo
Es un conjunto complejo de códigos de computadora que proporciona los protocolos de proceso de operatividad, o leyes de comportamiento. Sin un sistema operativo su computadora sería incapaz de interpretar los comandos que introduzca o de ejecutar un programa sencillo.

Sintaxis
Las reglas que rigen la formulación de las instrucciones en un programa de computación.

Servidor / Server
Programa que responde a las peticiones de un programa cliente usando el mismo protocolo.

SLIP
Serial Line Internet Protocol; Protocolo Internet en Línea serial. Protocolo antecesor al PPP que también permite el establecimiento de conexiones TCP/IP a través de una línea seriada.

SMTP
Simple Mail Transfer Protocol; Protocolo sencillo de transferencia de correo. Protocolo original de intercambio de correo en la Internet. Sólo permite el intercambio de mensajes ASCII, por lo que está sendo reemplazado por MIME.

SPAM
Correo masivo. Se refiere a mensajes publicitarios no solicitados que inundan nuestro buzón de correo electrónico.

SSH
Puede conectarse remotamente con el servidor a través del protocolo SSH (*Secure Shell*), que es análogo al protocolo TELNET pero más seguro, al estar cifradas todas las transmisiones. Esta es una característica pensada para usuarios avanzados y en realidad no es necesaria para el funcionamiento del sitio web.

T1
Conexión dedicada de alta velocidad (1.54 Mbps)

T3
Conexión dedicada de alta velocidad (44 Mbps)

TCP/IP (Transmission Control Protocol/Internet Protocol)
Conjunto de protocolos usados en la Internet para soportar servicios tales como acceso remoto telnet, transferencia de archivos FTP y correo electrónico SMTP.

Telnet
Telnet es un programa que permite acceder a computadoras distantes en la Internet a los cuales se tiene acceso. Una vez que se ha accedido a un sistema distante, se pueden descargar archivos y realizar las mismas funciones que si se estuviese directamente conectado a la computadora distante.

Trumpet
Programa de tipo Winsock que maneja la comunicación en conexiones telefónicas con enlaces SLIP o PPP.

URL
Siglas de Uniform Resource Locator. Es la dirección de un sitio o de una fuente, normalmente un directorio o un archivo en el World Wide Web y la convención que utilizan los navegadores para encontrar archivos y otros servicios distantes.

Usenet Newsgroups
Grupos de noticias, foros de discusión, comentarios y ayuda a través de mensajes públicos.

Virus
Programa malicioso que se mantiene oculto y se reproduce duplicándose en archivos de programas y disquetes, ocasionando daños, uno de los más comúnes es el borrado de información.

Winsock
Window Socket. Programa que permite establecer conexiones telefónicas con enlaces SLIP o PPP.

WAIS
Siglas de Wide Area Information Servers. El WAIS es un servicio de búsqueda de información en la red por palabra clave o frases. Contrariamente a Gopher, que busca archivos por sus títulos, los motores WAIS buscan en el texto completo de cada archivo y devuelve una lista de todos los documentos que contengan la palabra clave indicada. La mayor parte de los buscadores en el Web utiliza el método WAIS.

World Wide Web. Telaraña a lo ancho del mundo de documentos multimedia con ligas de hipertexto que apuntan a otros documentos HTML también llamado la Web. Colección Inmensa de páginas *Hypertexto en Internet*. Los conceptos World Wide Web fueron desarrollados en Suiza por el European Laboratory of Particle Physics (Conocido como CERN), ro el Web no es sólo una herramienta para cientificos, sino que también es una de las herramientas existentes más flexibles y excitantes para navegar en Internet.

Los enlaces hipertexto conectan piezas de información (texto, gráficas, audio o video) en páginas *HTML* separadas, ubicadas en el mismo lugar o en otro sitio Internet y se pueden explorar estas páginas y enlaces mediante un explorador Web, como *Netscape* o *Internet Explorer*. También se puede acceder al recurso WWW directamente, si se especifica el *URL* apropiado. El tráfico del World Wide Web está creciendo velozmente, más que muchos otros servicios Internet y la razón para ello se hace obvia una vez se prueba un explorador Web poderoso; es bastante fácil acceder a la información World Wide Web.

BIBLIOGRAFIA

[1] Eshed Robotec, 1982. "ACL Reference Guide", 3rd. Edition, Eshed Robotec, Ltd. Israel

[2] Eshed Robotec, 1982. "ACL Reference Guide", 3rd. Edition, Eshed Robotec, Ltd. Israel

[3] Jack Tackett, David Gunter y Lance Brown, 1996. "Linux Edición Especial", Prentice Hall Hispanoamericana, ISBN 968-880-580-7

[4] Chuck Musciano y Bill Kennedy, Mayo 1997, "HTML: The Definitive Guide" , Segunda Edición, ISBN: 1-56592-235-2

[5] Tesis Licenciatura (Ingeniero Mecanico)-UNAM, Facultad de Ingenieria "Software e infraestructura para la integracion de un sistema de manufactura flexible de caracter didactico", Jimenez Fabian, Rene Enrique, 1999.

Referencias tomadas de la Internet.

[W1] Febrero 2002, http://java.sun.com/docs/glossaries/glossary.es.html, Glossary of Java and Related Terms – Spanish

[W2] Febrero 2002, http://www.learnthenet.com/spanish/glossary/cgi.htm, Learn the Net

[W3] Febrero 2002, http://www.tejedoresdelweb.com/307/article-1883.html, Comparación entre lenguajes.

[W4] Febrero 2002, http://es.gnu.org/, ¿Qué es FSF/GNU?

[W5] Febrero 2002, http://www.gridsystems.com/forma/glosario.jsp

[W6] Febrero 2002, http://www.intelitek.com

[W7] Febrero 2002, http://www.beyondlogic.org/serial/serial.htm, Interfacing the Serial / RS232 Port

[W8] Noviembre 2001, http://www.php.net , PHP

[W9] Noviembre 2001, http://www.linux.org, Organización Linux Internacional

[W10] Noviembre 2001, http://www.apache.org, Organización Apache Foundation

[W11] Mayo 2002 http://www.usc.edu/dept/raiders/story/fmi.html Proyecto Mercury, Robot de excavación.

[W12] Mayo 2002 http://rr-vs.informatik.uni-ulm.de/rr/ Interactive Model Railroad, Distributed Systems, University of Ulm

[W13] Abril 2002 http://www.cs.uni-bonn.de/~rhino/tourguide/ RHINO, Universidad de Bonn

[W14] Abril 2002 http://dmtwww.epfl.ch/isr/asl/projects.html Autonomus System Labs

[W15] Marzo 2002 http://www-2.cs.cmu.edu/~minerva/web/iday/index.html Minerva Carnegie Mellon's robotic tourguide proyect

[W16] Marzo 2002 http://telerobot.mech.uwa.edu.au/ Telerobot Australiano

[W17] Mayo 2002 http://ranier.hq.nasa.gov/telerobotics_page/realrobots.html NASA Space Telerobotics Program