Diseño de un sistema domótico para control por Internet de aparatos sin acceso remoto

Osvaldo Nicolás Rodríguez Argüello

24 de septiembre de 2003

Índice general

Índice de figuras

Índice de cuadros

Objetivo

Diseñar un sistema que aproveche las capacidades de una PC común o estándar para permitir el control remoto de dispositivos caseros que actualmente no tiene acceso a Internet, con un costo mínimo; para cumplir con este requisito se plantea agregar el menor hardware posible y utilizar el máximo el hardware de la PC, el desarrollo del software se hará sobre una plataforma libre de regalías y licencias. Se realizará el diseño conceptual de un sistema que pueda utilizarse en las tareas más comunes que se definen en la Domótica.

Resumen

En el presente trabajo nos proponemos diseñar un sistema que nos permita controlar aparatos domésticos comunes de manera remota a través de Internet, 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. Actualmente existen varios productos que ofrecen este servicio pero usando sistemas embebidos propietarios, por consecuencia estos son costosos y difíciles de integrar. Nuestra propuesta es utilizar la computadora personal que cada vez se encuentra con más frecuencia en casas y oficinas para realizar el trabajo de la adquisición de los datos, la operación de los actuadores y la comunicación a través de Internet, estos procesos se pueden realizar mientras el usuario continúa con el trabajo que cotidianamente realiza en su PC y que normalmente no implican un uso extensivo de los recursos de la misma, en otras palabras, la computadora es capaz de atender las necesidades de nuestro sistema además de realizar las tareas que normalmente el usuario le solicita. Una vez que el sistema este funcionando, el usuario podrá solicitar en cualquier computadora conectada a Internet y con un pequeño programa "cliente", el estado de los aparatos conectados en su casa u oficina y de así desearlo, prender o apagar cualquiera de ellos.

Se tratará en lo posible de no incluir modificaciones muy grandes a la arquitectura original de la PC, ni tampoco agregar más hardware, con la finalidad de que el sistema no sea demasiado caro y que además pueda ser utilizado en cualquier computadora, adicionalmente a esto, el desarrollo del software se hará con herramientas libres de licencias y regalías para seguir en la idea de reducción de costos y extensión de uso.

El desarrollo de la interfase entre la computadora y los aparatos así como los programas cliente y servidor serán probados en diferentes condiciones y elemento por elemento, de esta manera podremos evaluar su comportamiento y corregir, si se presentan, errores u omisiones ocultos en el proceso original de diseño.

Capítulo 1

Introducción

En la actualidad las comunicaciones digitales y en particular las redes de computadoras se han extendido de manera muy intensa, participando ahora de manera fundamental en la vida cotidiana de las personas, inicialmente usábamos este tipo de tecnologías en nuestros trabajos que era el único lugar donde teníamos acceso a estos equipos. Con el incesante abaratamiento de las computadoras y de las redes de comunicación entre estas, poco a poco estas máquinas han entrado a muchos lugares que antes no imaginábamos; ahora es muy frecuente encontrar computadoras con acceso a Internet en nuestras casas, centros comerciales, tiendas e incluso cafés y restaurantes. Sin embargo, en la mayoría de los casos, estos equipos son subutilizados ya que solo realizan tareas principalmente de procesamiento de textos, edición de hojas de calculo y navegación por Internet.

Por otro lado la vida moderna de la gente de las ciudades propicia que pasen la mayoría del tiempo en su oficina o trabajo sin poder pasar el tiempo que desearían en su casa o en algún otro lugar más agradable, a este ingrediente se agregan algunos problemas como la inseguridad y el alto costo de los servicios de luz, gas y agua que no paran de crecer, lo cual genera la necesidad de conocer el estado de la casa en estos aspectos y de ser posible tomar acciones encaminadas a solucionar problemas como los mencionados hace un momento.

Es aquí donde la tecnología de Internet y las computadoras caseras se conjugan para tratar de resolver algunos de los problemas de seguridad y uso eficiente de los servicios. En nuestra casa nos interesaría conocer si las puertas y ventanas están cerradas o abiertas o si las luces, la calefacción o el ventilador se quedaron encendidos, todo esto a través de la computadora que tenemos en nuestra oficina o en un centro comercial, y si detectamos algún dispositivo que no esta en el estado adecuado sería muy conveniente poder cambiarlo remotamente. La computadora promedio que existe en casas y oficinas tienen las capacidades suficientes como para realizar estas funciones sin que tengamos que modificar substancialmente su arquitectura y que sea económico para que la mayoría de la gente tenga acceso a estas funciones.

La idea fundamental de este trabajo será desarrollar una serie de programas que puedan controlar alguno de los puertos de comunicación de la PC y que este programa utilice el tiempo de cómputo disponible corriendo en forma paralela a las aplicaciones que usualmente empleamos, sin afectar significativamente el desempeño del equipo de manera general. Para esto debemos de hacer una elección entro los distintos puertos de comunicación disponibles hoy en día en las computadoras personales, así como las herramientas de desarrollo y los métodos de comunicación remota entre PC's mas usados y mas convenientes para nuestro proyecto.

Así, este documento será dividido en ocho capítulos en los que se tratan los diferentes aspectos que forman este trabajo, empezaremos en el capítulo 1 con una introducción que definirá el problema a resolver, los conceptos básicos involucrados en el trabajo, los alcances y expectativas, en el capítulo 2 realizaremos un estudio de los productos que actualmente existen en el mercado y que ofrecen alternativas de solución similares al problema propuesto; en el capítulo 3 se estudian las diferentes alternativas de comunicación entre dispositivos que ofrecen las PC, también trataremos las comunicaciones entre PC a través de redes de computadoras en su esquema más común y ampliamente usado, Internet.

Esta investigación servirá para elegir entre los diferentes puertos de comunicación el más adecuado para los propósitos definidos en el objetivo y para determinar la manera en la que enviaremos los datos del "servidor" al "cliente". Mas adelante, el capítulo 4 establece el diseño conceptual que busca en principio, establecer las ideas centrales del sistema para que este cumpla con el objetivo propuesto basados en la investigación previa anotada en el capítulo 3; en el capítulo 5 se entra de lleno en el diseño del prototipo que será un modelo físico reducido del diseño conceptual, pero que sirve para demostrar la validez de la propuesta de solución, este modelo a diferencia del diseño conceptual, se verá influido por diversos factores físicos externos que de no ser analizados lo pueden llevar a manifestar algunos problemas en su operación, para llevar a cabo un estudio acerca de su comportamiento real.

El capítulo 6 establece algunas pruebas que serán aplicadas a las diferentes partes que conforman el sistema, con el fin de prevenir un funcionamiento inadecuado; mas adelante, en el capítulo 7 analizamos los resultados de las pruebas y en su caso, tomaremos medidas que mejoren el comportamiento general del sistema, finalizaremos con las conclusiones generales en el capítulo 8.

Es importante notar algunos puntos que aunque son importantes en el diseño de un sistema como este, no se tomarán en cuenta en el diseño final del prototipo, debido fundamentalmente a que nos desvían mucho del propósito general planteado en el objetivo, toman una cantidad de tiempo muy grande y además requieren de un estudio más detallado en otras disciplinas. Como ejemplo de estos puntos se puede mencionar: el análisis económico del mercado al que está orientado, la comercialización del producto y su imagen para tomar ventaja a la competencia, la protección intelectual de los componentes del sistema, la normatividad técnica a la que están sujetos este tipo de dispositivos y la seguridad

al consumidor al hacer uso del sistema.

No obstante la des incorporación de estos elementos, el sistema podrá cumplir totalmente con el objetivo planteado, sin detrimento de las características y funcionalidad básicas esperada en un sistema completo a este nivel.

Capítulo 2

Estudio de productos similares en el mercado

2.1. Introducción

Existen hoy en día un gran número de productos que realizan tareas de control, instrumentación o simplemente medición y que se auxilian de las comunicaciones digitales para transferir los datos adquiridos a una sede en la que se tomarán decisiones o que utilizará esta información para continuar con algún otro proceso.

En los último años, como consecuencia del crecimiento tan grande que a tenido Internet y todas las tecnologías asociadas a este concepto, muchas compañías han aprovechado esto para diseñar productos que agreguen nuevas funcionalidades a los servicios que antes prestaban, esta el caso por ejemplo de la telefonía celular, que aunque ya existe en algunos países desde la década de los 80s, no fue sino hasta años recientes cuando incorporó los servicios de mensajes instantáneos, correo electrónico, noticias y páginas Web.

Pensando en este mismo esquema otras compañías han incorporado acceso a Internet a todo tipo de dispositivos, desde electrodomésticos, hasta controladores industriales, pasando por dispositivos de información móviles, Pocket PCs y Agendas electrónicas.

Sin embargo todavía hay mucho por hacer y estas nuevas ideas están impulsando la creatividad y la imaginación de muchos diseñadores para fabricar nuevos dispositivos cada vez más complejos y sofisticados que satisfagan las necesidades que el consumidor se va creando.

Es este contexto ya se empieza a vislumbrar una división del mercado en varias corrientes que básicamente emplean la misma tecnología pero que atacan difer-

entes aspectos y proponen diferentes servicios. Está por ejemplo un campo dedicado al control y monitoreo del consumo de energía eléctrica, gas y agua en casas, comercios y otras instalaciones, además de ofrecer vigilancia de movimiento, de puertas y ventanas y un uso eficiente de la iluminación y en su caso la calefacción. A este campo de la tecnología se le llama Domótica y más adelante se hablará mas a fondo sobre esto.

También es muy marcada la influencia de las nuevas tecnologías de la información en el campo industrial, hoy en día no es raro encontrar productos y servicios que nos 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.

Aunque aun no podemos decir que estas tecnologías son de uso común y tampoco podemos garantizar que lo sean en el futuro, si existen indicios muy claros de que sí se pueden consolidar y crecer de manera muy notable. La reducción y proliferación de los accesos rápidos y permanentes a Internet que hoy en día se ofrecen, pueden ser un gran detonador de estos servicios, ya que el usuario que posee uno de estos enlaces tiene un ancho de banda lo suficientemente grande no solo para transmitir sus datos cotidianos como son el correo electrónico y páginas Web, sino además puede aprovechar este para enviar información útil para estos sistemas de monitoreo y control. Este tipo de enlaces se pueden instalar en los domicilios, oficinas y comercios, a través de el cable telefónico o el de televisión, existen además otras opciones como las antenas de microondas, y los enlaces satelitales, aunque estos últimos 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, obsolescentes 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, esto es así porque debido a que este campo de la ingeniería no esta muy desarrollado aun, no existen acuerdos ni estándares entre las muy pocas compañías que ofrecen este tipo de servicios, de esta manera casi todos son incompatibles entre si.

A continuación presentaré ejemplos representativos de algunos productos y servicios que se 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. 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 tome las mejores ideas de lo ya hecho para concentrar nuestra atención en nuevos problemas.

2.2. LabView (National Instruments)

Como ya mencionamos antes, existen algunas compañías que han aprovechado Internet para incorporar a sus productos nuevas posibilidades de comunicación de datos a otros sistemas. Este es el caso de National Instruments que tiene una larga historia en el desarrollo de productos de medición e instrumentación industrial, y que ahora quiere llevar este mercado y toda su experiencia a un nuevo concepto, la instrumentación virtual, y específicamente con un producto llamado Labview que agrega a la instrumentación y al control una gran flexibilidad, empleando para esto la idea de diseñar en una computadora un diagrama completo de todo el sistema que queremos monitorear, incluyendo actuadores, sensores, medidores y conectándolos entre si para definir el flujo de los datos.

Así el difícil proceso del control industrial y la programación de un sistema de control, se reduce a diseñar un flujo de datos entre los diferentes componentes del diagrama.

En otras palabras, Labview es un lenguaje de programación gráfico que utiliza iconos en lugar de líneas de texto para crear aplicaciones. Al contrario de los lenguajes de programación basados en texto, donde las instrucciones determinan la ejecución del programa, Labview usa una programación por flujo de datos, donde los datos determinan la ejecución del programa.

En el se construye la interfase de usuario utilizando un conjunto de objetos y herramientas. La interfase de usuario es conocida como Panel frontal. Después se agrega código usando representaciones gráficas o funciones de control a los objetos del Panel frontal. El diagrama de bloques contiene este código. Si este se organiza apropiadamente, mostrará el flujo de los datos.

Labview esta completamente integrado para la comunicación con hardware tales como, GPIB, VXI, PXI, RS-232, RS-485, y además para ser conectado con dispositivos de adquisición de datos, además tiene incluidas características para que tu aplicación se conecte a Internet usando el Labview Web Server y software estándar como el empleado el redes TCP/IP y ActiveX.

Se pueden crear aplicaciones compiladas de 32 bits que ofrecerán una rápida ejecución, necesaria cuando se realiza adquisición de datos, medición en tiempo real y soluciones de control. Se pueden además crear ejecutables, librerías compartidas y DLLs porque Labview es un verdadero compilador de 32 bits.

Contiene librerías para colecciones, análisis, presentación y almacenamiento de datos. Además incluye herramientas tradicionales de desarrollo, como los breakpoints y la ejecución del programa paso a paso en donde el programa te auxilia para hacer una depuración y así facilitar el desarrollo.

Labview provee además numerosos mecanismos para conectarse a código externo y software a través de DLLs y librerías compartidas, ActiveX y mas. Adicionalmente ofrece un conjunto de herramientas para una gran cantidad de necesidades en la aplicación. Ayuda a construir soluciones para el científico y el

ingeniero en sistemas. Labview ofrece la flexibilidad, rendimiento y potencia de un lenguaje de programación sin la dificultad y complejidad asociados a estos.

Tiene miles de usuarios satisfechos por encontrar en el, una vía rápida para el desarrollo de aplicaciones de instrumentación, adquisición de datos y sistemas de control. Se emplea en las etapas de prototipo, diseño, prueba e implementación del sistema de instrumentación, tú puedes reducir el tiempo de desarrollo del sistema e incrementar la productividad en un factor de 4 a 10 según los fabricantes.

2.2.1. Descripción de uso

Los programas de Labview son llamados Instrumentos Virtuales o VI por sus siglas en ingles, por que su apariencia y funcionamiento pretender imitar un instrumento físico real, como un osciloscopio o un multímetro. Cada uno de los VI usa funciones para manipular las entradas dadas por el usuario o por otro VI y despliegan el resultado en otros archivos o incluso en otras computadoras.

Un VI contiene los siguientes componentes:

Panel frontal.- Sirve como la interfase con el usuario

Diagrama de bloques.- Contiene el código fuente gráfico del VI que describe su funcionamiento.

Panel de iconos y conectores.- Identifica al VI, quien lo puede usar y si otro VI lo puede usar. Un VI que es usado por otro VI es llamado SubVI.

Un SubVi corresponde a una subrutina en un lenguaje de programación basado en código de texto.

2.3. Hometronic (Honeywell)

En la rutina diaria que se presenta en nuestras casas, encontramos que existen una serie de actividades técnicas que bien podrían ser delegadas, como por ejemplo, el manejo de la iluminación, la temperatura de la casa o de algún recinto de esta o simplemente la temperatura del agua, no solo en los momentos en los que estamos en ella, sino cuando salimos a trabajar o cuando nos ausentamos por vacaciones.

En muchos casos estas acciones se vuelven muy repetitivas y tenemos que dar una vuelta completa por toda la casa para ir ajustando a mano todos los parámetros de iluminación, temperatura, seguridad y confort, para llevar a nuestra casa a un estado más agradable o bien de bajo consumo de energía.

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. Comunicación inalámbrica

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 dispositivo.

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 adicionales

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

Durante las vacaciones, las funciones de ahorro de energía son activadas pero no solo 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 presenta un patrón de actividad dentro de la casa que aparenta estas ocupada.

Durante las horas de luz, los sensores de esta, pueden establecer que 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 a los rayos directos.

Como el usuario de Hometronic preprograma su regreso de vacaciones, el sistema puede llevar a toda la casa al mismo estado de funcionamiento, poco antes de que sus habitantes lleguen.

2.3.3. Equipo y especificaciones

El sistema Hometronic esta formado por un gran número de módulos, que pueden ser sensores o actuadores, además de una unidad central y otros componentes adicionales para conexión remota entre otras cosas.

Figura 2.1: Dispositivos que forman parte del sistema Hometronic

En la imagen que se presenta arriba, vemos algunos de los dispositivos existentes hasta ahora y que forman parte del sistema Hometronic, empezando por

la unidad central, el control remoto por radio, la interfase de voz, el controlador del radiador, el controlador del calentador, el modulo para recámaras que controlan la temperatura, el modulo de persianas, el control de cortineros de correa, modulo de control de iluminación, modulo de interruptores, sensor de viento, sensor de luz solar, medidor de energía eléctrica, gas y agua y módulo de extensión que integra interruptores y sensores con contactos de relevadores para conectarse a sistemas de seguridad en el propio sistema.

2.4. Domótica

En el caso de la automatización de las casas y oficinas existe recientemente una propuesta interesante por parte de varios fabricantes, esta es la llamada Domótica. La domótica es el uso simultáneo de la electricidad, la electrónica y la informática, aplicadas a la gestión técnica de las viviendas. Cuando se aplican estos conceptos en el mundo de las oficinas se usa el concepto de "inmótica".

Esta gestión técnica consiste en la modificación, local o remota, de los parámetros de funciones como:

- Gestión energética: regulación temperatura, gestión de los consumos de cada electrodoméstico y de la potencia contratada, etc.
- Seguridad: custodia y vigilancia frente a la intrusión, la inundación, el fuego, los escapes de gas, etc.
- Comunicaciones: telecontrol y telemetría, correo electrónico, etc.
- Confort: programaciones horarias, escenarios luminosos, riego automático, etc.

Para ello, la domótica usa multitud de dispositivos que pueden ser distribuidos por toda la vivienda en función de las necesidades de los propietarios. Básicamente estos dispositivos se pueden dividir en sensores y actuadores. Además, si la arquitectura es centralizada, se deben tener en cuenta los controladores.

Inicialmente, la única manera de construir una instalación domótica era con el uso de sensores y actuadores que se unían, con una arquitectura centralizada, a un autómata o controlador que tenía integrada toda la inteligencia que se exigía a la vivienda. Casi siempre eran sistemas propietarios, muy pocos flexibles y que hacían muy difícil y costoso el aumento de las prestaciones.

Pero desde hace pocos años, gracias al drástico descenso de los precios del hardware electrónico, es posible construir sensores y actuadores con inteligencia suficiente como para implementar "una red de área local" de control distribuido. Con una arquitectura distribuida y apoyándose en tecnologías o estándares como el X-10, el EIB, el Lonworks, entre otros, la domótica ha ganado en facilidad

de uso e instalación, en flexibilidad, en modularidad y en interconectividad a la vez que ha reducido su coste, y ampliado el abanico de productos, de fabricantes y de instaladores que trabajan en este campo.

En las arquitecturas distribuidas, las redes de control se pueden intercambiar los telegramas mediante cables de pares trenzados, con corrientes portadoras sobre la misma red de baja tensión (powerline communication), vía radio, por fibras ópticas, con cable coaxial, etc. Siendo las dos primeras las de uso más frecuente, el resto se usan allí donde alguna de sus prestaciones es imprescindible debido a los requisitos de la instalación.

A pesar de la aparición de algunos estándares y tecnologías que han abaratado y reducido la complejidad de las instalaciones domóticas, hasta la fecha esta industria no ha tenido la difusión y demanda esperada por parte de los propietarios de las viviendas. Muy poca gente estaba dispuesta a pagar los costes adicionales que implica construir una "vivienda inteligente", la sensación entre el valor añadido y los costes en que se incurren no justificaba, para la mayoría de los usuarios, la inversión.

Pero ahora, gracias a Internet, estamos viendo como están apareciendo multitud de fabricantes y proveedores de servicios que están desarrollando nuevos productos y servicios que conjugan lo mejor de Internet (bajo coste, amplia difusión, presentación Web y WAP) con tecnologías de redes de datos y control asequibles y estandarizadas que creemos que van a darle a la domótica el empujón definitivo para despegar.

Existen muchos más ejemplos en donde este esquema puede emplearse, por ejemplo en la industria, es muy posible que se requiera conocer el comportamiento de algún proceso sin la necesidad de estar presente en la planta y no solo eso, sino que en ocasiones será importante poder tomar alguna acción de control remotamente, esto será posible si nos apoyamos en el desarrollo de la domótica y extendemos sus prestaciones de manera mas robusta hacia las aplicaciones industriales.

En el mercado están surgiendo diversas ideas, innovadoras todas pero aun en proceso de adopción por parte de los consumidores y en proceso de evolución, sabemos bien que los productos nuevos siempre son por diversas causas caros y de difícil implementación, sin embargo si estos tienen éxito poco a poco irán tomando fuerza abaratando sus precios y siendo accesibles para mas personas.

2.5. Protocolos de comunicación empleados en domótica

2.5.1. CEBUS

El protocolo de comunicación CEBus (Cunsumer Electronics Bus) es un estándar vigente en los Estados Unidos que ha sido desarrollado por la Asociación de

Industrias Electrónicas (EIA - Electronic Industries Association). El estándar surgió en 1984 cuando la EIA se propuso unificar los protocolos de señalización para el control remoto de electrodomésticos. En 1992 el estándar se había extendido a todo el ámbito del control domótico.

Los objetivos principales del estándar son:

- Facilitar el desarrollo de módulos de interfaz de bajo costo que puedan ser integrados fácilmente en electrodomésticos.
- Soportar la distribución de servicios de audio y video tanto en formato analógico como digital
- Evitar la necesidad de un controlador central, disminuyendo la inteligencia de la red entre todos los dispositivos.
- Permitir añadir y quitar componentes de la red sin que afecte al rendimiento del sistema ni que requiera un gran esfuerzo la configuración por parte del usuario.
- Proporcionar un método adecuado de acceso al medio.

Medios físicos permitidos

- Red eléctrica convencional
- Cable trenzado
- Cable coaxial
- Infrarrojos
- Radio frecuencia
- Fibra óptica
- Bus audio - video

En todos los medios físicos, la información de control y datos se transmite a la misma tasa binaria, 8 Kb/s. Aunque también se permite canales para acomodar audio o video.

Funcionamiento

Los comandos y los informes de estados se transmiten por el canal de control en forma de mensajes. El núcleo de la especificación CEBus se centra en definir este canal de control. El formato de los mensajes CEBus es independiente del medio físico utilizado. Cada mensaje contiene la dirección de destino del receptor sin ninguna referencia sobre que medio físico esta situado el receptor o el transmisor. De esta forma CEbus forma una red uniforme a nivel lógico en forma de bus. CEBus soporta una topología flexible. Cualquier dispositivo se puede conectar a cualquier medio siempre que tenga la interfaz adecuada. Para comunicar segmentos de red que tienen diferente medio físico, se utilizan dispositivos llamados routers. Estos pueden estar integrados dentro de otro dispositivo con más funcionalidades.

Para facilitar la difusión de mensajes todos los dispositivos tienen una dirección a la que responden todos (broadcats address). Además , los dispositivos se pueden agrupar en grupos (group address). De esta forma se pueden mandar un único mensaje a varios dispositivos al mismo tiempo. Un dispositivo puede pertenecer a uno o más grupos.

CAL (Commun Appliance Languaje)

CAL es el lenguaje que utilizan los dispositivos CEBus para comunicarse. Es un lenguaje orientado a comandos que permite controlar dispositivos CEBus y asignar recursos. El lenguaje es un elemento de la capa de aplicación.

Las funciones de asignación de recursos permiten subir, usar y liberar recursos CEBus. Las funciones de control proporcionan la capacidad de enviar comandos CAL a dispositivos remotos, y responder a comandos CAL.

CAL utiliza el modelo de la programación orientada a objetos. Cuando un objeto recibe un mensaje se ejecuta alguno de los métodos disponibles. Un mensaje consiste en un identificar de método seguido de cero o más parámetros. Cuando se recibe el mensaje, se busca en la lista de métodos cual es el que tiene el identificador y si se encuentra, se ejecuta, por ejemplo, si se quiere subir el volumen de la radio en tres unidades, habrá que mandar un mensaje al objeto que controla la radio en cuestión en el que se invoque el método de subir volumen con el parámetro 3.

Los objetos CAL no se organizan en jerarquías (no existe el concepto de herencia tal como se entiende en POO) sino que el comportamiento depende del contexto en el que se encuentre, por ejemplo, si tenemos un objeto de control analógico, este se puede usar para representar un control de volumen, un termostato o un dimmer. La función exacta vendrá determinada por el contexto en el cual es instanciado el objeto.

2.5.2. EHS

Para cumplir los requerimientos de un sistema de autorización del hogar, EHS define un sistema de red completo, el cual soporta todas las funciones domésticas de forma modular, fácilmente expansible y configurable automáticamente. EHS es un sistema abierto con administración distribuida y funciones de control para todos los medios disponibles. El seguimiento de la norma asegura la interoperatividad de productos de fabricantes diferentes.

Direccionamiento

Existen varios niveles de direccionamiento. A nivel físico se reservan 256 direcciones de terminales físicos en cada sección. Separando el medio físico en varias secciones o empleando varios medios distintos, y uniéndolos mediante routers se puede llegar a millones de direcciones (alrededor de 1012). Esto asegura sin problemas expansión del sistema.

Fiabilidad de la comunicación

Las capas bajas del protocolo aseguran que las peculiaridades del medio físico no afecten a la fiabilidad de la comunicación. El protocolo decide cuando cada unidad puede comenzar una transmisión (arbitraje de bus CSMA). Cada mensaje debe ser confirmado con un mensaje de reconocimiento (ACK) si la recepción no ha detectado errores. En caso de no llegar este reconocimiento, se reintenta la emisión del mensaje. Los tiempos de este protocolo cumplen las regulaciones Europeas para transmisión de datos por la red eléctrica.

El medio red eléctrica tiene especial cuidado en la fiabilidad de la comunicación. Se emplean códigos redundantes para la detección y corrección de errores tanto a nivel de byte como a nivel de mensaje.

Además la modulación FSK permite unas variaciones de frecuencia muy estrictas que hace prácticamente imposible la posibilidad de que algún electrodoméstico pueda interferir precisamente en este estrecho rango de frecuencias.

Gestión de la Red

La red EHS mantiene internamente su administración. Cada unidad enchufada en la red negocia automáticamente su dirección de red, se da a conocer en la red, busca otras unidades que puedan estar interesadas en ella o que puedan interesarle, gestiona transparentemente la existencia de topologías complicadas detectado automáticamente los routers, etc. Todo ello sin la intervención del usuario ni del instalador.

Cada unidad (módulo con una dirección de red propia) puede contener una o varias subunidades. Estas pueden llevar a cabo tareas administrativas de la red:

routers para unir distintos medios físico o secciones de un mismo medio, coordinadores de dispositivos (Device coordinators), controladores de medio (medium controlers) para la asignación dinámica de direcciones de red, o pueden representar aplicaciones individuales como los Feature Controllers (controladores de una aplicación) y los Complex devices (dispositivos que gestionan un sensor o actuador concreto).

2.5.3. EIB

El European Installation Bus (EIB a partir de ahora) se ha pensado para ser utilizado como un sistema de gestión de la instalación eléctrica de un edificio. Su propósito comprende la monitorización y control de sistemas tales como el alumbrado, la calefacción, el aire-acondicionado, ventilación, persianas y alarmas de un edificio.

El estándar EIB ha sido propuesto por la EIBA (European Installation Bus Association). La EIBA es la organización que reúne a las empresas europeas de instalación eléctrica para impulsar el desarrollo de sistemas de edificios y conseguir ofrecer en el mercado europeo un sistema único de alta fiabilidad.

La EIBA tiene su cede en Bruselas, y es una Sociedad Cooperativa según la legislación Belga. Pertenecen a la asociación más de 100 miembros de la EIBA, que como fabricantes cubren 80 % de la demanda de aparatos de instalación eléctrica en Europa. Profesionales y usuarios continúan disponiendo de la libre elección de los productos de diferentes fabricantes y de sus soluciones técnicas. Aunque próximamente será estándar ANSI la penetración en USA es baja o incluso nula.

2.5.4. Konnex

La convergencia de EHS, BatiBUS y EIBUS

Un estándar solo tiene sentido si la práctica totalidad del mercado cumple ese estándar, de forma que equipos de distintos fabricantes pueden trabajar conjuntamente, sin interferir unos en el funcionamiento de otros e incluso colaborando entre sí.

Sin restringir la libre competencia entre empresas, se trata de fijar una norma que deban cumplir todos los fabricante, de forma que se asegure al consumidor final el funcionamiento correcto y conjunto de los productos.

En el campo de la domótica, a pesar de su relativa juventud, han surgido ya varios estándares, cada unos de ellos apoyado por alguna gran empresa. En Europa han destacado BatiBUS y EIBUS. Ambos sistemas han desarrollado productos domóticos en campos más o menos parciales, empleando distintos medios, topologías y protocolos, no compatibles entre sí.

Por otro lado EHS ha surgido avalado por la Comisión Europea y apoyado por grandes y pequeñas compañías Europeas, con el fin de definir un marco genérico que permita la estandarización de los sistemas domóticos.

En la primavera de 1996 EHSA (EHS association), BCI (BatiBUS Club International) y EIBA (European Installation Bus Association) iniciaron un diálogo que les condujo a la creación de un foro común para debatir los temas que interesan a las tres asociaciones. Se crearon comités encargados de aspectos técnicos, de marketing y de normalización, con el fin de alcanzar la convergencia de los tres sistemas, creando una norma común.

Tecnológicamente parece que los problemas que puedan surgir de la unión de los tres estándares son resolubles. Los buses de BatiBUS y EIBUS aparecen como dos nuevos medios en la norma EHS, y en ambos protocolos se realizan los cambios oportunos para asegurar la interconectividad de los distintos sistemas. El medio Red eléctrica de EHS no se ve afectado, y se conservará el protocolo actual.

Desde el punto de vista del marketing, el proceso de la convergencia abre la creación de mercado, aunando los esfuerzos de las tres asociaciones en promover el uso de la domótica hasta llegar a convertirla en un artículo de consumo.

2.5.5. LonWorks

Lonworks ofrece una plataforma completa que incluye no solo un protocolo o un tranceptor, sino también los estándares de interoperatividad y un API de software universal que funciona todo junto sin dificultad.

Medios físicos

En un entorno desafiante y variado como el doméstico, cualquier solución de conexión de redes domésticas tiene que ser capaz de adaptarse a las necesidades del dueño de la vivienda y a las condiciones especiales de la propia casa. Por ejemplo, es más rentable para la vivienda actual utilizar la energía eléctrica como medio de conexión, mientras que en las nuevas viviendas un doble cable entrelazado se puede sacar fácilmente de cualquier parte de la casa.

También sería deseable poder mezclar diferentes tipos de medios dentro de una sola vivienda, así por ejemplo, una parte de la red doméstica opera con energía eléctrica, otras partes con doble cable, alguna sección a través de la frecuencia de radio y otra incluso por Infrarrojo.

2.5.6. X-10

Siguiendo principios estratégicos, X-10 tiene patentes en aspectos clave de la tecnología PLC, que no ha tenido competidores desde los primeros productos X-10 introducidos en el mercado en 1978.

La tecnología X-10 de corrientes portadoras fue desarrollada entre 1976 y 1978 por ingenieros en Pico Electronics Ltd, en Glentothes, Escocia. Los ingenieros de Pico habían estado diseñando componentes microelectrónicos desde que se introdujeron los circuitos integrados en 1969. Hoy, el personal de Pico incluye ingenieros electrónicos, eléctricos y mecánicos. Todo el diseño y desarrollo de los productos X-10 se realiza en Pico.

Aunque el mercado principal de X-10 continúa siendo el americano, X-10 distribuye productos en Europa, Asia y África, Latinoamérica y Oceanía.

El formato de codificación X-10 es un estándar "de facto" usando transmisión de corrientes portadoras (Power Line Carrier = PLC). El formato de la codificación se introdujo en 1978 para el Sistema de Control del Hogar de Sears y para los sistemas Plug and power de Radio Shack.

Desde entonces, X-10 ha desarrollado y manufacturado versiones O.E.M. (Original Equipment Manufacturer) de su Sistema de Control del Hogar para muchas compañías incluyendo Leviton Manufacturing Co., General Electric, C&K Systems, Schlage Lock Co., Stanley Health/Zenith Co., Honeywell, NORWEB, Busch Jaeger, Ademco, DSC, IBM y unos cuantos más.

Todos estos sistemas utilizan el formato de codificación X-10. Todos son compatibles y prácticamente cualquier sistema para el hogar sin cableados utiliza X-10.

El formato de codificación X-10 está patentado.

Sin embargo, para que otras compañí,as puedan beneficiarse de los económicos Sistemas Modulares X-10, se dispone de una gama de interfases Power Line que sirven para crear señales compatibles X-10 y poder así usar la red eléctrica como medio de transmisión.

Las transmisiones X-10 se sincronizan con el paso por el cero de la corriente alterna. Las interfases Power Line proporcionan una onda de 50 Hz. con un retraso máximo de 100 seg. desde el paso por el cero de la corriente alterna. El máximo retraso entre el comienzo del envío y los pulsos de 120 KHz. es de 50 seg.

Un 1 binario se representa por un pulso de 120 KHz durante 1 milisegundo, en el punto cero, y el 0 binario se representa por la ausencia de ese pulso de 120 KHz. El pulso de 1 milisegundo se transmite tres veces para que coincida con el paso por el cero en las fases para un sistema trifásico.

La transmisión completa de un código X-10 necesita once ciclos de corriente. Los dos primeros ciclos representan el Código de Inicio. Los cuatro siguientes ciclos representan el Código de Casa (letras A-P), los siguientes cinco representan o bien el Código Numérico (1-16) o bien el Código de Función (Encender, Apagar, Aumento de intensidad, etc.). Este bloque completo (Código de inicio, Código de Casa y Código de Función o Numérico) se transmite siempre dos veces, separado cada 2 códigos por tres ciclos de la corriente, excepto para funciones

de regulación de intensidad, que se transmiten de forma continua (por lo menos dos veces) sin separación entre códigos.

Dentro de cada bloque de códigos, cada cuatro o cinco bits de código deben ser transmitidos en modo normal y complementario en medios ciclos alternados de corriente. Por ejemplo, si un pulso de 1 milisegundo se transmite en medio ciclo (1 binario), entonces no se transmitirá nada en la siguiente mitad del ciclo (0 binario).

2.6. Comentario final

Al mostrar estos productos que en general tienen características comunes con el proyecto que realizó, intento proponer una alternativa distinta, acotar los alcances de mi proyecto, tomando en cuenta que no se puede hacer una comparación directa entre los sistemas aquí mostrados y el que yo pretendo diseñar ya que las compañías involucradas son enormes y cuentan con un gran número de gente trabajando en esto y enormes inversiones detrás. Finalmente, esta claro que el objetivo de estas empresas es vender un producto y en este etapa el mío será investigas y definir la posibilidad de la implementación de estas tecnologías en una escala mas reducida pero con costos de la misma naturaleza, correspondientes a el entorno económico, social y cultural de nuestra comunidad.

Capítulo 3

Fundamentos de comunicación de la PC

En la actualidad es común encontrar computadoras personales (PC) con capacidades enormes en casi cualquier lugar, desde las oficinas de las empresas y el gobierno hasta nuestros hogares, pasando por las escuelas en casi cualquier nivel, estas son habitualmente empleadas para tareas rutinarias como son el procesamiento de textos, el cálculo de algunas tablas numéricas como la nómina o la facturación de la empresa, el diseño de algunas presentaciones electrónicas y en una escala menor se utilizan para la edición de imágenes, audio, video, animación, desarrollo de sistemas y mas recientemente para acceder a Internet en forma de páginas electrónicas, correo, chat o foros de discusión. Sin embargo, si comparamos todas estas tareas contra las capacidades de cálculo y almacenamiento de la PC's actuales nos damos cuenta de que en la mayoría de los casos, las computadoras están "ociosas" en periodos muy prolongados del día y con grandes espacios en memoria y disco duro disponibles, es decir, tenemos subutilizada la computadora; existen ya esfuerzos por parte de algunos institutos de investigación para aprovechar el tiempo de cómputo que la computadora desperdicia en el protector de pantalla, pero aun así tenemos desaprovechado todo el tiempo disponible para la computadora cuando estamos escribiendo un texto o pensando una nueva instrucción.

Por otro lado tenemos que el hardware de la computadora personal a ido creciendo conforme pasa el tiempo y de nuevo, las nuevas prestaciones del equipo no siempre son utilizadas por todos los usuarios, de hecho, en la mayoría de los casos la mayoría de los puertos de comunicación de la PC no se emplean y a medida de que nuevos puertos aparecen, como es el caso del USB, dejan a otros (Serial y paralelo) sin ninguna utilidad aparente.

A lo largo de este capitulo se mostraran las diferentes interfaces de comunicación que la PC ofrece para interactuar con otros dispositivos cercanos a el, se

mostrarán las características más importantes con la finalidad de establecer cual de todas estas posibilidades nos presentan una menor dificultad en su uso y un buen nivel de control ya sea para programarlo como para diseñar el hardware adicional necesario para este proyecto. Después continuaremos con lo referente a las comunicaciones a larga distancia entre computadoras y finalmente se estudiará el software disponible para el desarrollo del sistema y cual es el indicado para el proyecto basados en las necesidades propuestas en el objetivo principal de este trabajo.

3.1. Hardware de comunicación de la PC

3.1.1. Puerto serie

Uno de los componentes más universales de las PC, aparte de la CPU, es el puerto serie. En este se pueden conectar dispositivos tan diversos como el ratón, el MODEM, una impresora, un plotter y hasta otra computadora. No obstante, su funcionamiento a nivel hardware y software es uno de las partes que menos se han documentado en profundidad por parte de los fabricantes y expertos de las PC, no tanto por su uso, sino por su programación y rendimiento.

Desde que el puerto serie de comunicaciones hizo su aparición, la evolución de este no ha sido tan espectacular como la de otros dispositivos del ordenador, como pueden ser la CPU o las memorias, las mejoras del puerto serie siempre han llegado en forma de velocidad, aumentando desde los apenas 150 bps hasta los 115,200 bps que pueden tener hoy en día.

Cuando se habla del puerto serie cabe mencionar que siempre habrá un transmisor y un receptor, y entre ellos deben usar la misma velocidad de transmisión, que vendrá marcada por la misma velocidad que tenga uno de los dos. Genéricamente al transmisor se abrevia como TX y el receptor RX. Tanto el uno como el otro conocen el número de bits de información que manejan (probablemente con el bit de paridad incluido) y la marca de stop o bits de stop (1, 1.5 o 2 veces un bit de información).

Quien indica el comienzo de una nueva palabra de información es el bit de start. El bit de start es de vital importancia para sincronizar la transmisión y la recepción, y siempre será un "0" lógico, que equivale a 0 voltios en tecnología TTL o a 12 voltios reales, aproximadamente. El bit de paridad puede ser añadido a los bits de datos (información) para permitir la detección de errores en la serie de bits de información.

El bit de stop no indica el final de una palabra, aunque su nombre refleje lo contrario, su función es separar dos palabras consecutivas poniendo la línea de transmisión en un estado inactivo el mínimo tiempo posible. Por lo que el bit de stop será siempre un "1" lógico, -12 voltios reales, para que el bit de start siguiente tenga una probabilidad muy alta de ser detectado.

La paridad impar activa el bit cuando en la información hay un número impar de "1" o "0", según la lógica con la que se trabaje. Si usamos lógica negada, se activará cuando el número de ceros sea impar. Con lógica positiva, se activa cuando el número de unos es impar. La paridad par activa el bit cuando hay un número par de ceros o unos, según la lógica.

Diferencia entre bits por segundo y baudios

Por definición, un protocolo es una descripción clara y concreta de un método lógico de transmisión de información, pero nunca incluye cómo se debe realizar físicamente, este es el caso del puerto serie.

El nombre de Baudio se debe a J. M. Baudot, el impulsor del teletipo (TTY). Se define baudio como los cambios de estado que se producen en una línea de transmisión por segundo. Mientras que bits por segundo, son los bits que pasan por una línea de transmisión por segundo. Hay que tener en cuenta que al hablar de baudios y bits no existe ninguna diferencia si no hay más de dos estados. Es decir, los bits únicamente pueden ser "1" o "0", mientras que si se habla de baudios puede haber más de dos estados ya que no se trata de un sistema binario.

Acrónimos de protocolos de transmisión

Siempre se omite el bit de start, en todos los casos es uno.

- 8n1, 8 bits de información - sin paridad - 1 bit de stop
- 7e1, 7 bits de información - paridad par (even) - 1 bit de stop
- 7e2, 7 bits de información - paridad par (even) - 2 bits de stop

Los que se usan comúnmente son los dos primeros

Comunicación RS-232 C

Para que los datos circulen de un ordenador a otro o de un dispositivo al PC, se crea una línea de transmisión de lazo cerrado con una corriente estática de 20 mA y una corriente de caída de 0 mA. Estos datos son muy importantes para saber si hay una correcta conexión entre ambos dispositivos, ya que si la conexión es defectuosa, los dispositivos detectarán que la corriente estática no es de 20 mA, sino de 0 mA, ya que se trata de un lazo cerrado y por lo tanto darán error. Los voltajes que manejan las señales que corren por dentro del cable son de -3 a -15 voltios para el "1" lógico (normalmente se aproxima a -12 v), mientras que para el "0" lógico, las tensiones son positivas.

La impedancia de salida del puerto serie comúnmente es de 2Kohms (para 5 mA y 10v), mientras que la de entrada es de 4.3 Kohms, consiguiendo un fan-out de 5 (se puede conectar 5 entradas a 1 salida).

Aunque la especificación del cable RS-232, no permite que el cable que una dos dispositivos mida más de diez metros, lo cierto es que si se puede hacer siempre y cuando nada suficientemente importante o vital dependan del sistema, esto es por especificaciones eléctricas, del párrafo anterior se consigue la siguiente relación velocidad/distancia máxima del cable.

Baudios	Distancia max. cable blindado	Distancia max. cable sin blindar
110	1500 metros	1000 metros
300	1500 metros	1000 metros
1200	1000 metros	1000 metros
2400	300 metros	200 metros
4800	300 metros	100 metros
9600	100 metros	100 metros

Cuadro 3.1: Relación Velocidad / Distancia del cableado empleado para las comunicaciones RS-232

Cable RS-232 C

En la mayoría de las computadoras hay como mínimo un par de conectores al puerto serie. Para poder conectar cualquier dispositivo a estos conectores, hay que contar con terminales apropiados, estos son los terminales 9pin / 25 pin macho SUB-D.

Las líneas más importantes son las que hacen referencia a los datos, tanto la de transmisión como la de recepción y la de tierra. Las otras líneas son utilizadas por otros dispositivos, como un módem o impresora y sirven para indicar estados internos de estos dispositivos.

Los cables que conectarán cada pin son únicos, a continuación se muestra una pequeña descripción de cada uno de ellos.

Nombre	25 pin	9 pin	Dirección	Nombre completo
TxD	2	3	Salida	Datos transmitidos
RxD	3	2	Entrada	Datos recibidos
RTS	4	7	Salida	Permiso para transmitir
CTS	5	8	Entrada	Listo para transmitir
DTR	20	4	Salida	Terminal preparado
DSR	6	6	Entrada	Datos preparados
RI	22	9	Entrada	Detector de llamada
DCD	8	1	Entrada	Detector de portadora
GND	7	5	-	Tierra
-	1	-	-	Tierra de protección

Cuadro 3.2: Descripción de los pines que forman parte del puerto serie.

A continuación se muestran las funciones de las señales:

Señal	Descripción
TxD, RxD	Estas señales serán las encargadas de llevar los datos de un extremo a otro, uno para transmitir y el otro para recibir.
RTS, CTS	Son usados por los dispositivos para seguir o cortar las transmisiones de datos.
DTR, DSR	Mediante estas dos señales, los dos dispositivos que estén conectados, quedarán de acuerdo sobre la velocidad a usar, los bits de paridad, etc.
RI	Indica que hay una llamada entrante, muy importante en los MODEM's
DCD	También otra señal muy importante en los MODEM's, ya que indica si hay portadora, y por tanto si se puede transmitir o no.
GND	Es la señal de referencia, tras consultarla se sabrá si cualquier señal recibida es de nivel alto o bajo.

Cuadro 3.3: Significado de las siglas usadas para las señales del puerto serie.

Como podemos observar el puerto serie ofrece gran flexibilidad para la conexión de dispositivos periféricos con una relativa simplicidad, define entre sus características un protocolo que garantiza un traslado adecuado de la información e incluso detecta errores y los corrige, sin embargo incluso las más recientes versiones de este puerto son lentas en lo concerniente a la transmisión de datos, su

diseño se centra fundamentalmente en solo dos dispositivos a la vez, esto es, no puede ser concebido como un bus de comunicaciones para varios dispositivos, lo que nos hace pensar que no es la mejor opción para las necesidades que se plantean en este proyecto.

3.1.2. Puerto paralelo

Desde el punto de vista del software, el puerto paralelo son tres registros de 8 bits cada uno, ocupando tres direcciones de entrada/salida consecutivas de la arquitectura x86.

Desde el punto de vista de hardware, el puerto es un conector hembra DB25 con doce salidas latcheadas (que tienen memoria/buffer intermedio) y cinco entradas, con 8 líneas de tierra.

La función normal es transferir datos a una impresora a través de la línea de datos, usando las señales restantes como control de flujo.[6]

Descripción general

El puerto paralelo se identifica por su dirección de I/O (entrada/salida) base y se simboliza el MS-DOS como LPT y lp en UNIX o Linux. Cuando arranca la máquina, el BIOS verifica las direcciones específicas de I/O en busca de puertos paralelos y construye una tabla de las direcciones halladas en la posición de memoria 40h:8h (ó 0h:0408h).

Esta tabla contiene hasta tres palabras de 16 bits. Cada palabra es la dirección de I/O del puerto paralelo. La primera palabra corresponde a LPT1, la segunda a LPT2 y la tercera a LPT3. Hay que agregar que en MS-DOS tenemos el dispositivo PRN que es un alias a uno de los dispositivos LPT (generalmente es LPT1, pero se puede cambiar con la orden MODE).

Las direcciones estándar para los puertos paralelos son 0378h, 03BCh y 0278h respectivamente.[2]

El puerto, como se mencionó antes, consiste de tres registros de 8 bits ubicados en direcciones adyacentes del espacio de I/O de la PC. Los registros se definen relativos a la dirección de I/O base (variable de IO base) y son:

IO Base + 0: registro de datos

IO Base + 1: registro de estado

IO Base + 2: registro de control

Aclaración respecto a la nomenclatura

En las secciones siguientes, haré referencia a cada bit de los registros como una inicial que identifica el registro seguido de un número que identifica el número de bit, siendo 0 el LSB (bit menos significativo) y 7 el MSB (bit más significativo).

Para evitar confusiones, no pondré si es activo bajo o en alto en el nombre del bit porque no corresponde. En su lugar pondré la lógica que usa. Es decir, si es lógica negativa o positiva. Cuando se indica Alto o Bajo se refiere a la tensión en el pin del conector del puerto. Esto es porque la lógica puede ser positiva (un 1 lógico equivalente a Alto o 5 V en TTL) o negada (un 1 lógico equivale a Bajo o 0 V en TTL).

En aquellos pines que cumplen una función específica para la impresora (el periférico para el que fue diseñado el puerto paralelo), colocaré su nombre en ingles y la barra de negación si fuera una señal activa en bajo. Por lo tanto, no confundir la lógica que sigue el puerto, con la lógica que mantiene la impresora.

Registro de datos

Escribiendo un dato al registro causa que el mismo aparezca en los pines 2 a 9 del conector del puerto. Leyendo el registro, se lee el último dato escrito (NO lee el estado de los pines; para ellos hay que usar un puerto bidireccional).

El estándar es que las salidas sean LS TTL (low schottky TTL), aunque las hay que son de tipo OC (colector abierto). La corriente que pueden entregar (modo source) es de 2.6 mA máximo y pueden absorber (modo sink) un máximo de 24 mA. En el puerto original de IBM hay condensadores de 2.2 nF a tierra. Las tensiones para el nivel bajo son entre 0 y 0.8 V y el nivel alto entre 2.4 V y 5 V.

Registro de estado

El registro de estado está en IOBase+1. Es de sólo lectura (las escrituras serán ignoradas). La lectura da el estado de los cinco pines de entrada al momento de la lectura. Aquí se muestran los cinco pines que forman el registro de estado:

Bit	Nombre	Pin
7	Busy	11
6	/ACK	10
5	No paper	12
4	Selected	13
3	/Error	15

Cuadro 3.4: Asociación entre las señales del registro de estado y los pines del conector .

La línea Busy tiene, generalmente, una resistencia pull-up interna. El estándar es que sean entradas tipo LS TTL.

Registro de control

El registro de control se encuentra en IOBase+2. Es de lectura/escritura.

Bit	Nombre	Pin
5	Control bidireccional	
6	Interrupt control	
7	/Select	17
8	/Inicialize	16
9	/Auto feed	14
0	/Strobe	1

Cuadro 3.5: Asociación entre las señales del registro de control y los pines del conector .

Los cuatro bits inferiores son salidas. La lectura devuelve lo último que se escribió a dichos bits. Son TTL a colector abierto con resistencias de pull-up de 4700 ohms, por los que un dispositivo externo puede forzar el estado de los pines sin dañar el driver.

Esto permite utilizar estas cuatro líneas como entradas. Para ello, ponemos en alto las cuatro salidas (escribiendo 0000100b en IOBase+2) lo que hace que las salidas "floten". Ahora, un dispositivo externo puede forzar abajo alguna de las salidas con lo que, leyendo el puerto, sabemos si esto sucedió o no.

Es posible realizar esta técnica en salidas totem-pole (como D0-D7) pero no se recomienda su uso porque habría que tener un conocimiento preciso de la corriente ya que se puede sobrecargar los transistores de salida, dañando el driver (especialmente en puertos integrador LSI).

Bit de puerto bidireccional (compatible PS/2)

El bit C5, está disponible sólo si se trata de un puerto bidireccional; en los puertos comunes actúa como los bits C6 y C7. Si C5=1, el buffer de los datos de salida se pone en alta impedancia, "desconectado" dicho buffer de los pines 2 a 9 del conector del puerto (D0 a D7). Si se escribe al registro de datos, se escribe al buffer pero no a la salida. Esto permite que al leer el puerto, se lea el estado de las entradas y no lo que hay en el buffer.

En las computadoras IBM PS/2, para habilitar el puerto paralelo bidireccional, además de lo antes descrito, se debe poner a 1 el bit 7 del registro del puerto 102h (opciones de configuración)

En computadoras que no tengan puerto bidireccional compatible PS/2 hay que modificar uno o más bits de algún puerto especifico correspondiente al chipset de la placa. A veces se habilita por el Setup o por un jumper en la placa del puerto.

Bit de interrupción

En trabajos normales de impresión ni el BIOS ni el DOS hacen uso de la interrupción

El hecho de poseer una línea de interrupción que está conectada directamente al PIC (programmable Interrup Controller), lo hace muy útil para experimentación en data-loggers por ejemplo. El bit de interrupción está conectado al control de un buffer de tres estados. Cuando C4=1, se activa el buffer y su entrada, S6, se conecta a la línea IRQ (en general es IRQ7 o IRQ5). La lectura del bit, nos devuelve el estado del mismo (es decir si el buffer está en alta impedancia o no).

Se producirá una interrupción, cuando hay un flanco descendente en el pin correspondiente a S6. A continuación, se describen los pasos para poder utilizar interrupciones.

Interrupciones con el puerto paralelo

Primero que nada debemos habilitar el buffer que conecta la línea ACK con la línea IRQ. Esto lo hacemos poniendo a 1 el bit 4 del registro de control (IOBase+2). Luego debemos preparar una ISR (Interrupt Service Routine) que atienda la interrupción recordando enviar la señal EOI (20h) al registro de control del PIC (puerto 20h) al salir de la rutina. La interrupción software corresponde a la número 0Dh para IRQ5 y 0Fh para IRQ7. Finalmente se habilita con 0 la interrupción IRQ5 (o IRQ7) escribiendo al bit 5 (o 7) del registro de interrupciones del PIC (puerto 21h).

Para desinstalar la ISR, deshabilitamos la IRQ5 (o IRQ7) escribiendo un 1 al bit 5 (o 7) del registro de interrupciones del PIC (puerto 21h). Luego hacemos que C4=0.[6]

Velocidad

Un puerto paralelo ISA normal toma un ciclo-ISA para leer o escribir. Como en muchos sistemas, la velocidad del bus ronda los 1,3 Mhz, podemos decir que la lectura la podemos hacer cada 1 us (idealmente, ya que siempre tenemos otras instrucciones software, etc.; en la práctica pueden ser desde 1.2 us a 2 us).

Algunos puertos soportan un modo "turbo" que elimina los 3 estados de espera de la CPU, con lo que la velocidad de lectura/escritura del puerto se duplica (2,7 MHz).

Handshaking con impresora

El handshaking ("apretón de manos") viene a ser algo así como cuando se habla de protocolos en el software. Es decir, son un conjunto de reglas que ambos extremos de un sistema de comunicación tienen que seguir para que la comunicación sea exitosa.

El puerto paralelo, usado con una impresora, transmite datos y transmite/recibe las señales de handshaking. En la tabla anterior ya se ha descrito las señales utilizadas. Las principales son /Strobe, /Ack y Busy. La secuencia a seguir para enviar datos sería:

1. Colocar el byte a enviar en el registro de datos
2. Verificar que la impresora no esté ocupada (Busy = bajo, S7 = 1)
3. Indicarle a la impresora que acepte los datos (/Strobe = bajo , C0 = 1, pulso >5us)
4. En ese instante la impresora indica que está ocupada recibiendo los datos (Busy = alto, S7 = 0)
5. Finalmente, la impresora envía un pulso de aceptación indicándonos que se recibieron los datos y que podemos volver al paso 1 (/Ack = bajo, S6 = 0, pulso de entre 5 us y 15 us según impresora).

Las otras señales nos sirven para verificar el estado de la impresora (/Error, PaperEnd), para resetearla (/Init) y para "configurarla" (/AutoFd, /Select).

En los nuevos puertos, estas señales adquieren otra función, a veces parecida y otras totalmente distintas.[5]

3.1.3. USB

El USB es un sistema de transmisión de datos que usa cables para conectar equipos periféricos a las PC. Todas las nuevas computadoras tienen 2 o mas receptáculos USB y las predicciones dicen que con el tiempo reemplazará al resto de los puertos clásicos de la PC.

El USB puede proporcionar la corriente necesaria a los dispositivos conectados, cuenta con capacidades de “plug and play” y “hot swapping”, provee 12 Mbit/s y 1.5 Mbit/s en velocidad de transferencia (la especificación de USB 2.0 promete desarrollar velocidades más altas).

El estándar de los cables para dispositivos USB refieren un conector A-macho en un extremo y uno B-macho del otro, estos cables conectan de un lado a la computadora y del otro al dispositivo ya sea una impresora, un scanner o cualquier otro. el conector A y B son lo suficientemente distintos como para que no haya confusión al conectarlos.

Cinco metros es la máximo longitud del cable aceptada en la especificación del USB. Se puede ampliar esta distancia intercalando HUB cada 5 metros, otra alternativa es emplear "Cables de extensión activos" que no son mas que un cable USB normal de 5 metros mas un HUB con un solo puerto, puedes encadenar máximo cuatro de estos cables para lograr 25 metros de cable total.

En el mejor de los casos el USB trabaja muy bien con dispositivos plug & play cuando tienes incluidos en el sistema operativo los controladores o cuando el fabricante los provee, pero si no existen y se tienen que desarrollar por uno mismo la tarea se vuelve muy complicada, ya que el ambiente de desarrollo no es nada amigable. Primero por que el desarrollo del hardware USB es muy complicado y luego la programación del driver es un gran problema. Lo que normalmente se hace es usar la plataforma de un dispositivo USB genérico como un HID (Human Input Device) ya que de otra manera se tiene que programar un driver WDM y no es fácil.

El USB usa un conector con cuatro pines (en ambos casos A y B). Dos bits son para los datos (en el bus bidireccional diferencial) y los otros dos para la alimentación de los dispositivos (+5V y tierra, con corrientes entre 100mA o 500 mA dependiendo del dispositivo). El estándar USB define cuatro tipos de transferencia de datos: control, isochronous, interrupt y bulk. Todos los dispositivos USB soportan el tipo "control" para comandos y status. Los otros tres tipos se utilizan en diferentes aplicaciones, por ejemplo el tipo "bulk" se usa en dispositivos como impresoras, scanners y cámaras digitales donde se mueven grandes cantidades de datos hacia la PC.

El estándar 1.0 y 1.1 de USB define velocidades de 1.5 y 12 Mbps en sus modos de baja y alta velocidad, respectivamente. El último estándar 2.0 propone 480 Mbps como la velocidad transferencia alta.[9]

3.1.4. IEEE 1394 (Firewire)

El IEEE 1394 es un bus serial en principio similar al USB, pero este puede transferir mas de 400 Mbit/s y no esta enfocado a una PC, los dispositivos pueden estar conectados entre si sin PC o con varias PC en el mismo bus. Tiene un modo de transmisión que garantiza el ancho de banda y de esta manera se convierte en el ideal para dispositivos de video digital.

IEEE 1394 es un estándar que define al bus serie de alta velocidad, este bus es llamado FireWire por Apple o i.Link por Sony, todos estos nombres refieren la misma cosa pero el término formal debe ser IEEE 1394, en algunos sitios de Internet es común encontrarlo como 1394 simplemente.[10]

Este bus es de muy reciente introducción al mercado de las computadoras y al igual que el USB no esta presente en la PC común que se considera dentro del objetivo de este trabajo, sin embargo es importante notar que en un futuro puede ser empleado para ampliar las prestaciones de el sistema propuesto utilizando la flexibilidad, alta tasa de transferencia y la capacidad que tiene para conectar al mismo bus un sin número de dispositivos, que pueden en algún momento ser lo suficientemente inteligentes como para enviar peticiones e información entre ellos sin necesidad de pasar por la PC.[10]

3.2. Comunicación remota entre PC's

En esta sección abordaremos la manera en la que las computadoras pueden enviarse información unas a otras a través de las red de datos que actualmente se emplean que en su mayoría están basadas de forma general en los protocolos TCP/IP, el empleo de esta familia de protocolos nos permite extrapolar el esquema de conexión lógica de una red de computadoras de área local a redes más grandes o incluso hasta Internet sin la necesidad de ningún cambio. Posteriormente sentaremos las bases de la programación de estas comunicaciones utilizando los llamados "sockets", esta explicación servirá para iniciar la búsqueda de un sistema operativo y de las herramientas de desarrollo más adecuadas para el desarrollo final del prototipo.

3.2.1. Encapsulamiento de datos y construcción de paquetes

En el proceso de comunicación de información entre dos o más computadoras debemos primeramente establecer las reglas fundamentales con las cuales estas se organizarán para interpretar el flujo de datos, los protocolos realizan esta función en las diferentes capas que forman el esquema general de la comunicación de redes de computadoras (Modelo OSI), así cada uno de ellos agrega a la información inicial un pequeño bloque que describe el tipo de protocolo del que se trata, esto se realiza capa por capa, de tal forma que el paquete final que será transmitido esta formado por varias envolturas como si se tratara de una cebolla; a este proceso se le conoce como encapsulación de datos.

Figura 3.1: Ejemplo de empaquetado de datos bajo el modelo OSI.

En la imagen que se muestra arriba, podemos observar un ejemplo de un dato que será transmitido inicialmente por el protocolo TFTP, después por UDP, después por IP y finalmente a través de Ethernet.

La disposición de los protocolos en este paquete es resultado del mencionado modelo OSI que establece los niveles en los que las comunicaciones por redes de computadoras deben tener, a continuación la lista completa de estos niveles:

1. Aplicación
2. Presentación
3. Sesión
4. Transporte
5. Red
6. Enlace de datos
7. Físico

Sin embargo este modelo es tan general que para fines prácticos se acostumbra simplificarlo en el siguiente:

1. Aplicación (telnet, ftp, etc.)
2. Transporte (TCP, UDP)
3. Internet (IP y encaminamiento)
4. Acceso a la red (Ethernet, ATM, etc.)

Este último corresponde claramente con la imagen del paquete de información mostrada líneas arriba.

Construcción del paquete

En la programación de algún sistema que emplee este esquema de comunicación, el programador sólo deberá preocuparse de enviar los datos originales con las funciones send() o sendto() de lenguaje C, o las equivalentes si se utiliza otro lenguaje, la construcción del paquete en los niveles de Internet y Transporte es realizada por el kernel del sistema operativo y la de acceso a la red por el hardware. De esta manera podemos advertir que la programación de aplicaciones que empleen acceso a las redes y que utilizan este esquema de comunicación son independientes de la plataforma e incluso del sistema operativo que se utilicen, esto de manera conceptual, ya que los archivos ejecutables generados no siempre serán compatibles entre plataformas. La utilización de este modelo ofrece la ventaja de que el programador puede concentrarse en la manera en la que los datos serán enviados al kernel y no por todo el proceso involucrado en el envío de los datos hasta la otra computadora.

Recuperación de la información

Cuando el paquete de información es recibido por la computadora destino, esta procede a tomar la información de la primera capa y retirarla para así descubrir la capa siguiente y así sucesivamente hasta llegar a los datos originales para pasarlos a la aplicación en donde el usuario puede verlos de manera inteligible, nuevamente como en el proceso de construcción la aplicación final solo deberá concentrarse en la interpretación de los datos originales y no en los protocolos intermedio de las capas 2, 3 y 4 ya que esta tarea la realiza el kernel y el hardware respectivamente.

3.2.2. Programación de redes con Sockets

¿Qué es un socket?

Un Socket en muy breves palabras en una manera de comunicarse entre programas usando descriptores de archivo propios de UNIX.

Como sabemos, todos los dispositivos de hardware y unidades lógicas de información en una computadora UNIX es un archivo, cuando los programas de UNIX tienen que hacer cualquier función de entrada y salida, emplean para esto un descriptor de archivo, un descriptor de archivo no es mas que un número entero asociado al archivo abierto y claro, este archivo puede ser un archivo real en el disco duro, una terminal, la pantalla de la computadora o una conexión a la red.

Cuando nosotros hacemos una llamada a la función socket() propia del lenguaje C, esta nos devuelve un descriptor de archivo que apunta a una conexión de red. Este descriptor lo podemos utilizar para realizar lecturas y escrituras a la red con las funciones send() y recv(), como si fuera un archivo común y corriente.[8]

Tipos de sockets

Existen muchos tipos de sockets, dependiendo de la plataforma y las aplicaciones que se empleen, sin embargo, aquí nos concentraremos en dos tipos fundamentales pertenecientes a la familia de los sockets de Internet. El primer tipo lo definen los sockets de flujo [*stream sockets*] y el otro tipo son los sockets de datagramas [*datagram sockets*] a los que en lo futuro llamaremos SOCK_STREAM y SOCK_DGRAM, en ocasiones a los sockets de datagrama se les llama también sockets sin conexión.

Los sockets de flujo definen flujos de comunicación en dos direcciones, fiables y con conexión. Si envías dos datos a través del socket en el orden 1, 2 llegarán al otro extremo en el orden 1, 2 y llegarán sin errores.

Estos sockets tienen una aplicación muy común en el comando "telnet" ya que se requiere que todos los caracteres que escribamos lleguen en el mismo orden

al otro extremo, también los navegadores que usan el protocolo HTTP usan sockets de flujo para obtener las páginas.

Para conseguir que el nivel de calidad de transmisión sea alto estos sockets usan el "protocolo de control de transmisión", más conocido como "TCP". TCP asegura que la información llegue secuencialmente y sin errores, este protocolo es muy mencionado junto con el "IP" que significa "Protocolo de Internet", IP se encarga básicamente del encaminamiento a través de Internet y en general no es responsable de la integridad de los datos.

En el caso de los sockets de datagrama suceden cosas distintas, si se envía un datagrama puede que llegue, puede que llegue fuera de secuencia, pero si llega los datos que contiene el paquete no tendrán errores.

Los sockets de datagramas también usan IP para el encaminamiento, pero no usan TCP, usan en su lugar UDP "Protocolo de datagramas de usuario".

Se les llama sin conexión básicamente porque no se tiene que mantener una conexión abierta como se haría con los sockets de flujo. Simplemente se crea el paquete, se inserta la cabecera IP con la información de destino y se envía. No se necesita conexión. Generalmente se utilizan para transferencias de información por paquetes. Aplicaciones que usan este tipo de sockets son, por ejemplo, tftp y bootp.

Para controlar el flujo de información cuando se usan datagramas, un protocolo superior al UDP deberá implementar una función de verificación de la llegada de la información, si se envía un paquete, el receptor envía un paquete de "ACK" que le dice al emisor "lo tengo", si no recibe ninguna respuesta en un determinado tiempo, se retransmitirá el paquete hasta que finalmente reciba un "ACK". Este procedimiento es fundamental si se implementan aplicaciones basadas en datagramas.[8]

Capítulo 4

Diseño conceptual

4.1. Propuesta inicial

Como ya hemos visto en el capítulo anterior, existen en la actualidad un gran número de dispositivos domóticos que realizan tareas de control, monitoreo y gestión energética de una casa u oficina, sin embargo en todos los casos encontramos que los sistemas están conformados por una unidad central de procesamiento y con una gran variedad de actuadores y sensores que complementan el conjunto, esto implica que cada uno de los dispositivos deberá poseer una "inteligencia propia" y las capacidad para comunicarse entre ellos de manera fácil pero a la vez segura.

Aquí es donde nos planteamos la pregunta, ¿por qué no utilizar los recursos de comunicación y capacidad de cómputo de las computadoras personales que la mayoría de las personas con necesidades de monitoreo de su casa o negocio tiene?, al mismo tiempo de que tratamos de aprovechar la inversión echa en el equipo de cómputo al máximo, podemos pensar además en no modificar sustancialmente el hardware de este equipo en primer lugar porque esto implicaría un costo que no deseamos y en segundo, porque tratamos además de hacer un sistema que fácilmente podamos instalar en cualquier casa o negocio.

De lo anteriormente expuesto y de lo visto en los capítulos anteriores, podemos definir claramente las necesidades y requerimientos elementales que el sistema deberá cumplir para que realmente cubra con las necesidades planteadas en el objetivo de este trabajo.

4.2. Necesidades principales

A continuación se enlistan los puntos principales en lo que el diseño conceptual deberá fundamentarse, para de ahí ir descubriendo las especificaciones precisas

que el equipo tendrá

1. **Utilizar la PC como dispositivo de control y comunicación central.-** En la propuesta inicial mostramos que puede resultar muy útil emplear la computadora que ya se posee en la mayoría de las casas u oficinas que necesitan el servicio de monitoreo, así que el sistema procurará estar centralizado en la PC tanto para controlar los dispositivos conectados al puerto, como para comunicarse a Internet y avisar los cambios de estado. Es importante notar también que una parte importante del proyecto será intentar en lo posible, no modificar la configuración típica de una PC, el hardware que se agregará será muy elemental, externo y por lo tanto barato. En tanto al software aprovecharemos la familiaridad que la gente tiene para ingresar información y recuperarla con la PC para que la curva de aprendizaje sea lo menos pronunciada posible, además el programa de control y comunicación deberán correr de tal forma que la computadora pueda ser empleada por el usuario en otras tareas; en otras palabras, la computadora no deberá estar dedicada sólo al programa domótico.

2. **Aprovechar la infraestructura de la 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 responder a este problema el sistema sólo requerirá de líneas de comunicación de baja potencia lo cual implica cables muy delgados y fáciles de instalar. En una aproximación siguiente a este problema se pueden considerar comunicaciones inalámbricas aunque esto elevaría considerablemente el precio y la complejidad del sistema.

3. **Hacer el diseño independiente de la forma de conexión a Internet.-** 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, así que diseñar un sistema que esté atado a algún tipo de conexión en especial, creo que sería un error. Por ese motivo el sistema propuesto aquí, deberá estar diseñado sobre una capa superior en lo que se refiere a las comunicaciones, de esta manera, el programa empleará la plataforma que el sistema operativo tenga para comunicarse y no se tendrá que programar está parte, además de que con esto se gana independencia del tipo de conexión, en resumen, el sistema trabajará para comunicarse sobre el sistema operativo el cual abstraerá todas las funciones de comunicación requeridas en Internet a solo dos, lectura y escritura.

4. **Utilizar herramientas de desarrollo y producción libres y flexibles.-** Como parte del objetivo de este trabajo es hacer un sistema barato y con la posibilidad de extenderlo sin tener que pagar por tecnología propietaria, la utilización de software libre en la parte del desarrollo como en

la producción es fundamental. En el desarrollo del trabajo se requiere un lenguaje de programación que pueda manejar el hardware de manera sencilla y eficiente, pero a la vez las comunicaciones entre computadoras con protocolos estándar de Internet (TCP/IP), la respuesta trivial es el lenguaje "C", y de este tenemos de manera libre uno de los mejores compiladores el "GNU cc" o "gcc", con este compilador podremos hacer el programa servidor y el cliente necesarios para este proyecto. Además contamos con un sin número de aplicaciones libres ideales para el desarrollo de este tipo de programas, para la depuración, prueba, mantenimiento, etc.

5. **Emplear conexiones, cables, etc. estándar, no propietarios.-** Continuando con la misma idea del punto anterior, el empleo de conexiones o cables estándar nos permitirán abaratar los costos y permitir flexibilizar el sistema, para que terceras personas puedan modificar, reparar o incluso adicionar módulos al sistema.

De los puntos anteriores podemos ir definiendo un esquema general que pueda ir conformando lo que será el sistema domótico de manera integral y a la vez ir discriminando los módulos que lo conformarán, para posteriormente entrar de lleno al diseño de cada uno de estos módulo de manera conceptual y siguiendo las líneas fundamentales que se han planteado líneas arriba.

4.3. Esquema general

A grandes rasgos el sistema esta integrado por una interfase que se conecta a la PC a través de uno de los puertos de comunicación locales (Paralelo, USB, Serie, etc.) 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, etc.), la PC a la cual se conecta esta interfase correrá un programa que la controlará y además podrá comunicarse a través de Internet con otras computadoras para notificar el cambio de estado de la interfase, o en su caso tomar una orden de modificación proveniente de la computadora remota. Está última computadora que funciona como "cliente", correrá un programa que de igual forma se comunica a Internet para solicitar los datos que muestran el estado de la interfase y además puede capturar comandos dados por el usuario para modificar al 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 interfase.

De esta manera podemos ver que el proyecto se va dividiendo en varios módulos que se muestran en la siguiente imagen:

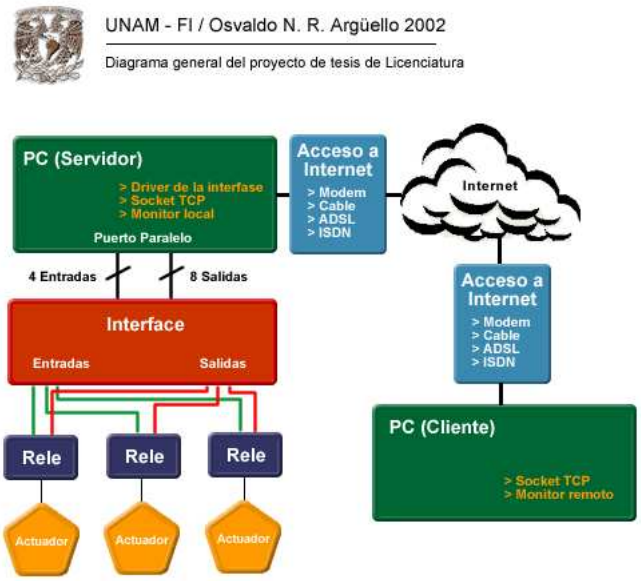

Figura 4.1: Esquema general del sistema por módulos.

4.4. Análisis por módulos

Para iniciar con la descomposición del sistema en módulos empezaremos por distinguir dos ramas principales, una será el software y hardware necesario en la computadora que estará encargada del control físico, directo de la interfase, y por otro lado estará el software requerido para el lado de la computadora que se conectará de manera remota para visualizar la información o para realizar una modificación en el estado de la interfase.

A la primera rama le llamamos "servidor" debido a que esta computadora estará encargada de escuchar las peticiones de la red y en caso de recibir información valida realizará una acción en la interfase paralela. A la segunda rama le denominaremos "cliente" y en general será el programa que interactúe con el usuario mostrando información de la interfase y tomando las ordenes para enviarlas al servidor.

4.4.1. La interfase y el puerto de conexión

Basados en la investigación de lo diferentes puertos de comunicación del capítulo 2 es importante apreciar que los puertos de comunicación que ofrecen una opción más viable para el actual proyecto son el puerto USB y el paralelo, el primero

es 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 la complejidad en la programación de un controlador y la aun escasa información acerca de la implantación en hardware y la necesidad de incluir un microcontrolador externo a la PC en la interfase no parecen darle la elección final, 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 introducidos recientemente ofrece varias ventajas que a continuación se mencionan y que nos pueden ser útiles tomando en cuenta el objetivo planteado:

1. **Compatibilidad con prácticamente todas las PC.-** Desde la PC original de IBM podas las computadoras de este tipo cuentan con un puerto paralelo, recientemente y como se menciona en el capítulo 1, se han desarrollado modos de comunicación más completos que el original, permitiendo ahora trabajar en modo bidireccional o controlar mejor los códigos de error, sin embargo el protocolo original define 8 bits de salida individuales y 4 de entrada, las cuales son suficientes para realizar la interfase. Por lo tanto es recomendable utilizar el puerto paralelo en modo estándar para mantener esta compatibilidad hacia atrás que puede resultar muy valiosa pensando en que en prácticamente en cualquier PC podríamos conectar nuestra interfase sin modificación alguna. Además es importante notar 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 en el, 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 tienen los permisos adecuados, estas dos funciones nos pueden servir para mandar o recibir datos del puerto con solo apuntar a la dirección correcta del puerto.

3. **Compatibilidad TTL.-** A nivel de electrónica, este puerto ofrece una salida TTL estándar, esto quiere decir que trabaja con 5Vcc aunque ofrece muy poca corriente, al rededor de 20 mA máximo, así que no puede manejar directamente dispositivos que requieran potencias altas, para esto se necesita que la interfase 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 bastante común y barato (Centronics y DB25)

y nos permite fabricar cables económicos, con cables sin blindar podemos garantizar buena comunicación hasta 10 metros.

La interfase entonces, estará conectada en este caso al puerto paralelo, aunque para futuros desarrollos no se descarta la posibilidad del USB. Ahora, ya que el puerto en modo estándar tiene 8 líneas de salida y 4 de entrada, debemos pensar que aparatos son lo que queremos que se conecten y que tipo de señales nos ofrecerán o cuales serán las que les tengamos que enviar para activarlas o desactivarlas.

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:

Salida

- Lámparas de cualquier tipo
- Ventiladores, calefactores o aire acondicionado
- Motores (bomba de agua, cortineros, persianas, puertas automáticas)

Entrada

- Interruptores de contacto (puertas, ventanas) ->**Digital**
- Sensor de luminosidad solar ->**Analógico**
- Sensor de temperatura ambiente ->**Analógico**
- Sensor de velocidad del viento ->**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 módulo de potencia que estará integrado a la interfase, este estará formado principalmente por un circuito basado en un transistor que trabajará en modo de corte - saturación el cual excitará a un relevador el cual finalmente funcionará como un interruptor común para los aparatos que conectemos a el, se puede 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 del circuito que puede ser empleado para la etapa de potencia de la interfase es el siguiente:

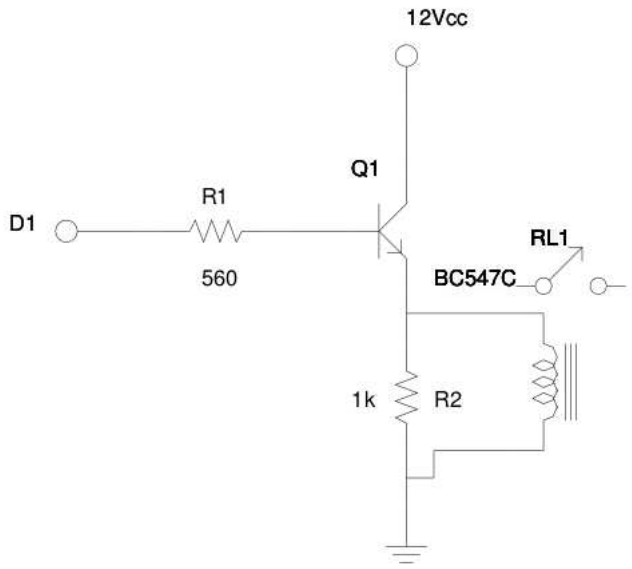

Figura 4.2: Ejemplo de circuito de salida que podrá ser empleado en el sistema.

Para los dispositivos de entrada existen varias alternativas para poder recibir los datos, en el caso de las entradas digitales la señal puede ser conectada directamente a uno de los bits del puerto y solo se deberá considerar un pequeño circuito para acondicionar la señal al voltaje y 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 precisión muy alta ya que la conexión a través de Internet nos impide garantizar una respuesta rápida, podemos acoplar un convertidor analógico - digital al mismo puerto paralelo, es deseable que se utilicen el menor número de bits posibles del puerto para que los demás se puedan emplear para el resto de las señales digitales, por esta razón, una alternativa muy atractiva para este tipo de sistemas son los convertidores analógico - digital "serie", que como su nombre lo indica transmite los datos resultantes de la conversión, de manera serie, esto quiere decir que sólo 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 esté 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 que acondiciona la señal de entrada para los bits de entrada del puerto:

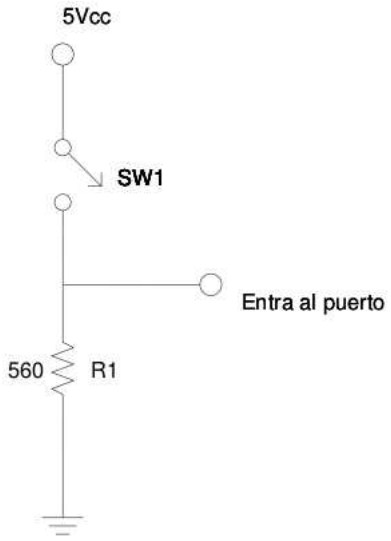

Figura 4.3: Ejemplo de un circuito de entrada que puede ser usado en el sistema.

Como vemos, el circuito es muy simple pero en realidad es todo lo que se necesita para que el pin del puerto se excite de manera confiable y a la vez para protegerlo, sin embargo, el circuito que proponemos para la adquisición de señales analógicas, se complica un poco más, debido a la inclusión del convertidor analógico - digital. El funcionamiento general de este circuito se basa en el DAC serial MAX186 que tiene la ventaja de utilizar solo 2 líneas de datos para enviar la señal, algunas más son usadas para la sincronia de los datos entre la interfase y la PC. Adicionalmente se presenta un 74LS5373 que es un buffer que sirve para adquirir las señales digitales directamente del conector; es importante aclarar que aunque los DAC seriales simplifican considerablemente las conexiones necesarias y las líneas de datos requeridas, también en este mismo grado complican la interpretación de los datos ya que estos se presentan como una secuencia de flancos de subida y bajada en una de las líneas de datos y el programa deberá implementar la interpretación del código usado en esta típica comunicación serial, en seguida el diagrama de este circuito:

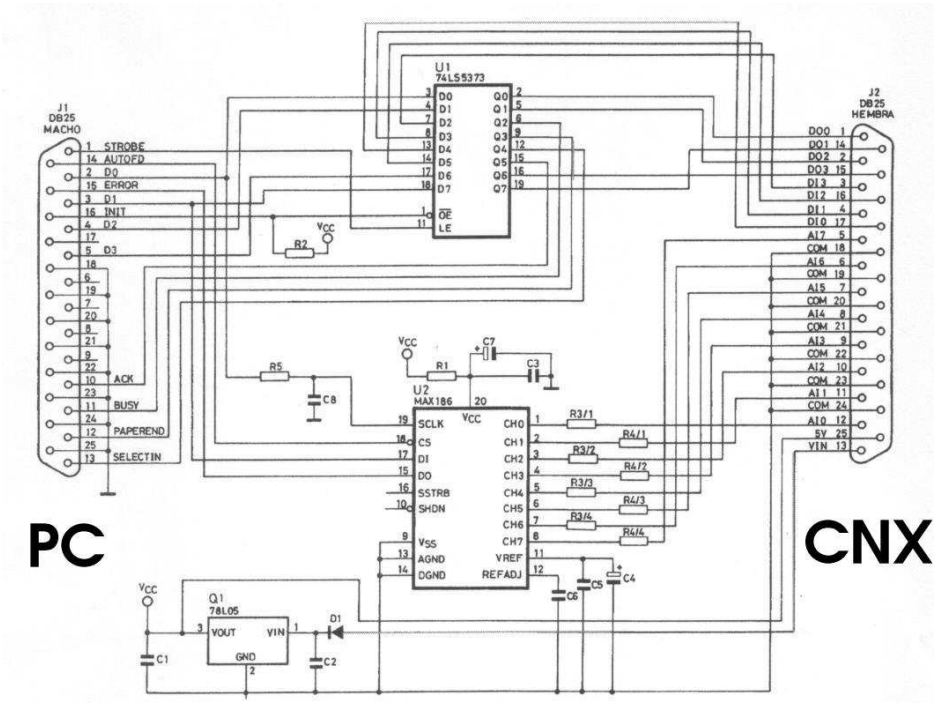

Figura 4.4: Circuito propuesto para la adquisición de señales analógicas.

Aunque esta es la primera aproximación al hardware necesario en la interfase ya se puede apreciar los módulos principales de los cuales consta está parte del sistema, más adelante se considerarán todos los puntos mencionados en esta sección que servirá para la implantación del prototipo de manera real. Quizá en esta implantación, algunas características se vean reducidas o canceladas por limitaciones de tiempo, dinero o técnicas, pero eso se analizará más adelante.

4.4.2. El servidor

Como se describió en líneas anteriores, el programa que llamamos servidor tendrá la tarea de controlar en primer lugar el puerto paralelo y en segundo la comunicación con Internet, este correrá en la computadora que tenga físicamente conectada la interfase al puerto paralelo. El control del puerto se hará principalmente a través de llamadas a las funciones inb() y outb(), definidas en el lenguaje "C" ANSI, para que estas funciones puedan emplearse en el programa tiene que realizar una petición de permisos al sistema que le den la facultad de escribir y leer directamente en el hardware, esto se realiza con la función ioperm(), esta última sólo puede ser utilizada por un programa que sea ejecutado por el administrador del sistema o "root", de está forma nos aseguramos de que

ningún usuario convencional pueda modificar el estado de la interfase de manera accidental o con propósitos negativos.

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 así. Primero abriremos un socket de red en el puerto TCP 6543 que "escuchará" 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 último tenemos el mensaje que pasaremos al interprete 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 que en este punto lo podemos considerar una línea de comandos dada por el cliente para separarla en los diferentes componentes que la forman, esto es, comando, argumento 1, argumento 2, etc. Luego tenemos una serie de opciones disponibles para el comando, como son, "apagar", "prender", "desc", "ayuda", "guarda", "recupera", etc., dependiendo el 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 Lampara1" esto es interpretado así: "desc" es el comando que cambia la descripción del aparato que esta 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 número 1. Algunos de los comandos utilizan uno o dos argumentos y sólo 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, más 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 mascara de bits que deseamos modificar y solo hacen referencia a los

bits de salida que son los que podemos alterar por software (como vimos en el capítulo 2, los bits de entrada sólo cambian de estado por hardware, en otras palabras son de solo lectura para el programador), la manera en la que la mascara modifica el estado es la siguiente:

Figura 4.5: 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 que modificarán el estado de la interfase 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 sistema operativo adecuado, en una primera aproximación se piensa en este programa como una aplicación hecha el lenguaje "C". Sin embargo, podemos hacer otra versión en "Java" con la finalidad de que pueda ser portable y por lo tanto pueda correr en cualquier S.O. que cuente con una máquina virtual "Java".

Este programa tendrá una estructura similar al servidor en lo que respecta a la conexión de la red y al interprete de comandos, sin embargo, cuenta con características especiales que lo harán más amigable e interactivo con el usuario. Esta dividido también en varias etapas las cuales enlistaré y explicaré a continuación:

- **Monitor del puerto paralelo.-** la función principal del cliente será ofrecer 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 que pasa con los aparatos que conectemos y así tomar decisiones de que prender y que apagar. Básicamente tendremos en cada renglón, la información del estado del bit, su número y la descripción de el aparato que

está conectado a el, esto será para los 8 bits de salida como los 4 de entrada, como hemos mencionado antes, los bits de entrada son solo de lectura y modificables por hardware, no así los de salida que son modificables por software.

- **Interprete de comandos.-** El usuario del sistema encontrará en esta sección del cliente una manera de interactuar con el sistema a través de comandos, 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 en archivos de texto, las descripciones de los bits o el estado de estos. Los comandos ingresados en la línea de comandos, son empaquetados para ser enviados al servidor, este sabemos que ejecutará el comando si es valido 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 línea de comandos es enviada al servidor a través de la red, este mensaje es encapsulado por parte del sistema operativo a solicitud 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 sientan las bases principales de lo que será el diseño general del sistema, vemos que está conformado tanto por software como por hardware, estas dos partes son a su vez divididas en otras etapas que forman las características principales del esqueleto principal del sistema, en el resto de este trabajo nos referiremos constantemente a este capítulo ya que es la parte que define las funciones del sistema que servirá para cumplir con nuestro objetivo.

Capítulo 5

Diseño y construcción del prototipo

En este capítulo iniciaremos el trabajo que definirá el prototipo que se realizará para comprobar que es posible construir un sistema domótico elemental con las características mínimas especificadas líneas atrás 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 capítulo anterior aunque no necesariamente seguirá fielmente la definición original ni las características, 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, pero por ningún motivo se deberá considerar como un producto terminado, ya que para esto se requiere de una infraestructura e inversión que están fuera del alcance de este trabajo, no obstante, el concepto puede ser usado en algún momento para crear un sistema verdaderamente capaz de cumplir con las especificaciones y características necesarias para un producto comercial.

A continuación se plantea la manera en que cada uno de los módulos se realizará, incluyendo las especificaciones esperadas y la explicación de como se decidió hacerlo de este modo y con esas herramientas. Iniciaremos en el mismo orden usado en el capítulo anterior, empezando por la interfase para seguir después con el servidor y el cliente.

Describe detalladamente el diseño final, que resulto de la evolución de una serie de pruebas preliminares basadas en el diseño conceptual del capítulo 4 y que no se describen completamente en este ni los capítulos posteriores

5.1. Interfase puerto paralelo - aparatos sin acceso a Internet

Basados en el diseño conceptual podemos decir que esta interfase será usada para 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 de 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 60Hz) y con un margen de potencia suficiente para aplicaciones sencillas (1000 watts). Además para los bits de entrada, la interfase, nos debe de ofrecer una señal de 5V y con una corriente de al rededor de 10 mA, 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 organización, se separará en varias etapas el circuito que formará la interfase, una será la encargada de los bits de salida, otra de los de entrada y finalmente una fuente de alimentación externa que alimentará al circuito y que proveerá la potencia necesaria para los relevadores, sin que esta sea demandada del puerto de la computadora.

5.1.1. Bits de salida

Como ya se dijo en el capítulo 2 sección 2.1.2, el puerto paralelo tiene 8 bits de salida, eléctricamente estos son salidas 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 está en "1" la salida estará en 5V y para el estado "0" la salida estará en 0 V, otra característica importante de este puerto es que el estado de los bits es mantenido 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 el programa este constantemente actualizando el valor de los bits.

En las referencias técnicas que describen el uso del 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 20 mA, de no ser así es muy probable que el puerto se queme, así que el objetivo más importante del circuito de salida será controlar esta corriente para no exceder el límite mencionado.

Por otro lado, usaremos un circuito basado en un transistor en modo de corte y saturación para amplificar la corriente del puerto y obtener la suficiente para excitar un relevador típico que será 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:

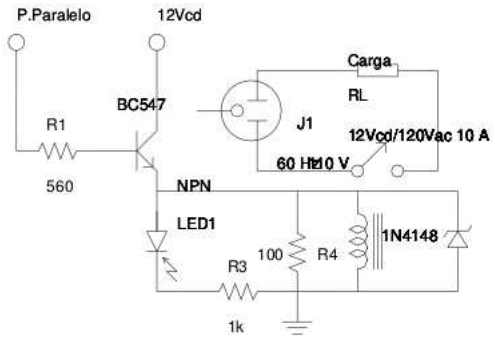

Figura 5.1: Circuito de salida para un bit del puerto paralelo.

Como vemos en el circuito, la salida del puerto paralelo entrará a través de una resistencia que limita la corriente, a un transistor común BC547, este está conectado de tal forma que funcionará en modo de corte o saturación, dependiendo del valor de voltaje que reciba del puerto paralelo, así en caso de que el voltaje sea 5 V en la entrada del circuito, el transistor entrará en saturación, y permitirá el paso de corriente entre su colector y el emisor, esto hará pasar corriente por el relevador y lo activará, por otro lado también pasará corriente por la resistencia de 1kohm y el led que servirá para indicar la activación del circuito. También es importante notar que se tiene un diodo 1N4148 ó 1N918 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 parásitas o espurias 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 su colector y emisor, el relevador se desenergiza y con esto se presenta un 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 127 VAC a 10A, que nos permite encender o apagar aparatos caseros pequeños, si embargo existen una gran variedad de relevadores que también se alimentan con 12 Vcc pero que soportan cargas de conmutación mucho mayores. Para fines ilustrativos, el relevador utilizado es suficiente.

A continuación se presenta una lista de los componentes necesarios para construir el 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 2.2 Kohms a 1/2 watt
- 1 Resistencia de 1 Kohm a 1/2 watt
- 1 Transistor BC547 (A,B,C)
- 1 Led común
- 1 Relevador 12 Vcc / 127 VAC, 1A
- 1 diodo 1N4148

Protección adicional

Con el fin de hacer mas confiable y seguro el circuito para la computadora, es posible agregar al circuito anterior un módulo adicional de protección de sobre corriente que consistiría en un circuito 74LS245 que es un Buffer a la salida del puerto y también un optoacoplador que podría sustituir al transistor BC547 realizando la misma función y además aislaría al circuito de potencia del de la computadora. El 74LS245 tiene 8 entradas y 8 salidas los que nos permite conectar en un solo paquete todos los bits de salida del puerto, en el caso del optoacoplador existen también paquetes con 4 entradas por 4 salidas independientes que disminuyen considerablemente el tamaño del circuito al no necesitarse un paquete DIP de 6 patas por cada optoacoplador, sin embargo, estos componentes son difíciles de conseguir.

Este sería el circuito modificado con la protección adicional.

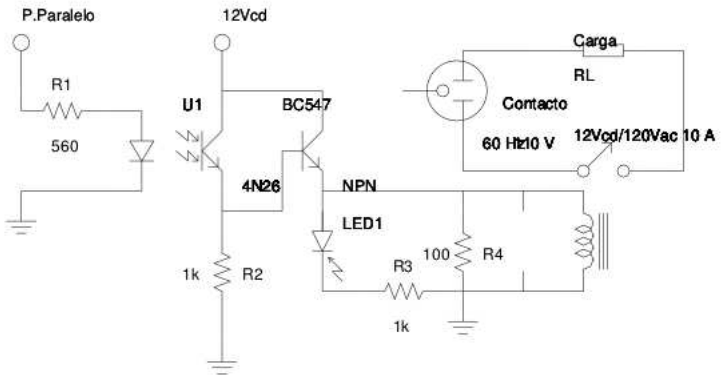

Figura 5.2: Circuito de salida modificado con protección para el puerto paralelo.

5.1.2. Bits de entrada

Para los bits de entrada digital mencionamos antes que se requiere un circuito que garantice un voltaje de 5 Vcc y al rededor de 10 mA que será aplicado al puerto para excitarlo, este circuito es un poco más simple que el de salida, consiste solo en una resistencia de pull-up conectada a una fuente de 5 Vcc 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 que sirven de entrada, el estado de estos bits es modificable únicamente por hardware, el software sólo se dedicará a leer los datos del registro del puerto destinado para los bits de estado (STATUS), el circuito sería así:

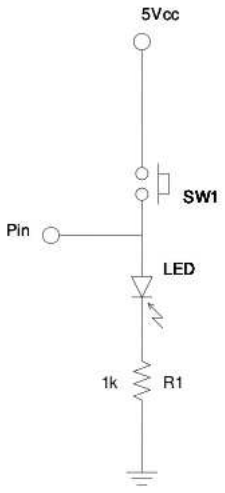

Figura 5.3: Circuito de entrada propuesto para el sistema.

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

5.1.3. Fuente de alimentación

La fuente de alimentación que requiere la interfase deberá proveer 5 y 12 Vcc regulados, y como la carga más importante que obtendremos de ella será la necesaria para excitar los relevadores, en general no requerimos más de 1A, el circuito propuesto es una clásica fuente con regulador de estado sólido LN7405 y LN 7412 para cada uno de los voltajes, aquí se muestra el diagrama del circuito:

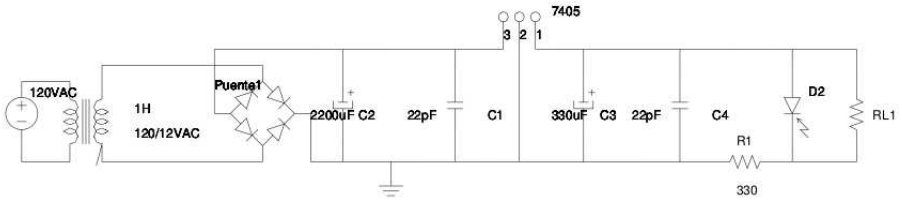

Figura 5.4: Diagrama del circuito de la fuente de alimentación de 5Vcc, la fuente de 12Vcc es identica solo cambia el C.I. regulador de voltaje del 7405 al 7412.

Vemos que la fuente inicia con la reducción del voltaje de 127 VAC a 15 VAC 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 poco de rizo, es aplanada más por dos capacitores para después pasar al regulador de estado sólido que a la salida genera un señal continua, se agrega un capacitor cerámico al final para el desacoplamiento.

Otra opción disponible para obtener la alimentación necesaria para la interfase es obtener los 5 y 12 Vcc directamente desde la fuente de la PC, esta tiene conectores que típicamente son usados para alimentar a los discos duros o CD-ROM que tienen precisamente un pin con 5 otro con 12 voltios y otro a tierra, el inconveniente de esta alternativa es que este conector esta en el interior del gabinete de la PC y para acceder a el se tendría que hacer una adaptación para conectarlo a la interfase, que estará en el exterior, aunque el ahorro de dinero y espacio obtenido es considerable.

5.2. Programa servidor

Ya en el capítulo anterior describimos 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 interfase 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 cuando se presenta una petición por parte del usuario, como vemos en el diagrama, el servidor estará esperando que desde la red llegue un paquete cuyo destino sea el, una vez que esto ocurre, el socket recibe un paquete de información que es descompuesto

en varias capas para determinar los parámetros de comunicación de las computadoras involucradas, como resultado del trabajo del socket obtendremos una cadenas de caracteres que contienen la dirección IP de origen, su longitud y la línea de comando que el usuario escribió en el cliente, de esta manera seguimos con el interprete de comandos. Los que este módulo hace es analizar la línea de comando para separarla 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 "prende" o "apaga" se modifiquen los bits correspondientes en el puerto. Por último el servidor enviará de regreso al cliente la misma cadena de caracteres que recibió con la finalidad de verificar que la aplicación del comando fue exitosa. En caso de que alguno de lo pasos antes mencionados no se realice con éxito, se devolverá una cadena de error que el cliente será capaz de interpretar para avisar al usuario.

Para el caso en el que la información provenga de la interfase, por ejemplo, cuando un sensor de una ventana se abra, la ruta de la información no varia 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 notificará 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 todos los módulos que se mencionaron 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 vemos el refresco de los datos es mas una tarea del cliente que del servidor.

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

5.2.2. La comunicación con Internet

La comunicación a través de Internet la realizaremos empleando los sockets, como se menciono anteriormente, los sockets no son más que descriptores de una conexión a red y funcionan de manera muy similar a los descriptores de archivo que son números enteros que apuntan a una localidad de memoria que representa al archivo, en el caso de lo sockets, representan la conexión a la red, de esta manera podemos llamar a las funciones de escritura y lectura y pasar como argumento este descriptor para hacer referencia a una conexión en particular, la manera en la que los sockets son implementados en "C" es la siguiente:

- Se incluyen las librerías: <sys/types.h>y <sys/socket.h>
- Se inicializa la estructura: struct sockaddr_in, con los datos de la IP y el puerto de comunicación en sockaddr_in.sin_addr y sockaddr_in.sin_port respectivamente.

- Se realiza una llamada a la función: int socket(int dominio, int, tipo, int protocolo>
- Ahora se llama a bind() para vincular el numero de puerto al descriptor devuelto por socket().
- Una vez echo esto se llama a connect() para esperar a que se realice un petición por la red.
- Si se recibe una conexión a la red, invocamos a la función accept() que como su nombre lo indica acepta el flujo de datos entrante.
- Por último una vez que la conexión se establece entre las dos computadoras, contamos con las funciones send() y recv() para poder enviar y recibir datos a través del socket que para este momento ya esta funcionando, en otras palabras estas son nuestras funciones de lectura y escritura para el descriptor de red.

Como podemos ver, existe ya una convención para establecer este tipo de conexiones a red en los programas hechos en lenguaje "C", así que solo es cuestión de seguir puntualmente los pasos mencionados arriba para contar con una conexión a Internet muy fácil de usar, con el fin de no extenderse en estos puntos que son por su naturaleza técnica ampliamente documentados no entramos en detalles finos de la programación de estas estructuras, fin embargo se ofrece en un apéndice de este documento, el código fuente del servidor completo 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 del paquete de información por parte del socket, este realiza un proceso de desincorporación de las capas superficiales del paquete hasta llegar al último bloque que contiene la línea de comando escrita por el usuario en el programa cliente.

5.2.3. El interprete de comandos

El siguiente paso en el diagrama del 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 esta formada por un comando y varios argumentos. La mayoría de los comandos solo tienen 1 o 2 argumento pero este interprete no está limitado a reconocer una cantidad mayor de argumentos.

En primer lugar el interprete de comandos separará 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", el interprete recorre de inicio a fin la cadena buscando espacios, cuando encuentra uno, todos los caracteres que estén detrás de este

espacio 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 primer elemento del arreglo siempre será el comando y los posteriores serán los argumentos.

El segundo paso del interprete de comandos será reconocer si el comando ahora alojado en el primer registro del arreglo es válido o no. Esto se realiza pasando al comando por una sería de estructuras condicionantes que determinan por comparación de cadena si el comando existe o no, en caso de que sí exista, o sea que el comando recibido por la red es igual que una de las cadenas de comparación, 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 y se devuelve al cliente un mensaje de error. Cabe aclarar que este tipo de errores se deben únicamente a problemas en la comunicación ya que el programa cliente antes de enviar una solicitud al servidor analiza el comando para verificar que es válido.

5.2.4. El manejo del puerto

El último módulo importante que forma parte del servidor es el encargado de manejar el puerto paralelo tanto para entrada como para salida, ya en el capítulo pasado se había mencionado que existen dos funciones implementadas en lenguaje "C" que nos permiten leer y escribir en algún especio del mapa de memoria de la PC, así conociendo la dirección del puerto paralelo podemos escribir y leer en el, estas funciones son inb() y outb() y requieren como parámetros la dirección en donde se desea leer o escribir y en el caso de outb() la mascara de bits que se escribirá, para inb() solo se requiere asignar un variable de tipo entera a la función, ya que esta devolverá el estado del puerto en esta variable.

Así en el caso de que el comando sea "prende" o "apaga", el programa utilizará la función outb() con la misma dirección (0x378) pero con una mascara diferente para cada caso, por ejemplo: si tenemos esta configuración en los 8 bits de salida del puerto 01110110 (118) y pedimos prende el bit 4, entonces se genera un mascara así 00001000 para que al realizar una operación OR bitwise se obtenga la siguiente configuración 01111110. En caso de querer apagar el bit 6 para ese mismo ejemplo inicial la mascara sería 00100000 para que con una operación XOR bitwise se tenga 01010110.

El valor resultante de la manipulación del puerto es almacenada 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.

5.2.5. El problema de la multitarea

Inicialmente se consideró 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 en el momento en el 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 escribiera un comando y entonces atenderlo, aquí es donde se presenta el problema, cuando el programa espera peticiones en la red se "detiene" en un ciclo hasta que la petición se presente, esto impide que se pueda poner atención en lo que el usuario escribe en la línea de comando ya que en ese momento el programa está "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 o programación multihilos que permite a un programa bifurcar el camino único que tradicionalmente recorren los datos, a dos o más, para que el 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 trabajo. Pero el argumento más fuerte por el cual no se usa la programación multihilos es que podemos cambiar la idea de que el servidor atienda comandos de manera local haciendo que el cliente pueda correr también en la máquina donde corre el servidor y así dejar la tarea del ingreso de comandos al cliente. De esta manera el servidor estará dedicado a controlar la conexión a la red y el puerto paralelo únicamente, este fue el esquema que finalmente se uso, 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 interfase

Un punto importante en lo que respecta a la utilización del sistema, será sin duda el control de los dispositivos conectados a la interfase 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 interfase y cuando la máquina 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 interfase deberá en principio, respectar el estado

de los dispositivos que tengamos conectados y no modificar este sino existe una petición expresa del usuario.

5.3. Programa cliente

El módulo que falta para completar el sistema es el programa cliente, sabemos que este programa se correrá en las computadoras remotas, o sea, en las computadoras que no tienen conectado directamente al puerto paralelo la interfase, estas computadoras estarán en cualquier punto del mundo, conectadas a Internet a través de diversos tipos de conexión, desde un MODEM hasta enlaces dedicados de varios Mbps. Este programa ofrecerá al usuario la información del estado de la interfase conectada al servidor de manera remota, además provee la posibilidad de que el usuario pueda ingresar en una línea de comandos, las instrucciones que desea sean transmitidas hasta el servidor, 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 interfase.

Este programa está programado en lenguaje "C" y con la inclusión de algunas librerías para el manejo de la pantalla, por el momento la versión que será entregada junto con este trabajo despliega la información en modo texto, sin embargo, ofrece todos los datos necesarios para conocer el funcionamiento general del sistema, en uno de los apéndices se ofrece información de como llevar esta versión a una programada en lenguaje "Java" y que tenga una interfase gráfica con las mismas funciones que ofrece la versión actual.

Vemos en el diagrama que el programa estará esperando que el usuario desde el teclado ingrese una línea de comando y que al final presione la tecla "Enter", en ese momento se inicia el proceso de reconocer si esa cadena esta formada correctamente y no contiene errores, esta tarea es realizada por un interprete de comandos, similar al incluido en el servidor, en caso de que esta instrucción sea válida se pasa la cadena de caracteres completa al socket que como hemos visto en la sección dedicada al servidor, se encarga de la conexión a la red. Este socket realiza las funciones de conexión, definición de puerto y transmisión del paquete al servidor, una vez hecho esto, espera que el servidor realice la tarea encomendada y en caso de que no exista ningún error ni en la transmisión ni en la realización de la orden, recibirá una cadena de caracteres de parte del servidor idéntica a la que se envió. Esta última es empleada para actualizar los datos mostrados en la pantalla y permiten al cliente visualizar que su orden fue ejecutada con éxito.

5.3.1. La información presentada

La pantalla del cliente esta dividida en varias partes de manera horizontal, la primera muestra los títulos en donde se incluye el nombre de la universidad, la

facultad de ingeniería y el autor y trabajo, posteriormente tenemos una sección en donde se muestra en cada renglón cada uno de los bits de salida y entrada del puerto, del lado derecho los 8 bits de salida y del izquierdo los 4 de entrada, cada línea esta formada por varios datos de la siguiente forma:

No. del bit:estado:descripción

Ejemplo:

Bit 5:1:Lámpara4

En el ejemplo vemos que esta línea se refiere al bit 5 de salida (debido a que se encuentra en la parte derecha de la pantalla), que esta encendido mostrado por el número 1 y que a este bit está conectada la lampara 4.

La siguiente parte de la pantalla muestra una pequeña bitácora que estará presentando información concerniente a la aplicación de los comandos en el servidor, así por ejemplo si solicitamos prender el bit 4, la bitácora presentará una línea que indica que el comando se ejecuto de manera exitosa, así:

Se encendió bit: 4, solicitado por: 192.168.7.198

Los mensajes mostrados en la bitácora variarán dependiendo del comando que se solicite y del resultado que se obtenga en el servidor, pero en general tendrán esta estructura. La bitácora esta formada por 4 líneas en la pantalla las cuales se estarán recorriendo conforme nuevos mensajes aparezcan.

Por último tenemos la sección en la que el usuario ingresa los comandos, está formada por dos líneas en la pantalla, la primera muestra el comando anterior que fue solicitado a manera de comprobación o recordatorio, la segunda y más importante muestra un "prompt"[1] que nos indica que podemos ingresar una línea de comando, es aquí donde el usuario escribe los comandos que serán enviados al servidor, por ejemplo:

gdomo>desc 1 Motor_bomba_de_agua

5.3.2. Comandos disponibles

Desde este programa tenemos la posibilidad de controlar todas las funciones que el sistema realice así que cada una de estas acciones deberá representarse por un comando, a continuación se presentan los comandos disponibles hasta el momento junto con los argumentos necesarios para su ejecución correcta en el servidor; en el futuro es posible que se incorporen nuevos comandos de acuerdo a las extensiones que el sistema sufra.

[1] Es un símbolo usado normalmente en el shell de algunos sistemas operativo que indica que la computadora esta lista para recibir una orden de manera escrita como línea de comando, en algunos casos ofrece información adicional como la localización del usuario en el árbol de directorios en ese momento o el tipo de usuario que se tiene en dicho sistema operativo.

Comando	Arg 1	Arg 2	Descripción
ayuda	[comando]	-	Ofrece una descripción de los comandos disponibles, si se agrega el argumento1, se muestra información especifica de ese comando.
prende	num_de_bit	-	Enciende el bit de salida que se indica en el argumento1 sin modificar los bits restantes.
apaga	num_de_bit	-	Apaga el bit de salida que se indica en el argumento1 sin modificar los bits restantes.
refresca	-	-	Solicita al servidor el estado actual de la interfase y actualiza los datos en la pantalla, es útil para reconocer cambios en los bits de entrada.
desc	num_de_bit	descripción	Modifica la descripción del aparato que está conectado al bit indicado en el argumento1.
guarda	desc	nombre_archivo	Guarda en un archivo en disco, la descripción de los bits.
recupera	desc	nombre_archivo	Recupera de un archivo en disco, la descripción de los bits.

Cuadro 5.1: Comandos disponibles en el programa cliente.

Nota.- los comandos deben introducirse exactamente con la palabra reservada.

5.3.3. Los argumentos de inicialización del programa

En este programa existen ciertos valores que pueden variar de usuario en usuario y de computadora en computadora, como la dirección IP del servidor o su nombre o incluso la descripción de los aparatos y su estado, debido a esto y a que es muy engorroso estar ingresando estos datos por parte del usuario cada vez que se utilice el programa, este ofrece la posibilidad de pasar estos datos a través de la línea de comando que se ingresa en el sistema operativo como argumentos, así cada uno de los datos tiene un valor estándar que en caso de que no se presente un argumento en su lugar, no será modificado, sin embargo si el usuario ingresa un argumento en cierto orden, este valor será usado en lugar del estándar para esa ocasión, los valores estándar usados son los siguientes:

- Dirección del servidor: 192.168.7.195 (Dirección del servidor usado para pruebas)
- Archivo de descripciones: /tmp/desc.gdomo

para mostrar esto mas claramente usaremos un ejemplo:

1. Imaginemos que la IP del servidor es 212.45.34.127, además en una sesión previa guardamos los valores de descripción en el archivo desc12.gdomo, entonces la línea de comando queda así: $cliente 212.45.34.127 desc12.gdomo.

Es evidente que el orden en el que los argumentos deben ser ingresados es el siguiente:

1°	dirección IP o nombre del servidor
2°	nombre del archivo de descripciones

5.3.4. Imagen de la pantalla

Por último y para mostrar con mayor detalle la forma del programa cliente se presenta a continuación una imagen que muestra la pantalla principal de este:

Figura 5.5: Imagen de la pantalla principal del programa cliente, muestra la información del estado del puerto y ofrece una linea de comandos para escribir las instrucciones.

5.4. Construcción del prototipo

5.4.1. Interfase puerto paralelo - aparatos

Configuración física

Todo el conjunto que forma la interfase está contenido en un gabinete plástico de 15.2 cm. X 7.6 cm., como se muestra en la siguiente foto:

Figura 5.6: Foto del gabinete de la interfase.

En la parte lateral podemos apreciar la conexión polarizada a la toma de corriente domestica, el botón de encendido en color rojo y un conector DB9 hembra que servirá para conectarse al puerto paralelo de la PC, en este punto es bueno mencionar que este prototipo solo usará dos entradas y dos salida, las cuales son suficientes para mostrar el funcionamiento del sistema y realizar las pruebas de verificación de funcionamiento.

Un cable DB25 - DB9 servirá para interconectar el puerto paralelo (que por estándar tiene un conector DB25 hembra en el gabinete de la PC) y la interfase, la configuración del cable es la siguiente:

DB25 (Macho)	DB9 (Macho)
(D0) Pin 2	(Salida 1) Pin 1
(D1) Pin 3	(Salida 2) Pin 2
(Busy) Pin 11	(Entrada 1) Pin 3
(Paper empty) Pin 12	(Entrada 2) Pin 4
(GND) Pin 18	(GND) Pin 5

Cuadro 5.2: Configuración del cable de conexión entre el puerto paralelo (DB25) y la interfase (DB9).

En la parte superior tenemos dos conectores que servirán para conectar los aparatos domésticos, estos tienes tornillos con los que fácilmente se pueden fijar cables de calibre hasta 10 AWG, los conectores con cables verdes son los de salida y los de cables rojos son los de entrada, a continuación se muestra el diagrama de conexiones que está colocado en el gabinete junto a estos conectores:

Figura 5.7: Panel de conexiones de la interfase

Como ya sabemos, esta interfase esta dividida en tres circuitos, la fuente de alimentación fue construida en una placa fenólica de 2.8 cm. X 6.9 cm. y junto

con el transformador se colocaron dentro del gabinete el cual contendrá también los circuitos de entrada y salida. La fuente tiene un conector con tres terminales ajustables con tornillo que proveen los 5Vcc, 12 Vcc y tierra requeridos, así las tarjetas de entrada y salida pueden fácilmente conectarse a la fuente sin necesidad de soldadura, esto es con la finalidad de darle a todo el aparato una división modular que nos permita sustituir cualquiera de las tarjetas sin mucha dificultad y sin el uso de herramientas especiales, aquí se muestra una foto de la tarjeta terminada:

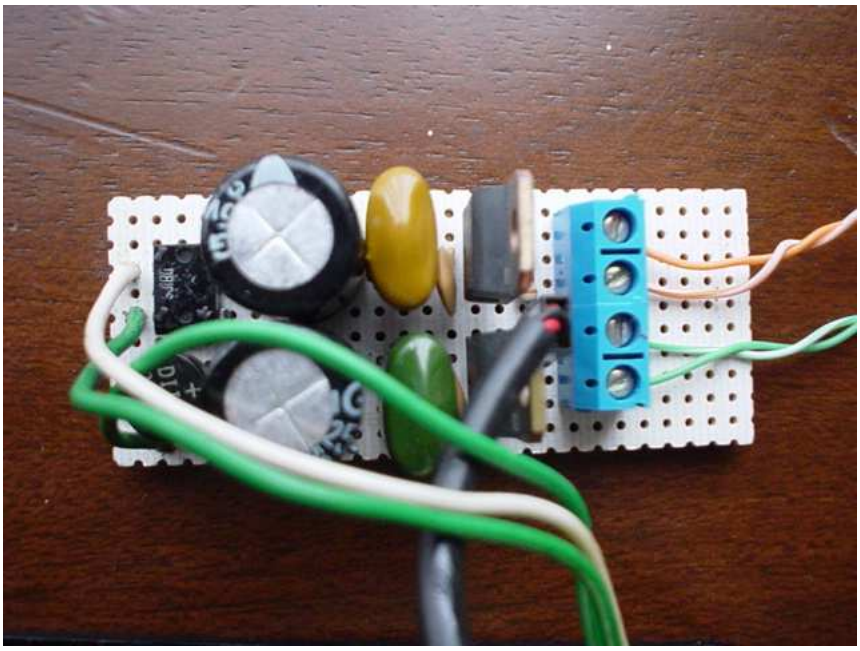

Figura 5.8: Vista final de la fuente de alimentación de 5Vcc y 12 Vcc

Otra tarjeta contiene los circuitos de entrada y salida, esta mide 3.8 cm. X 6.9 cm. y tiene tres conectores de tornillos, uno es para conectar el bit de salida del puerto, tierra y 12 Vcc y otro esta conectado a las terminales de salida del relevador, es evidente que estos dos conectores forman parte del circuito de salida, el tercero, recibe el bit de entrada, tierra y 5 Vcc y sirve para el circuito de entrada este es el aspecto de la tarjeta:

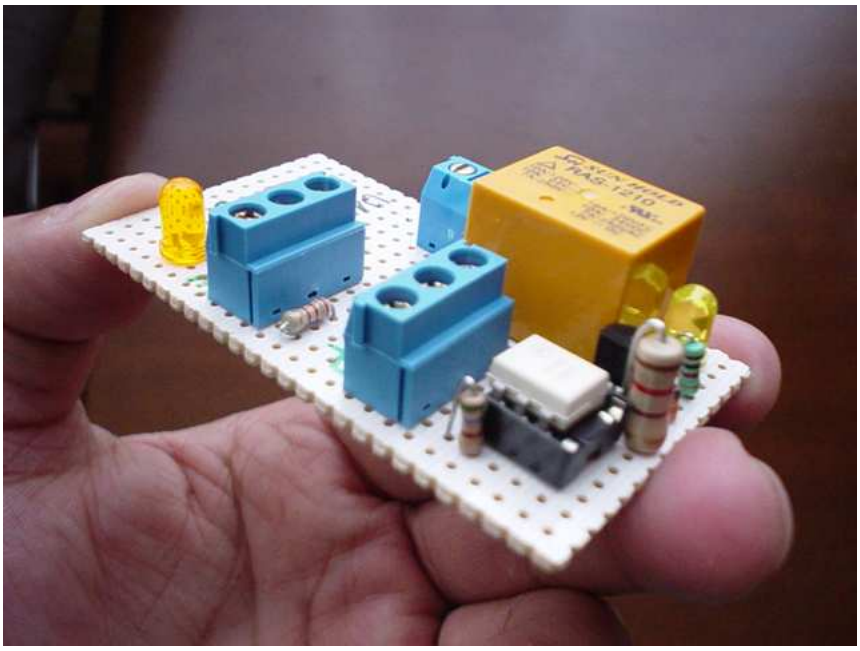

Figura 5.9: Vista final del módulo de entrada y salida.

Como el sistema puede ofrecer 8 entradas y 4 salidas, esta tarjeta al ofrecer solo una entrada y una salida puede repetirse para completar el conjunto, en el prototipo, solo se presentará dos veces, con lo cual, tendremos 2 entradas y 2 salidas. Esta diseñada geométricamente de tal manera que las dos tarjetas de entrada - salida caben perfectamente de manera vertical respecto al fondo del gabinete, así fácilmente se pueden realizar las interconexiones con la fuente de poder y en su caso, retirarlas para sustitución o reparación.

La vista final del interior del gabinete ya con las tarjetas y las conexiones apropiadas es la siguiente:

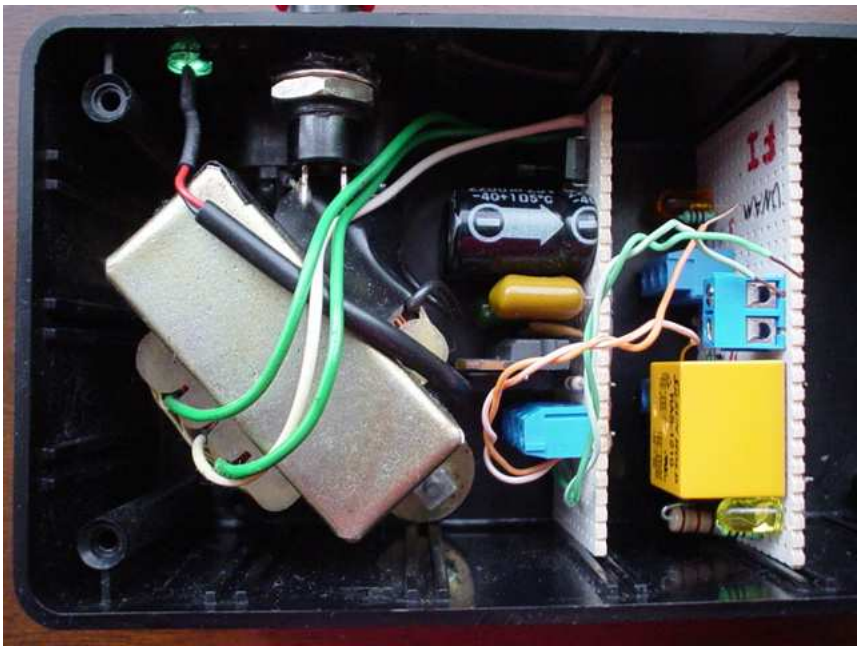

Figura 5.10: Vista del interior del gabinete donde se puede apreciar la disposición de los módulos de la fuente de alimentación y el de entrada - salida.

Circuitos definitivos y últimas modificaciones

Con respecto a los diagramas mostrados anteriormente, los circuitos definitivos sufrieron algunas modificaciones sencillas que fueron necesarias a partir del análisis del circuito en cuanto a potencia y dimensiones físicas, fundamentalmente se presentó un cambio en la potencia de la resistencia de 100 ohms que inicialmente se había considerado de 1/2 watt y que al final se instaló de 1 watt, esto debido a la cantidad de corriente que el relevador demanda para activarse, también se suprimió el 74LS245 y en cambio se empleó el optoacoplador 4N26 y un transistor BC547 común que sirve para alcanzar la corriente esperada ya que con el simple optoacoplador no alcanza, el diagrama final quedó así:

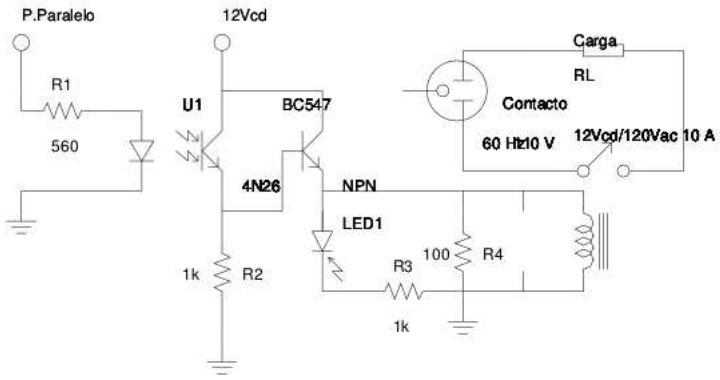

Figura 5.11: Circuito final de salida con protección.

En cuanto al circuito de entrada, solo se agrego un LED y una resistencia para indicar el estado del bit, si está encendido, el LED prende.

5.4.2. Programas cliente y servidor

Parte fundamental del sistema son los programas servidor y cliente ya que como se planteó en el objetivo el hardware que se agregó es mínimo y de esta manera la mayoría de las tareas fundamentales recaen sobre las piezas de software de las que ahora hablaremos.

Como ya sabemos, tanto el programa cliente como el servidor están escritos en "C" y por lo tanto siguen la ruta clásica del desarrollo de los lenguajes compilados, está es, escritura del código fuente, compilación, ligado, es algunos casos depuración y de nuevo al inicio del ciclo. Para la tarea de edición se emplearon fundamentalmente dos programas, "emacs" y "Anjuta"; ambos están diseñados para ser 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 sintácticas simples y permite compilar, depurar e incluso hasta ejecutar el programa sin la necesidad de salir del ámbito del editor.

La compilación y el ligado se realizaron con "gcc" el compilador de "C" de GNU, este es el compilador nativo de Linux y además de cumplir con el estándar ANSI y POSIX, incluye un sin número de extensiones y una API muy amplia, entre las librerías incluidas, está "ncurses" que es compatible con la original "curses" que se incluía en el conjunto de librerías del UNIX System V original de AT&T, actualmente casi todos los sabores de UNIX, poseen una versión de "curses", esta librería nos sirve para tener un mejor control de la terminal en cuanto a la

posición de los mensajes, la interpretación del teclado, y el despliegue de colores. Está librería se utilizó en el programa cliente para que la vista del programa fuera más clara y amigable para el usuario, cabe recordar que antes del auge del las interfases gráficas, este tipo de aplicaciones de texto con colores eran muy populares y aun hoy en algunas aplicaciones son ampliamente requeridas, sobre todo por requerir muy pocos recursos del hardware[1].

En el programa servidor usamos la API de "C" definida para el uso de conexiones a red, la librería "socket.h", contiene todas las funciones necesarias para desarrollar programas que usen las conexiones a redes de cualquier tipo, así con este conjunto de herramientas se pueden escribir desde servidores WEB hasta clientes de FTP[4].

En cuanto a las tareas de depuración, se uso "gdb" que es el depurador de GNU, esta herramienta nos permite hacer un análisis en tiempo de ejecución del comportamiento del programa, podemos ver el valor de las variables en cualquier punto, ejecutar línea a línea el programa, establecer punto de parada (breakpoints), modificar al vuelo valores de variables, etc. todo lo que comúnmente un depurador hace; además este programa puede ser invocado desde "emacs" el cual ofrece una integración muy interesante con el depurador que permite ver de manera más amigable el proceso de depuración.

Finalmente, los programas terminados pueden ser distribuidos en código fuente agregando un archivo "Makefile" que es usado por la utilería "make" para realizar la compilación de manera muy sencilla para el usuario. En el archivo "Makefile" se incluyen todos los parámetros que la línea de comando debe tener para llamar al compilador, así como una lista de las librerías que se usan y en algunos casos su localización, de esta manera, es usuario solo ejecuta el comando "make" para que tenga como resultado los archivos ejecutables particulares para su plataforma.

Capítulo 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 segundo término deberán identificar posibles defectos o errores probables en la operación. Este capítulo plantea la manera en la que los distintos módulos que forman el sistema serán probados, posteriormente se evaluarán los resultados obtenidos y así se llegará a las conclusiones finales del trabajo.

6.1. Interfase puerto paralelo - aparatos

Las pruebas que se efectuarán a la interfase tienen como objetivo principal verificar que los datos de la computadora, tanto de entrada como de salida lleguen a la interfase correctamente, que el circuito de salida pueda efectivamente interrumpir la corriente en circuito de corriente alterna casero, que el circuito de entrada mande el voltaje y corriente adecuados para excitar el puerto paralelo y detectar el accionamiento de algún sensor de posición y por último se realizará una prueba de estabilidad en el 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 que consistirá cada prueba y describiremos como se realizarán estas.

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

Existen dos puntos en los que la transferencia de datos entre la interfase y la PC pueden fallar más fácilmente, uno es en el cable DB25-DB9 y otro es en el interior del gabinete de la interfase, más específicamente entre las conexiones de cable del DB9 y la tarjeta del circuito. La prueba se llevó a cabo siguiendo estos pasos:

1. Comprobación de la continuidad de los cinco pines usados en el cable DB25-D9 con un multímetro digital.
2. Revisión de los puntos de soldadura o de fijación de los cables que conectan la tarjeta del circuito con el conector DB9 hembra colocado en el gabinete y comprobación de continuidad desde el conector DB25 hasta la tarjeta del circuito.
3. Comprobación de la atenuación del cable usando una señal de 1MHz cuadrada y un osciloscopio.

En general esta es la prueba más simple y debe verificar que todas las conexiones esten bien hechas y que la atenuación que sufra la señal aplicada al 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 de AC

En esta 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 un multímetro que el pin 2 del puerto paralelo se ponga en 5 Vcc.
3. Se apaga el bit desde el programa cliente y posteriormente se conecta la interfase a la PC.
4. Se pide nuevamente desde el programa que prenda el bit 1.
5. Con un multímetro 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 20 Watts entre la toma de corriente y las terminales de salida de la interfase, de tal forma que la interfase funcione como interruptor 1 polo 1 tiro y se verifica que la lámpara encienda.

7. Se vuelven a repetir los puntos 1 al 6 pero ahora con el bit 2 (pin 3).

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 sensor de posición

La adquisición de la señal digital que viene de un interruptor "push - button" es muy importante, de esta manera podemos verificar que una puerta o ventana esté 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 conecta el interruptor a las terminales del conector superior de la interfase siguiendo el diagrama de conexiones.

2. Se polariza la interfase y se acciona el interruptor, con un multímetro verificamos en el pin 11 o 12 del conector DB25 si se presenta voltaje, si este esta al rededor de 5Vcc y con una resistencia de 4.7 Kohms medimos la corriente que pasa (simulando la impedancia que el puerto va a presentar ya cuando el cable esté conectado a la PC).

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 utilizó en el prototipo marcan que la corriente máxima de interrupción es de 10A 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 serían de:

127 V x 10 A =1270 Watts (Solo para el caso puramente resistivo)

En un caso más real en donde los aparatos que conectamos presentan potencia reactiva, real y aparente podemos estimar que la capacidad máxima del circuito será de 800 watts.

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

1. Se busca un aparato casero que demande de la toma de corriente al rededor de 800 watts y se conecta a las terminales de salida según el diagrama de conexiones de tal forma que la interfase quede dispuesta como un interruptor 1 polo 1 tiro.
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 minutos.
3. Ahora se revisa la temperatura de los componentes y en especial del relevador.
4. De nuevo se enciende el bit pero ahora se mantendrá así durante 8 horas con carga, al final se revisa la temperatura y es estado físico de los componentes.
5. Se puede repetir la prueba para el otro 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 verifica 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, sí es la que parte que ofrece mayor complejidad en cuanto al diseño y por lo tanto existen mas factores que influyen en la buena o mala operación de los programas.

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

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

Para esta prueba usaremos inicialmente un programa que solo tiene implementado el control del puerto paralelo, con este programa que llamaremos "p-paralelo" podemos verificar que las funciones "inb()" y "outb()" junto con "ioperm()", efectivamente nos permitan manejar el puerto, posteriormente correremos el servidor que posee estas mismas funciones y se realizará la prueba:

1. Se inicia la sesión de trabajo como usuario "root" esto con el fin de utilizar el programa p-paralelo que llama a la función "ioperm()" que como sabemos requiere de los privilegios de este usuario para funcionar.
2. Ahora el programa encenderá y apagará alternativamente el bit 1 del puerto paralelo, esto puede ser verificado con la ayuda de la interfase y un multímetro colocado en las terminales de salida del bit 1.
3. Después se corre el programa servidor y el cliente en la misma máquina, de nuevo con privilegios de "root", desde el cliente se solicita el encendido del bit 1, y de nuevo se comprueba en la interfase que esto pase.
4. En el caso de la lectura, se coloca la interfase en el puerto y se unen las terminales de 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.

Con el fin de hacer esta prueba menos dependiente de la interfase, también se realizó la misma prueba pero conectando al puerto simplemente un LED en serie con una resistencia de 1Kohm, de esta manera la verificación es mas simple viendo prendiendo y apagando el LED.

6.2.2. Prueba 6 .- Interpretación de los comandos correctos e incorrectos

Con la intención de que esta prueba no dependa completamente de la conexión a la red, decidimos hacer uso del depurador gdb y escribir en tiempo de ejecución la línea de comandos en la variable destinada a esta tarea del servidor, con el fin de que localmente podamos manipularla y de esta manera no asociar los posibles errores a la conexión de red, en otras palabras, excluimos la conexión a la red de las posibles causas de error para este caso. La prueba puede realizarse de la siguiente manera:

1. Se ejecuta el "gdb" pasándole como argumento el binario del servidor, así: "$gdb servidor"

2. Se coloca un breakpoint en la línea anterior a la que llama al interprete de comandos.

3. Se corre el programa línea a línea hasta llegar al breakpoint, en este punto se ingresa en la variable "linea_de_comando" el comando con sus argumento, en este caso correctos (desc 1 lampara_techo).

4. Se continúa con la ejecución del programa, saltando la línea que lee desde la pantalla la línea de comando

5. Se verifica que el programa entre a la función de interpretación de comandos y que en este caso (correcto) mande llamar a la función que da servicio al comando.

6. Se repite el procedimiento solo que en el punto 3 se ingresa un comando incorrecto (des lampara3), para verificar que el interprete nos mande un error (comando no válido).

6.2.3. Prueba 7 .- Envío 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 probará en una red local y posteriormente en Internet, se usará en la otra máquina un cliente reducido desarrollado en Java que solo envía una cadena de caracteres y lo espera que regrese integro del servidor, los pasos son los siguientes:

1. Se corre el servidor como usuario "root" en una computadora perteneciente a una red local en este caso la 192.168.7.195

2. En otra computadora de la red, en este caso 192.168.7.198, se corre el programa cliente hecho en java.

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 llegó un mensaje, la IP de origen, el tamaño del mensaje y finalmente el mensaje.

5. En el cliente debe aparecer abajo de la línea donde se ingresó la cadena de prueba "probando probando", otra línea con el mismo mensaje exactamente, este último proveniente del servidor.

6. 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 realizar para diseñar las pruebas, estas funciones son, la validación de los comandos de la línea de comando, la comunicación por la red y el despliegue del estado de la interfase en la pantalla, así las pruebas quedan de la siguiente manera:

6.3.1. Prueba 9 .-Validación de comandos correctos y erróneos

Al igual que en la prueba 6 esta consiste en verificar al interprete de comandos, en este caso particular la idea es que si un comando es correcto y posee todos los argumentos necesarios, entonces es enviado al servidor y en caso de que no lo sea, se manda un error "comando no válido" y no se envía al servidor.

Los pasos a seguir en la prueba son exactamente iguales que en la número 6, por lo que no se mostrarán aquí.

6.3.2. Prueba 10 .- Envío de comandos por Internet

Esta prueba es complementaria a la prueba 7 hecha al servidor, aquí probaremos la transferencia de datos pero ahora con el programa cliente verdadero, esto es, el que finalmente se usará en el sistema. La prueba es así:

1. Corremos el servidor en una máquina con IP homologada.
2. Después desde otra computadora con acceso a Internet corremos el cliente y solicitamos un comando válido.
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.

Esta prueba solo se concreta a verificar la comunicación en un solo sentido, junto con la prueba 11 podemos saber si todo el ciclo completo de comunicación se da.

6.3.3. Prueba 11 .- Despliegue de los datos a partir de la contestación del servidor

Aquí veremos como el mensaje enviado por el servidor a manera de comprobación de que la petición se realizó con éxito, sirve en el cliente para actualizar

los datos mostrados en la pantalla, de esta manera se cierra el ciclo de comunicación que se presenta cuando un cliente pide un cambio de estado de la interfase, la prueba se lleva a cabo de la siguiente manera:

1. Se ejecuta el programa cliente y se toma nota de los datos que presentan el estado de la interfase.
2. Se realiza la prueba 10, una vez que se verifique que el servidor recibe el mensaje correctamente y realiza el envió de comprobación hacia el cliente se continúa con el siguiente punto.
3. Se revisa que en la bitácora aparezca el mensaje "comando ejecutado" y que los datos del estado de la interfase 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 su orden efectivamente fue realizada con éxito, se enviaron todos los comandos, "desc" que cambia la descripción de los bits, prende ya paga que hacer lo propio con los bits y refresca que actualiza los datos sin realizar cambios de estado.

6.3.4. Prueba 12 .- Guardar y recuperar los datos de descripción

Ahora probaremos la función de guardar y recuperar la descripción de los bits, esta función e llamada con la orden "guardar" seguida de "desc" y el nombre del archivo, una vez que el archivo es guardado puede ser recuperado explícitamente con el comando recupera o incluyendo en archivo en la línea que llama al programa cliente, la prueba entonces sería así:

1. Se corre el programa cliente y se cambia la descripción de varios bit, a través del comando "desc".
2. Se guarda la descripción de estos bits en un archivo en el directorio "/tmp".
3. Salimos del programa cliente.
4. Llamamos al programa de nuevo pero ahora incluyendo en la línea de comando el nombre del servidor y el archivo de descripciones.
5. Verificamos que la información de la descripción sea correcta y esté en los bits adecuados.
6. Ahora explícitamente llamamos al comando recuperar pasando como argumento el nombre del archivo de descripciones.

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

6.4.1. Prueba 13 .- general

En esta prueba revisaremos todo el sistema en su conjunto, la interfase conectada a la toma de corriente y a la computadora, los aparatos (lámpara y ventilador) y sensores (puerta y ventana) también dispuestos en su lugar. Los programas corriendo, el servidor en la misma computadora server.virtualio.com.mx pero ahora con la interfase conectada; 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 interfase y se 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 cierra la puerta y la ventana y se solicita al cliente que actualice los datos
4. Se cambian la descripción de los bits de salida por lámpara y ventilador
5. Se pide desde el cliente que la lámpara y el ventilados se enciendan
6. Se guardan los datos de descripción
7. Salimos del programa cliente
8. Ejecutamos de nuevo el programa pasando como argumento el archivo de descripción.
9. Verificamos que tengamos los mismo datos que en la sesión anterior.
10. Solicitamos apagar la lámpara y el ventilador y abrimos la puerta.
11. Se verifica que se actualice la pantalla incluyendo la apertura de la puerta.

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 14 .- Ejecución y compilación en varias plataformas

Lo que nos queda por probar es que tan portable y flexible son nuestros programas, como se dijo e el objetivo del trabajo, es muy importante que el sistema pueda ser usado en prácticamente todas las computadoras modernas compatibles con IBM, en un rango desde las 386 hasta las modernas Pentium IV, 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 de nuestro sistema en 3 distintas computadoras con características muy diversas.

La prueba consiste primero en ejecutar los programas en estas computadoras y aplicar la prueba 13, pero también probaremos si es posible compilar los programas con los compiladores y librerías que estas computadoras tengan. Aquí es donde aparece el sistema operativo, sabemos que nuestro sistema funciona en el sistema operativo Linux, pero este sistema a sufrido modificaciones desde su aparición hasta la fecha en lo que a compiladores se refiere, no obstante, el código de nuestros programas a sido cuidadosamente tratado para no incluir librerías o funciones que no están dentro del estándar ANSI, así que en teoría no debería haber ningún problema para compilar nuestros programas incluso en versiones muy viejas de Linux.

Por eso aquí se presenta una tabla con las 3 computadoras que se proponen para la prueba y las distribuciones de Linux.. Es claro que se busca coincidir la edad de las máquinas con la del sistema operativo. Se mostrará en la tabla el resultado obtenido en cuanto a la ejecución como a la compilación:

No.	Hardware	Distribución	Ejecución	Compilación
1	Acermate 486DX 33 MHz, 8 Mb RAM, 400 Mb D.D.	RedHat 4.0	Servidor: Sí, Cliente: Sí	Servidor: Sí, Cliente: No
2	Intel Celeron 300 MHZ, 32 Mb RAM, 1.2 Gb D.D.	Debian 3.0 Release 1 "Woody"	Servidor: Sí, Cliente: Sí	Servidor: Sí, Cliente: Sí
3	Compaq Athlon 1800, 256 Mb RAM, 40 Gb D.D.	RedHat 9.0 y SuSE LiveEval 8.1	Servidor: Sí, Cliente: Sí	Servidor: Sí, Cliente: Sí

Cuadro 6.1: Distribuciones de Linux usadas para las pruebas de compilación y ejecución.

La compilación del programa cliente no pudo realizarse en el equipo número 1, debido a que este programa usa la librería "ncurses" que aunque existe en

RedHat 4.0, el sistema operativo de esta máquina, no es la misma librería que en versiones posteriores. Existen 2 posibles soluciones a este problema que podría presentarse con seguridad en versiones antiguas de Linux, la primera es modificar en el código la ruta donde se encuentra la librería "ncurses" en RedHat 4.0 en la sentencia #include <ncurses.h>, esta acción en realidad podría hacerse automática con unas cuantas instrucciones al precompilador de "C" o desde el archivo "Makefile", sin embargo existe otra opción aunque no es tan real. Recordemos que el programa cliente que se entrega en este trabajo no es necesariamente el único que puede trabajar con este sistema, existe la posibilidad de que se pueda hacer un programa cliente incluso con mejores características en otro lenguaje como Java, así que existiendo una máquina virtual para cualquiera de las distribuciones de Linux usadas en la prueba, podríamos emplear un cliente Java en lugar del programado en "C".

Lo importante aquí es notar que los programas "corren" en las 3 plataformas esto quiere decir que se puede compilar en una máquina mas nueva y después llevar los binarios a un equipo antiguo; también queda demostrado que efectivamente los equipos obsoletos pueden volver a tener uso con este sistema, lo cual es muy importante dada la cantidad tan grande de equipos de estas características que continuamente se están desechando.

Capítulo 7

Evaluación de resultados

Con la idea de hacer un resumen de como las pruebas nos fueron dando indicios de que el objetivo se esta cumpliendo o no, haremos una evaluación mas 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 cuales 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 excito, podemos concluir que el objetivo se está alcanzando.

De nuevo dividiremos como en capítulos anteriores este proceso de evaluación en módulos, para que así podamos considerar hasta que punto cada uno de estos esta 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 está cubierto.

7.1. Evaluación de la interfase 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 interfase 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 interfase deberá cubrir, este es: Con el fin de corroborar la manera en la que una computadora personal puede manipular aparatos de uso doméstico con

el solo empleo de su 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 más corriente y voltaje que la dada por la PC y que además puede adquirir señal del exterior y pasarlas a la PC con seguridad con el fin de que sean sensadas.

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

Podemos decir entonces que la interfase maneja los aparatos domésticos, que puede adquirir señales digitales originadas en sensores de posición, que no son más que interruptores de pulso (push button), 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 interfase cubre un 80% del objetivo impuesto pero la idea de que es lo suficiente como para revisar los conceptos generales del sistema sigue firme.

7.1.1. Prueba 2

Los resultados de las pruebas fueron exitosos y en ambos casos (bit 1 y bit 2) los circuitos de salida funcionaron muy bien, se pudo prender y apagar la lámpara de 20W, sin ningún problema, además el circuito no la demanda al puerto más que 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 en esta prueba solo se probó la capacidad de interrupción de la interfase y no la potencia soportada, eso se realizará en una prueba posterior.

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 4.82 Vcc y 1.1 mA que efectivamente están dentro del margen adecuado para la operación del puerto paralelo.

7.1.3. Prueba 4

La prueba se realizó usando 3 aparatos de uso común que en suma consumían un total de 430 watts, estos eran un televisor de 21" con consumo aproximado de 80 watts, una lámpara de techo con un consumo de 300 watts y un monitor de computadora de 17" y 50 watts de consumo, en realidad los únicos aparatos domésticos que alcanzan los 800 watts son aquellos que generan calor y para la prueba no se tenia 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 3 aparatos. Con el primer encendido de 5 minutos los componentes se comportaron bien, ninguno sufrió daños y solo la resistencia de 100 ohms se calentó pero justamente esa resistencia es la única que es de 1 watt de potencia ya que se sabia que por ahí iba a circular una cantidad importante de corriente; el relevador funcionó perfectamente.

En la segunda prueba, la de 8 horas, las cosas no cambiaron, de nuevo el único componente que eleva su temperatura es la resistencia de 100 ohms, todos los equipos funcionaron durante las 8 horas sin problema y la interfase se mantuvo intacta después de esta prueba.

En resumen podemos decir que el circuito está 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 en este caso 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 a partir de las pruebas que el programa puede realizar las tareas fundamentales esperadas de el, puede manejar el puerto paralelo, interpretar comandos y enviar y recibir correctamente datos de la red. Particularmente en el aspecto de funcionalidad este programa no es el más amigable ni fácil de interpretar como se esperaría, la idea sería mejor ocultar totalmente de la vista del usuario los mensajes de respuesta del programa. Esto es, sería mejor que el servidor funcionara como un "demonio" que es un tipo especial de programa en los sistemas UNIX que corre en background y que el sistema puede echar a andar desde el arranque sin la intervención del usuario; en este caso los mensajes que el servidor genera son enviados a un archivo bitácora colocado en "/var/log" un directorio especial destinado específicamente para esta tarea.

Podemos ver que lo anterior no es un problema, es solo un detalle de funcionalidad que de ser implementado ofrecería una manera mas estándar de uso del servidor comparada con de lo otros servidores de sistemas UNIX, las tareas encargadas al servidor se están cumpliendo y de hecho al definir un conjunto de

instrucciones que manipulan la interfase ofrece la oportunidad de ver al servidor como una API (Interfase de programación de aplicaciones) para el desarrollo de aplicaciones domóticas. Veremos en cuanto evaluemos al cliente que las posibilidades de este sistema se multiplicarán en la medida de que se realice un trabajo más arduo en el cliente, con el fin de agregar nuevas funciones a este.

7.2.1. Prueba 5

La prueba demostró que las funciones antes mencionadas que forman parte de la API del lenguaje "C" funcionan satisfactoriamente, nos dejan controlar el puerto de manera sencilla y podemos controlar cada bit individualmente. En el caso de la lectura el trabajo delicado consiste en enmascarar los bits para seleccionar el deseado, pero fuera de eso no existe mucha dificultad.

Tanto el programa p-paralelo como el servidor funcionaron muy bien en este aspecto corriéndolos como usuario "root", en caso de que sean ejecutados como usuario normal sin privilegios, los programas abortan inmediatamente e indican un error "segmentation failure" o "violación de segmento".

7.2.2. Prueba 6

La prueba demostró que el interprete funciona muy bien, se probaron todos los comandos disponibles con los argumentos correctos y en ningún caso se presentaron errores, también se probaron todos los comandos pero con argumento incorrectos, de nuevo los resultados fueron exitosos, en todos los casos el interprete devolvió "comando no válido" y en el último caso que es comandos y argumentos equivocados, el resultado fue "comando no válido", en resumen, esta parte ya es funcional.

7.2.3. Prueba 7

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 informado la procedencia el tamaño correcto del mensaje y el mensaje. El cliente también registro que efectivamente el servidor estaba regresando el mensaje intacto cuando este contenía un comando válido, de no ser así devolvía un error "comando no válido".

En la prueba de Internet el servidor se corrió e un computadora propiedad del la compañía "virtualio.com.mx" y que normalmente sirve páginas WEB, tiene dirección 200.67.175.234 y nombre server.virtualio.com.mx cabe aclara que esta computadora esta conectada a Internet con una enlace de 512 Kbps lo cual no sería el común de los enlaces hogareños, Es una máquina AMD Athlon a 1.2 GHz 512 Mb de RAM y RedHat Linux 7.3.

La máquina cliente es una PC común Pentium III a 1 GHz con 128 Mb de RAM y RedHat 9.0, conectada por modem de 56 Kbps a un ISP y con una IP dinámica asignada por este último.

En este caso y como era de esperarse ocurrió un retraso en la comunicación, los mensajes enviados por el cliente tardaba más en llegar de regreso, de 10 primeros intentos 8 se realizaron con éxito, pero de los 10 siguientes, todos se recibieron, del lado del servidor todos los mensajes que llegaron fueron bien interpretados, es claro ver como el tráfico casi aleatorio que los paquetes encuentran en Internet afectan significativamente el comportamiento del sistema, esto ya se esperaba y por eso en ninguna ocasión se pensó en un sistema en tiempo real ya que los tiempos de respuesta del sistema no están bajo nuestro control.

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 del los comandos escritos por el usuario, la interpretación de estos, su posterior envío al servidor y finalmente, la recolección de la contestación del servidor para actualizar los datos en pantalla. No obstante, en este programa solo se consideran las funciones elementales ya que como se a 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, el cliente es el programa mínimo que se requiere para demostrar que el sistema funciona, pero no implementa de manera extensiva todas las posibles capacidades del programa cliente. Por otra parte también es importante aclarar que este programa está codificado en "C" que aunque se a mantenido en un 90% compatible con ANSI, es probable que su portabilidad a otros sistemas operativos diferentes a Linux, no sea directa, esto significa que posiblemente habrá que cambiar algunas librerías de nombre, las pruebas no incluyeron pruebas de compilación y ejecución en plataformas con Windows u otros sistemas, así que no conocemos con precisión la portabilidad de este programa en particular. Mas adelante en uno de los apéndices se ofrece una alternativa a este problema de portabilidad a través de un nuevo programa hecho en "Java" el cual deberá de correr en cualquier plataforma que tenga una máquina virtual.

En resumen podemos decir que el programa cliente cumple las funciones para las que fue hecho, sin embargo solo se cubren aproximadamente el 50% de las tareas que este tipo de programas podría tener para considerar que se tiene un sistema domótico completo, el resto de las opciones y su función dentro de un esquema más real de los sistemas domóticos se planteará en las conclusiones.

Quizá el punto más débil de este programa es la facilidad de uso y funcionalidad, en este punto el programa al estar programado en modo texto requiere que el usuario ingrese los comandos de manera escrita, esto se sabe, no es muy amigable

y menos aun cuando en los últimos años el uso de las interfases gráficas ha ido en aumento haciendo al usuario cada vez más dependiente de los menús, botones y demás elementos gráficos. Por otro lado el despliegue de los resultados aunque claro, no es lo suficientemente explicito o ilustrativo como podría ser. Aquí también podríamos auxiliarnos con las herramientas gráficas para hacer mucho más intuitiva la descripción de lo que el sistema está controlando. De nuevo todas estas características son cosméticas y aunque para el usuario puedan ser muy útiles, salen un poco del objetivo fundamental de este trabajo, sin embargo, están consideradas en las propuestas planteadas más adelante.

7.3.1. Prueba 9

Los resultados fueron buenos, se reconoció correctamente todos los comandos disponibles cuando estaban acompañados de los argumentos necesarios, y en el caso de comandos incorrectos y falta de argumentos, siempre se respondió con el error adecuado.

7.3.2. Prueba 10

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

7.3.3. Prueba 11

Todos funcionaron en caso de que el servidor no este prendido o que el paquete se pierda, el cliente marca un error después de 30 segundos de no recibir noticia, este procedimiento se probó desconectando al cliente de la red, sin embargo, no reportó el problema y se quedo esperando la contestación por mucho tiempo, así que se tendrá que verificar en el código por que se presenta este problema.

7.3.4. Prueba 12

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 de manera correcta, si por algún error se ingresa un nombre de archivo equivocado, el programa nos devuelve un error de lectura que por supuesto es recuperable, esto quiere decir que el usuario puede volver a ingresar el nombre correctamente sin necesidad de salir y volver a entrar al programa.

7.4. Evaluación general

Podemos decir que el sistema en su conjunto se comporta de manera aceptable considerando solo las pruebas que se definieron en el capítulo anterior. Sin embargo, no podemos dejar a un lado la posibilidad de errores u omisiones tanto en el sistema en sí, 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 presenten accidentalmente situaciones que en el diseño original nunca se pensaron.

Con esta última parte podemos decir que el trabajo esta concluido y que el prototipo demuestra la viabilidad de la tesis propuesta en el objetivo, existen muchas ideas que quedarán pendientes y otras 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 13

Todo funcionó muy bien, y el sistema en general realizó todas las tareas sin contratiempo, solo falta comprobar en diferentes plataformas si esto se cumple y de eso se trata la prueba 14.

7.4.2. Prueba 14

Podemos observar en la tabla que prácticamente en los 3 sistemas probados nuestros programas funcionan, en el 100 % de ellos los programas pueden ejecutarse y solo en uno de ellos, el más antiguo, el programa cliente no pudo compilarse.

Capítulo 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 un ambiente real de uso. Se utilizó efectivamente el mínimo de hardware adicional lo que se traduce en un bajo costo y una alta portabilidad a diferentes computadoras con características distintas, 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 cómputo no puede estar desligada de la facilidad de uso por parte del usuario, en este aspecto, el programa que nos interesaría que fuera muy fácil de usar sería el cliente, la versión presentada en este trabajo muestra una interfase en modo texto, que es operada a través de comandos, para algunas personas medianamente instruidas en el uso de computadoras. Esta característica ofrece la posibilidad de ampliar la funcionalidad del programa a través de lenguajes de programación interpretados que puedan generar "scripts" de comandos para después ser direccionados al cliente y con los cuales se podrían automatizar las tareas del sistema domótico. Para otras personas más acostumbradas a las interfases gráficas y a los modernos sistemas operativos operables a través de botones y ventanas, la línea de comandos del programa cliente podría parecer muy rudimentaria y hasta complicada, pero gracias a la manera en la que el servidor opera, es muy sencillo advertir que podrían hacerse una gran variedad de programas cliente gráficos, en cualquier lenguaje de programación que tenga incluido dentro de su API, funciones para el uso de sockets y conectividad con redes, ya en uno de los

apéndices mostramos dos propuestas una en JAVA y otra en C con GTK+. Queda entonces abierta la posibilidad de que alguien más pueda tomar este trabajo y ampliarlo desarrollando un programa cliente más amigable y funcional e inclusive con más funciones que las imaginadas originalmente.

Un sistema domótico completo en la actualidad es un conjunto muy complejo de elementos de comunicación, cómputo, electrónica y control; las empresas que ofrecen este tipo de productos invierten grandes cantidades de dinero al desarrollo de nuevas ideas en este rubro y aun así, la penetración de estos productos no es masiva y no lo será en un buen tiempo en los países en vías de desarrollo debido principalmente a los altos costos. 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 antes mencionadas, y lo mas importante, si mantenemos como prioridad el uso de software libre y el empleo de computadoras comunes y hasta descontinuadas, 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.

Aunque los resultados obtenidos en su conjunto son satisfactorios, hay algunos puntos que podrían mejorarse, pero esto representaría una inversión de tiempo y dinero mayor, sin repercutir sustancialmente en el cumplimiento del objetivo. Cosas como la adquisición de señales analógicas y la posibilidad de tener mas entradas y salidas disponibles, son inmediatamente vistas como deficiencias actuales en el prototipo, pero esto se salva considerando que en el diseño conceptual son tomadas en cuenta.

Trabajos adicionales pueden enfocarse a obtener un producto comercial, para esto se requiere el análisis económico del mercado, la comercialización e imagen del producto, protección intelectual de los componentes, normatividad técnica para este tipo de dispositivos, seguridad del consumidor, etc.

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, existen 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 parezcan menores no son siempre despreciables. Aquí es donde podemos resumir el trabajo de un Ingeniero, no se puede considerar a un sistema funcional solamente por el planteamiento teórico, necesariamente nos tenemos que enfrentar con la experimentación y la implementación práctica para eliminar estas sutiles diferencias entre la teoría y el empirismo.

Parte I

Código fuente

Servidor

```
/* D.R.(C) Osvaldo N. R. Argüello UNAM - FI / 2002 */
/* Tesis de Licenciatura en Ingeniería Eléctrica - Electrónica */
/* ------------------------------------------------------------------------ */
/* Programa : Servidor domótico y controlador del puerto paralelo */
/* Versión : 0.8.5 */
/* Fecha : 23 de septiembre de 2003 */
/* S.O. : GNU/Linux, Kernel 2.4.7-10, glibc 2.2.4, glib 1.2.10 */
/* Notas : Se compila así: $gcc -o servidor servidor.c */
/* Se corre simplemente con: $./servidor */
/* Notas de la versión */
/* ------------------------------------------------------------------------ */
/* */
/* 18/09/03 0.7.0 - Se integra el comando "estado" que permite */
/* solicitar desde el cliente, el estado de los bits */
/* de salida del puerto. */
/* 18/09/03 0.7.1 - Se agrego la libreria "math.h" por el uso de "pow" */
/* en la nueva función "EstadoLPT". */
/* 18/09/03 0.7.2 - Se elimina bug de inicialización del estado del */
/* puerto, se asigna Estado(Datos) a mascara, con eso */
/* se preserva el estado anterior de los bits. */
/* 23/09/03 0.8.0 - Envio de estado junto con el comando solicitado. */
/* 23/09/03 0.8.5 - Se envia al cliente el estado de los bits de salida*/
/* como de entrada */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <sys/io.h>
#include <math.h>
#define PUERTO 6543
#define MAXBUFLEN 150
int sockfd,newfd;
int retorno;
int paralelo=128;
int num_tokens=0,i=0,mascara;
int DirecBase=0x378; /* LPT1 */
int Datos,Estado,Control;
int addr_len, numbytes;
int status[8]={0,0,0,0,0,0,0,0};
int statusE[4]={0,0,0,0};
```

```
struct sockaddr_in dir_local; /* direccion IP y numero de puerto local */
struct sockaddr_in dir_remota; /* direccion IP y numero de puerto del cliente */
static char buf[MAXBUFLEN]; /* Buffer de recepcion */
static char *segmento[40];
char *version="0.8.5";
char bufferTemp[MAXBUFLEN];
char EstadoC[3],EstadoEntradaC[3];
int separa(char linea[80])
{
if(linea!=NULL)
{
i=0;
segmento[i]=strtok(linea," ");
i++;
while ((segmento[i]=strtok(NULL," ")) != NULL)
i++;
}
/* i=0; */
return(i);
}
/* Funcion para leer el estado de un bit del puerto paralelo */
int LeerLPTx(int mascara_local, int direccion)
{
int bit;
ioperm(DirecBase,3,1);
bit=inb(direccion) & mascara_local; /* enmascara el bit estudiado */
if (bit==mascara_local)
return(1);
else
return(0);
ioperm(DirecBase,3,0);
}
int EscribeLPTx(int mascara, int direccion)
{
ioperm(DirecBase,3,1);
outb(mascara,direccion);
ioperm(DirecBase,3,0);
}
int EstadoLPT(int direccion)
{
int SDatos=0;
int mascaraL=0;
int cont; /* Contador para el "for" local */
for (cont=0;cont<8;cont++)
{
mascaraL=pow(2,cont);
```

```
status[cont]=LeerLPTx(mascaraL,Datos);
if (status[cont]==1)
{
SDatos=SDatos+mascaraL;
}
/* printf("%i ",status[cont]); */
}
printf("\n");
return(SDatos);
}
int EstadoEntrada(int direccion)
{
int SDatosEntrada=0;
int mascaraE=0;
int contE;
for (contE=4;contE<8;contE++)
{
mascaraE=pow(2,contE);
statusE[contE]=LeerLPTx(mascaraE,Estado);
if (statusE[contE]==1)
{
SDatosEntrada=SDatosEntrada+mascaraE;
}
/* printf("%i ",statusE[contE]); */
}
printf("\n");
return(SDatosEntrada);
}
main()
{
Datos=DirecBase;
Estado=DirecBase+1;
Control=DirecBase+2;
/* se crea el socket */
printf("[1;37m");
printf("Tesis de Licenciatura en Ing. Eléctrica - Electrónica\nUNAM -
");
printf("[1;31m");
printf("FI");
printf("[1;37m");
printf(", Osvaldo N. R. Argüello 2002, versión:%s\n",version);
printf("-----------------------------------------------------\n");
mascara=EstadoLPT(Datos);
printf("Estado actual del puerto:%i\n",mascara);
printf("[0;37m");
while(1)
```

```
{
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
{
perror("Error al crear el socket");
exit(1);
}
/* Se establece la estructura dir_local para luego llamar a bind() */
dir_local.sin_family = AF_INET;
dir_local.sin_port = htons(PUERTO);
dir_local.sin_addr.s_addr = INADDR_ANY;
bzero(&(dir_local.sin_zero), 8);
/* Se le da un nombre al socket */
if (bind(sockfd, (struct sockaddr *)&dir_local, sizeof(struct sockaddr))
== -1)
{
perror("Error en bind");
exit(1);
}
/* Se reciben los datos */
addr_len = sizeof(struct sockaddr);
printf("[0;32m");
printf("\nEsperando mensaje del cliente");
printf("[0;37m");
printf("...\n");
if ((numbytes=recvfrom(sockfd, buf, MAXBUFLEN, 0, (struct sockaddr *)
&dir_remota, &addr_len)) == -1)
{
perror("Error al recibir mensaje [recvfrom]");
exit(1);
}
/* Se visualiza lo recibido */
printf("-> paquete proveniente de: ");
printf("[0;33m");
printf("%s\n",inet_ntoa(dir_remota.sin_addr));
printf("[0;37m");
printf("-> longitud del paquete en bytes: ");
printf("[0;33m");
printf("%d\n",numbytes);
printf("[0;37m");
buf[numbytes] = '\0';
printf("-> el paquete contiene: ");
printf("[0;33m");
printf("%s\n",buf);
printf("[0;37m");
/* El interprete de comandos */
num_tokens=separa(buf);
```

```
printf("No.de tokens:%i\n",num_tokens);
printf("Comando:%s\n",segmento[0]);
if(!strcmp(segmento[0],"prende"))
{
mascara|=atoi(segmento[1]);
EscribeLPTx(mascara,Datos);
/* strcpy(estado,"Se encendio el bit ");
strcat(estado,segmento[1]); */
}
else if(!strcmp(segmento[0],"apaga"))
{
mascara^=atoi(segmento[1]);
EscribeLPTx(mascara,Datos);
/* strcpy(estado,"Se apagó el bit ");
strcat(estado,segmento[1]); */
}
else if(!strcmp(segmento[0],"refresca"))
{
/* mascara^=atoi(segmento[1]); */
/* EscribeLPTx(mascara,Datos); */
/* strcpy(estado,"Se apagó el bit ");
strcat(estado,segmento[1]); */
}
printf("Nuevo estado:%i\n",EstadoLPT(Datos));
printf("Estado entrada:%i\n",EstadoEntrada(Estado));
printf("[0;37m");
/* Conversion de entero a cadena */
strcpy(bufferTemp,buf);
gcvt(EstadoLPT(Datos),3,EstadoC);
gcvt(EstadoEntrada(Estado),3,EstadoEntradaC);
strcat(bufferTemp," ");
strcat(bufferTemp,EstadoC);
strcat(bufferTemp," ");
strcat(bufferTemp,EstadoEntradaC);
printf("Caracter:%s",bufferTemp);
/* Envio del estado del puerto al cliente */
if ((retorno=sendto(sockfd, bufferTemp, strlen(bufferTemp), 0, (struct
sockaddr *) &dir_remota, addr_len)) == -1)
{
perror("Error al enviar retorno ");
exit(1);
}
/* Cerramos el socket y devolvemos recursos al sistema */
close(sockfd);
}
}
```

Cliente

```
/* D.R.(C) GPL Osvaldo N. R. Argüello UNAM - FI / 2002 */
/* Proyecto de Tesis de Licenciatura en Ingeniería Eléctrica - Electrónica */
/* Programa : Prototipo de cliente y monitor local domótico bajo terminal */
/* de texto, usando ncurses. */
/* Versión : 0.7.8 */
/* Fecha : 23 de septiembre de 2003 */
/* S.O. : GNU/Linux, Kernel 2.4.18-14, glibc 2.2.93-5, glib 1.2.10-8 */
/* Notas : Se compila así: $cc -lncurses -o cliente-tty cliente-tty.c */
/* Se corre simplemente con: $./cliente-tty puerto_par IP */
#include <curses.h>
/* #include <signal.h> */ /* Esta libreria es de prueba, quiza se elimine */
#include <stdlib.h> /* atoi() */
/* #include <sys/io.h> */ /* ioperm() */
#include <string.h> /* strncpy(), strsep() */
#include <stdio.h> /* fprintf() fscanf() */
#include <math.h> /* pow() */
/* Librerías incluidas para el cliente de red */
#include <errno.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/wait.h>
#define PUERTO 6543 /* el puerto donde se enviarán los datos */
#define BUFFER 256
#define IP "192.168.7.198" /* copelia.gam2.edu.mx */
#define version "0.7.8"
#define LONG_DESCRIPCION 28
#define LONG_BITACORA 80
#define MAXDATASIZE 100
int status[8];
int EstadoLPT;
int k;
int salida[8];
int entrada[4];
int DirecBase=0x378; /* LPT1 */
int Ebit; /* Esta de prueba */
int Estado,Datos,Control; /* Registros del puerto */
int i,j,num_tokens=0,indice_bitacora=0;
FILE *arch;
char bit; /* esta de prueba */
char descripcion[12][80];
char estado[80]="Bienvenido a gDomo";
```

```
char bitacora[5][80]; /* Buffer para escribir en la seccion bitacora */
static int mascara=0;
static char linea_de_comando[80],linea_al_servidor[80];
static char *segmento[40];
char buf[MAXDATASIZE]; /* Buffer donde se reciben los datos */
int separa(char linea[80])
{
if(linea!=NULL)
{
i=0;
segmento[i]=strtok(linea," ");
i++;
while ((segmento[i]=strtok(NULL," ")) != NULL)
i++;
}
i=0;
return(i);
}
int envia(char linea[80])
{
int sockfd;
struct sockaddr_in dir_local,dir_remota;
struct hostent *he;
int addr_len, numbytes,num_datos_recibidos,j,temp,temp2;
/* convertimos el hostname a su direccion IP */
if ((he=gethostbyname(IP)) == NULL)
{
herror("gethostbyname");
exit(1);
}
/* Creamos el socket */
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
{
strcpy(estado,"Error al crear el socket");
exit(1);
}
dir_local.sin_family = AF_INET; /* servidor orden byte */
dir_local.sin_port = htons(PUERTO); /* red orden byte */
dir_local.sin_addr = *((struct in_addr *)he->h_addr);
bzero(&(dir_local.sin_zero), 8);
if ((numbytes=sendto(sockfd, linea, strlen(linea), 0, (struct sockaddr
*)&dir_local, sizeof(struct sockaddr))) == -1)
{
strcpy(estado,"Error al enviar.");
exit(1);
}
```

```
/* printf("Se enviaron%d bytes hacia%s\n",numbytes,inet_ntoa(dir_local.sin_addr)); */
if ((numbytes=recvfrom(sockfd, buf, MAXDATASIZE, 0, (struct sockaddr *)
&dir_remota, &addr_len)) == -1)
{
strcpy(estado,"Error al recibir el retorno");
exit(1);
}
close(sockfd);
strcpy(estado,buf);
num_datos_recibidos=separa(buf);
temp=atoi(segmento[1]);
for(j=0;j<8;j++)
{
salida[j]=temp%2;
temp=temp/2;
}
temp2=atoi(segmento[2]);
temp2=(((temp2/2)/2)/2)/2;
for(j=0;j<4;j++)
{
entrada[j]=temp2%2;
temp2=temp2/2;
}
return 0;
}
void entra_descripcion(int indice, char contenido[80])
{
if (strlen(contenido)<=LONG_DESCRIPCION)
{
strcpy(descripcion[indice],contenido);
}
else
{
strncpy(descripcion[indice],contenido,LONG_DESCRIPCION);
}
}
int entra_bitacora(char contenido[80])
{
if (indice_bitacora<4)
indice_bitacora++;
else
indice_bitacora=0;
if (strlen(contenido)<=LONG_BITACORA)
{
strcpy(bitacora[indice_bitacora],contenido);
}
```

```
else
{
strncpy(bitacora[indice_bitacora],contenido,LONG_BITACORA);
}
return(indice_bitacora);
}
/* Escribe los titulos */
void titulos()
{
attron(A_BOLD);
attron(COLOR_PAIR(COLOR_WHITE));
clear();
move(1,2);
printw("UNAM - ");
attron(COLOR_PAIR(COLOR_RED));
move(1,9);
printw("FI,");
attron(COLOR_PAIR(COLOR_WHITE));
move(1,13);
printw("Prototipo de cliente y monitor local domótico, v.%s",version);;
move(2,2);
printw("(C) GPL, Osvaldo N. Rodríguez Argüello, 2002");
attroff(A_BOLD);
attron(COLOR_PAIR(COLOR_WHITE));
move(3,2);
printw("_____________________________________________________________________
____________");
move(4,2);
printw("Dirección del puerto: ");
move(4,24);
attron(COLOR_PAIR(COLOR_GREEN));
printw("0x%x, ",DirecBase);
move(4,31);
attron(COLOR_PAIR(COLOR_WHITE));
printw("Servidor: ");
move(4,41);
attron(COLOR_PAIR(COLOR_GREEN));
printw("%s",IP);
move(6,12);
attron(COLOR_PAIR(COLOR_YELLOW));
printw("Bits de salida");
move(6,52);
printw("Bits de entrada");
attron(COLOR_PAIR(COLOR_WHITE));
move(8,2);
printw("Bit 1: ");
```

```
move(9,2);
printw("Bit 2: ");
move(10,2);
printw("Bit 3: ");
move(11,2);
printw("Bit 4: ");
move(12,2);
printw("Bit 5: ");
move(13,2);
printw("Bit 6: ");
move(14,2);
printw("Bit 7: ");
move(15,2);
printw("Bit 8: ");
move(8,42);
printw("Bit 9: ");
move(9,42);
printw("Bit 10: ");
move(10,42);
printw("Bit 11: ");
move(11,42);
printw("Bit 12: ");
}
/* Refresca los datos de los bits del puerto */
void refresca_bits()
{
attron(COLOR_PAIR(COLOR_RED));
move(8,9);
printw("%i,%s",salida[0],descripcion[0]);
move(9,9);
printw("%i,%s",salida[1],descripcion[1]);
move(10,9);
printw("%i,%s",salida[2],descripcion[2]);
move(11,9);
printw("%i,%s",salida[3],descripcion[3]);
move(12,9);
printw("%i,%s",salida[4],descripcion[4]);
move(13,9);
printw("%i,%s",salida[5],descripcion[5]);
move(14,9);
printw("%i,%s",salida[6],descripcion[6]);
move(15,9);
printw("%i,%s",salida[7],descripcion[7]);
move(8,50);
printw("%i,%s",entrada[0],descripcion[8]);
move(9,50);
```

```
printw("%i,%s",entrada[1],descripcion[9]);
move(10,50);
printw("%i,%s",entrada[2],descripcion[10]);
move(11,50);
printw("%i,%s",entrada[3],descripcion[11]);
attron(COLOR_PAIR(COLOR_WHITE));
}
/* Bitacora */
void escribe_en_bitacora()
{
/* ++++++++++++++++++++ Bitacora de red ++++++++++++++++++++++++++++ */
move(16,2);
printw("_____________________________________________________________________
____________");
move(17,2);
entra_bitacora(" entrada: (01/12/02 12:24), <- 192.168.7.193, Enciende
bit 1");
printw("%s",bitacora[0]);
move(18,2);
attron(A_BOLD);
attron(COLOR_PAIR(COLOR_MAGENTA));
printw("-> salida: (01/12/02 12:35), 192.168.7.198 ->, Refresca 192.168.7.193");
attroff(A_BOLD);
attron(COLOR_PAIR(COLOR_WHITE));
move(19,2);
printw(" salida: (01/12/02 12:35), 192.168.7.198 ->, Refresca 192.168.7.193");
move(20,2);
printw(" salida: (01/12/02 12:35), 192.168.7.198 ->, Refresca 192.168.7.193");
}
void interprete_de_comandos()
{
/* ++++++++++++++++++++++++ Lista de comandos +++++++++++++++++++++ */
if (!strcmp(segmento[0],"desc"))
{
entra_descripcion(atoi(segmento[1])-1,segmento[2]);
}
else if(!strcmp(segmento[0],"prende"))
{
envia(linea_al_servidor);
mascara|=atoi(segmento[1]);
/* CambiaLPTx(mascara,Datos); */
strcpy(estado,"Se encendio el bit ");
strcat(estado,segmento[1]);
}
else if(!strcmp(segmento[0],"apaga"))
{
```

```
envia(linea_al_servidor);
mascara^=atoi(segmento[1]);
/* CambiaLPTx(mascara,Datos); */
strcpy(estado,"Se apagó el bit ");
strcat(estado,segmento[1]);
}
else if(!strcmp(segmento[0],"ayuda"))
{
strcpy(estado,"Comandos: prende, apaga, desc, guarda, recupera, salir,
refresca, ayuda");
}
else if(!strcmp(segmento[0],"refresca"))
{
strcpy(estado,"Pantalla actualizada");
}
else if(!strcmp(segmento[0],"guarda"))
{
if(!strcmp(segmento[1],"desc"))
{
arch=fopen(segmento[2],"w");
/* Bits de salida */
fprintf(arch,"%s*\n",descripcion[0]);
fprintf(arch,"%s*\n",descripcion[1]);
fprintf(arch,"%s*\n",descripcion[2]);
fprintf(arch,"%s*\n",descripcion[3]);
fprintf(arch,"%s*\n",descripcion[4]);
fprintf(arch,"%s*\n",descripcion[5]);
fprintf(arch,"%s*\n",descripcion[6]);
fprintf(arch,"%s*\n",descripcion[7]);
/* Bits de entrada */
fprintf(arch,"%s*\n",descripcion[8]);
fprintf(arch,"%s*\n",descripcion[9]);
fprintf(arch,"%s*\n",descripcion[10]);
fprintf(arch,"%s*\n",descripcion[11]);
fclose(arch);
strcpy(estado,"Descripcion guardada");
}
else if(!strcmp(segmento[1],"estado"))
{
arch=fopen(segmento[2],"w");
/* Bits de salida */
fprintf(arch,"%i\n",salida[0]);
fprintf(arch,"%i\n",salida[1]);
fprintf(arch,"%i\n",salida[2]);
fprintf(arch,"%i\n",salida[3]);
fprintf(arch,"%i\n",salida[4]);
```

```
fprintf(arch,"%i\n",salida[5]);
fprintf(arch,"%i\n",salida[6]);
fprintf(arch,"%i\n",salida[7]);
/* Bits de entrada */
fprintf(arch,"%i\n",entrada[0]);
fprintf(arch,"%i\n",entrada[1]);
fprintf(arch,"%i\n",entrada[2]);
fprintf(arch,"%i\n",entrada[3]);
fclose(arch);
strcpy(estado,"Estado de los bits guardado");
}
}
else if(!strcmp(segmento[0],"recupera"))
{
if(!strcmp(segmento[1],"desc"))
{
arch=fopen(segmento[2],"r");
fscanf(arch,"%s",descripcion[0]);
fscanf(arch,"%s",descripcion[1]);
fscanf(arch,"%s",descripcion[2]);
fscanf(arch,"%s",descripcion[3]);
fscanf(arch,"%s",descripcion[4]);
fscanf(arch,"%s",descripcion[5]);
fscanf(arch,"%s",descripcion[6]);
fscanf(arch,"%s",descripcion[7]);
fscanf(arch,"%s",descripcion[8]);
fscanf(arch,"%s",descripcion[9]);
fscanf(arch,"%s",descripcion[10]);
fscanf(arch,"%s",descripcion[11]);
fclose(arch);
strcpy(estado,"Descripcion recuperada");
}
if(!strcmp(segmento[1],"estado"))
{
arch=fopen(segmento[2],"r");
fscanf(arch,"%s",bit);
/* Ebit=(int) atoi(bit); */
strcpy(estado,"Se selecciono recuperar estado");
/* CambiaLPTx(atoi(bit1) | status_puerto,Datos); */
/* CambiaLPTx(Ebit,Datos); */
fclose(arch);
}
}
else
{
strcpy(estado,"Comando no reconocido");
```

```
}
/* Termina código en curses */
}
/* ------------------------- PROGRAMA PRINCIPAL ------------------------- */
int main(int argc, char **argv)
{
if (argc==2)
{
DirecBase=(int) atoi(argv[1]);
}
envia("refresca"); /* Solicita el estado del puerto paralelo al servidor */
initscr();
if (LINES>=24 && COLS>=80)
{
/* ioperm(DirecBase,3,1); */ /* Permite la lectura y escritura en el puerto */
/* Datos=DirecBase; */ /* Registro de datos */
/* Estado=DirecBase+1; */ /* Registro de estado */
/* Control=DirecBase+2; */ /* Registro de control */
/* outb(mascara,Datos); */ /* Inicializa el puerto, mascara=0 */
start_color();
keypad(stdscr, TRUE); /* Elimina los caracteres de control de la lectura */
init_pair(COLOR_WHITE,7,0); /* 4 = azul, 2 = verde, 3 = Amarillo */
init_pair(COLOR_YELLOW,3,0); /* 5 = Magenta, 6 = cyan, 7 = Blanco */
init_pair(COLOR_GREEN,2,0); /* 1 = Rojo */
init_pair(COLOR_BLUE,4,0); /* */
init_pair(COLOR_RED,1,0);
init_pair(COLOR_MAGENTA,3,4); /* Color inverso */
do{
titulos();
refresca_bits();
/* escribe_en_bitacora(); */
/* ++++++++++++++++++++ Interprete de comandos ++++++++++++++++++++ */
move(21,2);
printw("------------------------------------------------------------------
------------");
/* Imprime la linea de estado */
move(22,2);
printw("gdomo:%s",estado);
attron(COLOR_PAIR(COLOR_BLUE));
move(23,2);
printw("gdomo> ");
attron(COLOR_PAIR(COLOR_WHITE));
getstr(linea_de_comando);
strcpy(linea_al_servidor,linea_de_comando);
num_tokens=separa(linea_de_comando);
interprete_de_comandos();
```

```
}while(strcmp(segmento[0],"salir"));
} /* Final de if comprobacion de renglones y colunas */
else
{
endwin();
printw("La terminal no tiene la longitud mínima requerida\n");
}
/* ioperm(DirecBase,3,0); */ /* Deshabilita la lectura y escritura en el puerto */
endwin();
return(0);
}
```

Parte II

Propuesta de cliente en Java y GTK+

Como se mencionó en las conclusiones, el cliente que se propone en este trabajo no es el único que podría trabajar con el sistema y de hecho no incorpora todas las funciones que podría tener un sistema futuro, aquí se presentan dos opciones que en algún momento se plantearon como posibilidades para el programa cliente, uno de ellas esta escrita en Java y solo implementa las funciones de conexión al servidor, pero muestra que muy fácilmente se puede desarrollar un programa que se comunique con el servidor y opere con el.

```
import java.io.*;
import java.net.*;
public class cliente
{
public static void main(String[] args)
{
System.out.println("UNAM - FI / Osvaldo N. R. ArgÃ¼ello, 2003");
System.out.println("-------------------------------------");
try
{
Socket s=new Socket("127.0.0.1",6543);
InputStream is=s.getInputStream();
BufferedReader br=new BufferedReader(new InputStreamReader(is));
String line=br.readLine();
System.out.println("-> Mensaje recibido");
System.out.println(line);
br.close();
s.close();
}
catch(IOException ioe)
{
System.out.println("-> Ocurrio un error");
System.out.println(ioe);
}
}
}
```

La segunda propuesta, esta escrita en C con GTK+, esta última es una librería llamada (Gimp Tool Kit) que principalmente sirve para definir interfases gráficas al estilo del entorno de desarrollo GNOME propio de los sistemas Linux, además de dibujar toda clase de Widgets en la pantalla puede controlar todas las señales enviadas por estos Widgets para hacer una programación orientada a eventos[3]. El programa mostrado aquí tiene la capacidad de controlar el puerto paralelo a través de botones de encendido/apagado y no cuenta con las funciones de Red, pero ya que el lenguaje C cuenta con el uso de Sockets dentro de su API, fácilmente podremos agregar a este programa tales capacidades, aquí se muestra una imagen de su pantalla principal.

Figura 1: Imagen del programa cliente gráfico propuesto.

Algo muy importante dentro de este aspecto del programa cliente, es la posibilidad de correr este en diversas plataformas ya que será muy común que los usuarios en algunos casos tengan un sistema Windows o Macintosh o Linux. De las opciones mencionadas en el párrafo anterior, el programa hecho en JAVA nos provee la capacidad de correr en todas las plataformas que tengan una máquina virtual JAVA[7], la opción GTK+ solo correría en ambientes Linux, por ahora, ya que existen ya algunos intentos de portar la librería GTK+ a entornos Windows y Macintosh.

Estas dos opciones no son las únicas que se presentan para desarrollar otro programa cliente, solo son dos ejemplos, actualmente hay tantos lenguajes y casi todos con características de conectividad con redes que en casi todos ellos se podría desarrollar un programa útil en este sistema.

Parte III

Software empleado en el desarrollo de este trabajo

Para el documento

1. LyX (www.lyx.org).- procesador de textos basado en LaTex
2. GIMP (The Graphics Image Manipulator Program) (www.gimp.org) .- Editor de imágenes
3. Orégano (oregano.codefactory.se) .- Captura y simulación de circuitos compatible con Spice3
4. dvi2pdf .- Convertidor de formato DVI (Device independet format) a PDF (Portable document format)

Para el programa

1. gcc (www.gnu.org).- El compilador de "C" de GNU
2. gdb (www.gnu.org).- El depurador de "C" de GNU
3. emacs (www.gnu.org) .- Editor de texto y ejecución de macro operaciones
4. Anjuta (www.anjuta.org).- Entorno de desarrollo (IDE) basado en Python
5. make (www.gnu.org).- Ejecución de scripts de compilación e instalación

Todo el software empleado para este trabajo es libre y está disponible tanto en forma binaria como en código fuente en los sitios arriba listados, en el proceso de programación y edición del documento se emplearon las siguientes distribuciones de Linux: RedHat 7.3, RedHat 8.0, RedHat 9.0. Y para las pruebas del servidor y el cliente: RedHat 4.0, Debian 3.0 R1, SuSE 8.1 LiveEval.

Bibliografía

[1] "Red Hat Linux Survival Guide", Mohammed J. Kabir, Hungry Minds, EUA, 2001, 300 p.

[2] "El entorno de programación UNIX", Kernighan, Brian W., Prentice Hall, México, 1987, 361 p.

[3] "GTK+ / GNOME Application Development", Pennington, Havoc, New Raiders, EUA, 1999, 492 p.

[4] "El lenguaje de programación C", Kernighan, Brian W., Ritchie, Dennis M., Prentice Hall, México, 1991, 287 p.

[5] "Programming the paralel port: interfacing the PC for data acquisition and process control", Gadre, Dhananjay, V., CMP Books, EUA, 1998, 308 p.

[6] "Build your own lost-cost data acquisition and display devices", Hirst Johnson, Jeffrey, Mc Graw Hill / TAB Electronics, EUA, 1993, 305 p.

[7] "Java", Friedman - Hill, Hungry Minds, EUA, 2001, 300 p.

[8] "Guia Beej de programación de redes en UNIX" disponible en "http://www.arrakis.es/~dmrq/beej/"

[9] "Universal Serial Bus Specification" disponible en "http://www.usb.org/developers/docs/"

[10] "Linux y IEEE 1394" disponible en "http://www.linux1394.org/index.html"