UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"DISEÑO DE UN SISTEMA DOMÓTICO PARA EL CONTROL Y SUPERVISIÓN REMOTOS DE UNA CASA, POR MEDIO DE INTERNET."

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO ELÉCTRICO ELECTRÓNICO
P R E S E N T A N:

CANSECO ESPINOSA CHRISTIAN ALEXANDER
QUIRASCO RAMÍREZ FRANCISCO

FACULTAD DE INGENIERIA
U N A M

MÉXICO, D.F. 2008

AGRADECIMIENTOS Y DEDICATORIAS

Quiero agradecer y dedicar este trabajo a mis mamas por todo su apoyo y amor incondicional, por cuidarme desde pequeño, por orientarme, por su paciencia, por estar siempre junto a mí, por ayudarme en momentos difíciles y confiar en mí, por todos sus sacrificios y hacer todo lo posible para que hoy llegara este día. A mi mamá por escuchar y brindar ayuda cuando es necesario, por ser mi maestra de la vida, por enseñarme a nunca darme por vencido y por ganar el cariño, admiración y respeto de toda persona que te conoce. A mi abuelita Angelina, mi segunda mamá, por todas sus enseñanzas, buenos recuerdos y estar junto a mi desde donde quiera que se encontré. Siempre las querré y agradeceré la persona que formaron. A Dios por darme la vida y estas tres mujeres tan hermosas.

A Grace por ser una excelente hermana y amiga, por todo su amor y apoyo, por tu paciencia, por ser una persona de buenos sentimientos y enseñarme que los hermanos menores muchas veces parecen ser los mayores. Gracias por todo, siempre cuenta conmigo para lo que sea. Te quiero mucho.

A mis tías Esther y Dora, por siempre estar conmigo en momentos importantes de la vida, por darme su cariño y apoyo. Tía Esther, por tu apoyo al regularizarme para mis exámenes. Las quiero.

A mi prima Angélica, por los momentos padres que vivimos, por estar siempre cuando te eh necesitado y aunque no nos vemos seguido, por la distancia, cuento contigo y tú conmigo.

A mi primo Ale, por orientarme, por su cariño, paciencia, apoyo y comprensión, por todos esos momentos de la vida que has estado conmigo, siempre serás como un Padre para mí. Gracias y siempre cuenta conmigo. A tu esposa Tere y a mis sobrinitas Ale y Gaby por todos los momentos que hemos compartido.

A mis amigos, que son como mi segunda familia. Fabián, gracias a ti y a tu familia por su amistad y estar siempre que eh necesitado su apoyo. Lorena, gracias a ti y a tu familia por estar comigo en cualquier momento y brindarme su apoyo incondicional, especialmente a tu tía Elena y a tus papas. A Daniel, Tasha, Kiko, Gerardo, Ana, America, Kike, Carmen, Gil, Caro, Beto, Aurelio, Cesar y todas aquellas personas que no nombre y estuvieron presentes en varios momentos padres durante esta trayectoria. Gracias a todos por su apoyo y amistad sincera.

A mi asesor Eduardo Carranza por ser un excelente profesor y persona, por todo su apoyo y orientación para realizar este trabajo.

A nuestros sinodales, por supervisarnos y hacernos las observaciones necesarias para realizar un mejor trabajo.

A nuestra máxima casa de estudios la Universidad Nacional Autónoma de México en especial a la Facultad de Ingeniería y a sus profesores por el excelente nivel académico con el que forma a los futuros profesionistas.

A la Dirección General de Computo Académico (DGSCA) por contribuir ampliamente en mi formación profesional y humana.

A mis jefes Miguel Kelley y Roberto Chalon por su apoyo y contribución para mi desarrollo profesional.

Christian Alexander Canseco Espinosa

Quiero agradecer a mi familia por apoyarme todo el tiempo que duraron mis estudios universitarios, ya que fue un proceso bastante largo y difícil, y gracias a su apoyo se hizo más tolerable y por creer en que iba a terminar esta licenciatura no importando que tan difícil pareciera la situación.

Quiero agradecer a todos los amigos que estuvieron conmigo en todas las diferentes materias y que nos ayudamos mutuamente para poder aprobarlas y en especial a todos aquellos que confiaron en mí para realizar algún proyecto o un trabajo en equipo. También a todos aquellos que no pudieron terminar la carrera por sus propios problemas personales pero que aun así me apoyaron para que no lo dejara con su amistad.

A la Universidad Nacional Autónoma de México por darme una educación desde la preparatoria hasta el final de la licenciatura de bastante calidad mediante unos profesores de muy buenos conocimientos en sus materias.

Y por último quiero agradecer a mi compañero de tesis Christian Canseco y al asesor de la tesis el Ing. Eduardo Carranza por tenerme paciencia ya que no siempre tuve el tiempo necesario para dedicarle a este último trabajo.

Quirasco Ramírez Francisco

ÍNDICE

Capítulo 1:

Introducción.

1. Introducción

Anteriormente, la mayoría de los equipos domésticos que se desarrollaban eran totalmente independientes, es decir, que funcionan de forma autónoma, sin necesidad de comunicarse con otros dispositivos del mismo entorno. Por lo que se tuvo la necesidad de corregir esto, implementando hogares y edificios inteligentes, se empezó a considerar la integración de sistemas de este tipo en el ámbito comercial hasta los 80's. En el sector domestico la integración de estos sistemas, se desarrollo un poco después, coincidiendo con la evolución y despliegue de Internet. Empezó en los 90's en Japón, Estados Unidos y algunos países en el norte de Europa con el desarrollo de los nuevos métodos de acceso a residencias y pasarelas residenciales.

Debido a esta situación se fue originando la tecnología **domótica**[1], que es la automatización de equipos domésticos, que se realizaba mediante un control de su alimentación eléctrica, siendo una manera muy sencilla de gestión, y de poco atractivo tecnológico. Los equipos domésticos no tenían ningún tipo de comunicación eficiente con el sistema domótico. Por ello, la domótica estaba relegada a un mercado muy reducido, comparado con la totalidad del mercado de productos domésticos, y limitándose, por tanto, a dar respuesta a necesidades de control en la vivienda. Por ejemplo, las posibilidades de comunicación con el exterior se reducían a sencillas transmisiones de señales o avisos de alarma o al control remoto de un número reducido de sistemas o equipos.

Con la actual irrupción de Internet y las TIC (Tecnologías de la Información y las Comunicaciones) en el hogar, se ha forjado una nueva forma de entender la aplicación de la tecnología en la vivienda, mucho más positivista y realista, donde lo único importante es el propio usuario. Es decir, la tecnología adquiere un papel de soporte muy importante para cumplir con los servicios que se le brindan al usuario. Con ello, la tecnología es algo transparente para el usuario, el cual no tiene un interés técnico sino simplemente de utilidad. El usuario no está interesado en la tecnología sino en resolver su problema, necesidad o deseo. Por ello, se avanzo en el concepto de tecnología al servicio del usuario, y que permita aportar soluciones fáciles, útiles y económicas, con las finalidades claras de asegurar el bienestar y la seguridad.

Evidentemente, el desarrollo de este nuevo mercado requiere asegurar la capacidad de comunicación entre todos los equipos domésticos de la vivienda o lugar de trabajo. En el mercado internacional existen numerosas maneras de

[1]http://www.casadomo.com/noticiasDetalle.aspx?c=9&m=15&idm=15&pat=14&n2=14

denominar a esta nueva forma de concebir la comunicación en la vivienda (Digital Homes, Connected Homes, eHomes, Smart Homes y iHomes).
Las viviendas, hoy en día, todavía disponen de un gran número de equipos y sistemas, autónomos, y redes no conectadas entre ellas como la telefonía, los sistemas de acceso, la televisión, las redes de datos (cableados e inalámbricos), electrodomésticos, equipamiento de audio y video, calefacción, aire-condicionado, seguridad, riego e iluminación. Por lo que se ha buscado lograr la integración de estos sistemas en el Hogar, como se muestra en la figura 1.

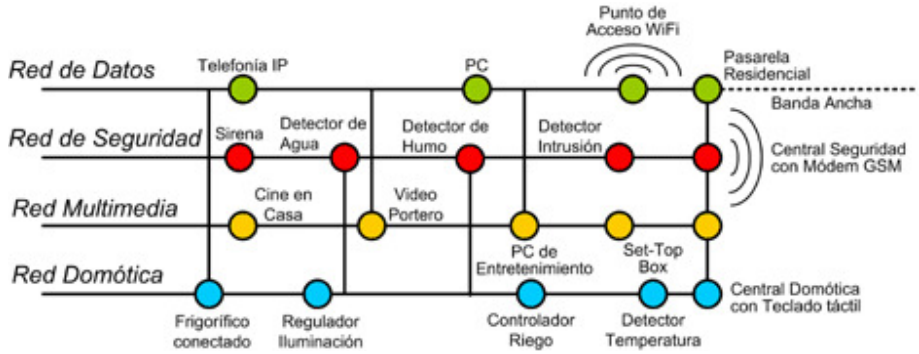


Figura 1. Modelo conceptual de la integración de sistemas del Hogar Digital

El Hogar Digital [2] es una vivienda que a través de equipos y sistemas, y la integración tecnológica entre ellos, ofrece a sus habitantes funciones y servicios que facilitan la gestión y el mantenimiento del hogar, aumentan la seguridad; incrementan el confort; mejoran las telecomunicaciones; ahorran energía, costos y tiempo, y ofrecen nuevas formas de entretenimiento, ocio y otros servicios dentro de la misma y su entorno.

Los productos y sistemas relacionados con el Hogar Digital pueden ser agrupados en las siguientes áreas:

- ✓ La Domótica es la automatización, el control local y remoto del hogar (apagar / encender, abrir / cerrar y regular) de aplicaciones y dispositivos domésticos, con instalaciones, sistemas y funciones para iluminación, climatización, persianas y toldos, puertas y ventanas, cerraduras, riego, electrodomésticos, control de suministro de agua, gas, y electricidad.

- ✓ La Multimedia son los contenidos de información y entretenimiento, relacionados con la captura, tratamiento y distribución de imágenes y sonido dentro y fuera de la vivienda, con instalaciones, sistemas y funciones como radio, televisión, audio / vídeo "multi-room", cine en casa, pantallas planas, videojuegos, porteros y video porteros.

[2] http://www.casadomo.com/noticiasDetalle.aspx?c=9&m=15&idm=15&pat=14&n2=14

- ✓ La Seguridad y Alertas son sistemas y funciones para alarmas de intrusión, cámaras de vigilancia, alarmas personales y alarmas técnicas (incendio, humo, agua, gas, fallo de suministro eléctrico y fallo de línea telefónica).

- ✓ Las Telecomunicaciones (red de datos) es la distribución de archivos de textos, imágenes y sonidos, compartiendo recursos entre dispositivos, el acceso a Internet y a nuevos servicios, con instalaciones, sistemas y funciones como red de telefonía, telefonía sobre IP, LAN, pasarelas residenciales, routers y acceso a Internet Banda Ancha.

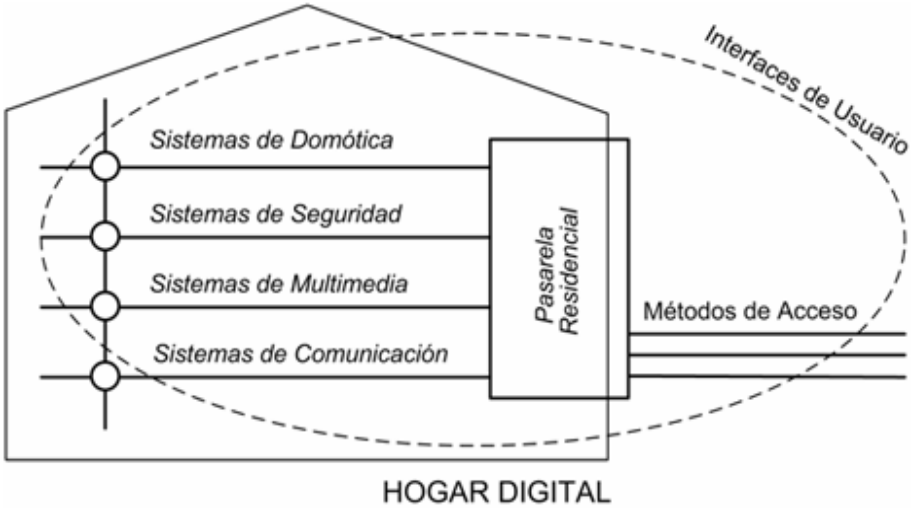


Figura 2. Esquema conceptual del Hogar Digital

Teniendo esta visión general del tema, como se ve en la figura 2, e investigando principalmente en Internet y en algunos libros, se puede notar que en México hay muy pocos proveedores de estos servicios, además los servicios que brindan estos proveedores tienen precios muy elevados, por ello, estos servicios que brindan algunas empresas, no están diseñados para un acceso a la mayoría de la población, sino para sectores cuyo nivel económico es alto o simplemente para empresas cuya solvencia económica sea suficiente para estos grandes gastos.

Sin embargo, al igual que la mayoría de las tecnologías crecen a pasos agigantados, se van desarrollando varios estándares que regulan estas tecnologías y por ende a asociaciones que controlan estos estándares como son:

Estándares[3]:

- ✓ KNX/EIB:
- ✓ X10

[3] http://es.wikipedia.org/wiki/Dom%C3%B3tica.
El Hogar Digital. Fernandez Valentin. Creaciones Copyright.

- ✓ ZigBee:
- ✓ OSGi:
- ✓ Lonworks: Estándar de carácter propietario.
- ✓ Universal Plug and Play (*UPnP*):
- ✓ CEBus, Batibus y EHS.

Asociaciones[4]:

- ✓ CENELEC:
- ✓ CENELEC/ENTR/e-Europe/2001-03
- ✓ ASIMELEC:
- ✓ AENOR

Por lo expuesto, el propósito de esta tesis es diseñar un sistema domótico que aproveche las capacidades de una PC común para permitir el control y supervisión remotos de dispositivos caseros, el desarrollo del software se pretende que sea con la característica de multiplataforma.

Ya se han desarrollado trabajos previos, pero tienen interfaces que no son amigables para el usuario, además de estar hechos en lenguajes que no son multiplataforma[5]. Por lo que en este proyecto se pretende corregir estos dos inconvenientes, además de optimizar lo mejor que se pueda el proyecto en cuanto al hardware.

Empleando como elemento principal una computadora personal con características mínimas y con un acceso a Internet que puede ir desde una conexión por modem hasta enlaces dedicados de varios Mbps[6] (Banda Ancha).

Actualmente existen varios productos que ofrecen servicios como apagar / encender, abrir / cerrar y regular los sistemas domésticos como la iluminación, climatización, persianas, puertas y ventanas, cerraduras, riego, electrodomésticos, suministro de agua, suministro de gas y suministro de electricidad. Pero son muy costosos y difíciles de integrar.

En nuestro caso vamos a adoptar las características más comunes de la domótica. El desarrollo de la interface entre la computadora y los aparatos así como los programas cliente y servidor serán probados en diferentes condiciones, de esta manera podremos evaluar su comportamiento y corregir, si se presentan, errores u omisiones ocultos en el proceso original de diseño.

[4] http://es.wikipedia.org/wiki/Dom%C3%B3tica
[5] http://es.wikipedia.org/wiki/Multiplataforma
[6] Megabit por segundo

OBJETIVO

Diseñar un sistema domótico que cumpla con las tareas más comunes en el área de la domótica, que aproveche las capacidades de una PC común para permitir el control y supervisión remotos de dispositivos caseros que actualmente son autónomos, es decir, no se comunican con otros dispositivos para una supervisión y control de ellos, por medio de una red o Internet, con un costo mínimo; para cumplir con este requisito se plantea agregar el menor hardware posible y utilizar al máximo el hardware de la PC, el desarrollo del software se hará sobre una un lenguaje multiplataforma, libre de regalías y licencias.

Objetivos específicos:

- ✓ Revisar los productos similares en el mercado.
- ✓ Revisar y entender los fundamentos de comunicación de la PC.
- ✓ Diseñar el software y el hardware.
- ✓ Construcción del prototipo.
- ✓ Probar el prototipo.
- ✓ Evaluar los resultados obtenidos.

Capítulo 2:

Estudio de productos similares en el mercado.

2. Estudio de productos similares en el mercado

2.1 Introducción

La mayoría de los entornos de nuestra vida cotidiana ya cuentan con un grado de automatización: los comercios, transportes y hoteles. Estas tareas de control, instrumentación o simplemente medición se auxilian de comunicaciones digitales para transferir los datos adquiridos, a una sede en la que se tomaran decisiones o que utilizara esta información para continuar con algún otro proceso.

En la actualidad la mayoría de los sistemas se conectan a Internet, ya que las empresas tratan de dar un mejor servicio a sus usuario; creando aplicaciones que sean más fácilmente accedidas desde cualquier lugar para una mayor satisfacción del cliente. Por ejemplo la telefonía celular que inicio por la década de los 70s y hoy en día ya no se limita a la función de comunicar a dos personas entre sí, sino que ahora ha evolucionado hasta incluir modalidades como el acceso a la Internet y con esto transmisión de datos, mp3, teleconferencia, transmisión de archivos fotográficos y videos, permite la transmisión de datos desde una PC hacia un teléfono móvil.

Pensando en este mismo esquema otras compañías han incorporado acceso a Internet a todo tiempo de dispositivos, desde electrodomésticos, hasta controladores industriales, pasando por dispositivos de información móviles, Pocket PC's y agendas electrónicas.

Este es el contexto que ya se empieza a vislumbrar una división del mercado en varias corrientes que básicamente emplean la misma tecnología pero que atacan diferentes aspectos y proponen diferentes servicios. Esta por ejemplo el campo dedicado al **Control de iluminación** (regulación, escenas preajustadas, simulación de presencia, iluminación exterior), **control y monitoreo de consumo de energía eléctrica, gas y agua** en casas, compañías y otras instalaciones, **Control de climatización** (calefacción y aire acondicionado), **Control de motorizaciones** (persianas, cortinas, toldos, puertas), **Control equipos A/V** (distribución de audio y video), **Seguridad** (integración de cámaras de TV y visión por Internet), **Comunicaciones** (sedes locales, centrales de teléfono), **Integración de sistemas** (control centralizado de todo el sistema).

También es muy marcado la influencia de las nuevas tecnologías de la información en el campo industrial, hoy en día no es raro encontrar productos y servicios que ofrecen monitorear y controlar un sin número de variables, actuadores y dispositivos industriales, para que un proceso se lleve a cabo de manera eficiente y

con costos más reducidos, y no solo eso, sino ahora con la comodidad de hacerlo desde una computadora personal, ya sea en el sitio o desde cualquier parte del mundo a través de Internet, cable telefónico o el de televisión , enlaces satelitales, aunque este último con costos significativamente mayores por ahora.

La mayoría de las tecnologías empleadas en estos productos y servicios son propietarias y muy cerradas en cuanto al acceso a su diseño electrónico, su software y sus protocolos de comunicación. Esto los hace caros, poco extensibles, obsoletos y con altos costos de mantenimiento a menos que se tenga un contrato con la compañía suministradora, en cuyo caso, el problema sería la complejidad que presenta migrar de dispositivos y de software todo el sistema si es que decidimos en algún momento cambiar de compañía.

A continuación presentaremos ejemplos representativos de algunos productos y servicios que ofrecen hoy en día en el mercado con algunas de las características que anteriormente hemos mencionado, también sus pros, sus contras y en algunos casos sus costos actuales. De esta manera sentaremos una base que nos sirva de antecedente para proponernos un sistema que de alguna manera cubra las deficiencias u omisiones de los sistemas actuales, complementar en su caso a estos mismos o tomar las mejores ideas de lo ya hecho para concentrar nuestra atención en nuevos problemas.

2.2 LabView

Acrónimo de Laboratory Virtual Instrumentation Engineering Workbench, es un ambiente gráfico y una plataforma de desarrollo para un lenguaje de programación visual de National Instruments. El lenguaje gráfico es llamado “G”. Originalmente desarrollado para la Apple Macintoh en 1986.

LabView es comúnmente usada para la adquisición de datos, control de instrumentos y automatización industrial en una variedad de plataformas incluyendo Microsoft Windows, UNIX, Linux y Mac OS. La última versión de LabView es la versión 8.20 liberada en el 2006.

El lenguaje de programación usado en LabView, llamada “G”, es un lenguaje de flujo de datos. La ejecución está determinada por la estructura de un diagrama de bloques gráfico en el cual el programador conecta diferentes nodos de funciones dibujando “cables”. Estos “cables” propagan las variables y cualquier nodo puede ejecutarse tan pronto todos sus datos de entrada se completan.
G es capaz de la ejecución paralela para procesar múltiples nodos a la vez, el multiprocesamiento del hardware es automáticamente explotado.

El flujo de datos define la secuencia de ejecución, y esta secuencia es completamente controlable por el programador, por lo que la secuencia de ejecución de la sintaxis gráfica del LabView esta tan bien definida como cualquier tipo de código de lenguaje de texto como C y Visual Basic. Una ventaja de LabView es que no requiere ningún tipo de definición de variables, todo es definido por los datos que son dados en los nodos y además soporta polimorfismo por lo que automáticamente se ajusta a diferentes tipos de datos.

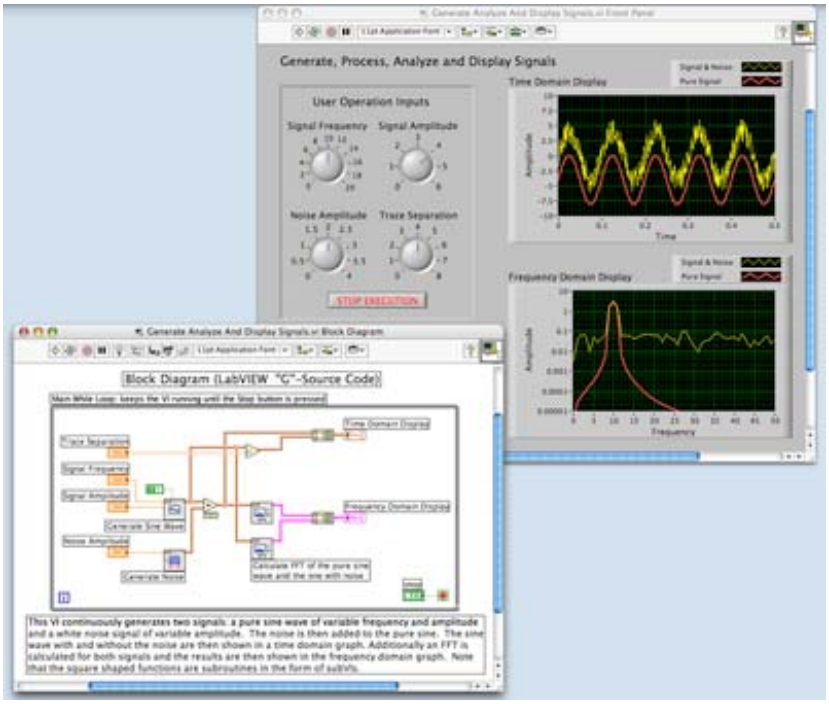


Figura 3

La figura 3 nos muestra un programa simple en LabView que genera, sintetiza, analiza y muestra formas de ondas, mostrando el diagrama de bloque en la parte frontal. Cada símbolo en el diagrama de bloques representa una subrutina que puede ser otro programa de LabView o una función de las librerías.

LabView usa la creación de interfaces de usuarios (llamadas paneles frontales) en el ciclo de desarrollo. Los programas y subrutinas son llamados instrumentos virtuales (VIs). Cada VI tiene tres componentes; un diagrama de bloques, un panel frontal y conectores. Los conectores pueden representar al VI como un subVI en un diagrama de bloques de varios VIs. Los controles y los indicadores en el panel frontal permiten al operador meter datos o extraer datos de un instrumento virtual que se este ejecutando. Sin embargo el panel frontal también puede servir como una interface para programar, por lo que un instrumento virtual puede ser ejecutado como un programa, con el panel frontal sirviendo como la interfaz del usuario, o cuando hay un nodo en el diagrama de bloques, el panel frontal define

las entradas y salidas para el nodo en el conector, esto ayuda a poder probar cada VI por separado para ser usados como subrutinas en programas mas grandes.

Este enfoque gráfico también permite a los no programadores a construir programas simplemente arrastrando y soltando representaciones virtuales del equipo de laboratorio con el cual ya están familiarizados. El ambiente de programación, con los ejemplos incluidos y la documentación incluida, permite hacer aplicaciones simples y pequeñas, esto es un beneficio pero aun así es mejor hacer los programas con la programación en "G" para varias aplicaciones.

Para algoritmos complejos o código a gran escala es importante que el programador posea conocimientos extensivos de las sintaxis de LabView y su forma de manejar la memoria. Es posible crear aplicaciones que se comuniquen como cliente/servidor.

Un **beneficio** de LabView sobre otros ambientes de desarrollo es su soporte extensivo para accesar a hardware de instrumentación. Cuenta con drivers para muchos tipos diferentes de instrumentos y los buses están incluidos o se pueden incluir, estos se presentan como nodos gráficos. También ofrece interfaces de software estándar para comunicarse con diferentes hardwares.

El objetivo de National Instruments es que hasta las personas que tengan experiencia con códigos de programación limitada pueda escribir programas y encontrar soluciones de prueba en un menor tiempo comparado a otros sistemas, además de proveer compatibilidad entre diferentes plataformas con la misma base de datos de los drivers llamada DAQmxBase.

En términos de desempeño, LabView incluye un compilador que produce código nativo para la plataforma del CPU. El código gráfico es traducido a código máquina interpretando las sintaxis y compilando. La sintaxis de LabView es estrictamente seguida durante el proceso de edición y compilado cuando se ejecuta el programa, generando un código fuente y un programa ejecutable en un solo archivo.

Hay varias opciones de LabView cada una con diferentes librerías para la adquisición de datos, generación de señales, matemáticas, estadísticas y análisis de señales, así como elementos gráficos de interfaz.

El sistema de desarrollo profesional de LabView permite crear ejecutables y estos pueden ser distribuidos de forma indefinida, el motor gráfico y las librerías pueden ser distribuidas gratuitamente con el ejecutable.

Existe una versión de bajo costo para estudiantes dirigida a instituciones educativas.

Una de las **desventajas** de LabView es que es un producto propietario de National Instruments, que a diferencia de otros lenguajes de programación como C o Fortran, no se puede distribuir de forma gratuita, tiene que ser comprado directamente a National, todavía no hay una fuente abierta (open source) o software libre que pueda implementar el código G.

Para crear una aplicación que se ejecute sola requiere del componente Application Builder el cual solo esta incluido en la versión Profesional, si se cuenta con otra versión hay que comprarla por separado

Hay un debate si es que LabView realmente es un lenguaje de programación general, comparándolo con otro tipo de aplicaciones específicas para la medición y automatización. Se le critica por su falta de aplicaciones, comunes en otros lenguajes como lo es la programación orientada a objetos, o el no estar bien preparada para las aplicaciones complejas que no requieran de medición o automatización.

2.3 Hometronic

En nuestras casas existen diariamente una serie de actividades que son rutinarias y bien podrían ser delegadas, como por ejemplo el manejo de la iluminación, la temperatura de la casa o del algún sitio de esta, no solo en los momentos en los que estamos en ella, sino cuando salimos a trabajar o estamos ausentes mucho tiempo.

Con Hometronic, Honeywell ofrece un sistema que realiza estas y otras tareas de manera automática, por un lado coordina la automatización de las tareas para lograr los más altos índices de confort, y por otro provee modos de economía de energía que pueden repercutir en ahorros muy significativos en el costo de los servicios.

2.3.1 Comunicaciones inalámbricas

Una de las ventajas más importantes de Hometronic es que la instalación del sistema no requiere necesariamente ser planeada junto con la residencia, para ser incluidos los conductos y requerimientos eléctricos, sino que el sistema puede ser utilizado en prácticamente cualquier casa sin la necesidad de instalaciones adicionales y costosas, solo se necesita pensar donde poner la unidad de manejo principal y en donde conectar esta a la alimentación eléctrica. El resto de los dispositivos que se van instalando en toda la casa se comunican vía radio a la

unidad central, con esto evitamos el uso e instalación de cables, y también reducimos los costos de la instalación notablemente, aunque esto se compensa con el precio de los dispositivos.

La potencia de transmisión de la señales de radio esta en el rango de 1mW, esto hace que la emisión de radio sea despreciable, mientras que la calidad y confiabilidad de la transmisión en un edificio promedio está garantizada. La posición y tamaño de las paredes normales no es problema para el aparato.

2.3.2 Capacidades y beneficios

El sistema Hometronic cuenta con 48 salidas a dispositivos que controlan la iluminación, la temperatura, las persianas y cortinas. También 16 entradas de sensores de consumo, que permiten monitorear y controlar todo tipo de casas habitación, desde las más pequeñas, hasta las más grandes.

Además Hometronic ofrece al usuario los siguientes beneficios, todas las funciones son consideradas independientes cuarto a cuarto y pueden ser monitoreadas desde un solo punto, ya sea desde la unidad de manejo principal o por radio a través de un control remoto.

El estado de la temperatura de los cuartos, los estados de los interruptores, las condiciones de iluminación, y la posición de las ventanas y cortinas, pueden ser fácilmente programadas en la unidad de manejo con un simple conjunto de instrucciones y un gran número de rutinas preprogramadas.

Además se cuenta con una gran cantidad de programas que definen diferentes estilos de vida estándar, y que permiten a Hometronic establecer diferentes condiciones en los cuartos, por ejemplo una temperatura media en la sala, luz y las persianas recogidas en el cuarto de los niños y en las recámaras.

Durante las vacaciones, las funciones de ahorro de energía son activadas pero no solo es eso, también se establece un modo de seguridad en el que se simula que la casa sigue estando habitada. Al cambiar al modo de vacaciones, el sistema baja la temperatura de toda la casa para ahorrar energía y para prevenir la observación de personas desde fuera de la casa, se prenden y apagan las luces y se corren y cierran las cortinas y persianas, como lo harían los habitantes todos los días, de esa manera se presentan un patrón de actividad dentro de la casa que aparenta estar ocupada.
Durante las horas de luz, los sensores de esta, pueden establecer que las persianas y cortinas son abiertas o cerradas, no solo para el control de la temperatura, sino para prevenir que la luz solar lastime muebles u otros objetos por una exposición prolongada.

2.3.3 Equipo y especificaciones.

Aquí mostraremos un resumen de los componentes y módulos de Hometronic[7]. Todos los módulos funcionan con tecnología de radiofrecuencia. En los módulos que funcionan con pilas, la duración media de éstas es de 2 años.

Hometronic Manager HCM 200

Figura 4

Es el panel central de control. Aquí puede establecer los programas automáticos, tales como caldear la habitación por la mañana, o regular todo automáticamente con solo pulsar un botón. Dispone de hasta 32 aplicaciones para iluminación, persianas, electrodomésticos, etc. El Manager puede registrar hasta 16 lecturas de contadores, y regular individualmente la temperatura hasta en 16 zonas. Se pueden establecer para cada habitación por separado, acciones diarias, semanales, durante las vacaciones, modos de vida personales y hasta la regulación de ciertos procesos. Un botón ECO facilita el ahorro de energía, disminuyendo la temperatura en toda la vivienda en 2 o 3 grados, por ejemplo. Los lugares más adecuados para situar el Manager son la entrada de la vivienda o el salón.
Dimensiones: 170 x 127 x 40 mm (Alto x Ancho x Fondo)

Mando a distancia HRD 20

Figura 5

Con el mando a distancia, puede controlar Hometronic desde cualquier lugar de su hogar, activando los Estilos de Vida automáticos o haciendo funcionar los módulos por separado.
Dimensiones: 115 x 55 x 25 mm (Alto x Ancho x Fondo)

[7] http://www.hometronic.es/technology/systemuebersicht/default.htm

Módulo Telefónico HCI 200

Figura 6

Con el interfaz telefónico puede controlar fácilmente su hogar o residencia de vacaciones, desde cualquier lugar. El interfaz puede conectarse a cualquier salida de teléfono de tipo analógico, no se requiere ningún tipo de señal de entrada específica. Puede llamar al Hometronic Manager y comprobar o cambiar los Estilos de Vida programados, con su Número de Identificación Personal (PIN) desde cualquier lugar. La señal puede ser digital o analógica, el usuario decidirá por él mismo. Así es como Hometronic se adapta de manera flexible a sus necesidades individuales.

Dimensiones: 163 x 207 x 53 mm (Alto x Ancho x Fondo)

Termostato de Radiador HR 50

Figura 7

Permite controlar la temperatura ambiente modulando el paso de agua por el radiador. La función de "Ventana abierta" evita el desperdicio de energía cuando se ventila la casa. Posible cambio manual de temperatura mediante la rueda selectora. Se acopla directamente a las válvulas de radiador Honeywell, Braukmann, MNG, Helmeier y Danfoss (otras marcas bajo petición). Dos pilas estándar permiten el funcionamiento sin cable. Los puntos de consigna de temperatura de confort y economía y las horas de cambio de temperatura se transmiten al Termostato de Radiador desde el Hometronic Manager.

Dimensiones: 82 x 52 x 100 mm (Alto x Ancho x Fondo)

Módulo de Conmutación HS 30

Figura 8

Controla electrodomésticos y aplicaciones del jardín hasta un máximo de 10 A/230 V c.a. El circuito eléctrico se instala en la caja de distribución, que sirve al mismo tiempo de suministro de corriente de 230 V. El Manager controla la situación deseada (encendido/apagado) por radiofrecuencia, en base a la programación horaria establecida. El control local o manual es posible en cualquier momento.

Dimensiones: 90 x 90 x 28 mm (Alto x Ancho x Fondo)

Módulo de Conexión/Conmutación HS 20

Figura 9

Conecta o desconecta los electrodomésticos que funcionan con un enchufe hembra con toma de tierra, hasta 16 A/230 V.

Dimensiones: 150 x 65 x 40 mm (Alto x Ancho x Fondo)

Grupo de Control de Persianas HA 70

Figura 10

Para elevar y bajar persianas con cinta. El módulo se instala en el espacio existente para el rollo de la cinta. El suministro de energía es a través de la red eléctrica, con enchufe estándar. Tamaño máximo de la persiana: 6 m^2 para persianas de plástico de 4,5 kg por m^2; 3 m^2 para persianas de aluminio/madera de 10 kg por m^2.

Dimensiones: 290 x 60 x 165 mm (Alto x Ancho x Fondo)

Antena de Recepción HRA 1

Figura 11

Antena receptora para Regulador de Suelo Radiante HCE 60, para recepción sin cables de señales.

Dimensiones: 103 x 99 x 30 mm (Alto x Ancho x Fondo)

Regulador de Suelo Radiante HCE 60

Figura 12

Centro de control y suministro de energía para hasta 8 circuitos de control de habitación. Se comunica sin cables con el Manager, y lleva a cabo sus órdenes y programas automáticamente.

Dimensiones: 82 x 315 x 75 mm (Alto x Ancho x Fondo)

Módulo de Seguridad HX 10

Figura 13

El módulo integra en el Sistema Hometronic interruptores, sensores con contactos de relé y salidas de sistema de seguridad. Puede activarse un Estilo de Vida específico. De esta forma, Hometronic proporciona un concepto integral en el área de la seguridad: simulación de presencia para prevención de robos, estado de la alarma en caso de robo y respuesta activa para protección contra siniestros en su hogar. El HX 10 necesita un suministro de energía de 12 Vca/Vcc +/- 20%. Esto puede conseguirse mediante el módulo HN 10 (no incluido en el embalaje), o también gracias a una fuente interna de energía de un sistema de seguridad.

Dimensiones: 90 x 90 x 28 mm (Alto x Ancho x Fondo)

Sensor de Ambiente HCF 22

Figura 14

Sensor de temperatura de habitación sin cables sin rueda de selección de temperatura.

Dimensiones: 103 x 99 x 30 mm (Alto x Ancho x Fondo)

Sensor de Habitación/ Módulo de Control Remoto HCW 22

Figura 15

Módulo de habitación sin cables, controla progresivamente la temperatura de la habitación. Una rueda de ajuste le permite seleccionar la temperatura deseada.

Dimensiones: 103 x 99 x 30 mm (Alto x Ancho x Fondo)

Módulo de Persianas y Toldos HA 30

Activa los motores de persianas y toldos. El Hometronic Manager ajusta progresivamente la posición de las persianas/venecianas o el ángulo de las lamas. Las posiciones de abierto/cerrado y cualquier otra posición intermedia, que usted desee, puede controlarse por programación horaria o por sensores de luz. Para montar en caja empotrada. Sobre la caja de distribución del circuito de la persiana, que sirve también como suministradora de corriente (230 V). El módulo de persianas sirve también para toldos y aplicaciones similares. Puede utilizarse también un contacto de puerta, para prevenir el cierre no deseado de la misma.

Dimensiones: 90 x 90 x 28 mm (Alto x Ancho x Fondo)

Sensor de Luz HB 05	Detecta la luz exterior para el control de persianas, venecianas y toldos, en base al amanecer y anochecer, para el control de los mismos dependiendo de la luz. Dimensiones: 72 x 76 x 50 mm (Alto x Ancho x Fondo)
Sensor de Viento/Luz HB 15	Detecta la luz exterior y la velocidad del viento para el control de persianas, venecianas y toldos, en base al amanecer y anochecer, para el control de los mismos dependiendo de la luz. También proporciona protección contra viento fuerte. Dimensiones: 72 x 76 x 50 mm (Alto x Ancho x Fondo)
Sensor de Viento HWS 40	Versión anemométrica con unidad de medida para la velocidad del aire. Sirve como protección contra daños, recogiendo los toldos en caso de viento fuerte. Dimensiones: 72 x 76 x 50 mm (Alto x Ancho x Fondo)
Módulo Atenuador de Luces HD 30	Válido para cualquier tipo de bombilla, lámparas halógenas de alto y bajo voltaje con cargas resistivas o inductivas, mínimo 60 W, máximo 300 W. El circuito eléctrico se instala en la caja de distribución, que sirve al mismo tiempo de suministro de corriente de 230 V. El Manager controla la atenuación deseada de las luces y el encendido y apagado de las mismas, de manera centralizada por radiofrecuencia. El control manual es posible en cualquier momento. Dimensiones: 90 x 90 x 28 mm (Alto x Ancho x Fondo)

Válvula de Corte	Válvula de bola con motor eléctrico que bloquee automáticamente la conducción principal de agua potable durante los períodos de ausencia, con el fin de evitar daños ocasionados por la rotura de tuberías. Dimensiones: 153 x 125 x 78 mm (Alto x Ancho x Fondo)
Medidor de Consumo	Recoge información de consumo de los radiadores, sin cable (coste, distribución), distribuidores de calor (medida de la cantidad de calor), y agua fría y caliente (medidor de caudal). La compañía Kundo ofrece productos para estos usos que pueden integrarse fácilmente en su sistema Hometronic. Sus técnicos especializados le aconsejarán con sumo gusto. Dimensiones: 90 x 90 x 28 mm (Alto x Ancho x Fondo)

2.4 Domótica

El término **domótica** proviene de la unión de las palabras *domus* (que significa *casa* en latín) y *robótica* (*robota*, *esclavo* en checo). Se entiende por domótica al conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas. Se podría definir como la *integración de la tecnología en el diseño inteligente de un recinto*.

Los sistemas domóticos consisten de uno o varios elementos. Se puede hacer la siguiente clasificación de los dispositivos de un sistema:

- ✓ Controlador: en instalaciones centralizadas, es la central que gestiona el sistema. En este reside toda la inteligencia del sistema y suele tener los

interfaces de usuario necesarios para presentar la información a este (pantalla, teclado y monitor).

- ✓ Actuador: es el dispositivo de salida capaz de recibir una orden del controlador y realizar una acción (encendido/apagado, subida/bajada de persiana y apertura/cierre de electroválvula).

- ✓ Sensor: es el dispositivo que está, de forma permanente monitoreando el entorno con objeto de generar un evento que será procesado por el controlador. Ejemplos de activación de un interruptor, los sensores son de temperatura, viento, humedad, humo y escape de agua o gas.

2.4.1 Ventajas de la domótica:

- ✓ Incremento en el confort.

- ✓ Automatización del control de luces, persianas, ventanas, cortinas y enchufes . A modo de ejemplo, un sistema domótico puede bajar las persianas cuando llueve, para que no se mojen los cristales o cuando hace mucho sol. O subirlas cuando es de noche.

- ✓ Climatización automática. Calefacción y refrigeración.

- ✓ Optimización en la gestión de consumos: energía eléctrica, gas, recursos hídricos.

- ✓ Uso de energías renovables: Energía solar, Energía geotérmica, Energía eólica.

- ✓ Automatización de tareas: riego, encendido de los servicios a ciertas horas y en función de eventos.

- ✓ Ubicuidad en el control tanto externo como interno, control remoto desde Internet, PC, mandos inalámbricos (p.ej. PDA con WiFi).

- ✓ Facilidad de uso (GUI: Interfaz de usuario gráfico, aplicación).

- ✓ Alarmas: Vigilancia anti-incendios, Temperatura, Detección de fugas de gas o agua.

- ✓ Control de accesos.

- ✓ Control de servicios para emular la presencia de gente durante las ausencias prolongadas.

- ✓ Gestión alarmas técnicas: corte de suministros, posibilidad de visualización remota de la vivienda.

Muchas personas desconocen el significado de la palabra “domótica” mientras que a otras si les resulta familiar aunque les sugiere una complejidad que no corresponde con la realidad.

La domótica tiene como objetivo obtener un mayor rendimiento de las diferentes instalaciones y equipos de nuestra vivienda (luces, persianas, calefacción, aire acond., toldos y electrodomésticos) haciendo que se comporten como un único sistema, con un control común para manejarlos, tanto individual como conjuntamente en escenas programadas (“ir a dormir”, “salir de casa” y “vacaciones”); tanto local como remotamente desde fuera de casa (por teléfono e Internet) y además, con la posibilidad de que funcionen automáticamente en base a programaciones horarias, cambios de estado en la vivienda (alarmas técnicas, intensidad de luz y viento.) o por nuestra orden directa (“simulación de presencia”).

Desde el punto de vista de donde reside la inteligencia del sistema domótico, hay dos arquitecturas diferentes:

- ✓ Arquitectura Centralizada: un controlador centralizado recibe información de múltiples sensores y, una vez procesada, genera las órdenes oportunas para los actuadores.

- ✓ Arquitectura Distribuida: en este caso, no existe la figura del controlador centralizado, sino que toda la inteligencia del sistema está distribuida por todos los módulos sean sensores o actuadores. Suele ser típico de los sistemas de cableado en bus.

Hay que destacar que algunos sistemas usan un enfoque mixto, esto es, son sistemas con arquitectura descentralizada en cuanto a que disponen de varios pequeños dispositivos capaces de adquirir y procesar la información de múltiples sensores y transmitirlos al resto de dispositivos distribuidos por la vivienda. Hoy en día hay buenos sistemas centralizados y distribuidos, todos ellos con elevadas prestaciones. Ambas arquitecturas tienen sus ventajas y sus inconvenientes, lo cual a priori no ayuda a decidir.

Se pueden clasificar las centrales según su nivel tecnológico en dos tipos:

- ✓ Centrales cableadas: todos los sensores y actuadores (sirenas), están cableados a la central, la cual es el controlador principal de todo el sistema. Esta tiene normalmente una batería de respaldo, para en caso de fallo del suministro eléctrico, poder alimentar a todos sus sensores y actuadores y así seguir funcionando normalmente durante unas horas.

- ✓ Centrales inalámbricas: en este caso usan sensores inalámbricos alimentados por pilas o baterías y transmiten vía radio la información de los eventos a la central, la cual está alimentada por la red eléctrica y tiene sus baterías de respaldo.

- ✓ Centrales mixtos: combinan el cableado con el inalámbrico.

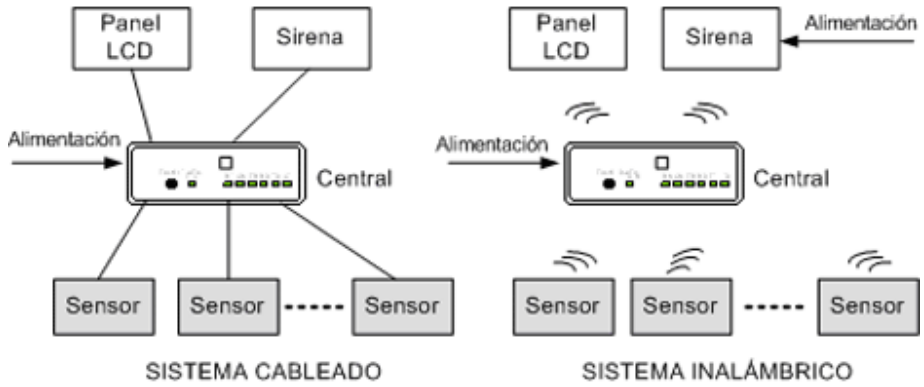


Figura 16

Las figura 16 ejemplifican los tipos de centrales, dependiendo del nivel tecnológico.

2.5 Protocolos de comunicación empleados en la domótica.

2.5.1 CEBUS

En 1984 varios miembros de la EIA norteamericana (Electronics Industry Association) llegaron a la conclusión de la necesidad de un bus domótico que aportara más funciones que las que aportaban sistemas de aquella época (ON, OFF, DIGMER y ALL OFF). Especificaron y desarrollaron un estándar llamado CEBus (Consumer Electronic Bus).

En 1992 fue presentada la primera especificación. Se trata de un protocolo, para entornos distribuidos de control, que está definido en un conjunto de documentos (en total unas 1000 páginas). Como es una especificación abierta cualquier

empresa puede conseguir estos documentos y fabricar productos que implementen este estándar.

En Europa una iniciativa similar en prestaciones, y en el mercado al que va dirigido, es el protocolo EHS (European Home System).
Se contemplan diversos protocolos para que los electrodomésticos y equipos eléctricos puedan comunicarse usando ondas portadoras por las líneas de baja tensión, par trenzado, cable coaxial, infrarrojo, radiofrecuencia y fibra óptica.

Para la transmisión de datos por portadoras, el CEBus usa una modulación en espectro expandido; estos se transmite uno o varios bits dentro de una ráfaga de señal que comienza en 100 kHz y termina en 400 kHz (barrido) de duración 100 microsegundos. La velocidad media de transmisión es de 7500 bps.

Las tramas definidas en CEBus pueden tener longitud variable en función de la cantidad de datos que se necesitan transmitir. El tamaño mínimo es 8 octetos y el máximo casi 100 octetos.

Al igual que los dispositivos EIB, los nodos CEBus tienen grabado una dirección física prefijada en fábrica, que los identifican de forma unívoca en una instalación domótica. Hay más de 4,000 millones de posibilidades.

Como parte de la especificación CEBus se ha definido un lenguaje común para el diseño y especificación de la funcionalidad de un nodo, a este lenguaje lo han llamado CAL (Common Application Language) y está orientado a objetos (estándar EIA-600).

La empresa Intellon Corporation dispone del hardware y el protocolo embarcados en un único circuito. Además proporcionan el entorno de desarrollo en lenguaje CAL compatible con sus propios circuitos así como Kits de inicio para aquellas empresas que deseen empezar a desarrollar productos CEBus.

La CIC (CEBus Industry Council) es una asociación de diferentes fabricantes de software y hardware que certifican que los nuevos productos CEBus que se lancen al mercado cumplan toda la especificación. Una vez que el producto pase todos los ensayos, el fabricante paga una tasa y es autorizado a poner el logo CEBus en ese producto.

2.5.2 EHS

El estándar EHS (European Home System) ha sido otro de los intentos que la industria europea (año 1984), auspiciada por la Comisión Europea, de crear una tecnología que permitiera la implantación de la domótica en el mercado residencial

de forma masiva. El resultado fue la especificación del EHS en el año 1992. Está basada en una topología de niveles OSI (Open Standard Interconnection), y se especifican los niveles: físico, de enlace de datos, de red y de aplicación.

Desde su inicio han estado involucrados los fabricantes europeos más importantes de electrodomésticos, las empresas eléctricas, las operadoras de telecomunicaciones y los fabricantes de equipamiento eléctrico. La idea fue crear un protocolo abierto que permitiera cubrir las necesidades de interconexión de los productos de todos estos fabricantes y proveedores de servicios.

Tal y como fue pensado, el objetivo de la EHS es cubrir las necesidades de automatización de la mayoría de las viviendas europeas cuyos propietarios que no se pueden permitir el lujo de usar sistemas más potentes pero también más caros (como Lonworks, EIB o Batibus) debido a la mano de obra especializada que exige su instalación.

El EHS viene a cubrir, por prestaciones y objetivos, al CEbus norteamericano y el HBS japonés y rebasa las prestaciones del X-10 que tanta difusión ha conseguido en EEUU.

Durante los años 1992 al 1995 la EHSA auspició el desarrollo de componentes electrónicos que implementaran la primera especificación. Como resultado nació un circuito integrado de ST-Microelectronics (ST7537HS1) que permitía transmitir datos por una canal serie asíncrono a través de las líneas de baja tensión de las viviendas (ondas portadoras o “powerline communications”). Esta tecnología, basada en modulación FSK, consigue velocidades de hasta 2400 bps y además también puede utilizar cables de pares trenzados como soporte de la señal.

En la actualidad, se están usando o se están desarrollando los siguientes medios físicos:

- PL-2400: Ondas Portadoras a 2400 bps.
- TP0: Par Trenzado a 4800 bps (idéntico a nivel físico del BatiBUS).
- TP1: Par Trenzado/Coaxial a 9600 bps.
- TP2: Par Trenzado a 64 Kbps.
- IR-1200: Infrarrojo a 1200 bps.
- RF-1100: Radiofrecuencia a 1100 bps.

Este protocolo está totalmente abierto, esto es, cualquier fabricante asociado a la EHSA puede desarrollar sus propios productos y dispositivos que implementen el EHS.

Con un filosofía Plug&Play, se pretende aportar las siguientes ventajas a los usuarios finales:

- Compatibilidad total entre dispositivos EHS.
- Configuración automática de los dispositivos, movilidad de los mismos y ampliación sencilla de las instalaciones.
- Compartir un mismo medio físico entre diferentes aplicaciones sin interferirse entre ellas.

Cada dispositivo EHS tiene asociada una subdirección única dentro del mismo segmento de red que además de identificar unívocamente a un nodo también lleva asociada información para el enrutado de los telegramas por diferentes segmentos de red EHS.

Después de la aparición de diversos productos y soluciones basadas en EHS, esta tecnología está convergiendo, junto con el EIB y el BatiBUS, en un único estándar europeo, llamado KNX .

La asociación EHSA (EHS Association) es la encargada de emprender y llevar a cabo diversas iniciativas para aumentar el uso de esta tecnología en las viviendas europeas. Además se ocupa de la evolución y mejora tecnológica del EHS y de asegurar la compatibilidad total entre fabricantes de productos con interface EHS.

2.5.3 EIB

El Bus de Instalación Europeo EIB es un completo sistema integrado de automatización y control de edificios y viviendas, destinado a la aplicación de soluciones gradualmente compatibles, flexibles y rentables. El Bus de Instalación EIB corresponde así a las necesidades y requerimientos de los instaladores eléctricos durante las diversas fases de un proyecto, desde la planificación, instalación, puesta en marcha y el funcionamiento normal del sistema, hasta el mantenimiento del mismo.

Las siglas EIB representan la tecnología de instalaciones de edificios más innovadora (“sistema bus”), promovida desde 1990 por el grupo de fabricantes que engloban la EIBA (Asociación EIB), con sede en Bruselas. EIBA está envuelta en la emisión de las marcas registradas relacionadas con el sistema, los estándares de comprobación y calidad de los productos, las actividades de marketing y estandarización. El EIB también es distribuido bajo varias denominaciones diferentes, por ejemplo: instabus, ABB I-Bus y Tebis.

Así, el EIB nació de las exigencias de mayor flexibilidad y comodidad en las instalaciones eléctricas, unidas al deseo de minimizar las necesidades de energía.

Las empresas participantes en EIBA garantizan que sus productos sean compatibles con el bus. Por ello se pueden emplear en una instalación EIB

aparatos de distintos fabricantes con total interoperabilidad. Gracias a esta variedad existe un mercado abierto y competitivo, donde el usuario final puede elegir y lo más importante, moverse en distintos presupuestos.

Entre sus características principales podemos destacar las siguientes:

El bus de control (medio de transmisión por pares trenzados - "Twisted Pair" - TP) se tiende paralelo al cableado de 230 V. Esto implica:

- Una reducción considerable de la cantidad total de cable instalada, en comparación con una instalación convencional (hasta un 60%).

- Un incremento del número de funciones posibles del sistema.

- Una mejora en la claridad de la instalación.

Este conductor:

- Conecta las cargas y los interruptores que las controlan.

- Suministra alimentación a los componentes bus, en la mayoría de los casos.

Al disponer todos los componentes bus de su propia inteligencia, no resulta necesaria una unidad central de control (p.ej. un ordenador) por tanto EIB es un sistema descentralizado. EIB es muy versátil y modulable, puede ser utilizado tanto para pequeñas instalaciones (viviendas) como en proyectos mucho más grandes (hoteles y edificios administrativos). Gracias esta flexibilidad de la tecnología cualquier instalación puede ser fácilmente adaptable a las necesidades cambiantes del usuario.

Se puede implementar EIB en distintos medios de transmisión a parte del propio que ya hemos comentado. Resulta posible implementar el sistema EIB en la red de fuerza de 230 V existente ("Medio de transmisión Powerline" - EIB PL) y vía radio (Medio de transmisión por "Radio Frecuencia" - EIB RF).

En las instalaciones tradicionales cada función requiere una línea eléctrica propia, y cada sistema de control precisa una red separada. Por el contrario, con el EIB se pueden controlar, comunicar y vigilar todas las funciones de servicio y su desarrollo, con una única línea común. Con esto se puede dirigir la línea de energía sin desvíos, directamente hasta el aparato consumidor.

Además del ahorro en el cableado se presentan adicionalmente otras ventajas:

La instalación en un edificio se puede realizar de un modo más sencillo desde el principio, y después se puede ampliar y modificar sin problemas. Ante cambios de uso o reorganización del espacio, el EIB consigue una adaptación rápida y sin problemas, mediante una fácil ordenación de los componentes del bus, sin necesidad de un nuevo cableado.

Este cambio de ordenación se realiza con un PC, conectado al sistema EIB, que tenga instalado el software ETS (EIB Tool Software) para proyecto y puesta en servicio, que ya se emplea en la primera puesta en marcha y/o futuras ampliaciones o mantenimiento. El EIB se puede conectar mediante los correspondientes interfaces con los centros de control de otros sistemas de automatización de edificios o con una red digital de servicios integrados (RDSI). De este modo el uso del EIB en una vivienda unifamiliar resulta tan rentable como en hoteles, escuelas, bancos, oficinas o edificios del sector terciario.

2.5.4 KONNEX

Es la primer plataforma abierta y libre independiente de estándares para el control de casas y edificios. Aprobada como un estándar Europeo y Mundial.

El estándar KOMEX (KNX) es una red para el control de casas y edificios:

- KNX puede ser usada para controlar varias funciones / aplicaciones en casas y edificios, por ejemplo: iluminación, calefacción, aire acondicionado, monitoreo, alarmas y sistemas de seguridad.
- KNX soporta varios medios de comunicación (Par Trenzado, Línea de Alimentación, Radio Frecuencia y Ethernet).
- KNX simplifica la instalación eléctrica y reduce las longitudes de los cables.
- KNX puede ser usada en sistemas de control de edificios automatizados.
- Puede ser usada tanto en edificios nuevos como viejos.
- Los equipos de diferentes constructores se instalan en una red KNX y pueden intercambiar información entre ellos.
- Soporta diferentes modos de configuración, permitiendo que tanto los usuarios con conocimientos como los que no puedan hacer una red KNX.
- KNX puede ser instalada en casas de familias pequeñas o en grandes edificios.
- Las instalaciones KNX se pueden extender fácilmente y adaptarse a nuevas necesidades, con poca inversión.
- KNX ayuda al ahorro de energía en casas y edificios.

Este estándar está basado en las comunicaciones EIB pero extendida con nuevos modos de configuración y la aplicación de BatiBUS y EHS.

Como KNX no fue diseñada desde cero, se basa en más de 10 años de experiencia en otros sistemas (Batibus, EIB y EHS), KNX es un estándar único para el control de casas y edificios.

El estándar KNX incluye 3 diferentes modos de configuración y 4 diferentes tipos de medios en los cuales pueden ser transportado.

Contando con 4 tipos diferentes de comunicación, los instaladores pueden usar el medio que más les convenga para los mensajes, por ejemplo: par trenzado para nuevas instalaciones, Líneas de Transmisión o Radio Frecuencias en renovaciones. Incluso se pueden mandar los mensajes KNX por medio de mensajes en IP por lo cual también se puede usar la Internet como medio de transmisión.

2.5.5 LONWORKS

LonWorks es una tecnología de control domótico de la compañía americana Echelon Corp[8].

Al igual que KNX, LonWorks puede utilizar una gran variedad de medios de transmisión: aire, par trenzado, coaxial, fibra, o red eléctrica. Requiere la instalación de "nodos" a lo largo de la red que gestionan los distintos sensores y actuadores. La instalación y configuración de estos nodos debe ser realizada por profesionales utilizando las herramientas informáticas apropiadas (Lonmaker).

LonWorks es una tecnología muy robusta y fiable por lo que está especialmente indicada para la automatización industrial, ámbito del que procede, aunque actualmente encaja perfectamente tanto en el control de edificios como en la propia automatización industrial orientada al hogar digital.
Por medio de un Bus (cable) de comunicaciones que une todas las plantas de la vivienda, los módulos a él conectados comparten información unos con otros.

La gran ventaja de este sistema es que queda abierto a la incorporación de nuevos elementos que se integren en la red, como pueden ser luces exteriores de jardín, riego automático y alarmas técnicas en calderas, así como el hecho de disponer de un cableado virtual mediante el cual, en cualquier momento se puede reconfigurar la instalación para conseguir actuaciones y funcionalidades diferentes.

2.5.6 X-10

X-10 es actualmente y ha sido una de las tecnologías más extendidas para aplicaciones domóticas gracias al bajo coste de los equipos, a la multitud de

[8] http://www.echelon.com

dispositivos disponibles y a la facilidad de instalación y configuración. Es una tecnología orientada a viviendas construidas debido al uso del cableado de la red eléctrica existente en las mismas, por lo que todos los dispositivos a controlar ya están interconectados entre sí, aunque también puede ir dirigido a obra nueva. X10 forma parte de los sistemas denominados de corriente portadora.

Fundamentalmente se basa en el envío de mensajes muy simples entre dispositivos compatibles. Permite combinar actuaciones con sistemas de radio frecuencia compatibles X-10.

La configuración de un sistema (protocolo) X-10 es sencilla. Asignando a cada uno de los dispositivos un código formado por una letra (A-P) y por un número (1-16) podemos llegar a controlar 256 dispositivos. Lo que se hace en la práctica es asignar una o dos letras a una vivienda, según necesidades, y especificar los dispositivos con los números. De esta manera tenemos todos los dispositivos de la vivienda direccionados. Estos códigos se seleccionan de forma manual en cada dispositivo. *No es necesario de ningún tipo de software de configuración, ahí reside su gran sencillez de instalación*, aunque hay que tener en cuenta la instalación de filtros en la entrada del cuadro eléctrico de la vivienda para evitar el paso de señales que puedan afectar a nuestro sistema o viceversa.

X10 contempla la integración con otros sistemas a través de varios interfaces así como el telecontrol de la vivienda vía radio con receptores de radio frecuencia, módulos temporizadores y reguladores de iluminación.

Podríamos incorporar todos los elementos de control que se desee en función de los elementos que se quiere controlar.

Para terminar hay que añadir que X10 fue uno de los primeros protocolos diseñados de corrientes portadoras. Es un sistema modular, ampliable y con multitud de adaptadores orientados a los dispositivos. Estos a su vez también se pueden instalar en cuadros eléctricos, en cajas de registro, módulos de enchufe (plug & play) todos ellos con el objetivo de interposicionar la toma de enchufe del dispositivo a controlar y el propio dispositivo.

Debido a su gran sencillez de instalación es un producto que sigue la filosofía hágaselo usted mismo.

2.5.7 Comentarios

Al mostrar estos productos que en general tiene características comunes con el proyecto que se va a diseñar, se intenta proponer una alternativa distinta, acotar los alcances del proyecto a realizar, tomando en cuenta que no se puede hacer

una comparación directa entre los sistemas aquí mostrados y el que pretendemos diseñar ya que las compañías involucradas son enormes y cuentan con un gran número de gente trabajando en esto y enormes inversiones.

Finalmente, está claro que el objetivo de estas empresas es vender un producto y en esta etapa el nuestro será investigar y definir la posibilidad de la implementación de estas tecnologías en una escala más reducida pero con costos también reducidos, correspondientes al entorno económico, social y cultural de nuestra comunidad.

2.6 Costos actuales

Contemplando que el día de hoy (09/ 10 / 2008) el tipo de cambio está en: 1 EUR = 18.506 pesos mexicanos. Los costos con los siguientes de algunos elementos generales que integran un sistema domótico:

 **Figura 17**	Actuador térmico Normal Open	29.23EUR
 **Figura 18**	Actuador Térmico (NC) Normalmente cerrado	29.23EUR
 **Figura 19**	Antena Receptor Hometronic RF	37.06EUR
 **Figura 20**	Aspiración Centralizada Sólidos (hasta 4 tomas)	1,745.80EUR
 **Figura 21**	Contactos Puerta (Reed)	17.91EUR
 Figura 22	Controlador Suelo Radiante RF	426.15EUR

Figura 23

Grupo Control Persianas ROLLOTRON RF 391.15EUR

Figura 24

Hometronic Manager RF Telecom CENTRAL 576.43EUR

Figura 25

KIT ADOSADO 1,582.00EUR

Figura 26

KIT PISO 1,190.86EUR

Figura 27

KIT RESIDENCIA 2,078.22EUR

Figura 28

KIT RESIDENCIA 2,410.48EUR

KIT RESIDENCIA 2,410.48EUR

Figura 29

KIT RESIDENCIA 2,078.22EUR

Figura 30

KIT RESIDENCIA 2,410.48EUR

Figura 31

Capítulo 3:

Fundamentos de comunicacion de la PC.

3. Fundamentos de comunicación de la PC

3.1 Hardware de comunicación de la PC

El Hardware de comunicación se usa para transmitir datos entre terminales (incluyendo la PC que emulan terminales) y computadoras, así como entre computadoras. Estos componentes fundamentales del hardware incluyen el puerto serie, puerto paralelo, puerto USB y el puerto Firewire (IEEE_1394).

En computación, un puerto es una forma genérica de denominar a una interfaz por la cual diferentes tipos de datos pueden ser enviados y recibidos. Dicha interfaz puede ser física, o puede ser a nivel software. Algunos de estos puertos que son de interés para esta tesis, se describen a continuación:

3.1.1 Puerto serie.

Un puerto serie es una interfaz de comunicaciones entre computadoras y periféricos en donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits a la vez. Entre el puerto serie y el puerto paralelo, existe la misma diferencia que entre una carretera tradicional de un sólo carril por sentido y una carretera con varios carriles por sentido.

El puerto serie por excelencia es el RS-232 (también conocido como COM) que utiliza cableado simple desde 3 hilos hasta 25 y que conecta computadoras o microcontroladores a todo tipo de periféricos, desde terminales a impresoras y módems pasando por ratones.

El RS-232 original tenía un conector tipo D de 25 pines, sin embargo la mayoría de dichos pines no se utilizaban, por lo que IBM incorporó desde su PS/2 un conector más pequeño de **solamente 9 pines que es el que actualmente se utiliza.**

El RS-232C es un estándar que constituye la tercera revisión de la antigua norma RS-232, propuesta por la EIA (Asociación de Industrias Electrónicas), realizándose posteriormente un versión internacional por el CCITT, conocida como V.24. Las diferencias entre ambas son mínimas, por lo que a veces se habla indistintamente de V.24 y de RS-232C (incluso sin el sufijo “C”), refiriéndose siempre al mismo estándar.

El RS-232C consiste en un conector tipo DB25 de 25 pines, aunque es normal encontrar la versión de 9 pines DB-9, mas económico e incluso más extendido para cierto tipo de periféricos (como el ratón serie del PC). En cualquier caso, los PC's no suelen emplear mas de 9 pines en el conector DB-25.

Las **señales con las que trabaja** este puerto serie son digitales, de +12V (0 lógico) y -12V (1 lógico), para la entrada y salida de datos, y a la inversa en las señales de control. El estado de reposo en la entrada y salida de datos es -12V.

Dependiendo de la velocidad de transmisión empleada, es posible tener cables de hasta 15 metros.

Cada pin puede ser de entrada o de salida, teniendo una función específica cada uno de ellos. Las más importantes son:

Pin	***Función***
TXD	(Transmitir Datos)
RXD	(Recibir Datos)
DTR	(Terminal de Datos Listo)
DSR	(Equipo de Datos Listo)
RTS	(Solicitud de Envío)
CTS	(Libre para Envío)
DCD	(Detección de Portadora)

Tabla 1

Las señales TXD, DTR y RTS son de salida, mientras que RXD, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es SG (Tierra de Señal). Finalmente, existen otras señales como RI (Indicador de Llamada).

Todas estas señales se pueden observar resumidas en la tabla 2, además de un esquema grafico de los puertos DB-25 y DB-9 en la tabla 3.

Numero	*de Pin*	*Señal*	*Descripción*	*E/S*
En DB-25	En DB-9			
1	1	-	Masa chasis	-
2	3	TxD	Transmit Data	S
3	2	RxD	Receive Data	E
4	7	RTS	Request To Send	S
5	8	CTS	Clear To Send	E
6	6	DSR	Data Set Ready	E
7	5	SG	Signal Ground	-
8	1	CD/DCD	(Data) Carrier Detect	E
15	-	TxC(*)	Transmit Clock	S
17	-	RxC(*)	Receive Clock	E
20	4	DTR	Data Terminal Ready	S
22	9	RI	Ring Indicator	E
24	-	RTxC(*)	Transmit/Receive Clock	S

Tabla 2

(*) = Normalmente no conectados en el DB-25

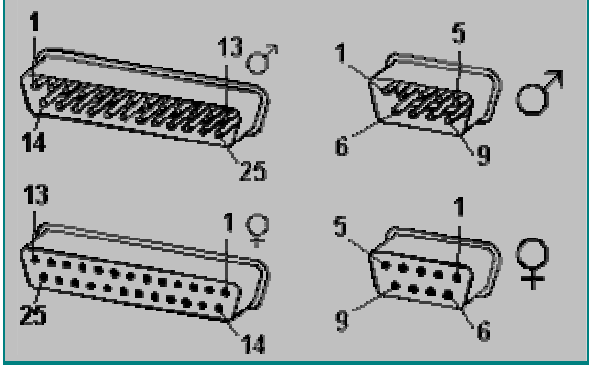
Conector DB 25	*Conector DB 9*
	

Tabla 3

La computadora controla el puerto serie mediante un circuito integrado específico, llamado UART (Transmisor-Receptor-Asíncrono Universal).

Normalmente se utilizan los siguientes modelos de este chip: 8250 (bastante antiguo, con fallos, solo llega a 9600 baudios), 16450 (versión corregida del 8250, llega hasta 115.200 baudios) y 16550A (con buffers de E/S). A partir de la gama Pentium, la circuitería UART de las placa base son todas de alta velocidad, es decir UART 16550A. De hecho, la mayoría de los módems conectables a puerto serie necesitan dicho tipo de UART, incluso algunos juegos para jugar en red a través del puerto serie necesitan de este tipo de puerto serie. Por eso hay veces que un 486 no se comunica con la suficiente velocidad con un PC Pentium. Las

computadoras portátiles suelen llevar otros chips: 82510 (con buffer especial, emula al 16450) o el 8251 (no es compatible).

Para controlar al puerto serie, el CPU emplea **direcciones** de puertos de E/S y líneas de interrupción (IRQ). En el AT-286 se eligieron las direcciones 3F8h (o 0x3f8) e IRQ 4 para el COM1, y 2F8h e IRQ 3 para el COM2. El estándar del PC llega hasta aquí, por lo que al añadir posteriormente otros puertos serie, se eligieron las **direcciones** 3E8 y 2E8 para COM3-COM4, pero las IRQ no están especificadas. Cada usuario debe elegirlas de acuerdo a las que tenga libres o el uso que vaya a hacer de los puertos serie (por ejemplo, no importa compartir una misma IRQ en dos puertos siempre que no se usen conjuntamente, ya que en caso contrario puede haber problemas). Es por ello que últimamente, con el auge de las comunicaciones, los fabricantes de PC's incluyan un puerto especial PS/2 para el ratón, dejando así libre un puerto serie.

Mediante los puertos de E/S se pueden intercambiar datos, mientras que las IRQ producen una interrupción para indicar a la CPU que ha ocurrido un evento (por ejemplo, que ha llegado un dato, o que ha cambiado el estado de algunas señales de entrada). El CPU debe responder a estas interrupciones lo más rápido posible, para que dé tiempo a recoger el dato antes de que el siguiente lo sobrescriba. Sin embargo, las UART 16550A incluyen unos buffers de tipo FIFO (first input first output), dos de 16 bytes (para recepción y transmisión), donde se pueden guardar varios datos antes de que la CPU los recoja. Esto también disminuye el número de interrupciones por segundo generadas por el puerto serie.

El RS-232 puede transmitir los datos en grupos de 5, 6, 7 u 8 bits, a unas velocidades determinadas (normalmente, 9600 bits por segundo o más).

Después de la transmisión de los datos, le sigue un bit opcional de paridad (indica si el numero de bits transmitidos es par o impar, para detectar fallos), y después 1 o 2 bits de Stop. Normalmente, el protocolo utilizado es el 8N1 (que significa, 8 bits de datos, sin paridad y con 1 bit de Stop).

Una vez que ha comenzado la transmisión de un dato, los bits tienen que llegar uno detrás de otro a una velocidad constante y en determinados instantes de tiempo. Por eso se dice que el RS-232 es síncrono por carácter y asíncrono por bit. Los pines que portan los datos son RXD y TXD. Las demás se encargan de otros trabajos: DTR indica que el ordenador esta encendido, DSR que el aparato conectado a dicho puerto esta encendido, RTS que el ordenador puede recibir datos (porque no está ocupado), CTS que el aparato conectado puede recibir datos, y DCD detecta que existe una comunicación, presencia de datos.

Tanto el aparato a conectar como la computadora (o el programa terminal) tienen que usar el mismo protocolo serie para comunicarse entre si. Puesto que el

estándar RS-232 no permite indicar en que modo se esta trabajando, es el usuario quien tiene que decidirlo y configurar ambas partes. Como ya se ha visto, los parámetros que hay que configurar son: protocolo serie (8N1), velocidad del puerto serie, y protocolo de control de flujo. Este ultimo puede ser por hardware o bien por software (XON/XOFF, el cual no es muy recomendable ya que no se pueden realizar transferencias binarias). La velocidad del puerto serie no tiene por que ser la misma que la de transmisión de los datos, de hecho debe ser superior. Por ejemplo, para transmisiones de 1200 baudios es recomendable usar 9600, y para 9600 baudios se pueden usar 38400 (o 19200).

Uno de los defectos de los puertos serie iníciales era su lentitud en comparación con los puertos paralelos, sin embargo, con el paso del tiempo, están apareciendo multitud de puertos serie de alta velocidad que los hacen muy interesantes ya que utilizan las ventajas del menor cableado; solucionan el problema de la velocidad y la economía usando la técnica del par trenzado. Por ello, el puerto RS-232 e incluso multitud de puertos paralelos están siendo reemplazados por nuevos puertos serie como el USB, el Firewire o el Serial ATA.

Los puertos serie sirven para comunicar la computadora con la impresora, el ratón o el módem; Sin embargo, específicamente, el puerto USB sirve para todo tipo de periféricos, como ratones o discos duros externos. Los puertos SATA (Serial ATA): tienen la misma función que los IDE, (a éstos se conecta, la disquetera, el disco duro, lector/grabador de CD's y DVD's) pero los SATA cuentan con mayor velocidad. Un puerto de red puede ser puerto serie o puerto paralelo.

3.1.2 Puerto Paralelo

Un puerto paralelo es un interfaz entre una computadora y un periférico cuya principal característica es que los bits de datos viajan juntos enviando un byte completo o más a la vez. Es decir, se implementa un cable o una vía física para cada bit de datos formando un bus. El puerto paralelo más conocido es el puerto de impresora que destaca por su sencillez y que transmite 8 bits. Un puerto paralelo sirve preferentemente para la impresora, siendo éste su uso más extendido. Sin embargo, dado que este puerto tiene un conjunto de entradas y salidas digitales, se puede emplear para hacer prácticas experimentales de lectura de datos y control de dispositivos.

Este puerto paralelo cumple la norma IEEE 1284 y también es denominado tipo Centronics. Se ha utilizado principalmente para conectar impresoras, pero también ha sido usado para programadores EPROM, escáners, interfaces de red Ethernet, unidades ZIP y para comunicación entre dos PC's (MS-DOS trajo en las versiones 5.0 ROM a 6.22 un programa para soportar esas transferencias).

Las líneas de comunicación de este puerto cuentan con un retenedor que mantiene el último valor que les fue escrito hasta que se escribe un nuevo dato, las características eléctricas son:

- ✓ Tensión de nivel alto: 3.3 o 5 V.
- ✓ Tensión de nivel bajo: 0 v.
- ✓ Intensidad de salida máxima: 2.6 mA.
- ✓ Intensidad de entrada máxima: 24 mA.

El sistema operativo gestiona las interfaces de puerto paralelo con los nombres LPT1, LPT2 y así sucesivamente, las direcciones base de los dos primeros puertos son:

LPT1 = 0x378
LPT2 = 0x278

El puerto paralelo está formado por 17 líneas de señales y 8 líneas de tierra.

Las líneas de señales están formadas por tres registros de 8 bits cada uno:

- ✓ 4 Líneas de control (Salidas de tipo Colector Abierto[9])
- ✓ 5 Líneas de estado (Salidas conectadas a un buffer 74LD240)
- ✓ 8 Líneas de datos (Salida de tipo Buffer 74LS374)

El **registro de datos**, se compone de 8 bits, que son de salida (aunque con algunas configuraciones actuales puede ser bidireccional, pero es común utilizarlo como salidas). Su dirección en el LPT1 es 0x378. Es el que utilizaremos para encender/apagar dispositivos. (Pin del 2 al 9)

El **registro de estado**, se trata de un registro de entrada (utilizaremos este debido a que no necesita ninguna configuración extra para que se comporte como entrada) de información de 5 bits, su dirección en el LPT1 es 0x379. En este registro las escrituras serán ignoradas. (Pin 11, 10, 12, 13, 15)

El **registro de control** es de salida (aunque puede ser de entrada con la nota de la pag. 50) en 4 bits, con 2 bit de configuración que no tiene conexión al exterior, su dirección en el LPT1 es 0x37A. Este no lo utilizaremos para leer los datos del exterior debido a que es más complicada su configuración que el registro de estado. (Pin 17, 16, 14, 1)

[9] Gadre, Dhananjay V. Programming the parallel port: interfacing the PC for data acquisition and process control. Lawrence, Kansas City. Ed. R&D Books, 1998. Pag. 32

En la tabla 4 se muestra la relación que existe entre las líneas físicas (pines) del conector de la PC y los registros.

DB25 pin	*Registro bit*		*Tipo (E/S)*	*Activo*	*Señal*	*Descripción*
1	**C0**	Control 0	**S**	bajo	**Strobe**	Si está bajo más de 0.5 µs, habilita a la impresora para que reciba los datos enviados.
2	**DO**	Dato 0	**S**	alto	**D0**	Bit 0 de datos, bit menos significativo (LSB)
3	**D1**	Dato 1	**S**	alto	**D1**	Bit 1 de datos
4	**D2**	Dato 2	**S**	alto	**D2**	Bit 2 de datos
5	**D3**	Dato 3	**S**	alto	**D3**	Bit 3 de datos
6	**D4**	Dato 4	**S**	alto	**D4**	Bit 4 de datos
7	**D5**	Dato 5	**S**	alto	**D5**	Bit 5 de datos
8	**D6**	Dato 6	**S**	alto	**D6**	Bit 6 de datos
9	**D7**	Dato 7	**S**	alto	**D7**	Bit 7 de datos, bit más significativo (MSB)
10	**S6** IRQ	Estado 6	**E**	alto	**Ack**	Un pulso bajo de ~11µs indica que se han recibido datos en la impresora y que la misma está preparada para recibir más datos.
11	**S7**	Estado 7	**E**	bajo	**Busy**	En alto indica que la impresora está ocupada.
12	**S5**	Estado 5	**E**	alto	**PaperEnd**	En alto indica que no hay papel.
13	**S4**	Estado 4	**E**	alto	**SelectIn**	En alto para impresora seleccionada.
14	**C1**	Control 1	**S**	bajo	**AutoFeed**	Si está bajo, el papel se mueve una línea tras la impresión.
15	**S3**	Estado 3	**E**	alto	**Error**	En bajo indica error (no hay papel, está fuera de línea, error no det.).
16	**C2**	Control 2	**S**	alto	**Init**	Si se envía un pulso en bajo > 50 µs la impresora se reinicia.
17	**C3**	Control 3	**S**	bajo	**Select**	En bajo selecciona impresora (en gral. no se usa, ya que SelectIn se fija a alto).
18-25					**GND**	Tierra retorno del par trenzado, tierra lógica y tierra chasis.

Tabla 4[10]

Notas:
Un dato en alto es un 1, un dato en bajo es un 0

[10] http://cfievalladolid2.net/tecno/cyr_01/control/puerto_paralelo.htm

Si encontramos que la dirección base es 378h, entonces las direcciones de los registros serán de datos = dirección base + 0, estado = dirección base + 1 y control = dirección base + 2:

- ✓ Base (datos)=378h
- ✓ Estado=379h
- ✓ Control=37Ah

Cada una de ellas permite accesar a los siguientes bits (descritos en la tabla 4):

- ✓ Base (datos)=D0, D1, D2, D3, D4, D5, D6, D7
- ✓ Estado=S3, S4, S5, S6, S7
- ✓ Control=C0, C1, C2, C3

No obstante existe otro puerto paralelo usado masivamente en las computadoras: el puerto paralelo IDE, también llamado PATA (Paralell ATA), usado para la conexión de discos duros, unidades lectoras/grabadoras (CD-ROM, DVD), unidades magneto-ópticas, unidades ZIP y SuperDisk.

Un tercer puerto paralelo, muy usado en las computadoras Apple Macintosh y en servidores, son las diferentes implementaciones del SCSI. Al igual que IDE ha sido usado para la conexión de discos duros, unidades ópticas lectoras/grabadoras (CD-ROM, DVD), unidades magneto-ópticas, unidades y SuperDisk, pero también de otros dispositivos como escáners.

En contraposición al puerto paralelo está el puerto serie, que envía los datos bit a bit por el mismo hilo, como ya lo mocionamos.

El puerto paralelo típico de una PC se puede ver en las siguientes figuras.

Figura 32

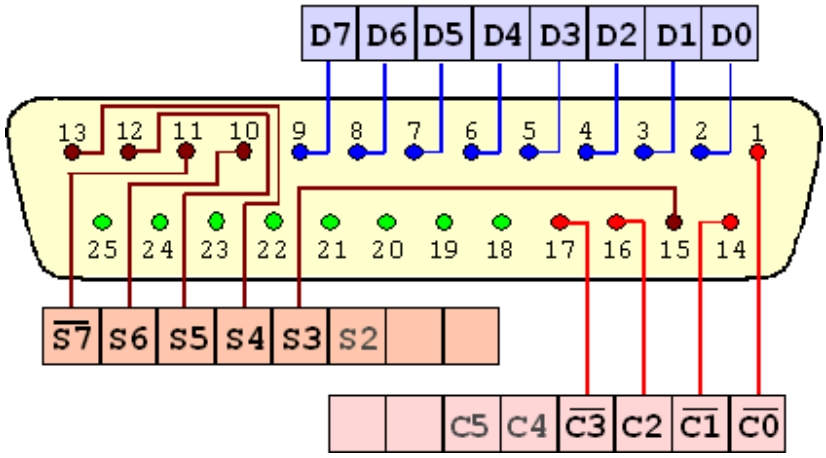


Figura 33

ENVIÓ Y LECTURA DE DATOS DEL PUERTO.

El puerto paralelo puede ser usado como una comunicación directa con el PC, de este modo es posible desarrollar aplicaciones en tiempo-real que necesiten un rápido tiempo de respuesta.

El **envió** se realiza escribiendo un byte (8 bits) en el registro de datos. Este byte debe referirse a cada uno de los bits de tal registro. Por lo que se debe definir el byte a escribir en sistema binario.

Leyendo el registro de datos, se lee el último dato escrito, NO LEE EL ESTADO DE LOS PINES.

Por ejemplo si queremos escribir en el registro de datos el bit D0 y el bit D2 entonces el byte es:

0	0	0	0	0	1	0	1	= 5 En decimal
D7	D6	D5	D4	D3	D2	D1	D0	Registro de datos

Tabla 5

Esto se verá reflejado en los pines correspondientes en la tabla 4 (del 2 al 9).

La **lectura** de datos se realiza acensando al byte del registro de estado. Al hacer la lectura se debe convertir el byte de decimal a binario para determinar él o los bits que interesen.

Nota:
Cabe destacar que las salidas del registro de control son TTL de tipo colector abierto con resistencias de Pull-up de 4700 ohms, por lo que un dispositivo externo puede forzar el estado de los pines sin dañar el driver ya que controlamos

la corriente con la resistencia que le pongamos. Para ello ponemos en alto las cuatro salidas (escribiendo 0000100b en el registro de control), lo que hace que las salidas "floten". Y podamos leer el estado de estos pines.
Es posible realizar esta técnica en salidas tótem-pole (ya que los buffer de los registros D0-D7 y S3-S7 son de este tipo) pero no se recomienda, a menos que se tenga un conocimiento preciso de la corriente, ya que sino se puede sobrecargar los transistores de salida, dañando el driver del puerto.

Por ejemplo, si el dato leído es igual a 96, significa que los bits S5 y S6 están activos (tienen un nivel alto, un 1).

0	1	1	0	0	0	0	0	= 96 En decimal
S7	S6	S5	S4	S3	S2	S1	S0	Registro de estado

Tabla 6

Por ultimo su **velocidad**[11] para leer o escribir un dato en la práctica, según su diagrama de tiempo, es de aproximadamente de 2 us.

Para tener una idea más clara de este puerto, se muestra un diagrama general en la figura 34, en donde las tierras se indican con color verde, las entradas de dialogo se indican con flechas que apuntan al conector y las salidas (tanto de datos como de dialogo) tienen flechas que apuntan hacia afuera del conector. (Note que algunas líneas pueden ser bidireccionales)

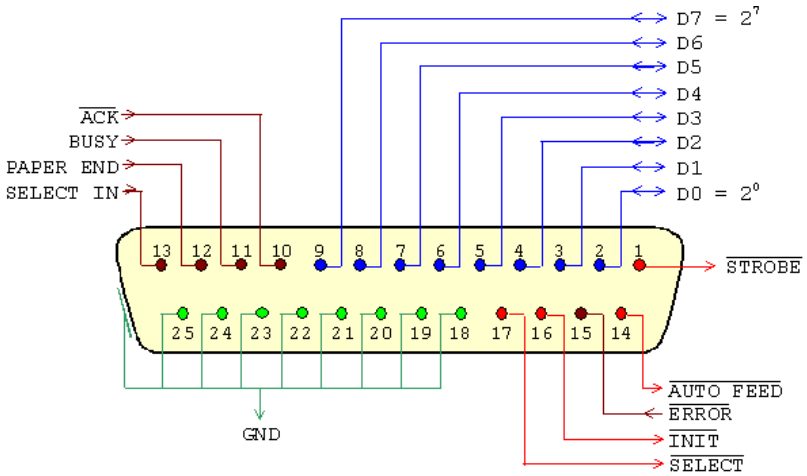


Figura 34

[11] Gadre, Dhananjay V. Programming the parallel port: interfacing the PC for data acquisition and process control. Lawrence, Kansas City. Ed. R&D Books, 1998. Pag. 36

3.1.3 USB

El **Bus de Serie Universal** (**USB**, de sus siglas en inglés *Universal Serial Bus*) es una interfaz que provee un estándar de bus serie para conectar dispositivos a una PC. En un principio teníamos la interfaz serie y paralelo, pero era necesario unificar todos los conectores creando uno más sencillo y de mayores prestaciones. Así nació el USB con una velocidad de 12Mb/seg. y evolucionando a llegado al USB 2.0 (de alta velocidad) con velocidades en este momento de hasta 480 Mb/seg, es decir, 40 veces más rápido que las conexiones mediante cables USB 1.1.

Fue creado en 1996 por siete empresas: IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC.

El estándar incluye la transmisión de energía eléctrica al dispositivo conectado. Algunos dispositivos requieren una potencia mínima, así que se pueden conectar varios sin necesitar fuentes de alimentación extra. Esta interfaz tiene la característica de ser plug-and-play, haciendo que la conexión sea fácilmente utilizada y permita instalar periféricos sin tener que abrir tu máquina para instalarle hardware o a pagar y encender la PC, para que reconozca este hardware.

Emplea una topología de estrellas apiladas que permite el funcionamiento simultáneo de 127 dispositivos a la vez. En la raíz o vértice de las capas, está el controlador anfitrión o host que controla todo el tráfico que circula por el bus. Esta topología permite a muchos dispositivos conectarse a un único bus lógico sin que los dispositivos que se encuentran más abajo en la pirámide sufran retardo. Como se puede ver en la pirámide.

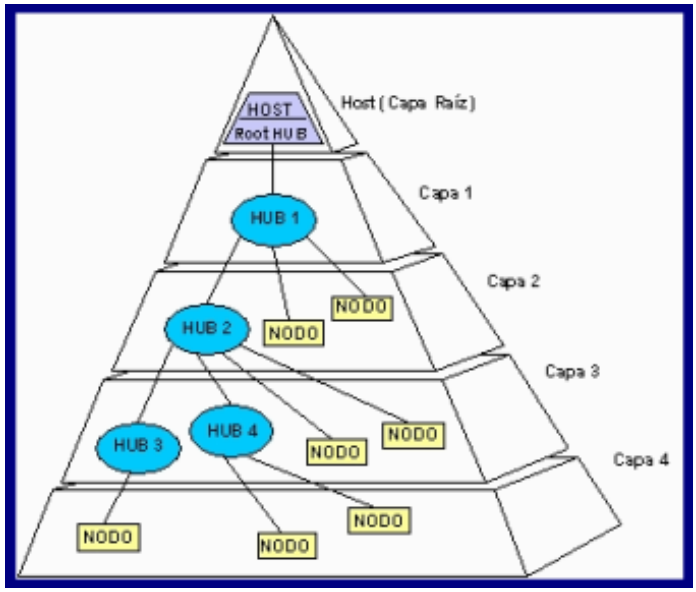


Figura 35

El sistema de bus serie universal USB consta de tres componentes:

- ✓ Controlador.
- ✓ Hubs o Concentradores.
- ✓ Periféricos

Controlador: Reside dentro del PC y es responsable de las comunicaciones entre los periféricos USB y la CPU del PC. Es también responsable de la admisión de los periféricos dentro del bus, tanto si se detecta una conexión como una desconexión. Para cada periférico añadido, el controlador determina su tipo y le asigna una dirección lógica para utilizarla siempre en las comunicaciones con el mismo. Si se producen errores durante la conexión, el controlador lo comunica a la CPU, que, a su vez, lo transmite al usuario. Una vez se ha producido la conexión correctamente, el controlador asigna al periférico los recursos del sistema que éste precise para su funcionamiento. El controlador también es responsable del control de flujo de datos entre el periférico y la CPU.

Concentradores o hubs: Son distribuidores inteligentes de datos y alimentación, y hacen posible la conexión a un único puerto USB de 127 dispositivos. De una forma selectiva reparten datos y alimentación hacia sus puertas descendentes y permiten la comunicación hacia su puerta de retorno o ascendente. Un hub de 4 puertos, por ejemplo, acepta datos del PC para un periférico por su puerta de retorno o ascendente y los distribuye a las 4 puertas descendentes si fuera necesario. Los concentradores también permiten las comunicaciones desde el periférico hacia el PC, aceptando datos en las 4 puertas descendentes y enviándolos hacia el PC por la puerta de retorno.

Además del controlador, el PC también contiene el concentrador raíz. Este es el primer concentrador de toda la cadena que permite a los datos y a la energía pasar a uno o dos conectores USB del PC, y de allí a los 127 periféricos que, como máximo, puede soportar el sistema. Esto es posible añadiendo concentradores adicionales. Por ejemplo, si el PC tiene una única puerta USB y a ella le conectamos un hub o concentrador de 4 puertas, el PC se queda sin más puertas disponibles. Sin embargo, el hub de 4 puertas permite realizar 4 conexiones descendentes. Conectando otro hub de 4 puertas a una de las 4 puertas del primero, habremos creado un total de 7 puertas a partir de una puerta del PC. De esta forma, es decir, añadiendo concentradores, el PC puede soportar hasta 127 periféricos USB.

Periféricos: USB soporta periféricos de baja y media velocidad. Empleando dos velocidades para la transmisión de datos de 1.5 y 12 Mbps se consigue una utilización más eficiente de sus recursos . Los periféricos de baja velocidad tales

como teclados, ratones, joysticks, y otros periféricos para juegos, empleando para ellos 1.5 Mbps, se puede dedicar más recursos del sistema a periféricos tales como monitores, impresoras, módems, y scanner que precisan de velocidades más altas para transmitir mayor volumen de datos o datos cuya dependencia temporal es más estricta .

En las figuras siguientes se puede ver cómo los hubs proporcionan conectividad a toda una serie de dispositivos periféricos.

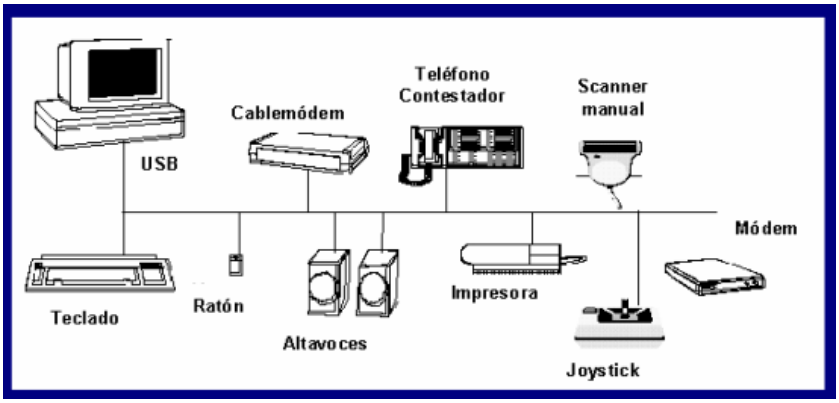


Figura 36

El USB transfiere señales y energía a los periféricos utilizando un cable de 4 hilos, trenzado para transmisiones a 12 Mbps y no trenzado para transmisiones a 1.5 Mbps . En la figura 35 se muestra un esquema del cable, con dos conductores para alimentación y los otros dos para señal, debiendo estos últimos ser trenzados o no según la velocidad de transmisión.

Figura 37

El calibre de los conductores destinados a alimentación de los periféricos varía desde 20 a 26 AWG, mientras que el de los conductores de señal es de 28 AWG. La longitud máxima de los cables es de 5 metros.

Por lo que respecta a los conectores hay que decir que son del tipo ficha (o conector) y receptáculo, y son de dos tipos: serie A y serie B. Los primeros presentan las cuatro patillas correspondientes a los cuatro conductores alineadas en un plano. El color recomendado es blanco sucio y los receptáculos se presentan en cuatro variantes: vertical, en ángulo recto, panel y apilado en ángulo recto. Se

emplean en aquellos dispositivos en los que el cable externo, está permanentemente unido a los mismos, tales como teclados, ratones, y hubs o concentradores.

Los conectores de la serie B presentan los contactos distribuidos en dos planos paralelos, dos en cada plano, y se emplean en los dispositivos que deban tener un receptáculo al que poder conectar un cable USB. Por ejemplo impresoras, scanner, y módems.

Los modelos más recientes de PC vienen hasta con 6 puertos USB, haciendo del PC una herramienta capaz de conectarse a todo por esta vía: Mouse, joysticks, impresoras, discos duros externos, scanner, PDA y copiadores de CD.

A mediados del 2001 se presentó la nueva maravilla de los puertos, USB 2.0. Con una velocidad de transferencia de 480 mbps, sobrepasó al estándar 1394.

La poderosa firma Intel no se demoró mucho en subirse al carro de la victoria y decir que sus chips vendrían integrados con esta nueva versión, que entre sus gracias está que es absolutamente compatible con la versión anterior. Si se tienen dispositivos USB 1.1, no hay problema en conectarlos al puerto USB 2.0.

3.1.4 IEEE 1394

Empezaron a salir nuevos estándares de USB y el USB 1.1 quedó medio obsoleto, pues no estaba acorde a las velocidades de transferencia del momento. Así, el puerto IEEE 1394, conocido en el ambiente Mac como FireWire y en los PC como iLink sobrepasó en velocidad al USB, por bastante velocidad: 400 mbps.

El IEEE 1394 o FireWire o iLink es un estándar multiplataforma para entrada/salida de datos en serie a gran velocidad.

El FireWire fue inventado por Apple Computer a mediados de los 90, para luego convertirse en el estándar multiplataforma IEEE 1394. A principios de este siglo fue adoptado por los fabricantes de periféricos digitales hasta convertirse en un estándar establecido. Sony utiliza el estándar IEEE 1394 bajo la denominación i.Link, que sigue los mismos estándares pero solo utiliza 4 conexiones, de las 6 disponibles en la norma IEEE 1394, suprimiendo las dos conexiones encargadas de proporcionar energía al dispositivo, que tendrá que proveerse de ella mediante una toma separada.

Es cierto que para muchos periféricos esta velocidad (400 mbps) es demasiada, no es necesaria, pero para algunos dispositivos es una cosa fundamental. Por ejemplo, los discos duros, los copiadores de CD, o las videocámaras digitales. La

cantidad de información que necesitan transferir en poco tiempo es mucha, y los 12 mbps no fueron suficientes.

Su identificación en la PC es como se muestra en la figuras 36 y 37:

Figura 38

Figura 39

3.1.4.1 Ventajas de FireWire:

- ✓ Alcanzan una velocidad de 400 megabits por segundo.
- ✓ Es hasta cuatro veces más rápido que una red Ethernet 100Base-T y 40 veces más rápido que una red Ethernet 10Base-T.
- ✓ Soporta la conexión de hasta 63 dispositivos con cables de una longitud máxima de 425 cm.
- ✓ No es necesario apagar un escáner o una unidad de CD antes de conectarlo o desconectarlo, y tampoco requiere reiniciar el ordenador.
- ✓ Los cables FireWire se conectan muy fácilmente: no requieren números de identificación de dispositivos, conmutadores DIP, tornillos, cierres de seguridad ni terminadores.
- ✓ FireWire funciona tanto con Macintosh como con PC.
- ✓ FireWire 400 envía los datos por cables de hasta 4,5 metros de longitud. Mediante fibra óptica profesional, FireWire 800 puede distribuir información por cables de hasta 100 metros, lo que significa que podrías disparar ese CD hasta la otra punta de un campo de fútbol cada diez segundos. Ni siquiera necesitas ordenador o dispositivos nuevos para alcanzar estas distancias. Siempre que los dispositivos se conecten a un concentrador FireWire 800, puedes enlazarlos mediante un cable de fibra óptica súper eficiente.

3.2 Comunicación remota entre PC's

En este apartado nos enfocaremos a la manera en que las computadores se comunican unas con otras a través de una red para enviarse datos.

Las redes actuales están basadas en la teoría de capas, a su vez existen dos modelos de referencia principales que describen la arquitectura de una red: OSI (Interconexión de Sistemas Abiertos) y TCP/IP. Aunque los protocolos asociados con el modelo OSI ya casi no se usan, el modelo en si es muy general y aun es valido, y las características tratadas en cada capa aun son muy importantes. El modelo TCP/IP tiene las propiedades opuestas: el modelo en si no se utiliza mucho pero los protocolos sí.

Posteriormente sentaremos las bases de la programación para este tipo de comunicación, utilizando "sockets", esto nos abrirá el panorama para saber que lenguaje de programación nos facilitara el desarrollo de nuestro proyecto.

3.2.1 Encapsulamiento de datos y construcción de paquetes

Para reducir la complejidad de su diseño, la mayoría de las redes está organizada como una pila de capas o niveles, cada una construida a partir de la que está debajo de ella. El número de capas, así como el nombre, contenido y función de cada una de ellas difieren de red a red. El propósito de cada capa es ofrecer ciertos servicios a las capas superiores, a las cuales no se les muestran los detalles reales de implementación de los servicios ofrecidos.

Este concepto se le conoce de diferentes maneras como: ocultamiento de la información, tipos de datos abstractos, **encapsulamiento de datos** y programación orientada a objetos. La idea básica es que una pieza particular de software (o hardware) proporciona un servicio a sus usuarios pero nunca les muestra los detalles de su estado interno ni sus algoritmos.

La capa n de una maquina mantiene una conversación con la capa n de otra máquina. Las reglas y convenciones utilizadas en esta conversación se conocen de manera colectiva como protocolo de la capa n. Básicamente un **protocolo** es un acuerdo entre las partes en comunicaciones sobre cómo se debe llevar a cabo la comunicación.

En realidad, los datos no se transfieren de manera directa desde una capa n de una maquina a la capa n de otra máquina, sino que cada capa pasa los datos y la información de control a la capa inmediata inferior, hasta que se alcanza la capa más baja. En la capa más baja se encuentra el medio físico a través del cual ocurre la comunicación real.

Ahora basándonos en el modelo principal que describe a las redes actuales, (OSI, que fue desarrollado por la Organización Internacional para la Normalización (ISO) en 1984) describiremos de manera breve como funciona la construcción de un paquete hasta su envió a otra máquina a través de un medio físico.

En el modelo de referencia OSI, hay **siete capas**, cada una de las cuales ilustra una función de red específica. Si la red se divide en estas siete capas, se obtienen las siguientes ventajas:

- ✓ Divide la comunicación de red en partes más pequeñas y sencillas.
- ✓ Normaliza los componentes de red para permitir el desarrollo y el soporte de los productos de diferentes fabricantes.
- ✓ Permite a los distintos tipos de hardware y software de red comunicarse entre sí.
- ✓ Impide que los cambios en una capa puedan afectar las demás capas, para que se puedan desarrollar con más rapidez.
- ✓ Divide la comunicación de red en partes más pequeñas para simplificar el aprendizaje.

El problema de trasladar información entre computadoras se divide en siete problemas más pequeños y de tratamiento más simple en el modelo de referencia OSI. Cada uno de los siete problemas más pequeños está representado por su propia capa en el modelo. Las siete capas del modelo de referencia OSI son:

Capa	***Descripción***
7	Aplicación
6	Presentación
5	Sesión
4	Transporte
3	Red
2	Enlace de datos
1	Física

Tabla 7

Cada capa individual del modelo OSI tiene un conjunto de funciones que debe realizar para que los paquetes de datos puedan viajar en la red desde el origen hasta el destino. A continuación, presentaremos una breve descripción de cada capa que aparece en la tabla 7.

- ✓ **La capa de aplicación:**

La capa de aplicación es la capa del modelo OSI más cercana al usuario. Suministra servicios de red a las aplicaciones del usuario. Difiere de las demás capas debido a que no proporciona servicios a ninguna otra capa (ya que es la primera). Algunos ejemplos de aplicaciones son los programas de hojas de cálculo, procesadores de texto y navegador de Internet

- ✓ **La capa de presentación:**

La capa de presentación garantiza que la información que envía la capa de aplicación de un sistema pueda leerse por la capa de aplicación de otro. Si es necesario, la capa de presentación traduce entre varios formatos de datos utilizando un formato común. Es decir, le corresponde la sintaxis y la semántica de la información trasmitida.

- ✓ **La capa de sesión:**

La capa de sesión establece, administra y finaliza las sesiones entre dos computadoras que se están comunicando, una vez conectados le envía los datos (capa 6) a los programas (capa 7). La capa de sesión proporciona sus servicios a la capa de presentación. También sincroniza el diálogo entre las capas de presentación de las dos computadoras y administra su intercambio de datos. Es decir es la que gestiona la conversación entre los dos equipos.

- ✓ **La capa de transporte:**

Su función básica es aceptar los datos provenientes de capas superiores, dividirlos en unidades más pequeñas si es necesario pasar estas a la capa de red y asegurarse de que todas las piezas lleguen correctamente al otro extremo. A partir de ahora las siguientes capas se encargan de cómo transportar los datos.

Es decir, que la capa de transporte intenta suministrar un servicio de transporte de datos que aísla las capas superiores de los detalles de implementación del transporte. Al proporcionar un servicio de comunicaciones, la capa de transporte establece, mantiene y termina adecuadamente las conexiones. Al proporcionar un servicio fiable, se utilizan dispositivos de detección y recuperación de errores de transporte.

- ✓ **La capa de red:**

La capa de red es una capa compleja que proporciona conectividad y selección de ruta entre dos computadoras que pueden estar ubicadas en redes geográficamente distintas. En este caso sería la selección de ruta, direccionamiento y enrutamiento.

- ✓ **La capa de enlace de datos:**

La capa de enlace de datos proporciona tránsito de datos a través de un enlace físico. Al hacerlo, la capa de enlace de datos se ocupa del direccionamiento físico

(comparado con el lógico que era de la capa anterior), la topología de red, el acceso a la red, la notificación de errores, entrega ordenada de tramas y control de flujo. Estaríamos hablando de tramas y control de acceso al medio (tarjeta de red).

- ✓ **La capa física:**

En esta capa se lleva a cabo la transmisión de bits puros a través de un canal de comunicación. Es decir, define las especificaciones eléctricas, de procedimiento y funcionales para activar, mantener y desactivar el enlace físico entre sistemas finales. Las características tales como niveles de voltaje, temporización de cambios de voltaje, velocidad de datos físicos, distancias de transmisión máximas, conectores físicos y otros atributos similares son definidos por las especificaciones de la capa física.

En conclusión, los datos para ser encapsulados tienen que pasar por estas siete capas, cuya función general es descrita en la figura 40.

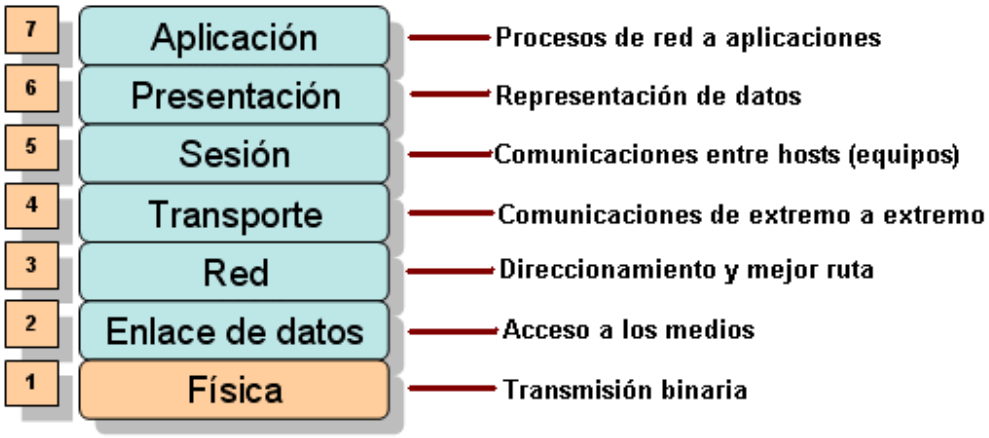


Figura 40

El encapsulamiento empaqueta los datos con la información de protocolo necesaria antes de que se una al tránsito de la red. Por lo tanto, a medida que los datos se desplazan a través de las capas del modelo OSI, reciben encabezados, información final y otros tipos de información. (Nota: La palabra “encabezado” significa que se ha agregado la información correspondiente a la dirección del destino). Cabe mencionar que actualmente se hace una combinación entre las capas de este modelo y los protocolos del modelo TCP/IP para tener una arquitectura eficiente de red. En general a esta combinación se le conoce como **TCP / IP**

3.2.2 Programación de Sockets.

Antes de explicar este tema, tendremos claros los siguientes conceptos:

El protocolo **TCP** (Transefer Control Protocol) es un protocolo orientado a conexión (establecer la conexión, utilizarla y abandonar la conexión) que utiliza los servicios de nivel IP. TCP permite la multiplexación, esto es, la capacidad de que una conexión TCP pueda ser utilizada simultáneamente por varios usuarios. La unidad de datos que maneja TCP se denomina segmento y la longitud de un segmento se mide en caracteres (octetos). La transmisión que ofrece TCP es fiable, permite la recuperación ante datos perdidos, erróneos o duplicados y garantiza la secuencia de entrega para lo que asigna al segmento de datos un número de secuencia y un código de control (checksum).

El protocolo **UDP** (User Datagram Protocol) es un protocolo del nivel de trasporte que se basa en el intercambio de datagramas. Permite el envió de datagramas a través de la red sin que se haya establecido previamente una conexión (ofrece un servicio no orientado a conexión), para lo que el propio datagrama incorpora la suficiente información de direccionamiento. Esto simplifica notablemente el protocolo, pero a cambio, no se confirman los datagramas recibidos ni se garantiza su orden, debiéndose a la aplicación que se encargue de su control. No proporciona control de flujo ni fiabilidad en las transmisiones o recuperación de algunos tipos de errores.

Y el **IP** (Protocolo de Internet) es el protocolo de la capa de red, que de alguna manera es el pegamento que tiene unida a Internet. Un datagrama (paquete) IP consiste en una parte de cabecera y otra de texto. El encabezado tienes una parte fija de 20 bytes y otra parte opcional de longitud variable. Es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes. Los datos en una red que se basa en IP son enviados en bloques conocidos como paquetes o datagramas El Protocolo de Internet provee un servicio de datagramas no fiable ya que IP no provee ningún mecanismo para determinar si un paquete alcanza o no su destino y únicamente proporciona seguridad (mediante checksums o sumas de comprobación) de sus cabeceras y no de los datos transmitidos. Por ejemplo, al no garantizar nada sobre la recepción del paquete, éste podría llegar dañado, en otro orden con respecto a otros paquetes, duplicado o simplemente no llegar. Si se necesita fiabilidad, ésta es proporcionada por los protocolos de la capa de transporte, como TCP. Las cabeceras IP contienen las direcciones de las máquinas de origen y destino (direcciones IP), direcciones que serán usadas por los enrutadores (routers) para decidir el tramo de red por el que reenviarán los paquetes.

La capa de transporte, como ya vimos, sirve como una interfaz entre las aplicaciones de la red y la propia red, y ofrece un método para direccionar los datos de la red hacia una aplicación en particular. En el sistema TCP/IP, las aplicaciones pueden direccionar los datos ya sea a través del modulo del protocolo TCP o del modulo del protocolo UDP, utilizando números de puertos.

Un **puerto** es una dirección interna predefinida que sirve como una ruta de acceso para una aplicación determinada. Por ejemplo, una computadora cliente comúnmente contacta una aplicación FTP en un servidor por medio del puerto TCP 21.

Una aproximación al esquema de direccionamiento especifico de la aplicación en la capa de transporte, revela que los datos del TCP y del UDP son direccionados a lo que se llama un **socket.** Un socket es una dirección formada por la unión de la dirección IP y el número de puerto. Por ejemplo el socket 192.168.9.1:5021

La **programación de sockets,** fueron desarrollados en los años ochenta en entorno de Unix como la interfaz de sockets de Berkeley. Básicamente, un socket permite la comunicación entre un proceso cliente y un proceso servidor y puede ser orientado a conexión o no orientado a conexión. Un socket cliente en una computadora utiliza una dirección para llamar a un socket servidor en otro computador. Una vez que se han enlazado los sockets apropiados, las computadoras pueden intercambiar datos.

Normalmente las computadoras son socket servidores mantienen un puerto TCP o UDP abierto, preparado para llamadas de entrada eventuales. El cliente generalmente determina la identificación del socket del servidor deseado.

Los sockets pueden crearse desde un lenguaje de programación como C o Java, permitiendo al programador proporcionar fácilmente funciones y aplicaciones de red, que nos servirán para desarrollar de una manera eficaz nuestro proyecto.

Capítulo 4:

Diseño Conceptual.

4. Diseño Conceptual.

4.1 Propuesta inicial.

Analizando toda la información que hemos recopilado, podemos observar que la mayoría de los sistemas domóticos actuales, cuentan con una unidad central (controlador o servidor) y de esta unidad central se desprenden varios sensores y actuadores que complementan el sistema que controla y monitorea una casa u oficina.

Actualmente, hay varios sistemas domóticos que tienen ciertas ventajas y desventajas, como vimos en capítulos anteriores, sin embargo, basándonos en nuestro objetivo principal, podemos realizar la propuesta de utilizar al máximo los recursos de una computadora personal, que hoy en día en la mayoría de los hogares u oficinas existe. Para aprovechar al máximo esta PC tenemos que pensar en no hacer grandes modificaciones a su hardware para no multiplicar el costo de nuestro proyecto y además para que sea fácil su instalación del sistema en cualquier casa u oficina.

Por lo tanto, tenemos que definir las necesidades y requerimientos principales que el sistema debe cumplir para satisfacer nuestro objetivo y lo expuesto en párrafos anteriores.

4.2 Necesidades principales.

En este apartado listaremos los puntos principales en los que el sistema deberá fundamentarse, para de aquí ir descubriendo las especificaciones precisas para la construcción de nuestro proyecto.

1. **Utilizar la PC como dispositivo de control y comunicaciones central:** En la propuesta inicial se menciono que resulta muy útil el utilizar la PC, debido a que se encuentra en la mayoría de las casas y oficinas que necesitan controlarse o monitorearse remotamente. Por lo que se procurara que el sistema este centralizado en la PC tanto para controlar los dispositivos conectados al puerto, como para comunicarse a Internet y actualizar los estados de estos dispositivos. En cuanto al hardware de la PC, no se le harán muchas modificaciones, mas bien se le conectara de manera externa hardware elemental, que haga que el control y la comunicación de la PC con los dispositivos sea eficaz y barata. En tanto al software se hará una interfaz grafica que facilite al usuario ingresar y recuperar la información. Además el programa el software diseñado para el control y

comunicación deberá correr en la computadora de tal forma que no impida al usuario realizar paralelamente otras tareas comunes en la PC, es decir, la computadora no deberá estar dedicada solo al programa domótico.

2. **Aprovechar la infraestructura de las casas sin grandes modificaciones:** Para que un sistema de este tipo sea fácil de implementar en un escenario real, la instalación de los sensores o actuadores en la casa u oficina deberá ser sencilla y barata, normalmente lo mas complicado es llevar los cables desde el sensor hasta la PC y de esta a los actuadores, para solucionar este problema el sistema sólo requerirá de líneas de comunicación de baja potencia lo cual implica cables muy delgados y fáciles de instalar.

3. **Hacer el diseño del software independiente de la forma de conexión a Internet y la plataforma donde se corra el sistema:** En la actualidad la forma más común para comunicarse a Internet es a través de la línea telefónica , sin embargo, poco a poco los accesos de banda ancha han ido ganando terreno, tanto por el descenso del precio como por la disponibilidad del servicio que es mas estable, así que diseñar un sistema que este atado a algún tipo de conexión en especial o algún tipo de plataforma, daría como resultado una aplicación no portable y también no eficiente de acuerdo al objetivo de esta tesis. Por este motivo el sistema que se implementara aquí, no estará atado a algún tipo de conexión a internet, simplemente que la conexión este basada en el protocolo TCP/IP. Tampoco estará atado a alguna plataforma en especial ya que el software será multiplataforma.

4. **Utilizar herramientas de desarrollo libres y eficientes:** Ya que es parte de este objetivo diseñar un sistema económico y con posibilidades de expandirse sin necesidad de tener que pagar por la tecnología empleada, la utilización de software libre en la parte del desarrollo es fundamental. Por lo tanto se requiere de un lenguaje de programación que pueda manejar el hardware de la PC de manera sencilla y eficiente, pero a la vez este lenguaje también tiene que implementar de de alguna forma la comunicación entre computadoras con algún modelo estándar (TCP/IP). La respuesta trivial para este requisito seria el lenguaje C pero lamentablemente este lenguaje no maneja interfaces graficas y amigables para el usuario, por lo que esta no es una solución viable. Por tal motivo se pretende diseñar el software del servidor y el cliente en el lenguaje de programación orientado a objetos Java, ya que cubre todas nuestras expectativas. Además contamos con un sin números de API's que están disponibles en este lenguaje para diseñar de forma libre, atractiva y eficiente nuestro sistema. También hay que destacar que se cuenta con una

infinidad de información de este lenguaje para facilitarnos las pruebas y posibles mantenimientos que se realicen.

5. **Emplear conectores y cables estándar:** De esta forma facilitaremos la posible expansión o modificación del proyecto, permitiendo reparar o incluso adicionar módulos al sistema. Además de continuar con la postura de nuestro objetivo, en la que decimos que nuestro sistema sea económico.

Con los cinco puntos anteriores podemos ir definiendo un esquema general que pueda ir conformando lo que será el sistema domótico de manera integral y a también ir construyendo los módulos que lo conformarán, para posteriormente entrar al diseño de cada uno de estos módulo de manera conceptual, siguiendo las ideas fundamentales que hemos estado planteando a lo largo de los capítulos anteriores.

4.3 Esquema general.

De forma general el sistema esta integrado por una interface que se conecta a la computadora a través de uno de los puertos de comunicaciones locales (Paralelo: por las características del puerto y nuestras necesidades), la cual recogerá señales de algunos sensores dispuestos en la casa u oficina, también podrá activar o desactivar actuadores que están conectados a aparatos de uso común (lámparas, ventiladores, cerraduras eléctricas, persianas, etc.). La PC a la cual se conecta esta interface correrá un programa que la controlara y además podrá comunicarse a través de Internet con otras computadoras para notificar y controlar el cambio de estado de la interface. Esta ultima computadora que funciona como cliente, correrá un programa que de igual forma se comunicara a Internet para solicitar los datos que muestran el estado de la interface y además puede capturar instrucciones dados por el usuario para modificar el sistema, estos datos los deberá manipular lo suficiente para que puedan ser enviados por la red al programa servidor que es justamente el programa que corre en la computadora que tiene conectada físicamente la interface.

De esta manera podemos ver que el proyecto se va dividiendo en varios módulos que se muestran en la figura 41

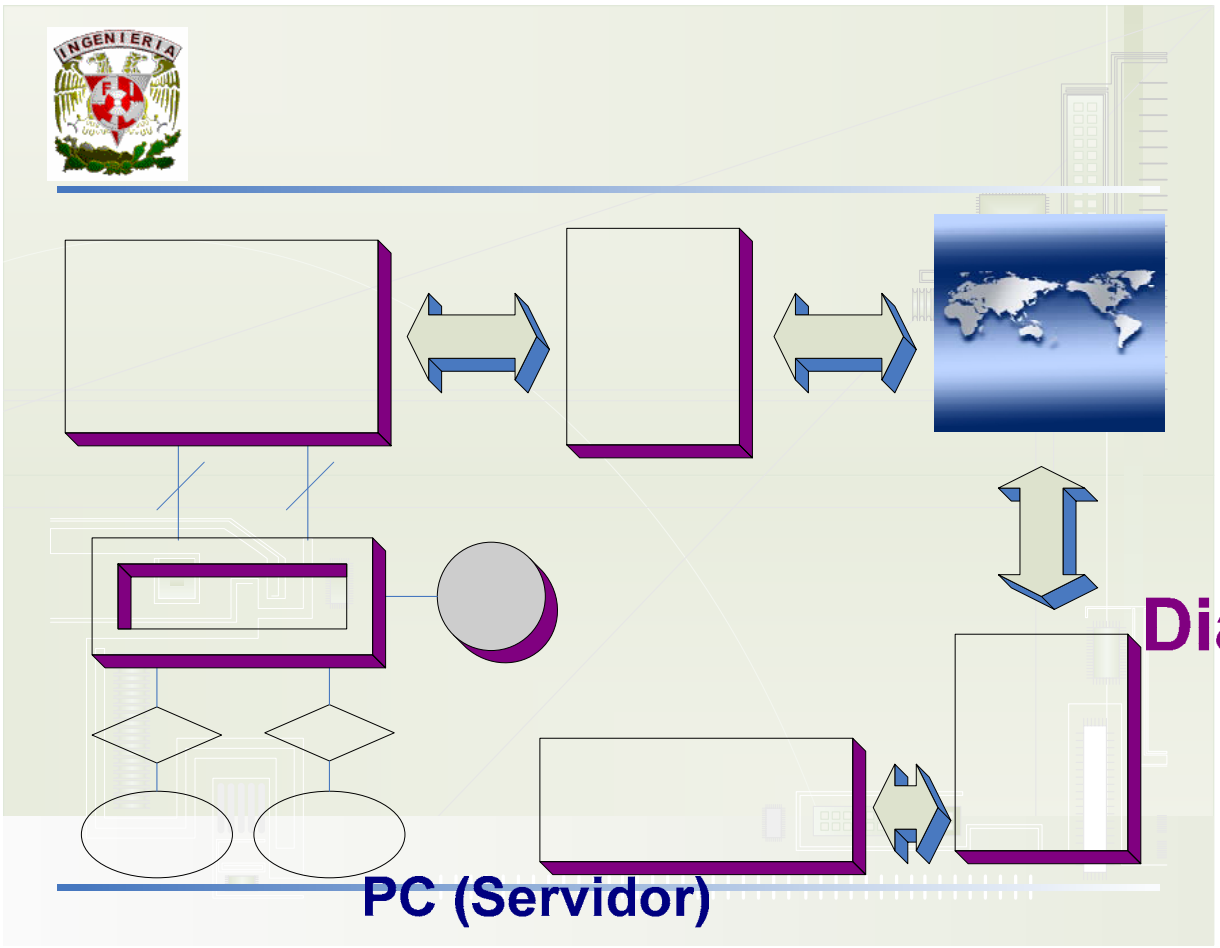


Figura 41

4.4 Análisis por módulos

Δ Driver de la Interfase
Δ Socket TCP/IP
Δ Monitor Local

Para analizar el sistema en módulos es necesario distinguir dos ramas principales, una será el software y el hardware necesario en la PC-Servidor, encargada del control físico de la interface, y por otro lado estará el software necesario para el lado de la PC-Cliente, que se encargara de conectarse de manera remota para visualizar la información o para realizar una modificación en el estado de la interface.

Puerto Paralelo

4 Entradas

8 Salidas

Por lo tanto, a la rama izquierda de la figura 41 le llamaremos **servidor** debido a que esta computadora estará encargada de escuchar las peticiones de la red y en caso de recibir información valida realizara una acción en la interface conectada al

Interface

Sensor

Rele

Rele

puerto paralelo. A la segunda rama del lado derecho le denominaremos **cliente** y en general será el programa que interactué con el usuario mostrando información de los estados de la interface y tomando órdenes para enviarlas al servidor y cambiar así los estados de esta interface.

4.4.1 La interface y el puerto de conexión.

Basados en la investigación de los diferentes puertos de comunicaciones del capítulo dos es importante apreciar que los puertos de comunicaciones que ofrecen una opción más viable para el actual proyecto son el puerto USB y el puerto paralelo, el primero e*s* un puerto de uso general que permite transferir información a ,velocidades relativamente altas y que se puede conectar a una gran cantidad de dispositivos, sin embargo, hay complejidad para la programación de un controlador, debido a que aun es escasa la información acerca de la implantación de este puerto en hardware y la necesidad de incluir un microcontrolador o un dispositivo externo a la PC en la interface para recibir y transmitir la información por este puerto no parece indicarnos que sea la mejor elección, así que considerando las necesidades reales y los alcances del proyecto podemos decir que el puerto paralelo nos puede ayudar a realizar las tareas necesarias para la adquisición de las señales y la excitación de los actuadores.

El puerto paralelo aunque viejo y lento en comparación a los puertos actuales ofrece varias ventajas que a continuación se enlistan para ver la utilidad de este puerto en nuestro objetivo planteado:

1. **Compatibilidad con prácticamente todas las PC:** Desde la PC original de IBM todas las computadoras de este tipo cuentan con un puerto paralelo, recientemente y como se menciona en el capítulo 3, se han desarrollado modos de comunicación más completos que el original, permitiendo ahora trabajar en modo bidireccional y controlar mejor los errores, sin embargo el protocolo original define 8 bits de salida individuales y 4 de entradas, las cuales son suficientes para realizar la interface de esta tesis. Por lo tanto es recomendable utilizar el puerto paralelo en la forma mas básica para garantizar la compatibilidad con cualquier PC en la que podríamos conectar nuestra interface sin modificación alguna. Además es importante destacar que en caso de que sea necesario aumentar la cantidad de bits de entrada o de salida, se podría recurrir a un circuito multiplexor externo que aumente las salidas hasta 256 y las entradas hasta 16.

2. **Facilidad de programación:** En muchos lenguajes de programación existen funciones con las cuales se puede establecer comunicación con el puerto para hacer escritura o lectura de datos sin mucha complejidad, específicamente en el lenguaje “C” estándar, existen dos funciones llamadas inb() y outb que permiten leer y escribir de un segmento de memoria

especifico, si se cuenta con los permisos adecuados. Otro lenguaje de programación que nos permite esto es C++, con sus dos funciones inp() y outp(). Sin embargo, el lenguaje que nosotros vamos a utilizar para este proyecto es Java, debido a las características que mencionaremos en el tema 4.4.2, y aunque no cuenta directamente con 2 funciones que realicen esta tarea, se puede fácilmente implementar una clase que las contenga.

3. **Compatibilidad TTL:** A nivel de electrónica este puerto ofrece una salida TTL estándar, esto quiere decir que trabaja con 5 VCC aunque ofrece muy poca corriente, alrededor de 20 mA máximo, así que no puede manjar directamente dispositivos que requieran potencias altas, para esto se necesita que la interface amplifique la señal del puerto. Otra ventaja es que el estado de los bits se mantiene hasta que no se realice un cambio por parte del programador, esto es muy útil debido a que de esta forma no tenemos que refrescar el valor constantemente.

4. **Conexión física genérica**: El conector usado tanto en la PC como en el otro extremo, es de tipo común y barato (DB25) y nos permite fabricar cables económicos y sin necesidad de blindarlos, garantizando buenos resultados de comunicación hasta 10 metros.[12]

Ya que el puerto que elegimos tiene 8 líneas de salida y 4 de entrada, debemos pensar que aparatos son los que queremos que se conecten y que tipo de señales nos tendrán estos aparatos para saber que señales les tengamos que enviar para activarlos o desactivarlos.

Considerando un ambiente domótico básico podemos pensar en varios elementos de salida y entrada indispensables y el tipo de salida que ofrecen en el caso de los de entrada:

Salidas:

- ✓ Lámparas de cualquier tipo encontradas comúnmente en una casa.
- ✓ Ventiladores, calefactores o aire acondicionado.
- ✓ Motores (bomba de agua, cortineros, persianas, puertas automáticas).

Entradas:

- ✓ Interruptor de contacto (puertas y ventas) → Digital
- ✓ Sensor de movimiento → Digital
- ✓ Sensor de luminosidad solar → Analógico

[12] http://es.wikipedia.org/wiki/Cable_paralelo
http://www.interfacebus.com/Design_Connector_1284.html

- ✓ Sensor de temperatura ambiente → Analógico

En el caso de los dispositivos de salida, todos requieren de una cantidad importante de potencia mucho más grande que la que el puerto ofrece, por lo tanto en todos los casos requerimos un modulo de potencia que estará integrado a la interface, este estará formado principalmente por un circuito basado en un transistor que trabajara en modo de corte – saturación el cual excitara a un relevador, el cual finalmente funcionara como un interruptor para los aparatos que conectemos a él, además vamos a agregar una etapa de protección para el puerto paralelo, ya que este es muy susceptible a picos de voltaje que se puedan originar por la conmutación del relevador o por demandas de corriente superiores a las que el provee.

Un ejemplo general del circuito que puede ser empleado para la etapa de potencia de la interface es el siguiente:

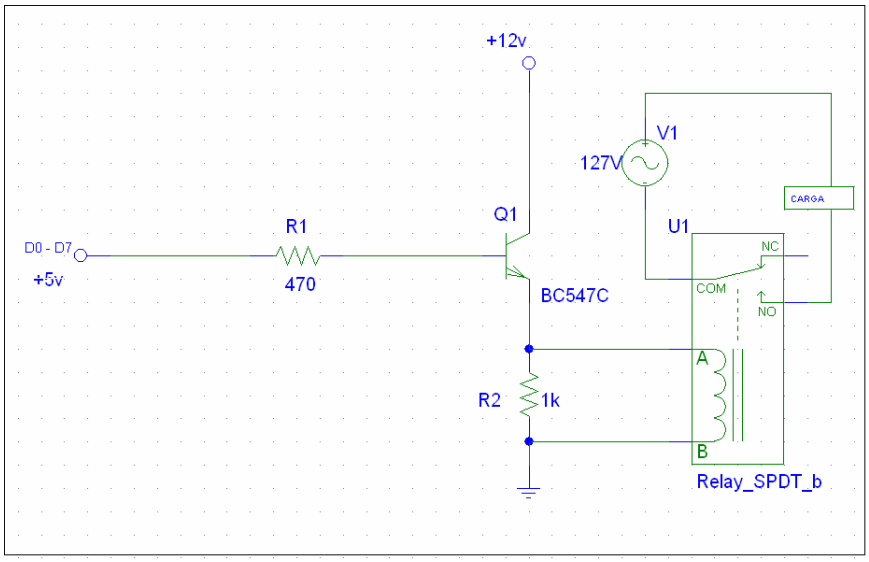


Figura 42. Ejemplo de circuito de salida para el puerto paralelo

Para los dispositivos de entrada existen alternativas para poder recibir los datos, en el caso de las entradas digitales, la señal puede ser conectada casi directamente a uno de los bits del puerto, solo se deberá considerar un pequeño circuito para acondicionar la señal al voltaje y la corriente necesarios para excitar al puerto y no quemarlo. Para las entradas analógicas y pensando que en principio no se requerirán altas tasas de actualización de datos, ni precisiones muy altas ya que la conexión a través de Internet nos impiden garantizar una repuesta rápida, podemos acoplar un convertidor análogo – digital al mismo puerto paralelo, es deseable que se utilicen el menor número de bits posibles del puerto para que los demás se pueda emplear para el resto de las señales digitales, por esta razón, otra alternativa muy atractiva para este tipo de sistemas son los convertidores análogo – digital serie, que como su nombre lo indica transmite los datos resultantes de la

conversión, de manera serie, esto quiere decir que solo utiliza una línea de transmisión que es modulada de tal forma que la secuencia en la que la señal sube o baja de voltaje, indica la transmisión de un bit, por supuesto, el trabajo de la interpretación de esta modulación en la línea corre a cargo del programa que este leyendo el puerto, sin embargo, de manera física se simplifica considerablemente la cantidad de bits necesarios para la conversión.

Para las señales digitales podemos considerar el siguiente circuito de la figura que acondiciona la señal de entrada para los bits de entrada del puerto (registro de control).

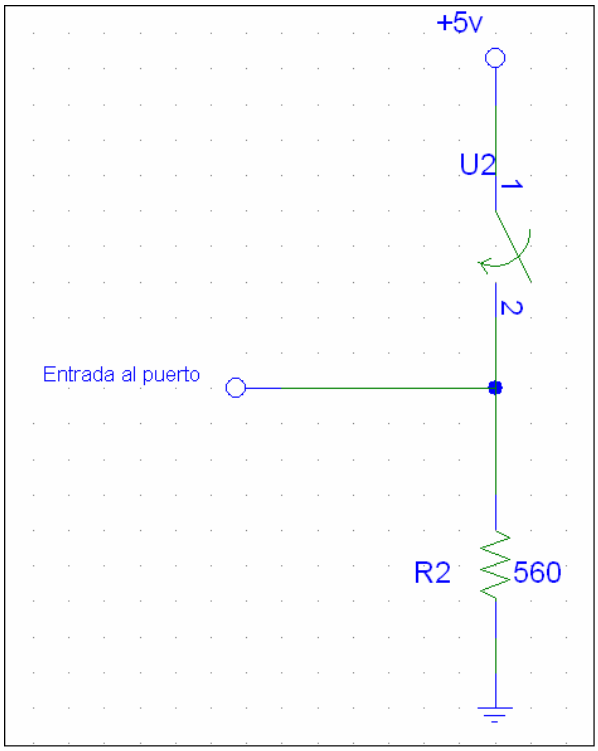
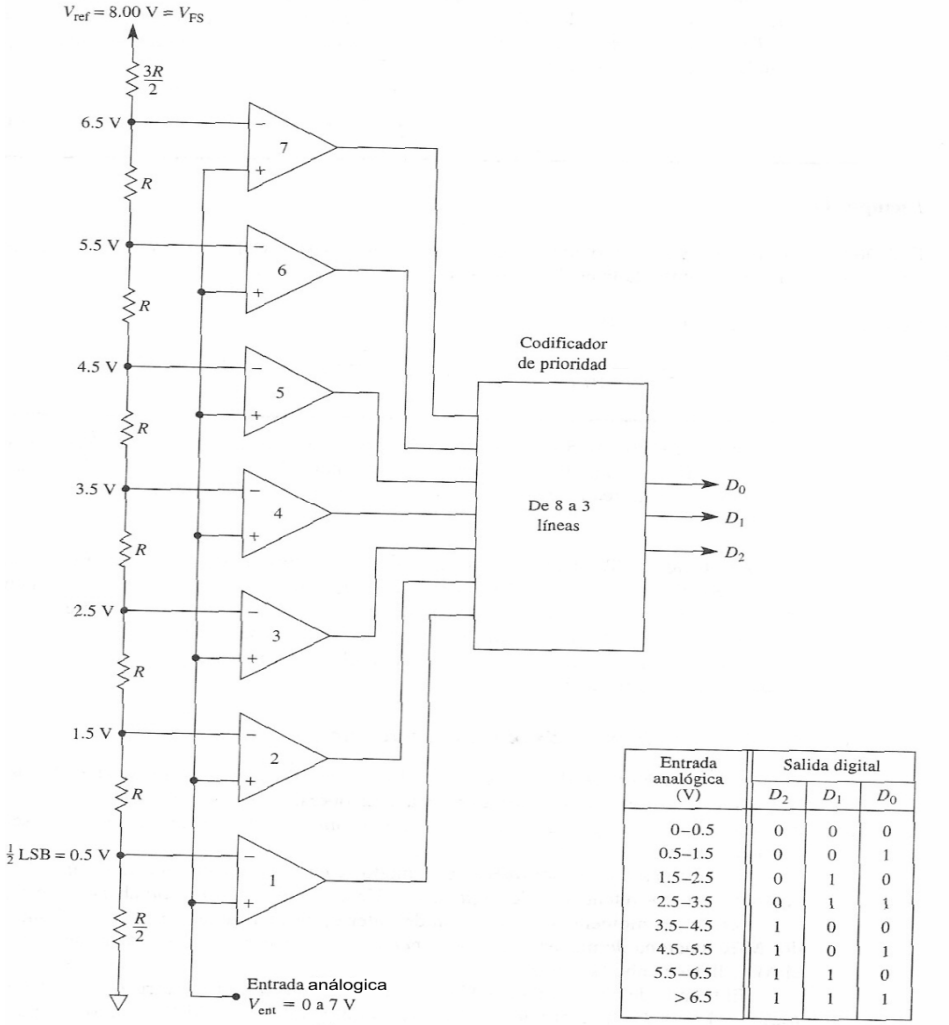


Figura 43. Ejemplo de un circuito de entrada

Como vemos, el circuito es muy simple pero en realidad es todo lo que se necesita para que el pin del puerto paralelo se excite de manera confiable y a la vez para protegerlo. Sin embargo, el circuito propuesto para la adquisición de señales analógicas, se complica un poco mas, debido a la inclusión del convertidor análogo – digital (ADC). Como mencionamos en párrafos anteriores el ADC serie nos complica su descodificación ya que tendríamos que sincronizar al programa que decodifique los datos que llegan vía serie del ADC al programa que controla la interface conectada al puerto paralelo. Por este motivo y también por limitaciones de tiempo, dinero o técnicas, podríamos utilizar un ADC bastante práctico y rápido. Es práctico debido a que ocupa pocos bits en la conversión y no necesita de una circuitería muy complicada, por lo que es la opción ideal para nuestro proyecto de

tesis. En la figura 44 se puede observar un esquema general del ADC paralelo (tipo flash).



Entrada analógica (V)	Salida digital		
	D_2	D_1	D_0
0–0.5	0	0	0
0.5–1.5	0	0	1
1.5–2.5	0	1	0
2.5–3.5	0	1	1
3.5–4.5	1	0	0
4.5–5.5	1	0	1
5.5–6.5	1	1	0
> 6.5	1	1	1

Figura 44. Convertidor Análogo – digital (flash) de 3 bits en paralelo.

Aunque esta es la primera aproximación al hardware necesario en la interface ya se puede observar los módulos principales de los cuales consta esta parte del sistema, mas adelante se considera todos los puntos mencionados en esta sección que servirá para la implementación del prototipo de manera real.

4.4.2 El servidor.

El programa que diseñaremos con el nombre de servidor tendrá la tarea de controlar en primer lugar el puerto paralelo y en segundo lugar la comunicación con Internet, este correrá en la computadora que tenga físicamente conectada la interface al puerto paralelo.

Este programa, al igual que el programa cliente, estarán hechos en el lenguaje Java debido a las siguientes razones:

- ✓ El lenguaje Java es multiplataforma, es decir, se puede correr en cualquier sistema operativo que cuente con la maquina virtual de java. Hoy en día en la mayoría de las computadoras se encuentra esta máquina virtual del lenguaje Java.
- ✓ Nosotros podemos crear las clases que necesitemos, sin embargo, java cuanta con muchas API's que nos facilitara la programación y nos ahorrara tiempo.
- ✓ En java se pueden crear aplicaciones graficas atractivas para los usuarios.

Java cuenta con el API de comunicaciones (javax.comm), que es una extensión Standard, que nos permite realizar comunicaciones con los puertos serie RS-232 y el paralelo IEEE-1284, esto nos permitirá realizar aplicaciones de comunicaciones que utilizan los puertos de comunicaciones (tarjetas inteligentes, fax) independientes de la plataforma en la que nos encontremos.

El API de comunicaciones no se encuentra incluido en el JDK (Kit de desarrollo de software para java) y es una extensión de este, así que antes de empezar a trabajar con el, debemos instalar este nuevo API en las maquinas que vayamos a realizar el desarrollo y que vayan a ejecutar programas realizados con este API.

Sin embargo, este API está orientado para aplicaciones más robustas en las que no nos involucremos con formas de transmisión de datos para los puertos serie y paralelo, como son estos estándares en el puerto paralelo:

1) Modo Compatible.
- Modo más típico de una impresora.
- Solo se puede enviar datos hacia periférico.
- Velocidad de transferencia 150 kBps.

2) Modo Nibble.
- Permite la lectura desde el periférico de un byte en forma de dos nibbles (realizable en cualquier puerto).
- Velocidad de transferencia 50 kBps.
- Usualmente se utiliza en combinación con el Modo Compatible para conseguir una transferencia bidireccional.

3) Modo Byte.
- Solo disponible en ciertas tarjetas, pues necesita de un puerto de datos bidireccional.
- Es un modo de solo entrada que permite leer un byte de una vez (se usa en vez del modo nibble).

- Velocidad de transferencia 150 kBps.
- Usualmente se utiliza en combinación con el Modo Compatible para conseguir una transferencia bidireccional.

4) Modo EPP
- Aumenta la velocidad de transferencia a 500 kBps- 2 MBps.
- Mantiene los registros del Modo Compatible lo que permite utilizar este modo.

5) Modo ECP
- Tiene capacidad de emular todos los modos anteriores y algunos más.
- El modo EPP difiere de este en que el PC controla todas las trasferencias, desde y hacia el periférico, en cambio en ECP el periférico debe negociar protocolo y canal de retorno.

Esto nos demuestra que la aplicación de esta API a nuestro proyecto nos proporcionaría recursos innecesarios para este sistema. Por lo que implementaremos una clase mas sencilla que nos permita de manera fácil acceder a los registros del puerto paralelo y utilizarlos ya sea para escribir o leer en ellos.

Por este motivo ocuparemos la clase ParallelPort que contiene los métodos **readOneByte** (int address), que nos permitirá leer un byte de una dirección de memoria especifica y **writeOneByte** (int address, int oneByte), que escribe un byte en una dirección especifica de memoria.

Dado que la versión de Window NT y superiores, han protegido de alguna forma el acceso a los puertos de comunicaciones tenemos que implementar alguna solución para que nuestros programas puedan acceder a los registros correspondientes a puerto paralelo.

Podemos describir en varios pasos las tareas que el servidor realizará con el propósito de explicar más adecuadamente esta etapa del sistema, a continuación se enlistarán estas tareas que forman el servidor y la manera en la que serán implementadas:

- ✓ **Conexión a Internet a través de un socket**: La primera tarea que se realizará es la recepción de datos a través de Internet, esto funcionará de la siguiente forma. Primero abriremos un socket de red en el puerto TCP 6543 que “escucharán” la red todo el tiempo esperando una petición de parte de un cliente, una vez que se recibe, *el mensaje se descompone en dirección IP origen, tamaño del mensaje y mensaje, de esta manera podemos validar a través de la IP la computadora que solicita algo al servidor, podemos además comprobar que el mensaje esté completo verificando el tamaño del paquete y por ultimo tenemos el mensaje que pasaremos al intérprete de*

comandos. Es bueno mencionar que como última etapa de la comunicación con la red, el socket nos servirá para enviar al cliente de nuevo el mensaje completo, con la finalidad de verificar no solo que el mensaje llegó, sino de que el comando solicitado fue ejecutado correctamente.

- ✓ **Interprete de comandos**: Lo que haremos en esta parte es tomar el mensaje proveniente de la red y lo consideraremos como una línea de comando dada por el cliente y estará separada en sus diferentes componentes que la van a formar, esto es, comando, argumento 1 argumento 2, etc. Luego tendremos una serie de opciones disponibles para el comando, como son: “apagar”, “prender”, “Desc”, “ayuda”, “guarda”, “recupera”, etc. Dependiendo del comando que sea obtenido por el interprete, se entra a una rutina que dará servicio a esa petición utilizando adicionalmente los argumentos, por ejemplo, si la línea de comando es “desc 1 Lamparaf” esto es interpretado así: “desc” es el comando que cambia la descripción del aparato que está conectado a un bit y requiere de dos argumentos para funcionar, el primero en este caso es el “1” que indica el bit al que deseamos cambiarle la descripción y el segundo es la cadena de texto de descripción en este caso “Lampara1”. De esta manera el programa al reconocer el comando “desc” entra a una función que cambia el texto descriptivo asociado a el bit numero 1. Algunos de los comandos utilizan uno o dos argumentos y solo algunos son usados por el servidor como: prende, apaga, refresca y los demás son más bien de uso del cliente que es quien ofrece más información al usuario, relativa a lo que tenemos conectado en el sistema, mas adelante en la sección que describe al cliente se hablará más de esto.

- ✓ **Manejo del puerto paralelo**: Esta sección del servidor solo es ejecutada cuando se solicitan los comandos “prende” y “apaga” y como su nombre lo indica sirve para prender o apagar uno o varios bits del puerto dependiendo de la máscara de bits que deseamos modificar y solo hacen referencia a los bits de salida que son lo que podemos alterar por software (como vimos en el capítulo 3, los bits de entrada solo cambian de estado por hardware, en otras palabras son de solo lectura para el programador), la manera en la que la máscara modifica al estado es la siguiente:

$$\begin{array}{ll} & 01001000 >> Estado\_Original \\ \text{OR} & 00100000 >> Mascara\_(Bit6) \\ \hline & 01101000 >> Bit6\_Encendido \\ \text{XOR} & 00100000 >> Mascara(Bit6) \\ \hline & 01001000 >> Estado\_Original \end{array}$$ Después de prender y apagar el bit 6

Tabla 8. Operación lógica que modifica el estado de los bits del puerto paralelo

4.4.3 El cliente.

Una vez que hemos analizado el servidor, pasamos a ver la última etapa del sistema pero que sin duda es con la que el usuario tendrá más relación, estamos hablando del programa que mostrará la información del estado de todos los bits de puerto paralelo y que además servirá para ingresar comandos que serán enviados al "servidor" y modificarán el estado de la interface o de la información mostrada en la pantalla, a este programa le llamaremos en lo posterior "cliente", este correrá ya sea en la computadora donde corre el servidor o bien en cualquier otra que cuente con conexión a Internet y con el cualquier sistema operativo que cuente con la maquina virtual de java. Como se menciono en el tema 4.4.2 este programa estará hecho en Java.

Este programa tendrá una estructura similar a la del servidor en lo que respecta a la conexión de a Internet y al intérprete de comandos, sin embargo, cuanta con características especiales que lo dividirán en varias etapas las cuales se enlistan y explicaran a continuación:

- ✓ **Monitor del puerto paralelo:** La función principal del cliente será ofrecer la información acerca del estado del puerto paralelo así como de que dispositivos están conectados físicamente a cada uno de los bits del puerto. La idea es tener en una sola pantalla todo la información necesaria para saber qué pasa con los aparatos que conectamos y así tomar decisiones de prende o apagar algún dispositivo. Básicamente tendremos en cada renglón, la información del estado del bit, su número y la descripción del aparato que está conectado a él, esto será para los 8 bits de salida como los 4 de entrada, como hemos mencionado anteriormente.

- ✓ **Interprete de comandos:** El usuario del sistema encontrará en esta sección del cliente una manera de interactuar con el sistema a través de comandos que estarán ligados a botones amigables para el usuario, estos son de varios tipos, unos son para encender o apagar un bit, otros son informativos, cambian la descripción del aparato conectado o refrescan la información pidiéndola al servidor, otros son para guardar o recuperar información en archivos de texto como las descripciones de los bits o el estado de estos. Los comandos ingresados a través de darle clic a un botón, son empaquetados para ser enviados al servidor, este sabemos que ejecutará el comando si es válido y devolverá la misma línea al cliente a manera de comprobación, una vez que el cliente recibe esta información, la utiliza para actualizar la pantalla y así cerrar el ciclo de un comando.

- ✓ **Conexión a Internet:** En el punto anterior señalamos que la cadena de texto que contiene al comando es enviada al servidor a través de la red, este mensaje es encapsulado por parte del sistema operativo a petición del

socket programado en nuestro cliente, en realidad la comunicación se realiza en ambas direcciones, primero se empaqueta el mensaje y se envía al servidor completado por la información necesaria para su enrutamiento, como la IP origen, destino y el número de puerto, después, en sentido inverso se recibe un paquete proveniente del servidor que contiene la misma información si es que no ocurrió error al ejecutar el comando, estas tareas de enviar y recibir son establecidas por el socket incrustado en el programa cliente.

4.5 Resumen.

En este capítulo se establecen las bases principales de lo que será el diseño general del sistema, vemos que está conformado por software y hardware, estas dos partes son a su vez divididas en otras etapas que forman las características principales de la composición del sistema, en el resto de esta tesis nos referiremos constantemente a este capítulo ya que es la parte que define las funciones del sistema que nos servirán para cumplir con nuestro objetivo.

Capítulo 5:

Diseño y construcción del prototipo.

5. Diseño y construcción del prototipo.

En este capítulo iniciaremos el trabajo que definirá el prototipo que se realizara para comprobar que es posible construir un sistema domótico elemental con software libre y con pocas modificaciones al hardware de una PC común. Esta versión del sistema se basa en la separación por módulos hecha en el capitulo anterior aunque no seguirá fielmente la definición original, se tratara de poner la mayoría de las características dichas anteriormente, ya que al ser este un circuito y software real, nos vemos influidos por las limitantes impuestas por los medios físicos, materiales y económicos. Por esto el prototipo se plantea como un elemento comprobador de las ideas expuestas en el diseño conceptual, mas no un producto comercial que requiera una infraestructura e inversión que estén fuera del alcance de este trabajo, no obstante, el concepto puede ser usado en algún momento para crear un sistema capaz de cumplir con los requerimientos actuales del mercado.

A continuación se plantea la manera en que cada uno de los módulos se realizara, incluyendo las especificaciones esperadas y la explicación de cómo se decidió hacerlo. Iniciaremos en el mismo orden usado en el capitulo anterior, empezando por la interface para seguir después con el servidor y el cliente.

5.1 Interface puerto paralelo – aparatos.

Basados en el diseño conceptual podemos decir que esta interface será usada conectar aparatos eléctricos comunes en nuestras casas u oficinas con los bits del puerto paralelo, básicamente lo que se requiere es ofrecer un circuito de potencia para los bits de salida, que permitan que la pequeña señal tanto en voltaje como en corriente que se obtiene del los bits del puerto, sea la encargada de excitar a los relevadores necesarios para encender o apagar un aparato que funcione con el voltaje y frecuencia nominal presente en una casa u oficina común (127 VAC 60 Hz) y con margen de potencia suficiente para aplicaciones sencillas (1000 watts). Además para los bits de entrada, la interface, nos debe de ofrecer una señal de 5V y con una corriente de alrededor de 10mA, para que sea ingresada al puerto sin peligro de dañarlo pero con la seguridad de que sea notado por el hardware de la computadora.

Por su organización, se separara en varias etapas el circuito que formara la interface, una será la encargada de los bits de salida, otra de los de entrada y finalmente una fuente de alimentación externa que alimente al circuito y que proveerá la potencia para los relevadores, sin que esta sea demandada de puerto de la computadora.

5.1.1 Bits de salida

Como ya se dijo en el capítulo 2, el puerto paralelo tiene 8 bits de salida, electrónicamente estos son salida TTL estándar que producen 5Vcc y entre 5 y 20 mA de corriente máxima, todas estas salidas tienen lógica positiva, esto quiere decir que si el bit esta en “1” la salida estará en 5V y para el estado “0” la salida estará en 0V, otra característica importante de este puerto es que el estado de los bits es manteniendo hasta que no se realice un cambio por software, en otras palabras tiene “memoria”, esto último nos ayuda mucho ya que así no se requiere que programa este constantemente actualizando el valor de los bits.

En las referencias técnicas que describen el uso de puerto paralelo, constantemente se encuentran recomendaciones para tener cuidado de cuanta corriente se toma del puerto, es muy importante nunca sobrepasar el límite de los 20mA, de no ser así es muy probable que el puerto se dañe, así que el objetivo más importante del circuito de salida será controlar esta corriente para no exceder el limite mencionado.

Por otro lado, usaremos un circuito basado en un transistor, que se conectara entre el puerto y una relevador, en modo de corte y saturación para tener una corriente del suficiente para excitar un relevador típico que sea el que maneje el voltaje y potencia nominales de la red eléctrica estándar.

Este circuito será igual para cada uno de los ocho bits de salida y a continuación se presenta el circuito.

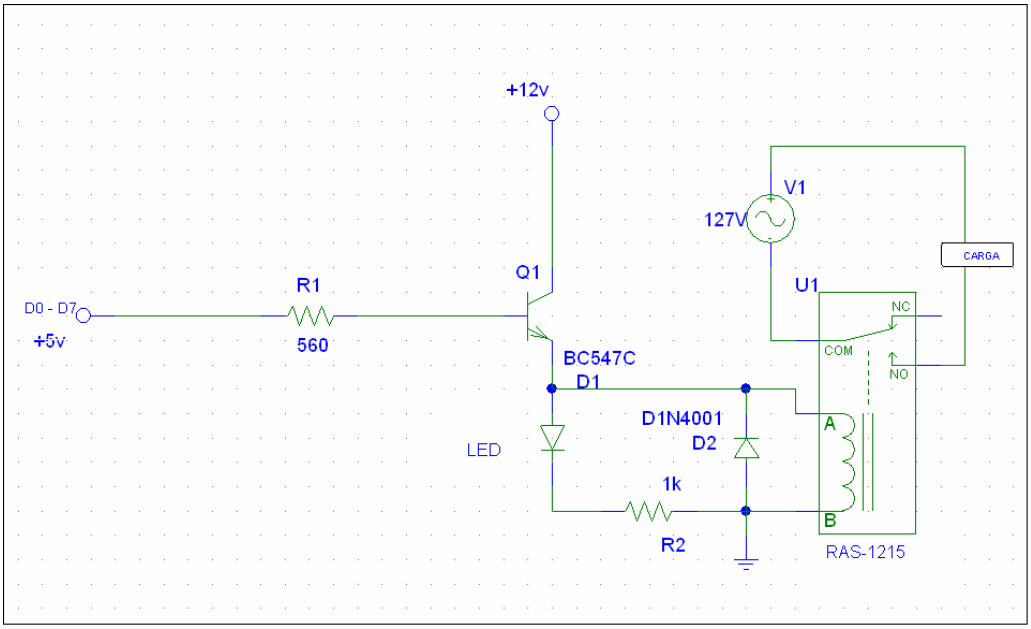


Figura 45. Circuito de salida para un bit del puerto paralelo.

Como vemos en el circuito, la salida de puerto paralelo entrara a través de una resistencia de 560ohm que limita la corriente, a un transistor común BC547C, este estará conectado de tal forma que funcione en modo de corte o saturación, de pendiendo del valor de voltaje que reciba del puerto paralelo, así en caso de que el voltaje sea 5V en la entrada del circuito, el transistor estará en saturación y permitirá el paso de corriente entre su colector y el emisor, esto hará pasar corriente por el relevador y lo activara, por otro lado también pasara corriente para la resistencia de 1Kohm y el led que servirá para indicar la activación del circuito. También es importante notar que se tiene un diodo 1N4001 conectado en posición inversa a la de la polarización del relevador, esto es con la finalidad de proteger al circuito y en especial al transistor de las corrientes parasitas que se presentan al conmutar las terminales del relevador, cuando la polarización está dada por la acción del transistor el diodo se opone al paso de la corriente proveniente de la fuente de poder. De esta manera, la única ruta disponible para cerrar el circuito es a través del relevador, sin embargo, al momento en que el transistor entra en corte e interrumpe el paso de corriente a través de si colector y emisor, el relevador se desenergiza y con esto se presenta el pulso de corriente en sus terminales de alimentación de la bobina, esta corriente siempre será inversa a la de la polarización normal, de esta manera, el diodo se presenta como un camino de baja resistencia para esta corriente, en otras palabras, el relevador estará en corto circuito con el diodo, para que así la corriente generada por la conmutación del relevador se elimine y nunca pase al transistor.

En este caso se emplea un relevador de 12Vcc con capacidad de conmutación de 127Vac a 15ª, que nos permite encender o apagar aparatos caseros pequeños, sin embargo, existen una gran variedad de relevadores que también se alimentan con 12Vcc pero que soportan cargas de conmutación mucho mayores. Para fines ilustrativos, el relevador utilizado es suficiente.

A continuación se muestra una lista de los componentes necesarios para construirle circuito y sus características básicas, en todos los casos existe una versión que puede trabajar con valores de voltaje y corriente mayores si es que es necesario hacer un circuito que maneje potencias más grandes.

- 1 Resistencia de 1 Kohms 1/2watt
- 1 Resistencia de 560 ohms 1/2watt
- 1 Transistor BC547C
- 1 Led
- 1 Relevador RAS-1215 (15A / 120 Vac)
- 1 Diodo 1N4001

Protección adicional

Con el fin de hacer mas confiable y seguro el circuito para la computadora, es posible agregar al circuito anterior un modulo adicional de protección de sobre corriente que consistirá en un circuito NTE3044 que es un optoacoplador con salida de transistor tipo Darlington, este se colocara entre la salida del puerto y el transistor BC547. Este circuito aislara al puerto paralelo del circuito de potencia, con el fin de protegerlo, también podría sustituir al transistor BC547C pero no lo hacemos para no forzar al NTE3044 al pedirle demasiada corriente.

La fig.46 será el circuito modificado con la protección adicional.

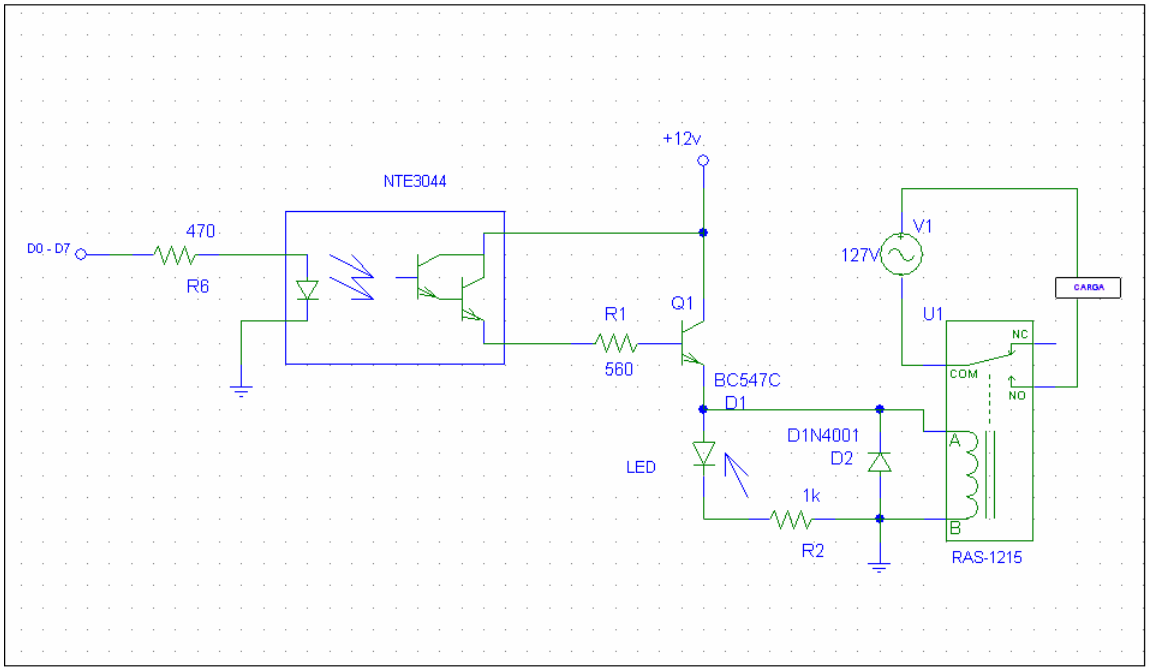


Figura 46. Circuito de salida modificado con protección para el puerto paralelo.

5.1.2 Bits de entrada

Para los bits de entrada digital mencionamos antes que se requiere un circuito que garantice un voltaje de 5Vcc y alrededor de 10mA que será aplicado al puerto para excitarlo, este circuito es un poco mas simple que el de salida, consiste en una resistencia de pull-up conectada a una fuente de 5Vcc regulada. El puerto paralelo posee 4 bits de entrada definidas de fábrica, aunque hay otros bits de control que pueden configurarse para servir tanto de entrada como de salida. En el caso de los bits definidos de fabrica que sirven de entrada, el estado de estos bits es modificable únicamente por hardware, el software solo se dedicara a leer los datos del registro del puerto designado para los bits de estado (STATUS), el circuito quedaría como el de la Fig. 47.

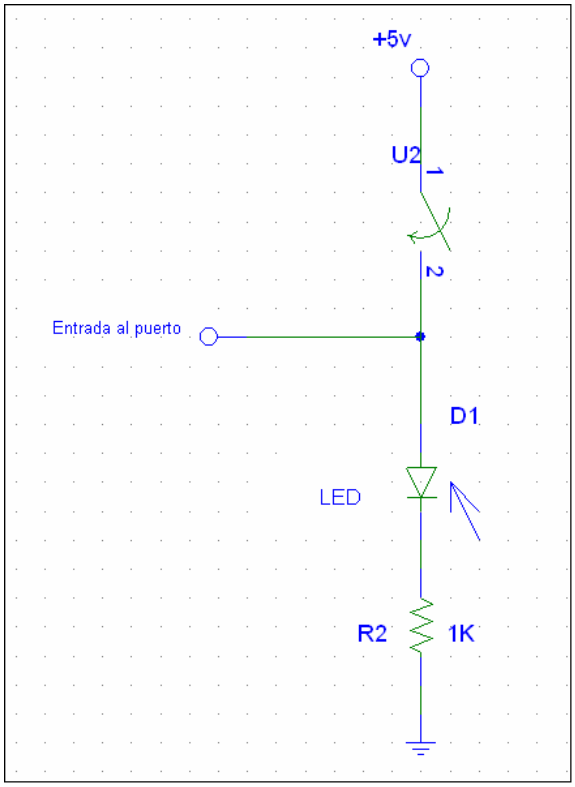


Figura 47. Circuito de entrada propuesto para el sistema.

En el circuito podemos apreciar un switch de contacto que cierra o abre el circuito, si este está cerrado circula corriente por la resistencia de pull-up y por lo tanto el bit del puerto paralelo se polariza en 5Vcc, si el switch se abre, la corriente se interrumpe y el bit se va a 0Vcc.

5.1.3 Fuente de alimentación

La fuente de alimentación que requiere la interface deberá proveer 5 y 12Vcc regulados, y como la carga más importante serán los relevadores, en general no requerimos mas de 1A, ya que cada relevador requiere de una corriente de aproximadamente 30mA, el circuito propuesto para satisfacer estas necesidades, es una clásica fuente con regulador de estado solidó LM317 (para regular una fuente variable de 1.35 a 37Vcc pero en nuestro caso solo necesitamos 12Vcc) y LN7405 (para regular una fuente fija de 5Vcc).

El circuito quedaría como el de la fig.48.

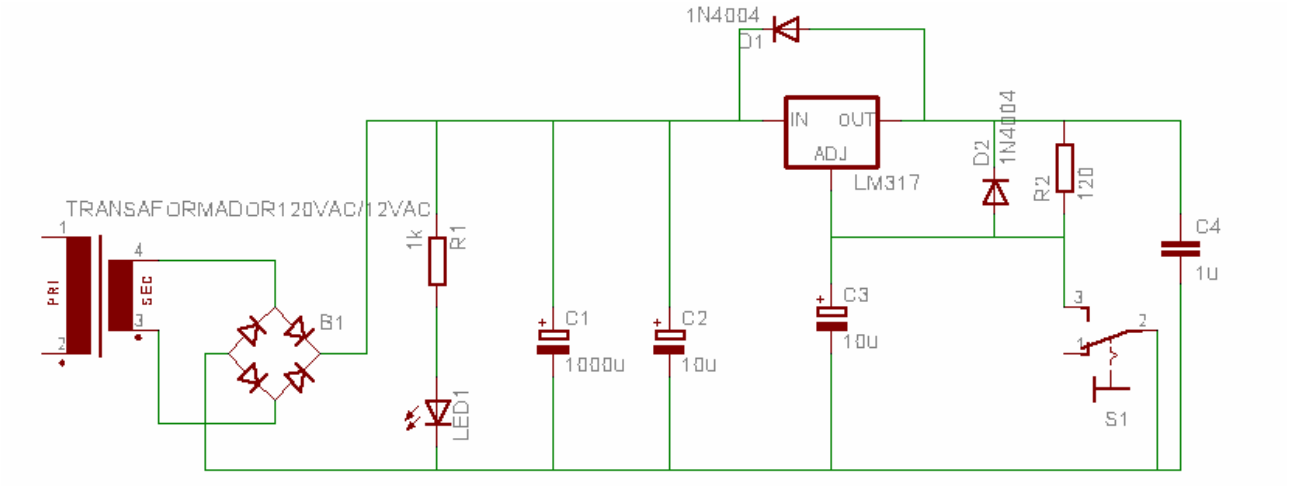


Figura 48. Diagrama del circuito de la fuente de alimentación de 12Vcc.

Podemos ver que esta fuente nos proporcionara un voltaje de 12Vcc (se puede variar) y para la fuente de 5Vcc hay dos opciones: la primera es tener otra fuente idéntica a la figura 48 pero solo cambiamos al circuito LM317 por el LN7405 y la segunda es tener a la salida de la fuente de la fig. 48 un regulador LN7405, esto hará que tengamos tanto la salida de 12Vcc como la de 5Vcc con la misma fuente de alimentación.

Vemos que la fuente inicia con la reducción del voltaje de 127Vac a 12Vac por parte de un transformador que es conectado a un puente de diodos que rectifican la señal a ciclo completo, esta señal que aun tiene un rizo, es rectificada aun mas por dos capacitores, para después pasar al regulador de estado solidó que a la salida genera una señal continua, se agrega un capacitor de tantalium para el desacoplamiento.

5.2 Programa del servidor.

Ya en el capitulo anterior describíamos las funciones básicas que este programa deberá de realizar y son en primer lugar la comunicación con Internet, el interprete de comandos y el manejo de la interface a través del puerto paralelo, en las siguientes líneas nos concentraremos en explicar como estas funciones serán implementadas.

5.2.1 Descripción general

Ahora vamos a analizar como el servidor trabaja, primero estará esperando que desde la red llegue un paquete cuyo destino sea él, una vez que esto ocurre, el socket recibe un paquete de información que es descompuesto en varias partes para determinar los parámetros de comunicación de las computadoras involucradas, como resultado del trabajo del socket obtendremos una cadena de caracteres que contienen la dirección IP o el nombre de host de origen, su longitud y la línea de comando que el usuario mando por medio de la aplicación cliente. Siguiendo esta proceso, el modulo que analiza esta cadena es el interprete de comandos, este se encargara de separar la cadena en comando y argumentos, posteriormente si el comando es válido junto con sus argumentos es pasado al manejador del puerto paralelo para que en el caso de que el comando recibido sea “prender” o “apagar” se modifiquen los bits correspondientes en el puerto. Por último el servidor enviara de regreso al cliente la misma cadena de caracteres que recibió con la finalidad de verificar que la aplicación de comando fue exitosa. En caso de que alguno de los pasos mencionado no se realice con éxito, se devolverá una cadena de error que el cliente será capaz de interpretar y cerrara la aplicación cliente.

Para el caso en el que la información provenga de la interface, por ejemplo, cuando un sensor de una ventana se abra, la ruta de la información no varía sustancialmente, el servidor al no contar con multitarea no puede estar dedicado al mismo tiempo a esperar una petición por la red y otra por parte del puerto, debido a esto, el servidor solo notificara al cliente un cambio en el estado del puerto, cuando este último se lo solicite, así cuando el cliente pida actualizar los datos, el paquete es recibido por el socket y conducido por todo el proceso que mencionamos recién. Esta característica del servidor no se debe ver como una desventaja, el cliente puede estar constantemente actualizando los datos y de esta manera detectar casi instantáneamente los cambios en el puerto, como veremos el refresco de los datos es más una tarea del cliente que del servidor.

Ahora describiremos más detalladamente el funcionamiento de los módulos que conforman el servidor.

5.2.2 Las comunicaciones con Internet

La comunicación a través de Internet la realizamos empleando los sockets, como se menciono anteriormente, los sockets no son más que descriptores de una conexión de red que nos sirven para saber a donde queremos conectarnos o donde estamos esperando una conexión. La manera en que los sockets son implementados en Java es la siguiente:

- Se importan las clases de los paquetes java.io. y java.net
- Se crea un objeto ServerSocket (ServerSocket server = new ServerSocket(12345, 100)), donde se especifica un numero de puerto disponible y después el número máximo de clientes que pueden esperar para conectarse al servidor, es decir, la longitud de la cola.
- El servidor escucha indefinidamente a que un cliente se conecte. Y para escuchar una conexión de un cliente el programa llama al método accept de ServerSocket. (Socket connection = server.accept())
- Después se obtienen los objetos OutputStream e InputStream que permiten al servidor comunicarse con el cliente, enviando y recibiendo bytes individuales o conjuntos de bytes. El servidor invoca al método getOutputStream en el objeto Socket para obtener una referencia al objeto OutputStream del objeto Socket e invoca al método getInputStream en el objeto Socket para obtener una referencia al objeto InputStream.
- El último paso consiste en cerrar la conexión, cuando se haya concluido la transmisión de datos. Esto se hace con el método close().

Como podemos ver, existe ya una convención para establecer este tipo de conexiones a red en los programas hechos en Java, así que solo es cuestión de seguir puntualmente los pasos que se acaban de mencionar para contar con una conexión a Internet muy fácil de usar, con el fin de no extenderse en estos puntos que son por su naturaliza técnica ampliamente documentadas no entraremos en detalles finos de la programación de estas funciones, sin embargo se ofrece en el apéndice de esta tesis el código fuente del servidor en donde se puede apreciar con todo detalle la manera en la que el socket necesario para este trabajo fue programado.

Al termino del proceso de conexión y recepción de los paquetes de información por parte del socket, este realizara un proceso de desencapsulamiento a través de su respectivas capas hasta llegar al último bloque que contiene la línea de comando enviada por el usuario en el programa cliente.

5.2.3 El intérprete de comandos

El siguiente paso en la aplicación servidor es el interprete de comandos, cuya función principal es la de analizar la cadena de caracteres que el socket le da, esta cadena de caracteres es una línea de comando que está formada por un comando y varios argumentos. La mayoría de los comandos solo tiene 1 o 2 argumentos pero este interprete no está limitando a reconocer una cantidad mayor de argumentos.

En primer lugar el interprete de comandos separa la cadena original en tokens que son en este caso subcadenas que no pueden ser divididas en unidades más pequeñas. La manera en la que determina que subcadenas separar es a partir de los caracteres “espacio” en algunos casos y en otros “,”. El intérprete de comandos recorre de inicio a fin la cadena buscando espacios o comas, cuando encuentra uno, todos los caracteres que estén detrás de este carácter son separados como un elemento de un arreglo de caracteres previamente creado, de esta manera, al final del proceso, la cadena original estará contenida en el arreglo de caracteres pero ahora cada elemento del arreglo será un token. El tercer elemento del arreglo será el comando y los posteriores serán los argumentos. Como se muestra en la tabla 9.

Token0	*Token1*	*Token2*	*Token3*	*Token4*	*Token(n)*
			Mensaje		
Ip origen o nombre del host.	Tamaño de la cadena.	Comando.	Argumento1	Agumento2	Argumento(n)

Tabla 9. Descomposición de la cadena que llega al servidor.

El segundo paso del intérprete de comandos será reconocer si el comando ahora alojado en el tercer registro del arreglo es válido o no. Esto se realiza pasando al comando por una serie de estructuras condicionales que determinan por comparación de cadena si el comando existe o no, en caso de que si exista, inmediatamente se llama a la rutina que da servicio a ese comando, no sin antes pasar como argumentos a esta rutina el resto de los índices del arreglo generado por el interprete de comandos. Si el comando no es reconocido, se hace una anotación en la bitácora con el fin de que en la pantalla del servidor se muestre el evento. Cabe aclarar que este tipo de errores se deben a problemas de comunicación ya que el programa cliente envía comandos predefinidos en cada botón, por lo que cada comando enviado es válido.

5.2.4 El manejo del puerto

El ultimo modulo importante que forma parte del servidor es el encargado de manejar el puerto paralelo tanto para entrada como para salida, ya en el capitulo pasado se había comentado que existen dos métodos implementados en la clase ParallelPort que nos permite leer y escribir en algún espacio del mapa de memoria de la PC, así conociendo la dirección del puerto paralelo podemos escribir y leer en el, estos métodos son **readOneByte** (int address), que no permitirá leer un byte de una dirección de memoria especifica y **writeOneByte** (int address, int oneByte), que escribe un byte en una dirección especifica de memoria.

Así en el caso del que el comando sea “prender” o “apagar”, el programa utilizara el método writeOneByte() con la misma dirección (0x378) pero con una manera diferente para cada caso, por ejemplo: si tenemos esta confirmación en los 8 bits de salida del puerto b01110110 (d118) y pedimos prende el bit 4, entonces se genera una máscara así 00001000 para que al realizar una operación OR se obtenga la siguiente configuración 01111110. En caso de querer apagar el bit 6 para ese mismo ejemplo inicial la máscara seria 00100000 para que con una operación XOR se obtenga 01010110.

El valor resultante de la manipulación del puerto es almacenado en dos variables una para la salida y otra para la entrada, las cuales serán enviadas al cliente para que actualice los datos que se muestran al usuario. (La variable de SALIDA es aByte y la de ENTRADA es currentState).

5.2.5 El problema de la multitarea

Inicialmente se considero para el servidor un esquema un poco diferente de trabajo, se pensaba que el servidor debería no solo realizar las tareas que se acaban de describir, sino que además debería mostrar en la pantalla el estado del puerto y las conexiones junto con una línea en la que se pudieran ingresar comandos de manera local, esto presenta un problema que no es sencillo de solucionar, como sabemos, los programas normalmente están siguiendo un camino en el que las operaciones se realizan una por una y aunque existen bifurcaciones y condiciones que cambian este camino, siempre se realiza una sola operación a la vez.

El programa que llamamos servidor tiene que estar pendiente de la conexión a la red, para que el momento en que se presente una petición la atienda sin embargo, cuando se pensó en que también el servidor atendiera comandos de manera local, el programa tendría que también esperar a que el usuario digitara un botón y entonces atenderlo, aquí es donde se presenta el problema , cuando el programa espera peticiones de red se “detiene” en un ciclo hasta que la petición se presente,

esto impide que se pueda poner atención en lo que el usuario digita localmente ya que en ese momento el programa esta “ocupado” con la red, de igual forma ocurriría si primero atendiéramos al usuario y después a la red.

Esto se podría arreglar si se empleara en la programación del servidor una técnica llamada threads (hilos) o programación multihilos que permite a un programa bifurcar el camino único que tradicionalmente recorren los datos, a dos o más caminos, esto para que programa pueda atender varias cosas a la vez, pero en general esta técnica complica significativamente el programa y en nuestro caso tomaría tiempo que podemos emplear en otras partes del proyecto. Pero el argumento mas fuerte por el cual no se usa la programación multihilos es que podemos cambiar la idea de que el servidor atienda comandos que se le envían remotamente, haciendo que el cliente pueda correr también en la maquina donde corre el servidor y así dejar la tarea del ingreso del comandos al cliente que esta localmente también. De esta manera el servidor estará dedicado a controlar la conexión a la red y al puerto paralelo únicamente, este fue el esquema que finalmente se utilizo, ninguna de las funciones planteadas en el sistema fue sacrificado solo se redistribuyeron las tareas entre los dos programas.

5.2.6 La inicialización del programa y la interface

Un punto importante en lo que respecta a la utilizaron del sistema, será sin duda el control de los dispositivos conectados a la interface en el momento en el que el sistema se encienda, hemos visto que la inicialización del puerto paralelo depende de la marca del BIOS, del sistema operativo, etc. El problema puede surgir si tenemos aparatos conectados a la interface y cuando la maquina arranque puede que arbitrariamente encienda o apague aparatos que no deseamos que cambien de estado, provocando accidentes o simplemente inconformidad.

Para resolver este problema la interface deberá estar apagada hasta que inicie nuestra aplicación servidor y también se debe tomar en cuenta que una vez inicie la aplicación todos los aparatos estarán iniciados en un estado de “apagado” para prevenir algún incidente.

5.3 Programa del cliente.

El modulo que falta para completar el sistema es el programa cliente, sabemos que este programa se correrá en las computadoras remotas, es decir, en las computadoras que no tienen conectado directamente al puerto paralelo la interface, estas computadoras estarán en cualquier punto del mundo, conectadas a Internet a través de diversos tipos de conexiones, desde un MODEM hasta enlaces dedicados de varios Mbit/s (Mbps). Este programa ofrecerá al usuario la información del estado de la interface conectada al servidor de manera remota, además provee la posibilidad de que el usuario pueda transmitir hasta el servidor una serie de comandos mediante botones, que facilitaran su manejo, ya sea para modificar algún bit del puerto y por consiguiente prender o apagar algún aparato, o bien solo para modificar la descripción que está asociada a cada bit de la interface.

Esta aplicación esta realizado con lenguaje Java con la inclusión de algunas librerías para el manejo de interfaces graficas. Ofrecerá todos los datos necesarios para conocer el funcionamiento general del sistema, cabe mencionar que a este versión del programa se le pueden añadir más librerías para poder visualizar con video lo que está pasando la casa u oficina. En esta versión no se agregaron debido a las características básicas de nuestro proyecto, pero se ofrece información en la página web de SUN [13] para instalar estas librerías.

El programa estará esperando que se dispare algún evento generado por el usuario y en ese momento se inicia la rutina correspondiente al botón que dispara el evento. Esta rutina se encarga de unir varias subcadenas para formar una cadena general, esta cadena se describe en la tabla 10.

Subcadena 0	Subcadena 1	Subcadena 2	Subcadena 3	Subcadena4	Subcadena(n)
			Mensaje		
Ip origen o nombre del host.	Tamaño de la cadena.	Comando.	Argumento 1	Agumento2	Argumento(n)

Tabla 10. Formación de la cadena total

Esta cadena antes de ser enviada al servidor se válida para ver si no tiene errores. Esta tarea es realizada por un intérprete de comandos, similar al incluido en el servidor, en caso de que esta cadena sea válida, se pasa al socket que como hemos visto en la sección dedicada al servidor, se encarga de la conexión de red.

[13] http://java.sun.com/products/java-media/jmf/index.jsp

Este socket realiza las funciones de conexión, definición de puerto y transmisión del paquete al servidor, una vez hecho esto, se actualiza los datos de la interface grafica del cliente y espera a que el servidor realice la tarea encomendada y en caso de que no exista ningún error en la transmisión o en la realización de la orden, se recibirá una cadena de caracteres de parte del servidor idéntica a la que se envió. De lo contrario el cliente mostraría un mensaje de error que terminaría la conexión entre el cliente y el servidor. Esto es para que si se vuelve a conectar el cliente, el servidor actualice la información mostrada al usuario por la aplicación cliente con los datos actuales que contiene el puerto paralelo.

5.3.1 La información presentada

La pantalla de cliente está dividida en varias partes, la primera (superior) muestra el escudo de la Facultad de Ingeniera, la segunda (izquierda) muestra el estado de cada uno de los ocho bits de salida del puerto paralelo, la tercera (derecha) muestra el estado de cada uno de los cuatro bits de entrada, la cuarta (central) muestra cuadros de texto donde se van a ir colocando los mensajes que se envían y reciben y la quinta (inferior) es donde se muestran diferentes botones con las funciones más importantes de la aplicación. Como se muestra en la figura 49.

Las etiquetas que se encuentran debajo de cada botón muestran la descripción que le corresponde a cada bit del puerto paralelo, es decir, describe que aparato tiene conectada la interface físicamente.

Se puede observar que las etiquetas mencionadas además de tener su respectiva descripción, cambian de color dependiendo si el bit se encuentra apagado o encendido. Por ejemplo: en el Botón OUT8 la etiqueta esta de color rojo lo que significa que el Bit8 de salida esta conectado a la “LAMPARA1” y esta “apagado” por lo que lógicamente tiene un “0”. Si el usuario le da click a este mismo botón el cliente le envía una cadena al servidor como el de la Tabla10 y la etiqueta cambia a color verde, lo que significa que el Bit8 de salida que esta conectado a la “LAMPARA1”, esta “encendido” por lo que lógicamente tiene un “1”.

5.3.2 Comandos disponibles

Desde este programa tenemos la posibilidad de controlar todas las funciones que el sistema ofrece, así que cada una de estas acciones deberán se representadas por un comando, a continuación se muestran los comandos disponibles en el sistema junto con sus argumentos necesarios para su correcta ejecución en el servidor.

Comando	Argumento1	Argumento2	Descripción
apagar	Numero de bit	-	Apaga el bit de salida que se indica en el argumento1, sin modificar los bits restantes.
prender	Numero de bit	-	Enciende el bit de salida que se indica en el argumento1, sin modificar los bits restantes.
guarda	Cadena	-	Guarda un archivo en el disco duro del servidor con la descripción de los bits, esta descripción va incluida en el argumento1
recupera	-	-	Recupera de un archivo en el disco duro del servidor la descripción de los bits.
refresh	-	-	Solicita al servidor el estado actual de la interface (bits de entrada y de salida) y actualiza los datos en la pantalla.
ayuda	-	-	Ofrece una descripción de las funcionalidades disponibles en la aplicación.

Tabla 11. Comandos disponibles en la aplicación Cliente.

5.3.3 Argumentos de inicialización del programa.

En este programa existen un parámetro que va a estar variando de usuario en usuario y de computadora en computadora, este es la dirección IP del servidor, debido a esto y a que es muy engorroso estar ingresando este dato por parte del usuario cada vez que se utilice el programa, este ofrecerá la posibilidad de pasar este dato a través de la línea de comando que se ingresa en el sistema operativo como argumento, así este dato tendrá un valor estándar en el caso de que no se introduzca ningún valor como argumento, en este caso el valor default será la IP **127.0.0.1**
Para mostrar esto más claramente usaremos un ejemplo: Imaginando que la IP del servidor sea la 192.168.9.1 la línea de comando quedara

>java Client 192.168.9.1

5.3.4 Imagen de la pantalla

Por último y para mostrar con mayor detalle la forma del programa cliente se presenta en la figura 49 una imagen que muestra la pantalla principal de este.

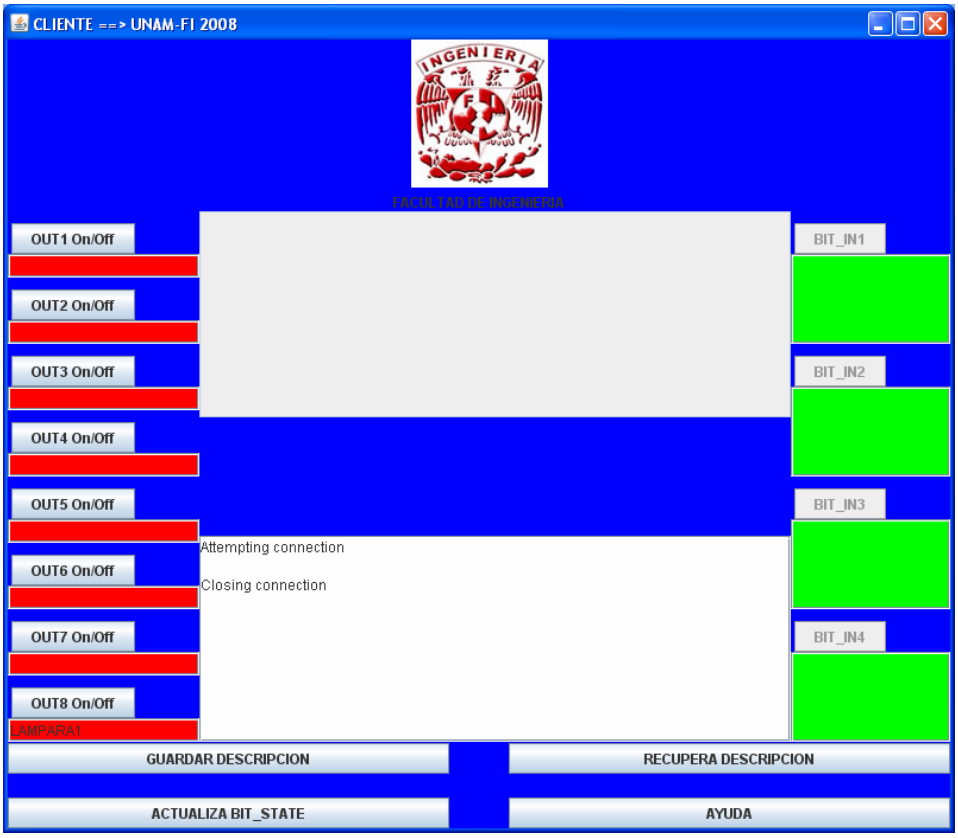


Figura 49. Información presentada en la aplicación Cliente.

5.4 Construcción del prototipo.

5.4.1 Interface puerto paralelo - aparatos

CONFIGURACIÓN FÍSICA

Todo el conjunto que forma la interface está contenido en un gabinete plástico de 22x 14x 9 cm. Como se muestra en la figura 50.

Figura 50. Gabinete de la interface.

- ✓ En la parte lateral izquierda podemos apreciar la conexión polarizada a la toma de corriente domestica de 127 Vac. Como en la figura 52.
- ✓ En la parte lateral derecha se encuentra el botón de encendido de color rojo y la protección del circuito la cual consiste en un fusible. Como en la figura 53.
- ✓ En la parte trasera, se encuentra un conector DB25 Macho para la conexión del puerto paralelo de la PC. Como en la figura 53.
- ✓ En la parte delantera se encuentra una ventana para la ventilación del circuito y la observación mediante leds del estado de los bits de salida, como en la figura 52. También se encuentra un micro switch para controlar los bits de entrada del puerto paralelo. Como en la figura 50.

- ✓ En la cara superior se encuentra el panel que servirá para conectar los diferentes aparatos que se van a controlar. Como en la figura 51.

Figura 51. Panel de conexiones.

Figura 52. Cara lateral izquierda.

Figura 53. Cara lateral derecha.

En este punto es bueno mencionar que el panel de conexiones (figura 51) se encontrara de izquierda a derecha los ocho bits de salida respectivamente.

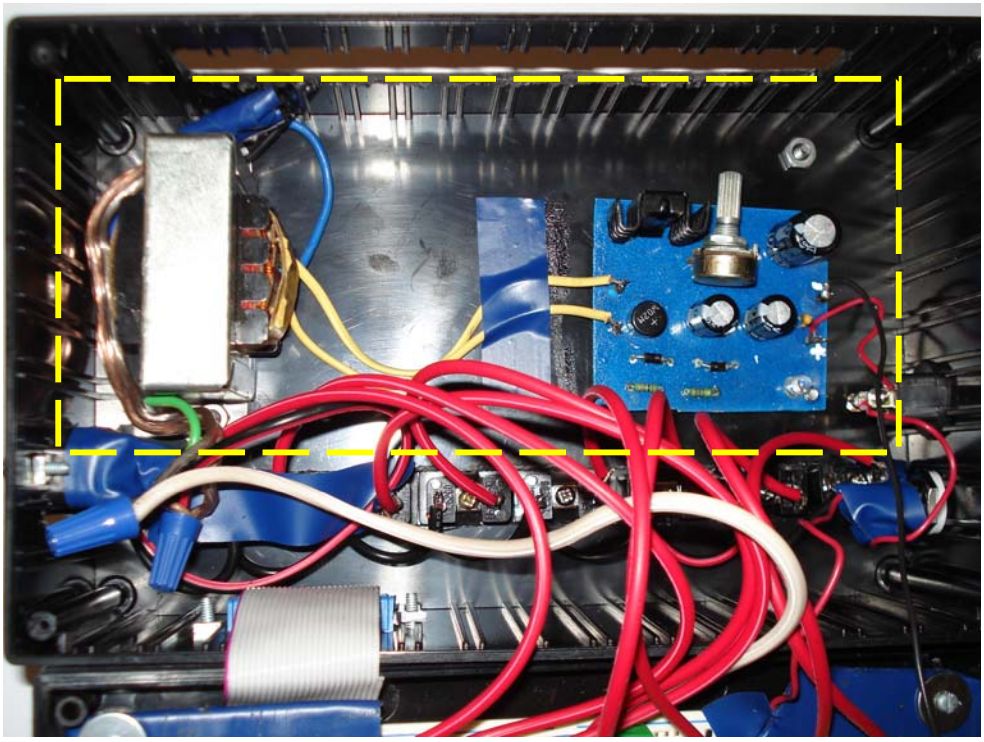


Figura 54. Fuente de alimentación.

Como ya sabemos esta interface está dividida en tres circuitos, la fuente de alimentación fue construida en una placa fenolica, cuyo diagrama se encuentra en la figura 55, y junto con el transformador se colocaron dentro del gabinete el cual contendrá también los circuitos de entrada y salida que están hechos sobre dos protoboards.

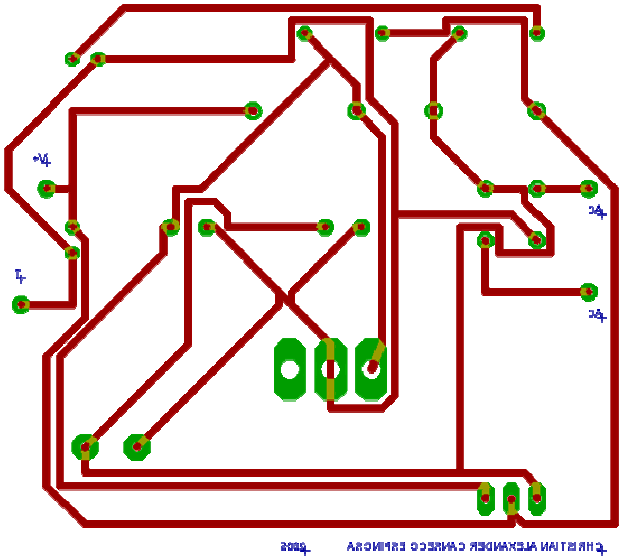


Figura 55. Diagrama de pistas de la fuente.

La fuente de alimentación tiene dos pines de salida que nos proporcionara un voltaje de 12 Vcc. Una derivación de estas dos salidas serán conectadas a un regulador de voltaje LN7405 para que nos proporcionen la salida de 5 Vcc. De esta forma el circuito de entrada y el de salida pueden conectarse fácilmente a la fuente, en la figura 54 se muestra la fuente terminada.

El circuito final de las cuatro entradas y las ocho salidas se muestra en la figura 56, este circuito está diseñado geométricamente de tal manera que se pueda adaptar al gabinete e interconectar las partes fácilmente para facilitar cualquier reparación o modificación.

La vista final del interior del gabinete ya con los circuitos y las conexiones apropiados las vemos en la figura 57.

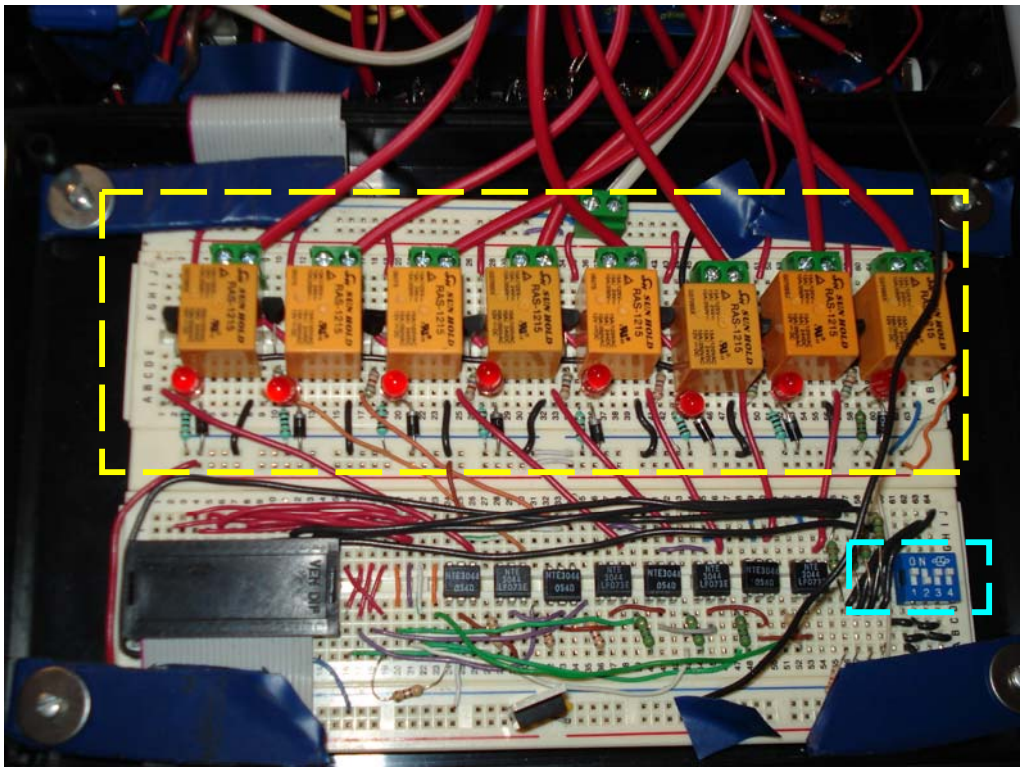


Figura 56. Circuito final de entradas y salidas.

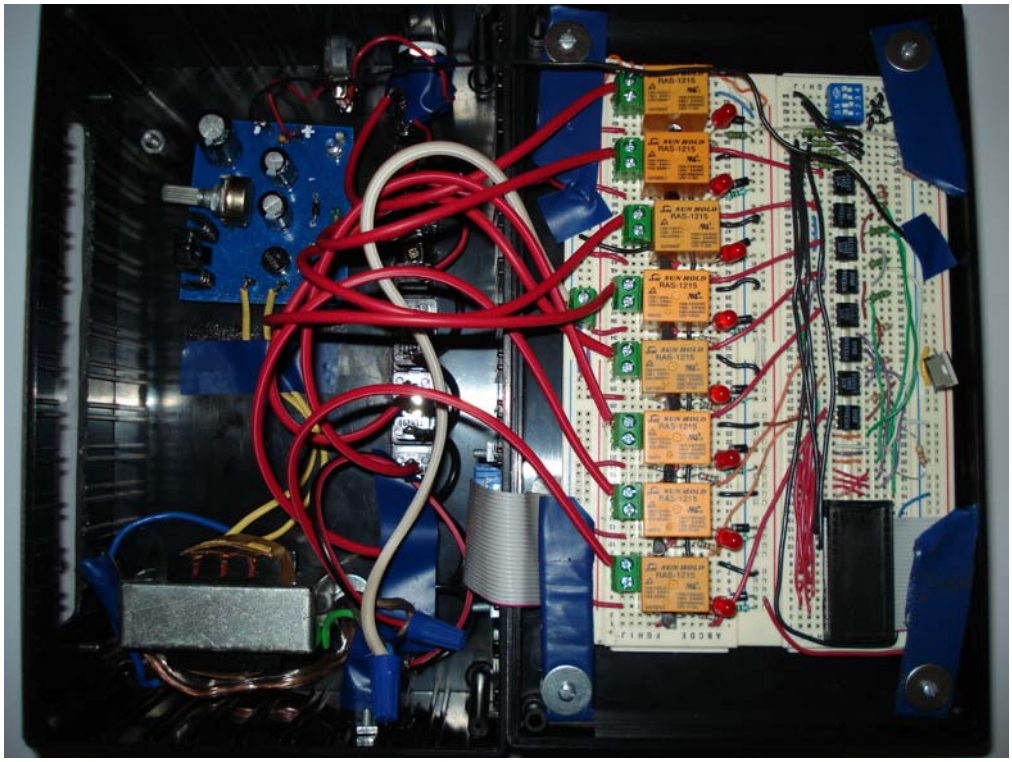


Figura 57. Vista final del interior del Gabinete.

5.4.2 Programas cliente y servidor

Parte fundamental del sistema son los programas servidor y cliente, ya que como se planteo en el objetivo, el hardware que se agrego es mínimo y de esta manera la mayoría de las tareas fundamentales recaen sobre las piezas de software de las que hablamos en los capítulos anteriores.

Como ya sabemos, tanto en el programa cliente como el servidor, están escritos en JAVA y por lo tanto siguen la ruta clásica para el desarrollo de la compilación de un programa, esta es, la escritura del código fuente, compilación, y en algunos casos depuración del código fuente. Para estas tareas se empleo fundamentalmente un software llamado "JCreator LE 4.00": este esta diseñado para ofrecer todas las facilidades en la edición del código fuente, resalta el texto en colores hace comprobación automática de apertura y cerrado de llaves y paréntesis, realiza comprobaciones sin tácticas simples y permite compilar, depurar e incluso ejecutar el programa sin necesidad de salir del programa del ámbito del editor.

Finalmente, los programas compilados (.class) pueden correrse con un archivo ".bat" que es usado para ejecutar un programa desde línea de comando, con todos los parámetros requeridos, de esta manera, el usuario solo ejecuta el archivo ".bat" para que tenga como resultado la ejecución del programa cliente o servidor.

Capítulo 6:

Diseño de pruebas.

6. Diseño de pruebas.

La idea principal de cualquier proyecto de este tipo es, en primer lugar alcanzar el objetivo planteado y resolver la hipótesis propuesta y para que esto pueda ser verificado, se requiere de un proceso de evaluación del sistema, esta evaluación se lleva a cabo a partir del diseño de pruebas a los diferentes componentes de este. Las pruebas deberán mostrar primeramente que las funciones para las que el sistema fue diseñado sean realizadas y en este segundo término deberán identificar posibles defectos o errores probables en la operación. Este capítulo plantea la manera en que los distintos módulos que forman al sistema serán probados, posteriormente se evaluaran los resultados obtenidos y si se llegara a las conclusiones finales del trabajo.

6.1 Interface del puerto paralelo- aparatos.

Las pruebas que se efectuaran a la interface tiene como objetivo principal verificar que los datos de la computadora tanto de entrada como de salida lleguen a la interface correctamente, que el circuito de salida pueda efectivamente interrumpir la corriente en el circuito de corriente alterna casera, que el circuito de entrada mande al voltaje y corriente adecuados para excitar el puerto paralelo y detectar el accionamiento de algún sensor y por último se realizara una prueba de estabilidad en tiempo y potencia máxima admitida, con la cual verificaremos si el circuito puede permanecer encendido por largos periodos de tiempo o incluso permanentemente y por otro lado sabremos el límite de potencia que puede interrumpir en el circuito de salida y si el cálculo de potencia de los componentes es adecuado. Ahora detallaremos en qué consiste cada prueba y describiremos como se desarrollara cada parte.

6.1.1 Prueba 1.- Comprobación de la transferencia de datos entre la interface y el puerto paralelo.

Existen dos puntos en los que la transferencia de datos entre la interface y la PC puede fallar más fácilmente, uno es en el Cable DB25 y otro es en el interior del gabinete de la interface, más específicamente entre las conexiones del cable DB25 y la Protoboard del circuito. La prueba se llevo a cabo siguiendo estos pasos:

1) Comprobación de la continuidad con un milímetro digital de los veinticinco pines usados en el cable DB25.
2) Revisión de los puntos de fijación de los cables que conectan la protoboard con el conector DB25 macho colocado en el gabinete y comprobación de continuidad desde el conector DB25 hasta la protoboard.

3) Comprobación de la atenuación del cable usando 1 Vcc .

En general esta es la prueba más simple y debe verificar que todas las conexiones estén bien hechas y que la atenuación que sufra el voltaje aplicado el cable no supere el 5%, con ello puede asegurarse que tanto el cable como las conexiones son correctos.

6.1.2 Prueba 2. - Interrupción del circuito AC.

En este prueba lo que nos interesa saber es que la señal que proviene del puerto paralelo y que es recibida por el circuito de salida puede efectivamente activar el relevador e interrumpir la corriente de los aparatos de AC. Estos son los pasos:

1) Se inician los programas servidor y cliente en la misma máquina y se pide prender el bit 1.
2) Se comprueba con el milímetro que el pin 2 del puerto paralelo se ponga en 5 Vcc.
3) Se apaga el bit desde el programa cliente y posteriormente de conecta la interface a la PC.
4) Se pide nuevamente desde el programa que prenda el bit 1.
5) Con un milímetro se verifica que la corriente demandada al puerto paralelo no exceda los 20 ma y que en las terminales del relevador y en las del conector externo efectivamente se presente continuidad.
6) Después se conecta una lámpara de 127 Vac a 150 Watts en el panel de conexiones de la interface y se verifica que la lámpara encienda.
7) Se vuelven a repetir los puntos del 1 al 6 pero ahora con cada uno de los 7 bit restantes.

Esta prueba es muy importante ya que la acción de prender y apagar un aparato de la red domestica es fundamental.

6.1.3 Prueba 3.- Circuito de entrada con un switch deslizable (dip switch de 4 posiciones).

La adquisición de la señal digital que viene de un interruptor es muy importante, debido a que de esta manera podemos verificar que una puerta o ventana está abierta o cerrada. El circuito que acompaña al interruptor es muy simple y solo debemos observar cuanta corriente y voltaje entran al puerto siguiendo estos pasos:

1) Se pone el interruptor que se encuentra en el interior del gabinete en posición ON.

Se polariza la interface y con un milímetro verificamos que alguno de los pines de entrada del conector DB25 presenta algún voltaje, y este tiene que ser alrededor de 5 Vcc.

6.1.4 Prueba 4.- Estabilidad en el tiempo y potencia máxima admitida.

En este caso haremos una prueba que podría tornarse destructiva si excedemos los márgenes de potencia admitidos por el relevador y los demás componentes que forman el circuito de salida. Las especificaciones del relevador que se utilizo en el prototipo marcan que la corriente máxima de interrupción es de **15 A a 127 Vac**, si consideramos idealmente que el aparato que conectamos es puramente resistivo, podemos ignorar la potencia reactiva y solo considerar la potencia aparente que matemáticamente se puede expresar como el producto del voltaje por la corriente, de esta manera la potencia que podríamos manejar con este relevador, sería de **1905 Watts**.

En un caso más real en donde los aparatos que conectamos presenten potencia reactiva, real y aparente podemos estimar que la capacidad máxima del circuito será de **1000 Watts**.

Para comprobar si el circuito puede estar encendido por largos periodos de tiempo con su carga máxima, la prueba contemplara un periodo corto primero para determinar si soporta la potencia y luego uno muy largo con la misma carga siguiendo los siguientes pasos:

1) Se busca un aparato casero que demande de la toma de corriente alrededor de 1000 Watts y se conecta al panel de conexiones de tal forma que la interface quede dispuesta como un interruptor.
2) Se activa a través del programa cliente uno de los bits de salida y se verifica que el aparato se encienda y así se mantiene 5 min.
3) Ahora se revisara la temperatura de los componentes y en especial del relevador.
4) De nuevo se enciende el bits pero ahora se mantendrá así durante ocho horas con carga, al final se revisa la temperatura y el estado físico de los componentes.
5) Se puede repetir la prueba para los otros bits de salida.

Esta prueba es muy importante para definir si este aparato puede ser usado en jornadas de trabajo comunes en la mayoría de las casas y oficinas y sin correr ningún riesgo, además verifican que la potencia calculada para cada componente sea la adecuada.

6.2 Programa servidor.

Ahora iniciaremos las pruebas al software, esta parte aunque no es tan riesgosa como la anterior en cuanto a la posible destrucción del prototipo, si es la parte que ofrece mayor complejidad en cuanto al diseño y por lo tanto existen más factores que influyan en la buena o mala operación del programa.

Empezamos con el servidor que como sabemos estará encargado de tres aspectos fundamentales, uno es el control del puerto paralelo, otro la interpretación de los comandos y por último la conexión a la red. Las pruebas que le haremos a este modulo serán en esos mismos aspectos y siguiendo el mismo orden, cabe señalar que al igual que en las pruebas hechas a la interface en algunos caso se verán involucrados tanto el programa cliente como la propia interface.

6.2.1 Prueba 5.- Acceso al puerto paralelo (entrada y salida).

Para esta prueba usaremos inicialmente un programa que solo tiene implementado el puerto paralelo, con este programa que llamaremos “SimpleIO.java” podemos verificar que las funciones “read()” y “write()” efectivamente nos permitan manejar el puerto, posteriormente corremos el servidor que posee estas mismas funciones y se realizara la prueba:

1) Se inicia el programa SimpleIO.java que llama a las dos funciones mencionadas y desplegamos en la pantalla en resultado que arroja la función read() y observamos en la interface los resultados de la función write() (dependiendo del número en decimal que le hayamos escrito al puerto paralelo).
2) En este caso el programa encenderá el bit 1 del puerto paralelo, esto puede ser verificado con la ayuda de la interface y un milímetro colocado en las terminales de salida del bit 1 o mediante el Led.
3) Después se corre el programa servidor y el cliente en la misma máquina, desde el cliente se solicita el encendido del bit 1, y de nuevo se comprueba en la interface que esto pase.
4) En el caso de la lectura, se coloca la interface en el puerto y se pone en posición ON una de las entradas, ahora desde el programa cliente se solicita refrescar la pantalla, en el estado del puerto se debe mostrar el bit respectivo encendido.

6.2.2 Prueba 6.- Envió y recepción de paquetes en red local e Internet.

Esta prueba consiste en verificar que el servidor es capaz de enviar y recibir datos hacia y desde otra computadora, primero se probara en una red local y posteriormente en Internet, se usara en la otra máquina nuestro programa cliente reducido y solo envía una cadena de caracteres y lo espera que regrese integro del servidor, los pasos son los siguientes:

1) Se corre el servidor en una computadora perteneciente a una red local en este caso la 192.168.1.0, el servidor tendrá la IP 192.168.1.2
2) En otra computadora de la red, en este caso 192.168.1.3 se corre el programa cliente.
3) Se escribe en el cliente la cadena "probando probando".
4) En la computadora que está corriendo el servidor se debe mostrar en la pantalla el aviso de que llego un mensaje, nombre de host, el tamaño del mensaje y finalmente el mensaje.
5) En el cliente debe aparecer debajo de la línea donde se ingreso la cadena de prueba "probando probando", otra línea con el mismo mensaje exactamente, este ultimo proveniente del servidor.

Posteriormente se repite la prueba, paro ahora corriendo el programa en un servidor con dirección IP homologada y el cliente en otra computadora conectada a Internet que no pertenezca a la red del servidor.

6.3 Programa cliente.

Continuando con las pruebas de software realizaremos la evaluación del programa cliente, de igual forma con el servidor nos guiaremos por las funciones principales que este programa realiza para diseñar las pruebas, estas funciones son, la validación de los comandos, la comunicación por la red y el despliegue del estado de la interface en la pantalla, así las pruebas quedan de la siguiente manera:

6.3.1 Prueba 7.- Envió de comandos por Internet.

Esta prueba es complementaria a la prueba 6 hecha al servidor, aquí probaremos la transferencia de datos pero ahora con el programa cliente completo, esto es, el que finalmente se usara en el sistema. La prueba es así:

1. Correremos el servidor en una maquina con IP homologada.
2. Después desde otra computadora con acceso a Internet corremos el cliente y damos clic en un botón para enviar un comando valido.

3. Del lado del servidor verificamos que el mensaje haya llegado y contenga la línea de comando original y que la IP de origen sea la correcta solo se concreta a verificar la comunicación en un solo sentido, junto con la prueba 8 podemos saber si todo el ciclo completo de comunicación se da.

6.3.2 Prueba 8.- Despliegue de los datos a partir de la contestación del servidor.

Aquí veremos como el mensaje enviado por el servidor a manera de comprobación y la actualización de los datos en la pantalla del cliente son correctos y de esta manera se cierra el ciclo de comunicación que se presenta cuando un cliente pide un cambio de estado de la interface, la prueba se lleva a cabo de la siguiente manera:

1) Se ejecuta el programa cliente y se toma nota de los datos que se presentan el estado de la interface.
2) Se realiza la prueba 7, una vez que se verifique que el servidor recibe el mensaje correctamente y realiza el envió de comprobación hacia el cliente se continua con el siguiente punto.
3) Se revisa que los datos del estado de la interface se actualice de acuerdo a la petición hecha.

Esta es una de las partes más importantes de toda la comunicación ya que es donde el usuario podrá apreciar que el comando efectivamente fue realizado con éxito. Esto se realizara con todos los comandos disponibles en el Cliente.

6.3.3 Prueba 9.- Guardar y recuperar los datos de descripción.

Ahora comprobaremos la función de guardar y recuperar la descripción de los bits, este botón llama al comando “guarda” para guardar la descripción de todos los bits de la interface, una vez que el archivo es guardado puede ser recuperado explícitamente con el botón “Recupera Descripción” , la prueba entonces sería así:

1) Se corre el programa servidor y cliente, en el programa cliente se cambia la descripción de varios bits.
2) Se guarda la descripción de esto bits.
3) Salimos del programa cliente.
4) Llamaremos al programa de nuevo pero ahora y presionamos el botón “Recupera Descripción”.
5) Verificamos que la información de la descripción sea correcta y este en los bits adecuados.

6.4 General (en 3 computadoras representativas de las que existen en el mercado).

6.4.1 Prueba 10.- General.

En esta prueba revisaremos todo el sistema en su conjunto la interface conectada a la toma de corriente y a la computadora, los aparatos (lámpara y ventilador) y sensores (dip switch de 4 posiciones) también dispuestos en su lugar, los programas corriendo, el servidor en una computadora y el cliente en otro lugar geográfico de la ciudad con la computadora conectada por MODEM de 56 Kbps, ahora realizaremos las siguientes tareas.

1) Se enciende la interface y corren los programas.
2) Se cambia la descripción de los bits de entrada por ventana y puerta de acuerdo a las conexiones.
3) Ahora se cambia el estado del dip switch y se solicita al cliente que actualice los datos.
4) Se cambian la descripción de los bits de salida por la lámpara y ventilador.
5) Se pide desde el cliente que la lámpara y el ventilador se enciendan.
6) Se guardan los datos de descripción.
7) Salimos del programa cliente.
8) Ejecutamos de nuevo el programa y recuperamos la descripción de los bits.
9) Verificamos que tengamos los mismos datos que en la sesión anterior.
10) Solicitamos apagar la lámpara y el ventilador y regresamos al estado original el dip switch.
11) Se verifica que actualice la pantalla incluyendo la modificación del dip switch.

Esta es la prueba final que demuestra que el sistema funciona en su conjunto y con todos los elementos que lo conforman, como era de esperarse observando los resultados de las pruebas pasadas.

6.4.2 Prueba 11.- Ejecución en varias plataformas y compilación.

Lo que nos queda por probar es que tan portable y flexible son nuestros programas como se dijo en el objetivo del trabajo, es muy importante que el sistema pueda ser usado en prácticamente todas las computadoras modernas compatibles con el modelo de IBM, en un rango desde 386 hasta las modernas Core 2 duo, esto además es importante porque se desea demostrar que podemos usar computadoras antiguas que se consideran obsoletas para las tareas comunes de oficina y hogar. Así nos daremos a la tarea de probar los programas en 3 distintas computadoras.

La prueba consiste primero en ejecutar los programas en estas computadoras y aplicar la prueba 10, pero también probaremos si es posible compilar los programas en diferentes sistemas operativos con el The Java SE Development Kit (JDK) que estas computadoras tengan. Entonces aquí es donde aparece el sistema operativo, sabemos que nuestro sistema funciona en el sistema operativo Windows XP, pero este S.O. a sufrido modificaciones desde que aparición hasta la fecha en lo que actualizaciones se refiere, no obstante, el código objeto que generamos en Java es independiente de la plataforma, así que en teoría no debería haber ningún problema para ejecutar nuestros programas e incluso en versiones también de LINUX que contenga la maquina Virtual de Java actualizada. Por eso aquí se pretende una tabla con las tres computadoras en la que se hizo la prueba junto con los S.O. que tienen instalados y el resultado obtenido en cuanto a la ejecución como en la compilación.

Numero	*Hardware*	*S.O.*	*Ejecución*	*Compilación*
1	Pentium M725, 1.6GHz, 1Gb en RAM, 60Gb de D.D.	Windows XP	Servidor: Si Cliente: Si	Servidor: Si Cliente: Si
2	Pentium IV, 2.66GHz, 1Gb en RAM, 80Gb de D.D	Wisndows XP	Servidor: Si Cliente: Si	Servidor: Si Cliente: Si
3	Pentium 250MHz, 256Mb en RAM, 40Gb de D.D	Fedora 9	Servidor: Si Cliente: Si	Servidor: Si Cliente: Si

Tabla 12. Diferentes computadoras con diversas características en las que se probó el sistema.

Capítulo 7:

Evaluación de resultados.

7. Evaluación de resultados.

Con la ida de hacer un resumen de cómo las pruebas nos fueron dando indicios de que el objetivo se está cumpliendo o no, hacemos una evaluación más detallada de los resultados en esta parte.

La mejor manera a mi parecer de evaluar los resultados de un proyecto es partir del diseño y de cuáles son las funciones que cada parte debe de realizar, a su vez, este diseño se planteo como una alternativa para cumplir el objetivo principal del trabajo; entonces, siguiendo esta lógica podemos advertir que si las pruebas verifican que las funciones principales del prototipo se están realizando con éxito, podemos concluir que el objetivo se está alcanzando.

De nuevo dividiremos como en capítulos anteriores en módulos, para que así podamos considerar hasta que punto cada uno de estos está cubriendo sus funciones, después haremos una evaluación final del sistema en su conjunto para determinar si el nivel alcanzado por el sistema es satisfactorio para decir que el objetivo esta cubierto.

7.1 Evaluación de la interface del puerto paralelo- aparatos.

Partiendo de los resultados de las pruebas y siempre considerando las limitantes en la construcción manual y las simplificaciones que se hicieron desde el diseño conceptual al del prototipo, podemos decir que la interface es en general suficiente para realizar la demostración de la viabilidad del uso de las computadoras en problemas domóticos.

En principio podemos definir a partir del objetivo general, un objetivo especifico que la interface deberá cubrir, este es: Con el fin de corroborar la manera en la que una computadora personal puede manipular aparatos de uso domestico con el solo puerto paralelo, se deberá contar con un circuito electrónico que tenga la función de que a partir de una pequeña señal proveniente de la PC pueda controlar el encendido o apagado de aparatos que demandan mas corriente y voltaje que la dada por la PC.

Entonces la pregunta que definirá la evaluación es, ¿ La interface construida cumple con los requerimientos del objetivo particular destinado a este circuito?. La respuesta es sí, de hecho las pruebas realizadas en el capitulo anterior lo enfatizan, sin embargo, es importante no solo definir se el circuito funciona o no, además habrá que calificarlo y esta es una función mas cuantitativa que cualitativa.

Podemos decir entonces que la interface maneja los aparatos domésticos, que pueden adquirir señales digitales originadas en sensores o interruptores dip switch, pero no puede adquirir señales analógicas, esto se debe a la reducción hecha en el prototipo en relación al diseño conceptual, no obstante, la propuesta de un circuito que realice esta tarea de adquirir señales analógicas está hecha precisamente en el capítulo relativo al diseño conceptual.

Con esto podemos decir que la interface cubre un 80% del objetivo impuesto pero la idea de que es lo suficiente para revisar los conceptos generales del sistema sigue firme.

7.1.1 Prueba 2

Los resultados de las pruebas fueron exitosos y en todos los casos los circuitos de salida funcionaron muy bien, se pudo prender y apagar la lámpara de 150Watts, sin ningún problema, además el circuito no le demanda al puerto mas de **1.2 mA**. Es claro que parte de esta prueba depende del funcionamiento correcto de los programas cliente y servidor, así que podemos adelantar que por lo menos de manera local los programas parecen funcionar bien. Otra parte importante que debemos notar es que esta prueba solo se probó la capacidad de interrupción de la interface y no la potencia soportada, eso se realizara en una prueba posteriormente.

7.1.2 Prueba 3.

La prueba nos ofreció resultados positivos tanto en la detección del interruptor como en los valores de voltaje y corriente obtenidos en el conector DB25, estos fueron **4.82Vcc** y **1.1mA** que efectivamente están dentro del margen adecuado para la operación del puerto paralelo.

7.1.3 Prueba 4

La prueba se realizo usando 3 focos de uso común que en suma consumían un total de 450Watts, cada uno de estos tenían un consumo aproximado de 150Watts, en realidad los únicos aparatos domésticos que alcanzan los 1000Watts son aquellos que generan calor y para la prueba no se tenía ninguno cerca y además considerando que la prueba duraría muchas horas la factura por el servicio eléctrico se iba a incrementar mucho, por eso se prefirió la opción de los focos. Con el primer encendido de 5 minutos los componentes se comportaron bien, ninguno sufrió daños.

En la segunda prueba, la de 8 horas, las cosas no cambiaron, de nuevo el único componente que elevo la temperatura un poco fue el regulador de 12Vcc pero todos los componentes funcionaron bien durante las 8 horas sin problema y la interface se mantuvo intacta después de esta prueba.

En resumen podemos decir que el circuito esta en condiciones de uso domestico, y considerando que la construcción es manual se puede calificar como un buen prototipo que nos permitirá demostrar que el objetivo puede ser alcanzado.

7.2 Evaluación del programa servidor.

De nuevo la evaluación de los resultados obtenidos por este programa se basa en lo obtenido en las pruebas y en la percepción que el usuario obtiene al utilizarlo.

Es claro que a partir de las pruebas, el programa puede realizar las tareas fundamentales esperadas de él, puede manejar el puerto paralelo , interpretar comandos, enviar y recibir correctamente datos de la red. Particularmente el sistema es funcional y amigable para el usuario. Cabe mencionar que sería mejor ocultar totalmente de la vista del usuario los mensajes que emite el programa, es decir, sería mejor que el servidor funcionara como un “servio” (en Windows) o un “demonio” (en Linux) que es un tipo especial de programa que corre en background y que el sistema puede echar a andar desde el arranque del S.O. sin intervención del usuario; en este caso los mensajes que el servidor genera son enviados a un archivo bitácora.

Podemos ver que lo anterior no es un problema, es solo un detalle de funcionalidad, que de ser implementado ofrecería una manera mas estándar del uso del servidor.

7.2.1 Prueba 5

La prueba demostró que los métodos antes mencionados que forman al programa “SimpleIO.java” funcionan satisfactoriamente, nos dejan controlar el puerto paralelo de manera sencilla y podemos controlar cada bit. En el caso de la lectura el trabajo delicado consiste en trasformar de decimal a binario el byte que arroja el método para poder ver el contenido de los bits deseados, pero fuera de eso no existe mucha dificultad.

Por lo que el programa “SimpleIO.java” como el servidor funcionan muy bien.

7.2.2 Prueba 6

En la primera prueba no ocurrió ningún problema el programa cliente envió varias cadenas a la red, algunas formadas por comandos correctos y otras no, el servidor recibió correctamente los paquetes, siempre informando la procedencia, el tamaño y el mensaje. El cliente también registró que efectivamente el servidor estaba regresando el mensaje intacto cuando este contenía un comando valido, de no ser así el servidor devolvía un mensaje de error "EL COMANDO NO FUE RECONOCIDO".

En la prueba de Internet se también se obtuvieron pruebas satisfactorias, sin embargo, el único inconveniente era que había un retraso en la comunicación, los mensajes enviados por el cliente tardaban mas en llegar de regreso. Del lado del servidor los comandos se interpretaron correctamente.

7.3 Evaluación del programa cliente.

El programa cliente demuestra en las pruebas que su funcionamiento es aceptable, realiza las tareas fundamentales esperadas como son: la captura de los comandos introducidos mediante un clic en alguno botón de la interface, la interpretación de estos, su posterior envió al servidor y finalmente, la recolección de los datos del servidor para actualizar los datos en pantalla. No obstante, en este programa solo se consideran las funciones elementales ya que como se ha mencionado en varias ocasiones a lo largo de este texto, existen muchas más funciones que en un futuro se podrían implementar en el programa cliente. Es claro que como mencionamos en la sección dedicada al servidor, es cliente es el programa mínimo que se requiere para demostrar que el sistema funciona correctamente, pero no implementa de manera extensiva todas las posibles capacidades del programa cliente, que se ha hablado en los capítulos anteriores.

7.3.1 Prueba 7

4. Se obtuvieron buenos resultados ya que en todos los casos se recibió correctamente el mensaje aunque como en la prueba 7, se presentaron retrasos debidos al tráfico por Internet.

7.3.2 Prueba 8

Todo funciono correctamente en caso de que el servidor no esté prendido o que el paquete se pierda, el cliente marca un error instantáneamente cuando el cliente no

recibe noticias del servidor, este procedimiento se probo desconectando al cliente de la red, sin embargo.

7.3.3 Prueba 9

La prueba demostró que la función trabaja bien, cuando se sigue al pie de la letra las instrucciones de la prueba, la recuperación de las descripciones se realiza correctamente.

7.4 Evaluación general

Podemos decir que el sistema es su conjunto se comporta de manera aceptable considerando solo las pruebas que se definieron en el capitulo anterior. Sin embargo, no podemos dejar a un lado la posibilidad de errores u omisiones tanto en el sistema en si, como en el diseño de las pruebas realizadas en este trabajo. Estas omisiones por supuesto no tratan de ser voluntarias pero no es raro que en la práctica se presente accidentalmente situaciones que en el diseño original no se tenían contempladas.

Con esta ultima parte podemos decir que el trabajo está concluido y que el prototipo demuestra la viabilidad de la tesis propuesta en el objetivo, existen muchas ideas que quedaran pendientes y otras que aunque se muestran aquí, muchas ideas que quedaran pendientes y otra que aunque se muestran aquí, son susceptibles de mejora o ampliación, ahora nos concentraremos en resumir en el siguiente capítulo las conclusiones a las que se llego gracias a este trabajo.

7.4.1 Prueba 10

Todo funciono muy bien, y el sistema en general realizo todas las tareas sin contratiempo, solo falta comprobar en diferentes plataformas si esto se cumple y de eso de trata la prueba 11.

7.4.2 Prueba 11

Podemos observaren la tabla que prácticamente en los 3 sistemas probados, los programas funcionan, en el 100% de ellos los programas pueden ejecutarse y compilarse correctamente.

Capítulo 8:

Conclusiones.

Capítulo 8:

Conclusiones.

8. Conclusiones.

Inicialmente nos planteamos la posibilidad de manejar aparatos domésticos a través de Internet con una PC común y un par de programas, y al final de este trabajo podemos decir que efectivamente lo logramos aunque como en cualquier proyecto existen sus modificaciones. Si pensamos únicamente en la funcionalidad mínima que un sistema como estos debe cumplir, entonces nuestro sistema funciona muy bien y de hecho las pruebas lo confirman , ya que algunas de estas se realizaron en ambiente real de uso. Se utilizo efectivamente el mínimo hardware adicional lo que se traduce en un bajo costo y una alta portabilidad a diferentes plataformas y computadoras, también en lo que respecta al software, logramos desarrollar todo con productos libres, desde el documento y los diagramas, hasta los programas que en el caso del servidor puede muy fácilmente ser visto como una plataforma de manejo del puerto paralelo a través de Internet de forma general.

Es muy claro que la funcionalidad del los sistemas de computo no puede estar desligada de la facilidad de uso por parte del usuario, en este aspecto, el programa cliente se trato de que fuera lo mas amigable posible.

Queda abierta la posibilidad de que alguien más pueda tomar este trabajo y ampliarlo colocándole diversas características, como el multiprocesamiento del servidor, un convertidor análogo-digital, poder ver desde el cliente por medio de video lo que pasa alrededor del servidor desde el cliente, entre otras funcionalidades que las imaginadas originalmente.

Un sistema domótico completo en la actualidad es un conjunto muy complejo de elementos de comunicaciones, computo, electrónica y control. La propuesta mostrada aquí aunque en comparación con los sistemas comerciales se presenta como muy elemental, plantea las ideas y conceptos necesarios para el desarrollo de nuevas investigaciones que produzcan soluciones mas completas y con todas las prestaciones ofrecidas por las compañías grandes, y lo mas importante, si mantenemos como prioridad el uso de software libre y el empleo de computadoras comunes y hasta discontinuadas, podemos lograr que las ventajas que la domótica propone puedan estar al alcance de mas gente sin necesidad de grandes cantidades de dinero, ni para el desarrollo del sistema ni para su consumo.

Por último podemos decir que la experiencia del desarrollo de este sistema, principalmente en lo referente a los programas, nos sirvió para darnos cuenta que por cada función que se plantea originalmente, existe un sin número de opciones que se presentan ya en el trabajo de la implementación y que en ocasiones no se habían considerado, y aunque parezca menores no son siempre despreciables. Aquí es donde podemos resumir el trabajo del Ingeniero , no se puede considerar

a un sistema funcional solamente por el planteamiento teórico, necesariamente nos tenemos que enfrentar con la experiencia y la implementación practica para eliminar estas sutiles diferencias entre la teoría y el empirismo.

9. Glosario

- ✓ **Actuadores:** Aquellos elementos que pueden provocar un efecto sobre un proceso automatizado. Los actuadores son dispositivos capaces de generar una fuerza a partir de líquidos, de energía eléctrica y gaseosa. Por ejemplo: Válvulas, Resistencias calefactores, Motores eléctricos, Bombas,ventiladores, etc.
- ✓ **AENOR:** Subcomisión del Hogar Digital, dentro de la Comisión 133 (AEN/CTN 133 Telecomunicaciones) a fin de definir estándares.
- ✓ **API:** (Application Programming Interface - Interfaz de Programación de Aplicaciones). Grupo de rutinas (conformando una interfaz) que provee un sistema operativo, una aplicación o una biblioteca, que definen cómo invocar desde un programa un servicio que éstos prestan. En otras palabras, una API representa un interfaz de comunicación entre componentes software. El software que provee la funcionalidad descrita por una API se dice que es una implementación del API. El API en sí mismo es abstracto, en donde especifica una interfaz y no da detalles de implementación. Un API a menudo forma parte de SDK (Kit de desarrollo de software).
- ✓ **ASIMELEC**: La Comisión Multisectorial del Hogar Digital de ASIMELEC es la organización encargada de definir el servicio, los agentes involucrados y las tecnologías de la domótica.
- ✓ **Banda Ancha:** Se le llama así a la transmisión de datos en el cual se envían simultáneamente varias piezas de información, con el objeto de incrementar la velocidad de transmisión efectiva, este envío de datos debe ser mayor a 600 baudios o símbolos por segundos (bps) para que se le considere como banda ancha. En ingeniería de redes este término se utiliza también para los métodos en donde dos o más señales comparten un medio de transmisión.
- ✓ **Bifurcar:** Separarse en dos ramales, brazos o puntas.
- ✓ **CCITT:** son las siglas de Comité Consultivo Internacional Telegráfico y Telefónico (Consultative Committee for International Telegraphy and Telephony - Comité Consultatif International Télégraphique et Téléphonique), antiguo nombre del comité de normalización de las telecomunicaciones dentro de la UIT (Unión Internacional de Telecomunicaciones) ahora conocido como UIT-T.
- ✓ **CEBus: CEBus**, siglas de **Consumer Electronic bus**, también conocido como **EIA-600**, es un conjunto de estándares y protocolos de comunicación para que los aparatos electrónicos transmitan comandos y datos.
- ✓ **CONSIGNA:** poner en depósito una cosa, señalar y destinar una cantidad determinada.

- ✓ **BatiBUS**: otro conjunto de estándares y protocolos de comunicación para aparatos electrónicos.
- ✓ **EHS:** El estándar EHS (European Home System) se usa para crear una tecnología que permita la implantación de la domótica en el mercado residencial de forma masiva.
- ✓ **CENELEC:** Comité Europeo de Normalización Electrotécnica. La Comisión CENELEC/ENTR/e-Europe/2001-03 es la encargada de elaborar normas a nivel internacional y la organización que ha promocionado el Smart House Forum.
- ✓ **Datagrama:** Es la estructura interna de un paquete de datos. Paquetes de datos que se transfieren en una conexión.
- ✓ **DOMOTICA:** La domótica es la automatización y control centralizado y/o remoto de aparatos y sistemas eléctricos y electrónicos en la vivienda. Los objetivos principales de la domótica es aumentar el confort, ahorrar energía y mejorar la seguridad.
- ✓ **EHSA:** La asociación EHSA (EHS Association) es la encargada de emprender y llevar a cabo diversas iniciativas para aumentar el uso de esta tecnología en las viviendas europeas. Además se ocupa de la evolución y mejora tecnológica del EHS y de asegurar la compatibilidad total entre fabricantes de productos con interface EHS.
- ✓ **FSK:** El FSK (Frequency-shift keying) es un tipo de modulación de frecuencia cuya señal modulante es un flujo de pulsos binarios que varía entre valores predeterminados. En los sistemas de modulación por salto de frecuencia, FSK, la señal moduladora hace variar la frecuencia de la portadora, de modo que la señal modulada resultante codifica la información asociándola a valores de frecuencia diferentes.
- ✓ **FTP:** FTP (File Transfer Protocol) es un protocolo de transferencia de archivos entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar archivos desde él o para enviarle nuestros propios archivos independientemente del sistema operativo utilizado en cada equipo.
- ✓ **HBS:** HBS (Home Bus System). Un consorcio de compañías Japonesas soportado por agencias gubernamentales y asociaciones de negocio con el objetivo de especificar estandards de comunicación en dispositivos domóticos, además de asegurar vía pares trenzados y cables coaxiales la unión de estos con dispositivos telefónicos y audio/video.
- ✓ **IEEE 1284:** Este estándar proporciona una comunicación bidireccional de alta velocidad entre un PC y un periférico externo, estableciendo una comunicación entre 50 y 100 veces más rápida que el original puerto paralelo. Por supuesto es totalmente compatible con todos los periféricos existentes para puertos paralelos. El estándar 1284 define 5 modos de transferencia de datos. Cada modo proporciona un método de transferencia

de datos hacia el exterior (PC a periférico), hacia el interior (periférico a PC) o bidireccional (dúplex).

- ✓ **IP:** El Protocolo de Internet (IP, de sus siglas en inglés Internet Protocol) es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados. Los datos en una red basada en IP son enviados en bloques conocidos como paquetes o datagramas (en el protocolo IP estos términos se suelen usar indistintamente). En particular, en IP no se necesita ninguna configuración antes de que un equipo intente enviar paquetes a otro con el que no se había comunicado antes.
- ✓ **IRRUPCIÓN:** invasión, entrada impetuosa.
- ✓ **KNX/EIB:** Se trata de un protocolo de comunicaciones estándar, multimedio y abierto, Normalizado en Europa (no en América) cuyo ámbito actuación se reduce a viviendas y, en menor medida, edificios. KNX es la resultante de unir otras tres tecnologías, BATIBUS, EIB y EHS, buscando obtener una resultante más potente de lo mejor de cada una de ellas, aunque la base sobre la que se sustenta es EIB (European Installation Bus).
- ✓ **LAMA:** Tira lisa y delgada de una materia dura, especialmente madera, metal o cristal, que se utiliza para dejar pasar más o menos luz o aire, en ventanas, escaparates, etc.
- ✓ **LAN:** Una red de área local, o red local, es la interconexión de varios ordenadores y periféricos. (LAN es la abreviatura inglesa de Local Area Network, 'red de área local'). Su extensión está limitada físicamente a un edificio o a un entorno de hasta 100 metros. Su aplicación más extendida es la interconexión de ordenadores personales y estaciones de trabajo en oficinas, fábricas, etc., para compartir recursos e intercambiar datos y aplicaciones. En definitiva, permite que dos o más máquinas se comuniquen. El término red local incluye tanto el hardware como el software necesario para la interconexión de los distintos dispositivos y el tratamiento de la información.
- ✓ **JDK:** Kit de desarrollo de software para java.
- ✓ **Lonworks:** Estándar de carácter propietario.
- ✓ **Multiplataforma:** Multiplataforma es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86 (solo para equipos Apple) o en un PowerPC.
- ✓ **Multi-room:** son aplicaciones que te dan la posibilidad de llevar la señal de los aparatos digitales como radio, DVD, CD, TV, etc., a varios sectores de la casa. Existen diferentes tipos de aplicaciones pero, en general, todas consisten en un controlador central que recibe las señales y las redistribuye.
- ✓ **Nibble:** En arquitectura de computadoras, un nibble equivale a 4 bits, lo que permite 16 posibles valores (2 elevado a 4). En el contexto de redes o

telecomunicaciones, el nibble es a veces llamado "semiocteto" o "cuarteto" Los nibbles son importantes en las representaciones hexadecimal y BCD, pues ambas utilizan 4 bits para representar cantidades.

- ✓ **OSGi:** Open Services Gateway Initiative. Especificaciones abiertas de software que permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios. Ha sido pensada para su compatibilidad con Jini o UPnP.
- ✓ **Pasarela Residencial:** es en general, la conexión entre los sistemas del interior de la vivienda y el mundo exterior, es decir, redes de comunicaciones (Internet, GSM o tecnologías similares).
- ✓ **Plataforma:** Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos. La plataforma más conocida es probablemente Microsoft Windows en una arquitectura x86; otras plataformas conocidas son GNU/Linux y Mac OS X
- ✓ **Plug-and-play:** Conocida también por su abreviatura **PnP** es la tecnología que permite a un dispositivo informático ser conectado a un ordenador sin tener que configurar jumpers ni proporcionar parámetros a sus controladores. Para que eso sea posible, el sistema operativo con el que funciona el ordenador debe tener soporte para dicho dispositivo.
- ✓ **Polimorfismo:** En programación orientada a objetos se denomina polimorfismo a la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación. Un objeto polimórfico es una entidad que puede contener valores de diferentes tipos durante la ejecución del programa. En la práctica esto quiere decir que un puntero a un tipo puede contener varios tipos diferentes, no solo el creado. De esta forma podemos tener un puntero a un objeto de la clase Trabajador, pero este puntero puede estar apuntando a un objeto subclase de la anterior como podría ser Márketing, Ventas o Recepcionistas (todas ellas deberían ser subclase de Trabajador). El concepto de polimorfismo se puede aplicar tanto a funciones como a tipos de datos. Así nacen los conceptos de funciones polimórficas y tipos polimórficos. Las primeras son aquellas funciones que pueden evaluarse o ser aplicadas a diferentes tipos de datos de forma indistinta; los tipos polimórficos, por su parte, son aquellos tipos de datos que contienen al menos un elemento cuyo tipo no está especificado.
- ✓ **Positivista:** Consiste en no admitir como validos científicamente otros conocimientos, sino los que proceden de la experiencia, rechazando, por tanto, toda noción a priori y todo concepto universal y absoluto.
- ✓ **SCSI:** Acrónimo inglés Small Computer System Interface, es un interfaz estándar para la transferencia de datos entre distintos dispositivos en el bus de la computadora.

- ✓ **TCP:** TCP (Transmission-Control-Protocol, en español Protocolo de Control de Transmisión) es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 - 1974 por Vint Cerf y Robert Kahn. Muchos programas dentro de una red de datos compuesta por computadoras pueden usar TCP para crear conexiones entre ellos a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. TCP da soporte a muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP, SSH y FTP.
- ✓ **TOKEN:** Es un elemento individual en un lenguaje de programación. Es un bloque de texto categorizado. Por ejemplo una marca de puntuación, un operador, un identificador, un número, etc.
- ✓ **TTL:** TTL es la sigla en inglés de Transistor-Transistor Logic o "Lógica Transistor a Transistor". Es una familia lógica o lo que es lo mismo, una tecnología de construcción de circuitos electrónicos digitales. En los componentes fabricados con tecnología TTL los elementos de entrada y salida del dispositivo son transistores bipolares.
- ✓ **UBICUIDAD:** capacidad de estar en todas partes al mismo tiempo.
- ✓ **UDP:** User Datagram Protocol (UDP) es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción. Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos
- ✓ **X10:** Protocolo de comunicaciones para el control remoto de dispositivos eléctricos.

ZigBee: Protocolo de comunicaciones inalámbrico.

10.Bibliografía

1. Fernández Valentín y Ruz Enrique. El hogar digital: Necesidades que atiende servicios que presta Tecnologías que utiliza. España. Ed. Creaciones Copyright, 2005.

2. Gordon Meyer. Domótica Los mejores trucos. Madrid, España. Ed. Anaya, 2005

3. Gadre, Dhananjay V. Programming the parallel port: interfacing the PC for data acquisition and process control. Lawrence, Kansas City. Ed. R&D Books, 1998.

4. Buchanan William. PC interfacing, Communications and Windows Programming. England. Ed. Addison – Wesley, 1999.

5. Andrew S. Tanenbaum. Redes de computadoras, cuarta edición. México. Ed. Pearson Educación, 2003.

6. Casad y Willsey. Aprendiendo TCP/IP en 24 hrs. Ed. Prentice Hall. Pag.88-89

7. Garcia Tomas y Raya Cabrera. Alta velocidad y calidad de servicio en redes IP. Ed. Alfaomega. Pag 369-390.

8. William Stallings. Comunicaciones y redes de computadoras. Séptima edición. Ed. Prentice Hall. Apéndice C

11. Mesografía

1. http://www.adrformacion.com/cursos/wserver/leccion1/tutorial4.html
2. http://www.interfacebus.com/Design_Connector_1284.html
3. http://cfievalladolid2.net/tecno/cyr_01/control/puerto_paralelo.htm
4. http://es.wikipedia.org/wiki/Cable_paralelo
5. http://java.sun.com/products/javacomm/
6. http://java.sun.com/products/java-media/jmf/index.jsp

12. Anexos

12.1 Programa Cliente

```
/*CLIENTE
 *UNAM - FI 2008
*TESIS DE LICENCIATURA EN INGENIERIA ELECTRICA ELECTRINICA MOD. COMUNICACIONES
******ESTE PROGRAMA LEERA Y DESPLEGARA INFROMACION QUE ENVIE EL SERVIDOR, PARA TENER INFORMADO**********
******AL USUARIO DEL ESTADO ACTUAL DE LOS DISPOSITIVOS CONECTADOS AL PUERO PARALELO DEL SERVIDOR **********
*
**********TESIS: Diseño de un sistema domótico para el control y supervisión **********
**********remotos de dispositivos caseros por medio de Internet**********
*****************************************************************************************************
*****************************************************************************************************/

// EL CLIENTE LEE Y DESPLIEGA INFROMACION QUE ENVIA EL SERVIDOR.
import java.io.*;
import java.util.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Client extends JFrame implements ActionListener {
   //DECLARACION DE VARIALBES
   private JLabel label1;
   private JPanel centerPanel;
   private JTextField enterField;
   private JTextArea displayArea;
   private ObjectOutputStream output;
   private ObjectInputStream input;
   private String message = "";
   private String chatServer;
   private Socket client;
   private String arrayTokens[] = new String[30]; /*Arreglo que guarda los tokens de la cadena que viene del cliente*/
   int i=0;   //VARIABLE PARA EL INDICE DE arrayTokens
   int j=0; //NUMERO DE TOKENS
   private int bitIN1;
   private int bitIN2;
   private int bitIN3;
   private int bitIN4;
   //BOTONES DE LA APLICACION
   private JButton bit_IN1;
   private JButton bit_IN2;
   private JButton bit_IN3;
   private JButton bit_IN4;
   private JButton bit_OUT1;
   private JButton bit_OUT2;
   private JButton bit_OUT3;
   private JButton bit_OUT4;
   private JButton bit_OUT5;
   private JButton bit_OUT6;
   private JButton bit_OUT7;
   private JButton bit_OUT8;
   private JButton guarda;
   private JButton recupera;
   private JButton refresh;
   private JButton ayuda;
   //ETIQUETAS DE LA APLICACION
   private JTextField lbit_IN1;
   private JTextField lbit_IN2;
   private JTextField lbit_IN3;
   private JTextField lbit_IN4;
   private JTextField lbit_OUT1;
   private JTextField lbit_OUT2;
   private JTextField lbit_OUT3;
   private JTextField lbit_OUT4;
   private JTextField lbit_OUT5;
   private JTextField lbit_OUT6;
   private JTextField lbit_OUT7;
```

```
   private JTextField lbit_OUT8;

   // INICIALIZA EL CLIENTE Y CONFIGURA LA GUI
   public Client( String host )
   {
     super( "CLIENTE ==> UNAM-FI 2008" );

     chatServer = host; // COLOCA EL SERVIDOR PARA QUE EL CLIENTE SE CONECTE

     Container container = getContentPane();

/*******************************************************************************/
                        // CREA LA ETIQUETA CON EL ESCUDO
                        Icon imagen = new ImageIcon( "ESCUDO ING.jpg" );
                        label1 = new JLabel();
       label1.setText( "FACULTAD DE INGENIERIA" );
       label1.setIcon( imagen );
       label1.setHorizontalTextPosition( SwingConstants.CENTER );
       label1.setVerticalTextPosition( SwingConstants.BOTTOM );
       label1.setToolTipText( "UNAM - TESIS" );
                        label1.setHorizontalAlignment( SwingConstants.CENTER ); //CENTRA LA ETIQUETA EN LA VENTANA

                        Box vertical1 = Box.createVerticalBox();
       Box vertical2 = Box.createVerticalBox();

        // CREA strut Y AÑADE buttons A EL  Box vertical1 (IZQUIERDO)*********

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT1 = new JButton( "OUT1 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT1 = new JTextField( 10 ) );
                        lbit_OUT1.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT2 = new JButton( "OUT2 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT2 = new JTextField( 10 ) );
                        lbit_OUT2.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT3 = new JButton( "OUT3 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT3 = new JTextField( 10 ) );
                        lbit_OUT3.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT4 = new JButton( "OUT4 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT4 = new JTextField( 10 ) );
                        lbit_OUT4.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT5 = new JButton( "OUT5 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT5 = new JTextField( 10 ) );
                        lbit_OUT5.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT6 = new JButton( "OUT6 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT6 = new JTextField( 10 ) );
                        lbit_OUT6.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT7 = new JButton( "OUT7 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT7 = new JTextField( 10 ) );
                        lbit_OUT7.setBackground(Color.red);

              vertical1.add( Box.createVerticalStrut( 10 ) );
              vertical1.add( bit_OUT8 = new JButton( "OUT8 On/Off" ) );
              //AÑADE LA ETQUETA DE TEXTO
              vertical1.add( lbit_OUT8 = new JTextField( 10 ) );
                        lbit_OUT8.setBackground(Color.red);
```

```
// CREA strut Y AÑADE buttons A EL  Box vertical2                (DERECHO)**********
     vertical2.add( Box.createVerticalStrut( 10 ) );
        vertical2.add( bit_IN1 = new JButton( "BIT_IN1" ) );
                bit_IN1.setEnabled( false );
        //AÑADE LA ETQUETA DE TEXTO
        vertical2.add( lbit_IN1 = new JTextField( 10 ) );
                lbit_IN1.setBackground(Color.green);

        vertical2.add( Box.createVerticalStrut( 10 ) );
        vertical2.add( bit_IN2 = new JButton( "BIT_IN2" ) );
                bit_IN2.setEnabled( false );
        //AÑADE LA ETQUETA DE TEXTO
        vertical2.add( lbit_IN2 = new JTextField( 10 ) );
                lbit_IN2.setBackground(Color.green);

        vertical2.add( Box.createVerticalStrut( 10 ) );
        vertical2.add( bit_IN3 = new JButton( "BIT_IN3" ) );
                bit_IN3.setEnabled( false );
        //AÑADE LA ETQUETA DE TEXTO
        vertical2.add( lbit_IN3 = new JTextField( 10 ) );
                lbit_IN3.setBackground(Color.green);

        vertical2.add( Box.createVerticalStrut( 10 ) );
        vertical2.add( bit_IN4 = new JButton( "BIT_IN4" ) );
                bit_IN4.setEnabled( false );
        //AÑADE LA ETQUETA DE TEXTO
        vertical2.add( lbit_IN4 = new JTextField( 10 ) );
                lbit_IN4.setBackground(Color.green);


//SE CREA UN PANEL PARA LOS BOTONES VERTICALES
JPanel panel = new JPanel();
panel.setLayout( new BoxLayout( panel, BoxLayout.Y_AXIS ) );

// CREA panelBotones Y AÑADE buttons A EL   (INFERIOR)******************************
JPanel panelBotones = new JPanel();
panelBotones.setLayout ( new GridLayout(2,2,50,20) );
        panelBotones.add(guarda = new JButton( "GUARDAR DESCRIPCION" ));
        panelBotones.add(recupera = new JButton( "RECUPERA DESCRIPCION" ));
        panelBotones.add(refresh = new JButton( "ACTUALIZA BIT_STATE" ));
        panelBotones.add(ayuda = new JButton( "AYUDA" ));

        //SE AÑADEN LOS PANELES AL CONTENEDOR
        container.add(vertical1, BorderLayout.WEST);
        container.add(vertical2, BorderLayout.EAST);
        container.add( label1, BorderLayout.NORTH  );
        container.add( panelBotones, BorderLayout.SOUTH  );

        //PANEL CENTRAL
          centerPanel = new JPanel();
          centerPanel.setLayout( new GridLayout( 2, 1, 20, 100) );

// CREA enterField Y EL register listener
enterField = new JTextField();
enterField.setEditable( false );
enterField.addActionListener(
  new ActionListener() {

    // ENVIA MENSAJE AL SERVIDOR
    public void actionPerformed( ActionEvent event )
    {
      sendData( event.getActionCommand() );
      enterField.setText( "" );
    }
  }
);

centerPanel.add( enterField );

// CREA displayArea
displayArea = new JTextArea();
centerPanel.add( new JScrollPane( displayArea ) );


        //SE AÑADEN LOS PANELES AL CONTENEDOR
        container.add( vertical1, BorderLayout.WEST);
```

```
            container.add( vertical2, BorderLayout.EAST);
            container.add( label1, BorderLayout.NORTH  );
            container.add( centerPanel, BorderLayout.CENTER);

              //PERZONALIZAR EL CONTENEDOR
   setSize( 800, 700 );
   container.setBackground(Color.BLUE);
   panelBotones.setBackground(Color.BLUE);
   centerPanel.setBackground(Color.BLUE);
   setLocationRelativeTo(null); //PARA CENTRAR LA VENTANA

   setVisible( true );
   //setExtendedState(MAXIMIZED_BOTH);
   //setExtendedState(ICONIFIED);

} // TERMINA EL constructor DEL CLIENTE

// SE CONECTA AL SERVIDOR Y PROCESA LOS MENSAJES DEL SERVIDOR
private void runClient()
{
   // connect to server, get streams, process connection
   try {
      connectToServer(); // Step 1: Create a Socket to make connection
      getStreams();      // Step 2: Get the input and output streams
      processConnection(); // Step 3: Process connection
   }

   // server closed connection
   catch ( EOFException eofException ) {
      System.err.println( "Client terminated connection" );
   }

   // process problems communicating with server
   catch ( IOException ioException ) {
      ioException.printStackTrace();
   }

   finally {
      closeConnection(); // Step 4: Close connection
   }

} // end method runClient

// connect to server
private void connectToServer() throws IOException
{
   displayMessage( "Attempting connection\n" );

   // create Socket to make connection to server
   client = new Socket( InetAddress.getByName( chatServer ), 12345 );

   // display connection information
   displayMessage( "Connected to: " +
      client.getInetAddress().getHostName() );
}

// get streams to send and receive data
private void getStreams() throws IOException
{
   // set up output stream for objects
   output = new ObjectOutputStream( client.getOutputStream() );
   output.flush(); // flush output buffer to send header information

   // set up input stream for objects
   input = new ObjectInputStream( client.getInputStream() );

   displayMessage( "\nGot I/O streams\n" );
}

// process connection with server
private void processConnection() throws IOException
{
   // enable enterField so client user can send messages
   setTextFieldEditable( true );

   //**************************** añade EL LISTENER A CADA BOTON
```

```
bit_OUT1.addActionListener(this);
bit_OUT2.addActionListener(this);
bit_OUT3.addActionListener(this);
bit_OUT4.addActionListener(this);
bit_OUT5.addActionListener(this);
bit_OUT6.addActionListener(this);
bit_OUT7.addActionListener(this);
bit_OUT8.addActionListener(this);
guarda.addActionListener(this);
recupera.addActionListener(this);
refresh.addActionListener(this);
ayuda.addActionListener(this);

        //INICIALIZA EL CLIENTE CON LOS ESTADOS DEL SERVIDOR
        message = "refresh";
        sendData( message );
        //*************************************************

do { // process messages sent from server

  // read message and display it
  try
  {
       message = ( String ) input.readObject();
    displayMessage( "\n" + message );

    //INTERPRETE
                /**********ALMACENANDO CADA TOKEN Y LO PROCESA***********/
                StringTokenizer tokens =
                  new StringTokenizer( message , "," );

                                j=tokens.countTokens();
                System.out.println( "Numero de tokens: " + tokens.countTokens() + "\nLos tokens son:\n" );

                while ( tokens.hasMoreTokens() )              //CICLO PARA LEER LOS TOKENS Y ALLACENAR C/TOKEN EN UN
ARREGLO
                {

                  arrayTokens[i] = tokens.nextToken() ;

                  System.out.println( i + " " + arrayTokens[i] + "\n" );
                  i++;

                }

                i=0; //RESET THE COUNTER i TO THE NEW ARRAY_TOKEN

                if  (j > 10)
                {
                   System.out.println( "EL Numero de tokens ES MAYYOR A 10: " + j);
                        if ( arrayTokens[1].equals("guarda") ) //FILTRO DE COMANDOS
                        {
                                System.out.println( "EL COMANDO DEL ARCHIVO ES= " + arrayTokens[1] + "PERO EL ACTUAL
COMANDO ES recupera");
                                        lbit_IN1.setText(arrayTokens[10]);
                                                                lbit_IN2.setText(arrayTokens[11]);
                                                                lbit_IN3.setText(arrayTokens[12]);
                                                                lbit_IN4.setText(arrayTokens[13]);
                                                                lbit_OUT1.setText(arrayTokens[2]);
                                                                lbit_OUT2.setText(arrayTokens[3]);
                                                                lbit_OUT3.setText(arrayTokens[4]);
                                                                lbit_OUT4.setText(arrayTokens[5]);
                                                                lbit_OUT5.setText(arrayTokens[6]);
                                                                lbit_OUT6.setText(arrayTokens[7]);
                                                                lbit_OUT7.setText(arrayTokens[8]);
                                                                lbit_OUT8.setText(arrayTokens[9]);
                        }
                        else
                        {
                                System.out.println( "EL COMANDO NO FUE RECONOCIDO");
                        }
                }
                else if ( j > 2  ) //FILTRO DE COMANDOS
                 {
                        if(arrayTokens[1].equals("EL_BYTE_DE_SALIDA_Y_ENTRADA_ES:")) //FILTRO DE COMANDO  refresh
                         {
```

```
                                        System.out.println( "BYTE_OUT: " + arrayTokens[2]);
                                        decBinRefresh (arrayTokens[2], "OUT");
                                        System.out.println( "BYTE_IN: " + arrayTokens[3]);
                                        decBinRefresh (arrayTokens[3], "IN");
                                }

                        }

      }

      // catch problems reading from server
      catch ( ClassNotFoundException classNotFoundException ) {
         displayMessage( "\nUnknown object type received" );
      }

   } while ( !message.equals( "SERVER>>> TERMINATE" ) );

} // end method processConnection

// close streams and socket
private void closeConnection()
{
   displayMessage( "\nClosing connection" );
   setTextFieldEditable( false ); // disable enterField

   try {
      output.close();
      input.close();
      client.close();
   }
   catch( IOException ioException ) {
      ioException.printStackTrace();
   }
}

// send message to server
private void sendData( String message )
{
   // send object to server
   try {
      output.writeObject( "CLIENT>>> " + message.length() + " " +  message );
      output.flush();
      displayMessage( "\nCLIENT>>> " + message.length() + " " + message );
   }

   // process problems sending object
   catch ( IOException ioException ) {
      displayArea.append( "\nError writing object" );
   }
}

// utility method called from other threads to manipulate
// displayArea in the event-dispatch thread
private void displayMessage( final String messageToDisplay )
{
   // display message from GUI thread of execution
   SwingUtilities.invokeLater(
      new Runnable() {  // inner class to ensure GUI updates properly

         public void run() // updates displayArea
         {
            displayArea.append( messageToDisplay );
            displayArea.setCaretPosition(
               displayArea.getText().length() );
         }

      }  // end inner class

   ); // end call to SwingUtilities.invokeLater
}

// utility method called from other threads to manipulate
// enterField in the event-dispatch thread
private void setTextFieldEditable( final boolean editable )
{
   // display message from GUI thread of execution
```

```
   SwingUtilities.invokeLater(
      new Runnable() {  // inner class to ensure GUI updates properly

         public void run()  // sets enterField's editability
         {
            enterField.setEditable( editable );
         }

      }  // end inner class

   ); // end call to SwingUtilities.invokeLater
}

//PARA ESCUCHAR LOS EVENTOS DE LOS BOTONES
public void actionPerformed (ActionEvent evento)
{
            if ( evento.getSource() == bit_OUT1 )
            {
                        if ( lbit_OUT1.getBackground() == Color.red )
                        {
                                    message = "prender 1";
                                    sendData( message );
                                    lbit_OUT1.setBackground(Color.green);
                        }
                        else
                        {
                                    message = "apagar 1";
                                    sendData( message );
                                    lbit_OUT1.setBackground(Color.red);
                        }
            }
            else if       ( evento.getSource() == bit_OUT2 )
            {
                        if ( lbit_OUT2.getBackground() == Color.red )
                        {
                                    message = "prender 2";
                                    sendData( message );
                                    lbit_OUT2.setBackground(Color.green);
                        }
                        else
                        {
                                    message = "apagar 2";
                                    sendData( message );
                                    lbit_OUT2.setBackground(Color.red);
                        }

            }
            else if       ( evento.getSource() == bit_OUT3 )
            {
                        if ( lbit_OUT3.getBackground() == Color.red )
                        {
                                    message = "prender 4";
                                    sendData( message );
                                    lbit_OUT3.setBackground(Color.green);
                        }

                        else
                        {
                                    message = "apagar 4";
                                    sendData( message );
                                    lbit_OUT3.setBackground(Color.red);
                        }
            }
            else if ( evento.getSource() == bit_OUT4 )
            {
                        if ( lbit_OUT4.getBackground() == Color.red )
                        {
                                    message = "prender 8";
                                    sendData( message );
                                    lbit_OUT4.setBackground(Color.green);
                        }
                        else
                        {
                                    message = "apagar 8";
                                    sendData( message );
                                    lbit_OUT4.setBackground(Color.red);
```

```
                }
        }
        else if ( evento.getSource() == bit_OUT5 )
        {
                if ( lbit_OUT5.getBackground() == Color.red )
                {
                        message = "prender 16";
                        sendData( message );
                        lbit_OUT5.setBackground(Color.green);
                }
                else
                {
                        message = "apagar 16";
                        sendData( message );
                        lbit_OUT5.setBackground(Color.red);
                }
        }
        else if ( evento.getSource() == bit_OUT6 )
        {
                if ( lbit_OUT6.getBackground() == Color.red )
                {
                        message = "prender 32";
                        sendData( message );
                        lbit_OUT6.setBackground(Color.green);
                }
                else
                {
                        message = "apagar 32";
                        sendData( message );
                        lbit_OUT6.setBackground(Color.red);
                }
        }
        else if ( evento.getSource() == bit_OUT7 )
        {
                if ( lbit_OUT7.getBackground() == Color.red )
                {
                        message = "prender 64";
                        sendData( message );
                        lbit_OUT7.setBackground(Color.green);
                }

                else
                {
                        message = "apagar 64";
                        sendData( message );
                        lbit_OUT7.setBackground(Color.red);
                }

        }
        else if ( evento.getSource() == bit_OUT8 )
        {
                if ( lbit_OUT8.getBackground() == Color.red )
                {
                        message = "prender 128";
                        sendData( message );
                        lbit_OUT8.setBackground(Color.green);
                }

                else
                {
                        message = "apagar 128";
                        sendData( message );
                        lbit_OUT8.setBackground(Color.red);
                }
        }
        else if ( evento.getSource() == guarda )
        {
                message = "guarda ,guarda," + lbit_OUT1.getText() + "," + lbit_OUT2.getText() + "," + lbit_OUT3.getText() + "," +
lbit_OUT4.getText()
                                + "," + lbit_OUT5.getText() + "," + lbit_OUT6.getText() + "," +
lbit_OUT7.getText() + "," + lbit_OUT8.getText()
                                + "," + lbit_IN1.getText() + "," + lbit_IN2.getText() + "," +
lbit_IN3.getText() + "," + lbit_IN4.getText();
                System.out.println (message);
                sendData( message );
        }
```

```
		else if ( evento.getSource() == recupera )
		{
			message = "recupera";
			sendData( message );
		}
		else if ( evento.getSource() == refresh )
		{
			message = "refresh";
			sendData( message );
		}
		else if ( evento.getSource() == ayuda )
		{
			JTextArea areaOut = new JTextArea( 17, 20 );
			JScrollPane desplazador = new JScrollPane( areaOut );
			String ruta = "./hlp";
			File help = new File(ruta);
				areaOut.setText( fileToString(help) );
				JOptionPane.showMessageDialog( null, desplazador, "Help and About Tesis FI...",
JOptionPane.INFORMATION_MESSAGE );

		}

		else
		{
			System.out.println("EL BOTON NO SE ENCONTRO");
		}

	}

		//CONVIERTE UN NUMERO DECIMAL A BINARIO Y ACTULIZA EL ESTADO EN EL CLIENTE
	public void decBinRefresh (String convertirNUM, String type)
	{

		int num = Integer.parseInt(convertirNUM);
		int z,x;
		Integer nn2[] = new Integer[50];

		nn2[0]=0;
		nn2[1]=0;
		nn2[2]=0;
		nn2[3]=0;
		nn2[4]=0;
		nn2[5]=0;
		nn2[6]=0;
		nn2[7]=0;

			for (x=0;x<50;x++)
			{
				nn2[x]=num%2;
				num=num/2;
				System.out.println( " nn2=" + nn2[x] +"  x=" + x +  "num=" + num );
				if(num==0)
					break;
			}
			System.out.println("LA CONVERCION DE " + convertirNUM + " ES ");
			for(z=x;z>=0;z--)
			{
			System.out.print( nn2[z] );
			}

		//ALALIZA CADA BIT PARA PONER LAS ETIQUETAS EN ROJO O EN VERDE.

			if ( type.equals("IN") )
			{
				if (  nn2[6] == 0)

					lbit_IN1.setBackground(Color.red);
				else
					lbit_IN1.setBackground(Color.green);

				if (  nn2[5] == 0)

					lbit_IN2.setBackground(Color.red);
				else
					lbit_IN2.setBackground(Color.green);
```

```
                        if (  nn2[4] == 0)

                                lbit_IN3.setBackground(Color.red);
                        else
                                lbit_IN3.setBackground(Color.green);

                        if (  nn2[3] == 0)

                                lbit_IN4.setBackground(Color.red);
                        else
                                lbit_IN4.setBackground(Color.green);

                }else if ( type.equals("OUT") )
                        {
                        if (  nn2[0] == 0)

                                lbit_OUT1.setBackground(Color.red);
                        else
                                lbit_OUT1.setBackground(Color.green);

                        if (  nn2[1] == 0)

                                lbit_OUT2.setBackground(Color.red);
                        else
                                lbit_OUT2.setBackground(Color.green);

                        if (  nn2[2] == 0)

                                lbit_OUT3.setBackground(Color.red);
                        else
                                lbit_OUT3.setBackground(Color.green);

                        if (  nn2[3] == 0)

                                lbit_OUT4.setBackground(Color.red);
                        else
                                lbit_OUT4.setBackground(Color.green);

                                if (  nn2[4] == 0)

                                lbit_OUT5.setBackground(Color.red);
                        else
                                lbit_OUT5.setBackground(Color.green);

                                if (  nn2[5] == 0)

                                lbit_OUT6.setBackground(Color.red);
                        else
                                lbit_OUT6.setBackground(Color.green);

                                if (  nn2[6] == 0)

                                lbit_OUT7.setBackground(Color.red);
                        else
                                lbit_OUT7.setBackground(Color.green);

                                if (  nn2[7] == 0)

                                lbit_OUT8.setBackground(Color.red);
                        else
                                lbit_OUT8.setBackground(Color.green);

                                }
}//FIN decBinRefresh

//READ A FILE TO MAKE A STRING
public String fileToString(File file)
{
        StringBuffer buffer = new StringBuffer();
        String line;
        FileReader fReader;
        BufferedReader bReader;

                try {
                                fReader = new FileReader(file);
                                bReader = new BufferedReader(fReader);
```

```
                                        while ((line = bReader.readLine()) != null){
                                                buffer.append(line + "\n");
                                        }
                                        bReader.close();
                                        fReader.close();
                        } catch (FileNotFoundException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }

                return buffer.toString();
    }//END fileToString


    public static void main( String args[] )
    {
      Client application;

      if ( args.length == 0 )
        application = new Client( "127.0.0.1" );
      else
        application = new Client( args[ 0 ] );

      application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      application.runClient();
    }

} // end class Client
```

12.2 Programa Servidor

```
/*SERVIDOR
 *UNAM - FI 2008
*TESIS DE LICENCIATURA EN INGENIERIA ELECTRICA ELECTRINICA MOD. COMUNICACIONES
******ESTE RECIBIRA LA CONEXION DEL CLIENTE, LE ENVIARA UNA CADENA Y CERRARA LA CONEXION**********************
*
**********************TESIS: Diseño de un sistema domótico para el control y supervisión **********************
**************************remotos de dispositivos caseros por medio de Internet*******************************
*****************************************************************************************************************
*****************************************************************************************************************/

import parport.ParallelPort;
import java.util.*;
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Server extends JFrame {
   private JTextField enterField;
   private JTextArea displayArea;
   private ObjectOutputStream output;
   private ObjectInputStream input;
   private ServerSocket server;
   private Socket connection;
   private int counter = 1;
   private String arrayTokens[] = new String[10]; /*Arreglo que guarda los tokens de la cadena que viene del cliente*/
   private String message = "prender LA TV";
   private String stringtState = "";    //ESTADO ACTUAL DEL BYTE DE SALIDA Y ENTRADA
   int i=0;   //VARIABLE PARA EL INDICE DE arrayTokens
   int aByte=0; // VARIALBE PARA ALMACENAR EL BYTE_OUT ACTUAL DEL PARALLEL PORT
   int mByte=0; // VARIALBE PARA ALMACENAR EL BYTE MASCARA
   private int currentState = 0;   //ESTADO ACTUAL DEL BYTE_IN DEL PARALLEL PORT

   // CONFIGURA LAGUI
   public Server()
   {
      super( "SERVIDOR ==> UNAM-FI 2008" );
```

```
   Container container = getContentPane();

   // CREA enterField Y EL register listener
   enterField = new JTextField();
   enterField.setEditable( false );
   enterField.addActionListener(
     new ActionListener() {

       // ENVIA MENSAJES AL CLIENTE
       public void actionPerformed( ActionEvent event )
       {
         sendData( event.getActionCommand() );
         enterField.setText( "" );
       }
     }
   );

   container.add( enterField, BorderLayout.NORTH );

   // CREA displayArea
   displayArea = new JTextArea();
   container.add( new JScrollPane( displayArea ),
     BorderLayout.CENTER );

   setSize( 300, 150 );
   setVisible( true );

} // TERMINA EL CONSTRUCTOR DELSERVIDOR

// CONFIGURA Y ARRANCA EL SERVIDOR
public void runServer()
{
   // PREPARA AL SERVIDOR PARA RECIVIR CONECCIONES Y LAS PROCESA
   try {

     // Step 1: Create a ServerSocket.
     server = new ServerSocket( 12345, 100 );

     while ( true ) {

       try {
         waitForConnection(); // Step 2: Wait for a connection.
         getStreams();        // Step 3: Get input & output streams.
         processConnection(); // Step 4: Process connection.
       }

       // process EOFException when client closes connection
       catch ( EOFException eofException ) {
         System.err.println( "Server terminated connection" );
       }

       finally {
         closeConnection();   // Step 5: Close connection.
         ++counter;
       }

     } // end while

   } // end try

   // process problems with I/O
   catch ( IOException ioException ) {
     ioException.printStackTrace();
   }

} // end method runServer

// wait for connection to arrive, then display connection info
private void waitForConnection() throws IOException
{
   displayMessage( "Waiting for connection\n" );
   connection = server.accept(); // allow server to accept connection
   displayMessage( "Connection " + counter + " received from: " +
     connection.getInetAddress().getHostName() );
}
```

```
// get streams to send and receive data
private void getStreams() throws IOException
{
   // set up output stream for objects
   output = new ObjectOutputStream( connection.getOutputStream() );
   output.flush(); // flush output buffer to send header information

   // set up input stream for objects
   input = new ObjectInputStream( connection.getInputStream() );

   displayMessage( "\nGot I/O streams\n" );
}

// PROCESA CONEXION CON EL CLIENTE
private void processConnection() throws IOException
{
            File Description = new File("Description.txt"); //SAVE THE DESCRIPTION OF THE DATAS OF CLIENT
            ParallelPort lpt1 = new ParallelPort(0x378); // 0x378 is normally the base address for the LPT1 port

            if (counter == 1)
                  lpt1.write(0); // write a byte to star the port's DATA pins with 00000000

   // send connection successful message to client
   String message = "Connection successful";
   sendData( message );

   // enable enterField so server user can send messages
   setTextFieldEditable( true );

   do { // process messages sent from client

      // read message and display it
      try {
         message = ( String ) input.readObject();
         displayMessage( "\n" + message );

            //INTERPRETE
                  /**********ALMACENANDO CADA TOKEN Y LO PROCESA**********/
                  StringTokenizer tokens =
                     new StringTokenizer( message );

                  System.out.println( "Numero de tokens: " + tokens.countTokens() + "\nLos tokens son:\n" );

                  while ( tokens.hasMoreTokens() )          //CICLO PARA LEER LOS TOKENS Y ALLACENAR C/TOKEN EN UN
ARREGLO
                  {

                    arrayTokens[i] = tokens.nextToken() ;

                    System.out.println( i + " " + arrayTokens[i] + "\n" );
                    i++;

                  }

                  i=0; //RESET THE COUNTER i TO THE NEW ARRAY_TOKEN

                  if ( arrayTokens[2].equals("apagar") ) //FILTRO DE COMANDOS
                  {
                              System.out.println( "EL COMANDO= " + arrayTokens[2] + "ARG1(mascara)= " + arrayTokens[3]);
                              mByte = Integer.parseInt(arrayTokens[3]);
                              System.out.println("LOS VALORES DE aByte= " + aByte + "  mByte= " + mByte );
                              aByte = aByte ^ mByte;
                              System.out.println("EL RESUTADO DE LA OPERACION Xor = " + aByte );

                              lpt1.write(aByte); // write a byte to the port's DATA pins
                              System.out.println("EL BYTE SE APAGO CORRECTAMENTE EN EL PUERTO...");
                              sendData( message );

                  }
                  else if     ( arrayTokens[2].equals("prender") )
                  {
                              System.out.println( "EL COMANDO= " + arrayTokens[2] + "ARG1(mascara)= " + arrayTokens[3]);
                              mByte = Integer.parseInt(arrayTokens[3]);
                              System.out.println("LOS VALORES DE aByte= " + aByte + "  mByte= " + mByte );
                              aByte = aByte | mByte;
                              System.out.println("EL RESUTADO DE LA OPERACION or = " + aByte );
```

```
                        lpt1.write(aByte); // write a byte to the port's DATA pins
                        System.out.println("EL BYTE SE PRENDIO CORRECTAMENTE EN EL PUERTO...");
                        sendData( message );
                }
                else if ( arrayTokens[2].equals("refresh") )            //REFRESCA EL ESTADO DE LOS BITS DE ENTRADA y SALIDA
                {
                        System.out.println( "EL COMANDO FUE " + arrayTokens[2]);
                        System.out.println( "EL OUT_BYTE ACTUAL ES " + aByte);
                        currentState = lpt1.read();          //READ a byte from the port's STATUS pins (0x378 + 1)
                        System.out.println( "EL IN_BYTE ACTUAL ES " + currentState);
                        stringtState = ",EL_BYTE_DE_SALIDA_Y_ENTRADA_ES:," + Integer.toString(aByte) + "," +
Integer.toString(currentState) ;
                        sendData( stringtState );

                }
                else if ( arrayTokens[2].equals("guarda") )             //GUARDA LA DESCRIPCION DE LAS ETIQUETAS EN EL FILE
Description.txt
                {
                        System.out.println( "EL COMANDO FUE " + arrayTokens[2]);
                        stringToFile(message , Description);
                        System.out.println( "LA DESCRIPCION SE GUARDO EN EL ARCHIVO CORRECTAMENTE..." );
                        sendData(message);
                }
                else if ( arrayTokens[2].equals("recupera") )           //RECUPERA LA DESCRIPCION DE LAS ETIQUETAS, GUARDA
EN EL FILE Description.txt
                {
                        System.out.println( "EL COMANDO FUE " + arrayTokens[2]);
                        sendData(fileToString(Description));
                        System.out.println( "EL ARCHIVO SE LEYO CORRECTAMENTE");
                        //sendData(message);
                }
                else
                        System.out.println( "LA PALABRA " + arrayTokens[2] + " NO FUE RECONOCIDA COMO COMANDO." );


        } //end try

        // catch problems reading from client
        catch ( ClassNotFoundException classNotFoundException ) {
           displayMessage( "\nUnknown object type received" );
        }

     } while ( !message.equals( "CLIENT>>> TERMINATE" ) );

  } // end method processConnection

  // close streams and socket
  private void closeConnection()
  {
     displayMessage( "\nTerminating connection\n" );
     setTextFieldEditable( false ); // disable enterField

     try {
        output.close();
        input.close();
        connection.close();
     }
     catch( IOException ioException ) {
        ioException.printStackTrace();
     }
  }

  // send message to client
  private void sendData( String message )
  {
     // send object to client
     try {
        output.writeObject( "SERVER>>> " + message );
        output.flush();
        displayMessage( "\nSERVER>>> " + message );
     }

     // process problems sending object
     catch ( IOException ioException ) {
        displayArea.append( "\nError writing object" );
```

```
   }
}

// utility method called from other threads to manipulate
// displayArea in the event-dispatch thread
private void displayMessage( final String messageToDisplay )
{
   // display message from event-dispatch thread of execution
   SwingUtilities.invokeLater(
      new Runnable() {  // inner class to ensure GUI updates properly

         public void run() // updates displayArea
         {
            displayArea.append( messageToDisplay );
            displayArea.setCaretPosition(
               displayArea.getText().length() );
         }

      }  // end inner class

   ); // end call to SwingUtilities.invokeLater
}

// utility method called from other threads to manipulate
// enterField in the event-dispatch thread
private void setTextFieldEditable( final boolean editable )
{
   // display message from event-dispatch  thread of execution
   SwingUtilities.invokeLater(
      new Runnable() {  // inner class to ensure GUI updates properly

         public void run()  // sets enterField's editability
         {
            enterField.setEditable( editable );
         }

      }  // end inner class

   ); // end call to SwingUtilities.invokeLater
}

//READ A FILE TO MAKE A STRING
public String fileToString(File file)
{
            StringBuffer buffer = new StringBuffer();
            String line;
            FileReader fReader;
            BufferedReader bReader;

                        try {
                                                fReader = new FileReader(file);
                                                bReader = new BufferedReader(fReader);
                                                while ((line = bReader.readLine()) != null){
                                                            buffer.append(line + "\n");
                                                }
                                                bReader.close();
                                                fReader.close();
                        } catch (FileNotFoundException e) {
                                    e.printStackTrace();
                        } catch (IOException e) {
                                    e.printStackTrace();
                        }

            return buffer.toString();
}

//READ A STRING TO MAKE A FILE
public static void stringToFile(String string, File file)
            {
                        FileWriter fWriter;
                        BufferedWriter bWriter;

                        try {
                                    fWriter = new FileWriter(file);
                                    bWriter = new BufferedWriter(fWriter);
                                    bWriter.write(string);
```

```
                                bWriter.close();
                                fWriter.close();
                        } catch (IOException e) {
                                e.printStackTrace();
                          }
                }

  public static void main( String args[] )
  {
    Server application = new Server();
    application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    application.runServer();
  }

}  // end class Server
```