UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

Control de una red de Automatización X10 a través del Correo Electrónico

Alejandro Balbás Contreras

Guatemala
2004

Control de una red de Automatización X10 a través del Correo Electrónico

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ciencias y Humanidades

Control de una red de Automatización X10 a través del Correo Electrónico

Alejandro Balbás Contreras

Trabajo de investigación presentado para optar al grado académico de
Licenciado en Ingeniería Electrónica

Guatemala
2004

Vo.Bo.:

_____

Ing. Gonzalo Palaréa
Asesor

Tribunal:

_____

Ing. Gonzalo Palaréa

_____

Dr.-Ing. Manuel A. Lopez V.

_____

Ing. Julio Vásquez

Fecha de aprobación: Guatemala, 10 de junio del 2004

CONTENIDO

Página

LISTA DE CUADROS

Página

LISTA DE ILUSTRACIONES

Página

RESUMEN

Al haber cada vez más computadoras conectadas a Internet y ser las direcciones IP de versión cuatro las que prevalecen (IPv4) es cada vez más costoso contar con una dirección IP pública para uso doméstico. Los sistemas de control de redes de automatización a distancia existentes necesitan una dirección IP pública.

El propósito de este trabajo es permitir el control de una red de automatización X10, por ser de las más económicas y de uso más extendido, sin necesidad de una dirección IP pública llevando a cabo la comunicación a través del correo electrónico tradicional.

El objetivo se logró utilizando un microcontrolador eZ80F91 y un módulo de interfaz RS232 para red X10 *PowerLinc II*. El resultado al que se llegó fue un módulo compacto independiente capaz de recibir órdenes de automatización en lenguaje natural (español) por correo electrónico y enviar respuestas o notificaciones por esta misma vía.

I. INTRODUCCIÓN

La idea de una casa automatizada no es nueva. La ciencia ficción ha alimentado desde hace tiempo la imaginación de muchas personas, llevándolas a idear un ambiente en el que todo lo que los rodea obedece a sus deseos y necesidades sin que ellos tengan que hacer mayor esfuerzo para que suceda. Sueñan con una casa en la que luces, puertas y todo tipo de aparato que utilicen obedezca a sus órdenes o a sus acciones. Por ejemplo, que al entrar a una habitación las luces se enciendan sin necesidad de que ellos lo hagan y que se apaguen o bajen de intensidad con un simple comando de voz, o una casa en la que las ventanas o las cortinas se abran al haber calor y se cierren si hace frío.

Aunque todas estas ideas son normalmente relacionadas con ciencia ficción, y es ésta la que las ha ido avivando con el pasar del tiempo, la tecnología actual ya pone a nuestra disposición todas estas comodidades. No es raro encontrar ya en hoteles, centros comerciales, supermercados u otros sitios públicos puertas que se abren automáticamente con la proximidad de las personas.

Sin embargo, debido al costo de la automatización, ésta ha estado por lo general ligada a sitios públicos y no a residencias particulares. Pero conforme la industria avanza, la tecnología baja su costo y ya hay en el mercado productos de automatización destinados precisamente a mercados residenciales.

El conjunto de estos sistemas que automatizan las diferentes instalaciones de una vivienda, que más que las instalaciones en si, es la interacción entre éstas, se llama domótica.

La domótica tiene como propósito, entre otros, facilitar el control integral de la casa controlando por ejemplo desde un solo punto la iluminación de toda la vivienda e incluso puertas, ventanas, calefacción y todo tipo de electrodomésticos y equipo de entretenimiento. Aumentar la seguridad tanto contra acceso no autorizado, incluyendo control de acceso automático, sistemas de vigilancia automatizados y notificación en caso de algún evento, como para prevenir y detectar accidentes mediante sensores (por ejemplo de temperatura, humo, gases y humedad) y actuadores como válvulas electromecánicas y notificación de los problemas.

La domótica pretende también incrementar el confort de los habitantes. Controlan por ejemplo la iluminación para que se encienda con la presencia de los habitantes o sincronizan distintas lámparas y aparatos para distintas actividades. Por ejemplo, lograr con presionar un

botón que las luces disminuyan su intensidad, deje de sonar la música ambiental, una película empiece a reproducirse y que el teléfono y el intercomunicador de la entrada avisen solo visualmente cuando alguien llame o toque a la puerta. Además la domótica tiene también como objetivo ayudar al ahorro de energía, tiempo y dinero, controlando el encendido de aparatos (como luces y aire acondicionado) solo cuando son necesarios y en los lugares en los que se necesitan.

Antiguamente, los sistemas de domótica eran centralizados con sensores remotos. Este tipo de arquitectura hacía los sistemas muy poco flexibles y su actualización o ampliación significaba cambiar prácticamente el sistema completo. El cambio en la arquitectura de los sistemas de domótica es precisamente lo que ha permitido el descenso de precios y el alza en el número y tipo de dispositivos ofrecidos comercialmente.

El tipo de sistemas utilizados actualmente se basan en dispositivos inteligentes conectados a una red formando un sistema distribuido. Las ventajas que ofrece este tipo de arquitectura es que al no haber un dispositivo central que controle todo el sistema, se pueden agregar módulos nuevos que realicen distintas funciones sin necesidad de cambios en el sistema existente. Esta arquitectura fomenta además la creación de protocolos de red abiertos (cualquiera es libre de utilizar el protocolo en sus aplicaciones) que contribuyen a la independencia de soluciones propietarias (sólo la empresa que la desarrolló conoce su funcionamiento) y favorecen a la competencia entre empresas empujándolas a mejorar la calidad y precios de sus productos para conseguir su puesto en el mercado.

Bajo este modelo de redes de sistemas distribuidos han surgido varios protocolos, muchos de ellos abiertos, cada uno con sus puntos fuertes y sus debilidades.

El protocolo más aceptado en Europa se llama KNX. Este protocolo, desarrollado por el grupo Konnex, se basa en tres protocolos anteriores llamados EIB (European Installation Bus) de SIEMENS, Batibus y EHSA (European Home Systems Association) y trata de reunir en tres modos de operación, instalación profesional, instalación simplificada e instalación automática, lo mejor de los tres protocolos. La capa física del protocolo de red permite a los sistemas ser instalados en distintos tipos de entornos pues las señales pueden enviarse por cable trenzado, Radio Frecuencia y a través de la red de alimentación. La primera versión oficial de este protocolo es muy reciente (diciembre 2003) y su propósito es dar en Europa a la domótica el empuje que no ha tenido en ese continente.

El protocolo Lonworks, de Echelon, desarrollado en 1992 es sumamente versátil ya que puede montarse sobre cable coaxial, par trenzado, fibra óptica, radio frecuencia y sobre la red de alimentación. Sin embargo, a pesar de estar fuertemente implantado en aplicaciones de oficinas, hoteles e industrias, no ha logrado introducirse exitosamente en el mercado residencial. Probablemente esto se debe a que a pesar de ser autodenominada una tecnología abierta, los sistemas necesitan utilizar un integrado llamado Neuron Chip para implementar el protocolo que es fabricado únicamente por Echelon, lo que impide la competencia y el descenso de los precios de esta tecnología.

X10 es la tecnología más antigua en este sector. Fue desarrollada entre 1976 y 1978 en Escocia y la transmisión de sus señales está limitada a las líneas de alimentación, aunque esto, más que cerrarle el mercado residencial, la ha vuelto líder en él ya que no es necesario crear nueva infraestructura en las residencias que no fueron diseñadas con fines de ser automatizadas. El protocolo tiene sus limitaciones: trabaja solo a 60 bps (50 bps en Europa) y tiene un conjunto de comandos bastante limitado. Sin embargo la tecnología para implementarlo es sumamente simple y de bajo costo y además el protocolo es completamente abierto. Esto ha permitido el surgimiento de muchas empresas que desarrollan productos con este protocolo y la competencia entre ellas mejorando precios, calidad y diversidad de sus productos.

Después de X10, en 1982 surgió CEBus (Consumer Electronic Bus) como un estándar abierto que proporcionaba más comandos y soporta más medios de comunicación: por la red de alimentación, par trenzado, cable coaxial, infrarrojos, radiofrecuencia y fibra óptica. Además, sobre las líneas de alimentación alcanza velocidades de 7500 bps, mucho más que X10. Aún a pesar de las ventajas que CEBus presenta frente a X10, este último sigue acaparando el mercado de la domótica debido a que al ser más sencillo el protocolo, los dispositivos pueden tener procesadores menos potentes y además, al ser menor la velocidad de transmisión y más sencilla la modulación de las señales, los circuitos para X 10 pueden implementarse con costos mucho menores.

Bajo la teoría de redes distribuidas hay todavía más protocolos que han sido diseñados para la automatización, como por ejemplo SCP (Microsoft y General Electric), Jini (Sun Microsystems), Bacnet, HAVi, uPnP (Universal Plug and Play), HAPI y ZigBee (inalámbrico) entre otros. Pero los protocolos que acaparan los mercados son en Europa el del grupo Konnex a nivel residencial y ECBus a nivel industrial y fuera de Europa Lonworks para industria y edificios y X10 en residencias.

La idea detrás de X10 es poder utilizar la red de alimentación y las lámparas y aparatos ya existentes sin necesidad de modificarlos para su automatización. X10 manda entonces sobre la línea de alimentación tradicional los comandos necesarios de tal suerte que los distintos aparatos de la casa interactúen entre sí.

Existen todo tipo de módulos X10: interruptores y controladores de intensidad para lámparas, interruptores para cualquier tipo de dispositivo, sensores de movimiento, de luz, de temperatura, sensores de apertura o cierre de puertas o ventanas, motores para cortinas y persianas y detectores de comandos de voz entre otros. La combinación de todo este tipo de módulos resulta en la fácil automatización de una casa dependiendo de las necesidades o deseos de cada usuario sin la necesidad de dispositivos muy especializados.

Se encuentran además dispositivos de control remoto para regular el funcionamiento de la red X10 a distancia, incluyendo entre ellos los que pueden hacerlo a través de Internet.

Con la introducción por parte de las compañías de telecomunicación de conexiones permanentes a Internet de banda ancha a nivel residencial, un dispositivo para acceder a la red X10 a través de Internet se vuelve la solución ideal para controlar y monitorear una casa a distancia, incluso cuando se está de viaje.

Sin embargo existe un problema. Las conexiones residenciales a Internet por lo general no cuentan con una dirección IP pública, por lo que acceder a ellas desde fuera no es posible.

La idea del módulo que se describe en el presente trabajo es precisamente para lograr acceder a la red X10 desde Internet en cualquier sitio, a pesar de no tener disponible una dirección IP pública. Para lograrlo, el correo electrónico es la solución idónea. A través de él se pueden enviar y recibir cualquier tipo de información, incluyendo comandos para controlar la red X10.

El uso de una computadora para este fin no es práctico, pues necesitaría estar encendida todo el tiempo aunque no se tenga necesidad de utilizarla para otro propósito, lo que provoca desaprovechamiento de energía y desgaste. El módulo mencionado pretende conectarse directamente a la red LAN, evitando de esta manera incluso la necesidad de una computadora.

II. OBJETIVOS

- Armar un sistema compacto con la capacidad de establecer una conexión a Internet a través de una red Ethernet. El sistema debe ser a la vez independiente, es decir, utilizará un programa corriendo en una PC remota (cliente) para su configuración, pero su funcionamiento no dependerá de él y una vez configurado el sistema éste debe efectuar sus tareas sin asistencia de ningún otro componente externo más que la conexión a la red Ethernet y a la red X10.

- Desarrollar sobre el sistema un cliente de correo POP3 capaz de recuperar los correos recibidos y seleccionar los comandos contenidos dentro de ellos.

- Crear un lenguaje para la interacción con el sistema X10 a través del correo electrónico.

- Agregar al sistema del módulo un cliente SMTP capaz de enviar correos electrónicos para informar sobre el estado de la red.

- Implementar una interfase X10 para poder interactuar con módulos enviándoles y recibiendo de ellos comandos X10.

- Desarrollar un servidor HTTP que corra sobre el módulo con el fin de ofrecer al usuario los medios necesarios para la configuración del módulo.

III. MÉTODOS DE TRABAJO

Para lograr los objetivos arriba expuestos, se utilizó un módulo de microcontrolador que incluía entre otros periféricos y controladores una interfase de red (NIC: *Network Interface Controller*) para lograr la interconexión a una red Ethernet.

De esta manera quedó establecida la capa física y sus servicios para desarrollar sobre ella la capa de red y de transporte las cuales fueron implementadas en software para correr sobre el microcontrolador y formar la pila TCP/IP para los servicios y puntos de acceso a la capa de aplicación que se programó sobre el microcontrolador del sistema.

Dentro del programa del microcontrolador también se montó un servidor HTTP que es el encargado de ofrecer a los usuarios del módulo los medios necesarios para la configuración a través de la red Ethernet.

Además, se desarrolló un cliente de correo POP3 capaz de manejar las instrucciones del protocolo necesarias para la conexión al servidor, autenticación del usuario, recuperación de los correos y extracción de los comandos incluidos en estos.

El servidor HTTP y el cliente POP3 formaron la capa de aplicación de la pila de red del sistema.

Por último se creó un protocolo o lenguaje que consta de comandos y parámetros que sirve para que el usuario pueda hacerle llegar al sistema sus instrucciones.

A continuación se describen los componentes, estrategias y protocolos que forman la base del proyecto y la justificación de por que fueron seleccionados.

A. MICROCONTROLADOR

Una de las partes más importantes del proyecto es el acceso a la Red de Área Local (LAN) ya que es a través de ésta que el sistema se conectará a Internet para recibir órdenes y enviar notificaciones. La conexión a la LAN es también la que brinda acceso al módulo para su configuración.

Para lograr la conexión y mantener el sistema compacto e independiente de cualquier

otra computadora, fue necesario que el proyecto se desarrollara sobre un microcontrolador que pudiera interactuar con un controlador de acceso al medio (MAC) para la conexión a la red.

El sistema debía contar además con suficiente memoria para almacenar las páginas HTML de configuración y las definiciones para el lenguaje que utiliza el usuario a través del correo electrónico.

El microcontrolador elegido fue el Módulo *eZ80F91* de la familia *eZ80Acclaim!* de microcontroladores *ZiLOG*. Este módulo incluye dentro de la misma tarjeta impresa un Microcontrolador *eZ80F91*, un Controlador de Acceso al Medio y suficiente memoria para las necesidades del proyecto entre otros elementos de utilidad.

El Microcontrolador *eZ80F91* trabaja a 50 MHz y cuenta entre sus periféricos con un Controlador de Acceso al medio para Ethernet que soporta velocidades 10/100Mbps que sirve para el acceso a Internet, dos puertos seriales UART de los cuales uno se utiliza para la interacción con la red X10. Además cuenta con un Reloj de Tiempo Real (RTC) que es empleado para la programación de comandos X10 que el sistema efectúa en un momento determinado.

El módulo en el que se encuentra el microcontrolador, además de contener el MAC y el conector RJ45 para la red Ethernet, incluye un circuito de respaldo de poder para el Reloj de Tiempo Real para que éste no pierda la hora incluso si falla la alimentación del resto del sistema, lo que permite que las tareas programadas se realicen en el momento adecuado incluso después de una falla de poder.

El microcontrolador *eZ80F91* tiene incorporados 256 KB de memoria FLASH para el programa y 8 KB de memoria RAM para el MAC. A estas memorias se suma la que está ubicada en el módulo del microprocesador que aporta otros 512 KB de RAM para las variables del programa y 1 MB de memoria FLASH para almacenar el resto del programa y las variables que deben conservarse incluso después de una falla en la alimentación.

Otra de las características principales por las que se escogió este microcontrolador ante otros con características de Hardware similares es el Software que ZiLOG ha desarrollado para sus Microcontroladores. La familia de Microcontroladores FLASH *eZ80Acclaim!* cuenta con compilador ANSI C y un sistema operativo de fuente abierta llamado XINU. El compilador ANSI C facilita y acelera mucho el desarrollo de software. En C pueden escribirse programas más sofisticados sin que el código resultante sea demasiado complejo o enredado.

XINU es un sistema operativo (OS) que tiene la simplicidad como fundamento, aunque no por esto deja de ser un sistema operativo completo. Tiene todas las funciones necesarias para implementar un sistema multitarea y además brinda soporte para red. Al ser un OS multitarea, el desarrollo de aplicaciones se vuelve mucho más robusto y simple ya que permite independizar entre sí las diferentes tareas del sistema.

B. ACCESO A LA RED X10

Como se comentó ya en la introducción, el protocolo X10 se escogió para el proyecto debido a la simplicidad de instalación al no requerir ningún tipo de cableado o conexión diferente al cableado estándar de alimentación de cualquier casa y por ser el protocolo más utilizado en el continente americano.

X10 transmite todos los comandos a través de las líneas de alimentación de corriente alterna (AC) de las casas. Las señales son transmitidas durante el cruce por cero de la línea de alimentación de 110V que es cuando se logra una mayor relación de señal a ruido para la transmisión. Durante cada cruce por cero de la señal portadora de 110V puede ser enviado un bit de información. Un 1 lógico es representado por una señal de 120kHz montada sobre la señal de alimentación; la ausencia de esta señal en el momento de un cruce por cero significa un 0 lógico. Por lo general, las líneas de alimentación que llegan a las casas son parte de líneas trifásicas y los 110V presentes en los tomacorrientes es la diferencia de voltaje entre una de las fases y tierra. Como en la gran mayoría de las casas se utiliza más de una de las fases en la instalación para balancear las cargas, el cruce por cero puede no coincidir entre todas las líneas de la casa. Para que la señal pueda ser recibida por módulos ubicados en líneas a diferente fase, un 1 debe ser enviado no solo en el cruce por cero, sino también dos veces más coincidiendo con los cruces por cero de las otras dos fases.

Los comandos que se envían a través de la línea están compuestos por un código de inicio de cuatro bits (1110) seguido de un código de casa de cuatro bits y cinco bits representando un número de unidad o comando.

El protocolo X10, a pesar de no estar orientado a conexión, tiene como principal objetivo que la recepción de los datos sea fiable y evitar que el ruido en la línea sea interpretado como comandos. Para conseguirlo introduce redundancia en dos aspectos de la transmisión. En primer lugar, después de todo bit enviado, a excepción de los del código de inicio, se envía también el

complemento. Y en segundo lugar, el comando completo, incluyendo el código de inicio debe ser enviado dos veces y los receptores no tomarán un comando transmitido como válido hasta haberlo recibido dos veces. De esta manera, las posibilidades de que el ruido en la línea sea malinterpretado como cualquiera de los comandos o direcciones reconocidas se ven drásticamente reducidas.

X10 también cuenta con detección de colisiones. Cualquier módulo puede usar la línea para transmisión, siempre y cuando no haya sido transmitido un comando en los últimos tres pasos por cero. Si en dado caso dos módulos comienzan la transmisión en el mismo momento, el primero que encuentre un 1 en la línea cuando está enviando un 0 debe retirarse y esperar que la línea vuelva a estar desocupada por lo menos durante tres cruces por cero.

Cada módulo tiene asignada una dirección y obedecerá sólo a los comandos que sean enviados a esa dirección. Las direcciones consisten de una casa (de la casa A a la P) y de un número de unidad (de la unidad 1 a la 16) permitiendo así 256 direcciones diferentes. Para enviar un comando a un módulo debe enviarse primero un mensaje que contenga el código de casa y el código de unidad y luego uno con el código de casa y la orden que el módulo debe obedecer.

Los diferentes comandos de los que se conforma el protocolo X10 son los siguientes:

Códigos X10	
00001	Apagar todas las unidades
00011	Encender todas las luces
00101	Encender
00111	Apagar
01001	Obscurecer
01011	Aclarar
01101	Apagar todas las luces
01111	Código extendido
10001	Petición de saludo
10011	Saludo
101X1	Prefijar intensidad (X es el bit más significativo de la intensidad)
11001	Datos extendidos
11011	Estado = encendido
11101	Estado = apagado
11111	Petición de estado

Cuadro 1: Códigos X10

A pesar de que X10 como protocolo da soporte a todos estos comandos, con el fin de

mantener al X10 como una opción económica de automatización al alcance de todo el mundo la mayoría de los módulos comerciales receptores (como controladores de luces, electrodomésticos, cortinas, válvulas, etc.) no están equipados con un transmisor X10 y sólo un muy pequeño porcentaje puede responder a las peticiones de saludo y de estado. Debido a esto, la inmensa mayoría de redes X10 utilizan sólo los comandos de encender, apagar, aclarar y obscurecer, dejando los comandos de petición de saludo y estado sólo a un reducido número de controladores de dos vías.

Para la aplicación de controlar una red por el correo electrónico, los comandos que realmente interesan son los de encender y apagar ya que los comandos de aclarar y encender son utilizados prácticamente en exclusiva por los módulos para controlar lámparas incandescentes y estando a distancia poca importancia tiene para el usuario si la luz de su casa está encendida al 100 ó al 50%.

Para poder enviar y recibir comandos a través de la red X10 que se desea controlar, se escogió el *PowerLinc II* de *SMARTHOME*. Éste es un receptor/transmisor X10 que traduce los comandos X10 que recibe por la línea de alimentación y avisa por un puerto serial RS232 el comando X10 que fue recibido. A la vez es capaz de recibir mensajes por el puerto RS232 y convertirlos a comandos X10 para enviarlos por la línea de alimentación. Esto permite al procesador enviar y recibir comandos X10 a través de uno de sus periféricos de puerto serial UART sin tener que utilizar mayor tiempo de procesamiento controlando circuitos externos. Otra razón por la que se escogió el *PowerLinc II* fue porque junto con su interfase serial RS232 tiene una fuente no regulada de 700 mA que servirá para alimentar al circuito del proyecto.

C. ENVÍO DE CORREOS ELECTRÓNICOS

El envío y tráfico de correos electrónicos a través de Internet se lleva a cabo utilizando el Protocolo de Transferencia Simple de Correo (Simple Mail Transfer Protocol) SMTP. Éste es un protocolo de la capa de aplicación que permite la comunicación entre clientes de distintas redes para el envío de correos.

SMTP funciona de la siguiente manera: cuando un usuario envía un correo electrónico, el programa de correo o el servidor de webmail en el que ha lo ha escrito se conecta a un servidor SMTP y le envía el correo junto a la información necesaria para que el correo pueda llegar a su destino. El servidor que recibió el correo revisa entonces la dirección o direcciones destino que

recibió junto al correo que pueden ser direcciones propias de ese servidor o ajenas a él. Si el servidor SMTP tiene una casilla de correo para la dirección de destino, éste guarda una copia del correo en la casilla. Si la dirección no es de ese servidor, éste se conecta a otro servidor SMTP que puede ser el servidor que contiene la casilla del destinatario o un servidor intermedio. El correo sigue viajando a través de servidores SMTP hasta llegar al servidor que contiene la casilla de la dirección del destinatario. Al llegar a este servidor, se guarda una copia del correo en la casilla y el correo está disponible para ser recuperado por el usuario final a través de un cliente de correo.

Una de las mayores ventajas del SMTP es que al correr en la capa de aplicación es independiente del tipo de red por donde circulan los correos, dependiendo únicamente de un transporte que le proporcione un servicio en donde los datos sean recibidos en el orden en que se envían. De esta manera se hace posible el envío y recepción de mensajes a través de redes de diferentes tecnologías.

En una red TCP/IP, el servicio de transporte que le brinda a SMTP un flujo ordenado de datos es el TCP. Todas las conexiones y comunicaciones entre servidores SMTP en redes TCP/IP se realiza entonces a través de conexiones TCP al puerto 25 del servidor que recibe la conexión.

Para el proyecto de este trabajo, el programa corriendo en el Microcontrolador es el encargado de establecer esta conexión al puerto 25 del servidor SMTP para el envío de información al usuario.

D. RECEPCIÓN DE CORREOS ELECTRÓNICOS

Una vez los correos electrónicos han sido transportados a través de servidores SMTP hasta la casilla de su destinatario final, éste tiene que acceder a ellos o extraerlos para poder leerlos. Esto se lleva a cabo a través de clientes de correo.

Existen diversos tipos de clientes de correo. Muchos son simplemente aplicaciones corriendo en el mismo servidor que contiene la casilla que utiliza páginas HTML para presentar al usuario la información contenida en la casilla y permitirle borrar los mensajes que no desea conservar. Este tipo de cliente de correo, a pesar de ser uno de los más difundidos en estos días no sigue ningún estándar y tanto la presentación como la manipulación de la información de los correos dependen completamente de la implementación del cliente que use cada servidor en particular. Además, los correos nunca son copiados al cliente donde desean ser vistos sino que permanecen en el servidor, que por lo general tiene una cuota máxima de espacio para cada

usuario, limitando así el número y tamaño de los mensajes que un usuario pude tener en un momento determinado.

Otro tipo de clientes de correo son los que siguen el protocolo IMAP. Éste, a diferencia de los clientes a través de páginas HTML, sí es un protocolo estándar que sirve para acceder y manipular los correos de un usuario almacenados en un servidor. Aún así, presenta el mismo inconveniente de cuotas de espacio en el servidor ya que toda la información está guardada en el servidor y no en el cliente.

Otro protocolo usado por clientes de correo es el protocolo POP3 (Post Office Protocol) que es uno de los más difundidos en la actualidad. POP3, a diferencia de los protocolos anteriores, está diseñado para ser un protocolo simple que permita descargar los archivos del servidor al cliente, en donde son leídos e interpretados sin necesidad de mantener una conexión al servidor. Este protocolo, al igual que el SMTP es un protocolo de capa de aplicación que utiliza los servicios confiables que le brinda TCP en una red TCP/IP para establecer una conexión al servidor. El servidor POP3, corriendo en el servidor que ha recibido los mensajes a través de SMTP, espera conexiones de los clientes POP3 (desde el sistema del usuario receptor del correo) a través del puerto 110.

En el caso de este proyecto, el programa corriendo en el microcontrolador es el encargado de comunicarse con el servidor POP3 que contenga los correos para pedirle información acerca de correos entrantes y descargarlos para ser interpretados.

E. LENGUAJE DE USUARIO

Una vez recibidos los mensajes de correo, el sistema debe ser capaz de entender el mensaje que el usuario le ha enviado y para eso es necesario que ambos hablen un mismo idioma o lenguaje.

Hacer que el usuario final aprenda un nuevo lenguaje para hablar con el sistema era factible y probablemente la solución más simple, pero definitivamente no era la idónea. La mejor solución era crear un lenguaje que el programa pueda interpretar correctamente pero que sea lo más parecido posible al lenguaje normal del usuario.

Para lograr esto se tenían que tomar en cuenta muchos aspectos del lenguaje normal de un usuario. En primer lugar, no hay una sola forma de definir una acción sino que existen

distintos sinónimos con los que la persona que desea ordenarle algo al sistema puede utilizar. Además, al dar la orden el usuario puede formular los comandos en distintas conjugaciones verbales y dirigirse al sistema utilizando distintas personas. Por ejemplo, para pedir que se encienda una lámpara puede decir "Enciende la lámpara", "Encender la lámpara", "Encienda la lámpara", "Quiero que se encienda la lámpara" o incluso "La lámpara, enciéndela." Todas estas oraciones, a pesar de ser diferentes, encierran el mismo mensaje y con el fin de ajustarse lo más posible al lenguaje normal de cualquier usuario, el sistema debía ser capaz de interpretarlo correctamente.

El programa encargado de interpretar las órdenes contenidas dentro de los correos recibidos, debía poder identificar el mensaje que pretendía comunicar el usuario dentro de las oraciones sin importar las formas verbales que éste haya usado para escribirlas ni el orden en el que haya formulado la oración.

De esta forma, quien use el sistema no deberá aprender un nuevo idioma para hablar con la red de automatización ni deberá preocuparse de expresar sus órdenes en una manera demasiado rígida. Podrá enviar sus comandos tal y como lo haría al hablar con cualquier persona que pudiera manipular los aparatos conectados a la red de automatización en la misma forma que el sistema de este proyecto.

F. CONTROL DE LA RED X10

Con el fin de poder controlar la red de automatización aceptablemente el sistema debe ser capaz de entender y llevar a cabo cierto número de comandos diferentes.

En primer lugar, la acción más elemental dentro del control de la red es poder encender o apagar cualquier módulo cuando se recibe la orden del usuario y en la secuencia en la que éste lo ordene. Segundo debe poder programar tiempos de encendido o apagado de módulos de la red X10 según sea indicado por el usuario del sistema. Para esto debe además llevar un control fiable del tiempo. También tiene que ser capaz de reportar, al ser cuestionado al respecto, del estado de los módulos de la red. Debido a que la mayoría de módulos comerciales no proporcionan comunicación en doble vía, el sistema sólo debe informar de los módulos de los que conozca su estado por haber recibido o enviado un mensaje X10 para modificarlo. Además tiene que poder enviar notificaciones de eventos específicos a través de correo electrónico cuando el usuario le solicite saber de ese cambio en particular.

Con el fin de poder prestar todos estos servicios para el usuario, el sistema debía mantener una base de datos del estado de los módulos en la red y una serie de listas para guardar la información acerca de que notificaciones a enviar y de las acciones que debe realizar en momentos específicos.

G. CONFIGURACIÓN

Para la configuración del módulo, incluso antes de haber especificado el servidor y cuenta que usará para recibir correos, éste deberá ser accesado a través de la red de área local. Para poder localizar el dispositivo dentro de la LAN sin necesidad de conocer su dirección IP, que puede haber sido obtenida dinámicamente, se tendrá que hacer uso del servicio de nombres de NetBIOS.

NetBIOS es un protocolo que permite identificar y compartir recursos dentro de una red. Por lo general, el servicio de nombres de NetBIOS permite a un sistema registrar un nombre dentro de la red en que se encuentra y unirse a uno o varios nombres de grupo. Una implementación completa del protocolo de NetBIOS incluye servicios para registrar nombres, competir por un nombre ya presente en la red, buscar un nombre dentro de la red, cancelar el uso de un nombre previamente registrado y responder cuando algún otro integrante de la red pregunta por el nombre del sistema. Para este proyecto, el único servicio necesario era la respuesta cuando alguien pregunte por su nombre. No interesaba tener el nombre registrado dentro de toda la red para que cualquiera pueda encontrar el sistema en cualquier momento. Lo importante es que alguien que sepa que está presente en la red pueda encontrarlo cuando lo busca.

El procedimiento funciona de la siguiente manera: cuando un usuario desea configurar el módulo, escribe el nombre del sistema en el campo de dirección de su buscador y el sistema envía un paquete NetBIOS a la dirección de difusión de la LAN preguntando quien tiene el nombre por el que pregunta el usuario. Cuando este paquete llega al módulo del proyecto, este responde con un paquete NetBIOS al sistema que envió el paquete de pregunta indicándole que el nombre es suyo y su dirección IP. A partir de ese momento, la comunicación entre el buscador en el que el usuario introdujo el nombre y el módulo del proyecto se lleva a cabo a través de paquetes de petición HTTP desde el sistema del buscador al IP del módulo.

Los recursos que el módulo pone disponibles a través de HTTP debían ser páginas

HTML a través de las cuales el usuario pueda monitorear y modificar los parámetros de funcionamiento del módulo incluyendo los servidores e información de cuenta que el módulo utilizará para enviar y recibir correos electrónicos.

IV. DESARROLLO DEL TRABAJO

A. RECEPCIÓN DE CORREOS ELECTRÓNICOS

Como ya se comentó en la sección de métodos de trabajo, el protocolo escogido para la recepción de correos electrónicos es POP3.

Los comandos del protocolo POP3 son palabras de hasta cuatro caracteres ASCII, seguidas algunas de parámetros compuestos también de letras ASCII y separados del comando por un espacio y terminados por un retorno de carro seguido un cambio de línea (CRLF). Los comandos y los parámetros son enviados como texto a través de la conexión TCP al servidor POP3. Después de cada comando recibido, el servidor responde con un mensaje empezando con +OK si el comando tuvo éxito o –ERR si hubo algún error y terminado siempre con un par CRLF. Para mensajes multilínea, se indica el final de la respuesta con una línea conteniendo un único punto seguida de una línea vacía (CRLF.CRLF)

Los comandos necesarios para este proyecto son 7: USER, PASS, STAT, TOP, RETR, DELE y QUIT. Los primeros dos, USER y PASS son utilizados para iniciar la sesión y enviar la información de usuario al servidor. USER se envía con el nombre del usuario. Si el servidor conoce al usuario, responde positivamente pidiendo la contraseña de ese usuario. Ésta es enviada como argumento del comando PASS y si es la correcta, el servidor responde con un mensaje que comienza con +OK indicando que el inicio de sesión se ha completado con éxito.

A partir del momento en el que se ha iniciado la sesión, el resto de comandos POP3 pueden ser usados. Al comando STAT, que no necesita parámetros, el servidor responde con un +OK seguido de dos números, el número de mensajes en la casilla y el número de bytes que ocupan.

El comando TOP sirve para recuperar las cabeceras y parte de un mensaje sin recuperarlo todo. Necesita dos parámetros, el número de mensaje y la cantidad de líneas del cuerpo de mensajes que se desea que envíe el servidor.

RETR sirve para descargar un mensaje del servidor y necesita un solo parámetro: el número que identifica el mensaje dentro de la casilla. Si el mensaje existe, el servidor responde con una línea que comienza con +OK seguida de todas las líneas de cabecera del mensaje, una línea vacía y el mensaje seguido inmediatamente por una línea con un solo punto y una línea en

blanco (la secuencia de escape).

DELE es el comando para borrar mensajes y toma como único parámetro el número del mensaje que se desea borrar. Los mensajes no son realmente borrados, sino marcados como eliminados y permanecen en el servidor hasta que la sesión ha termina.

El último de los comandos utilizados en el proyecto es QUIT que indica al servidor el fin de sesión para que borre los mensajes que han sido marcados como borrados con el comando DELE y desconecte la sesión.

Comandos POP3 utilizados	
USER	Enviar nombre de usuario
PASS	Enviar contraseña de usuario
STAT	Información acerca del número de mensajes en casilla
TOP	Descargar parte de un mensaje
RETR	Descargar un mensaje
DELE	Marca un mensaje como eliminado
QUIT	Termina la sesión y borra los mensajes marcados como eliminados

Cuadro 2: Comandos POP3

La rutina encargada de revisar si hay nuevos correos en el servidor y descargar un correo cuando llegue utiliza estos siete comandos para comunicarse con el servidor.

Lo primero que hace es conectarse al servidor POP3 y, tras recibir una respuesta positiva anunciando que el servidor está listo, envía la información de usuario con los comandos USER y PASS. Si hay algún error en cualquiera de estas dos operaciones, la rutina desconecta la conexión y termina con un código de error LOGIN_ERROR.

Si no hay error en la etapa de autenticación de usuario, envía el comando STAT e interpreta la respuesta para averiguar el número de mensajes en la casilla. De no haber ningún mensaje, se desconecta del servidor y termina con un mensaje de NO_DATA.

En el caso de haber mensajes en la casilla de correo, se utiliza el comando TOP para recuperar la información de cabecera del primer mensaje de la casilla. De entre estos encabezados se extrae la dirección de correo de quien envía el correo. Si la dirección no está dentro de la lista de usuarios permitidos, se borra el correo, se desconecta el servidor y se devuelve el código USER_ERROR. Sin embargo, si la dirección es válida, se utiliza el comando RETR para descargar el mensaje de la casilla y tras descartar los encabezados, el mensaje es guardado en un buffer para

poder ser interpretado más tarde por otras partes del programa. Una vez leído y guardado el mensaje, éste es marcado como eliminado con el comando DELE la conexión con el servidor se termina con el comando QUIT. Entonces, la función termina devolviendo el código POP3_DATA para indicar que hay información en el buffer lista para ser interpretada.

La Ilustración 1 muestra el diagrama de flujo de la rutina de revisión de correos.

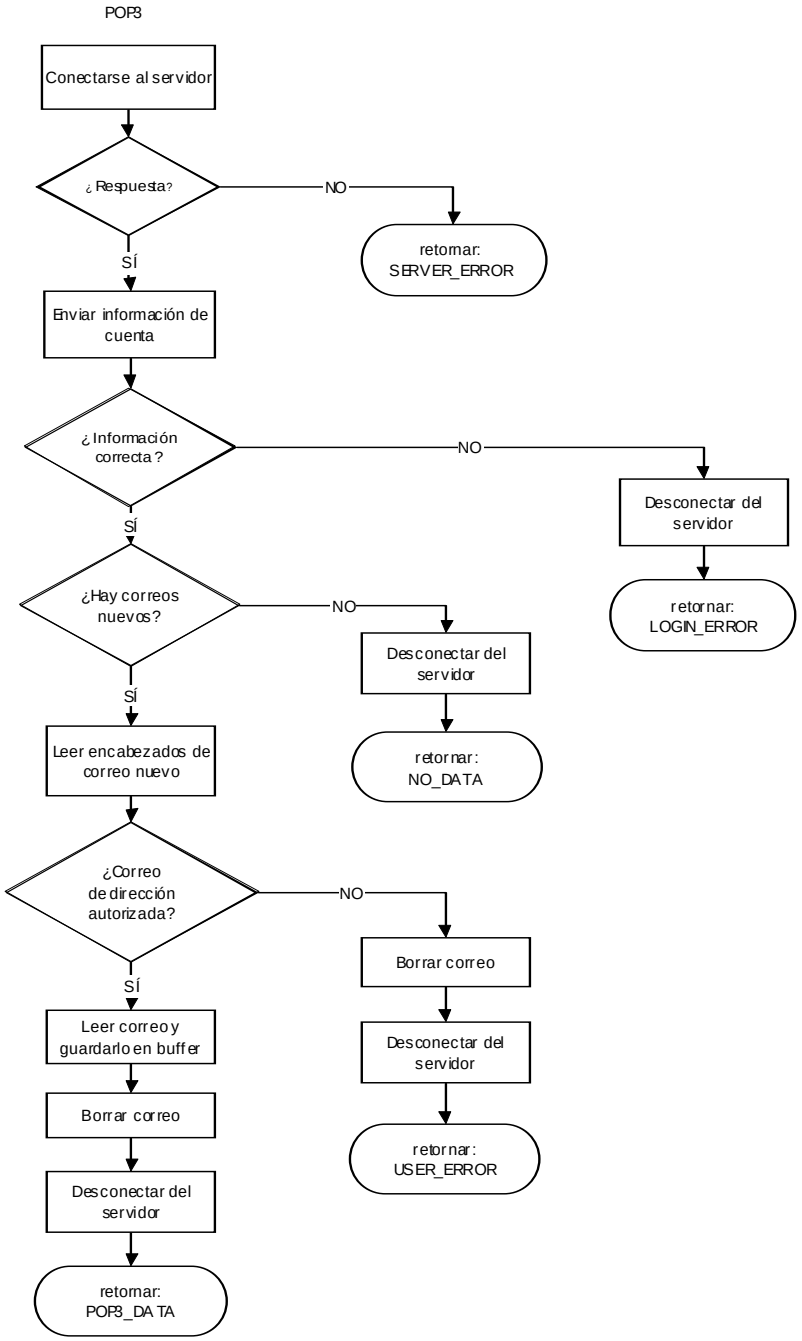


Ilustración 1: Diagrama de flujo de POP3

B. ENVÍO DE CORREOS ELECTRÓNICOS

Al igual que los comandos POP3, SMTP utiliza mensajes de caracteres ASCII terminados en un retorno de carro seguido de un cambio de línea (CRLF). Los comandos SMTP utilizados en este proyecto para comunicarse con el servidor son HELO, MAIL, RCPT, DATA y QUIT.

A estos comandos, el servidor SMTP responde con un código de estado de tres dígitos seguido por un mensaje de descripción del código. El primer dígito del código de estado indica si la operación pudo ser realizada con éxito o no. Si éste dígito es un cinco significa que hubo un error y no pudo llevarse a cabo el comando solicitado. Los otros dos dígitos del código indican qué tipo de error fue el que se dio. Si la orden pudo ser interpretada y obedecida correctamente, el código de respuesta comienza con un dos o un tres. El dos significa que la operación a concluído y el servidor está listo para recibir más comandos. Si el primer dígito es un tres; sin embargo, el servidor está esperando más información del usuario para poder finalizar la acción. Este tipo de códigos sólo se presenta después de un comando DATA, que es utilizado para enviar el cuerpo y las cabeceras del correo.

El comando HELO es el que se utiliza una vez establecida la conexión TCP para iniciar la sesión con es servidor. Utiliza un parámetro separado por un espacio que es el nombre de red o dirección del sistema que lo emite y sirve para identificarse con el sistema remoto quien responde con su nombre de red o dirección. Esta dirección no es comprobada de ninguna manera por lo que no es utilizada por los servidores SMTP para autenticar usuarios. De hecho, la implementación básica de SMTP, que es la que se usa la mayoría de los casos, carece de un sistema de autenticación de usuarios ya que el protocolo pretende que los correos puedan circular libremente entre servidores para llegar a su destino. Sin embargo para impedir que cualquiera pueda utilizar los servidores para enviar correos sin autorización, se ha recurrido a otros métodos de autenticación entre los cuales el más utilizado es la comprobación del IP para asegurarse que venga de una red específica o que haya sido usado con anterioridad para acceder a la casilla con el protocolo POP3, que sí utiliza autenticación de usuarios.

Para enviar la información del remitente del mensaje se utiliza el comando MAIL. Éste lleva como argumentos, separados por espacios, la palabra FROM seguida de dos puntos y la dirección de quien envía el mensaje.

La información de el o los destinatarios del mensaje se hace llegar al servidor por medio

del comando RCPT , al que le sigue la palabra TO y un carácter de dos puntos. Detrás viene la lista de destinatarios, separadas por el carácter punto y coma.

Las direcciones utilizadas tanto en el comando RCPT TO: como en el comando MAIL FROM: siguen un mismo formato. Consisten en una cadena arbitraria de caracteres seguida por otra cadena de caracteres delimitada por un carácter < antes y un > después que es la dirección. La primera cadena de caracteres, que es opcional, suele representar el nombre del usuario de la dirección en una forma en la que pueda ser identificado por la persona que recibe el correo

El comando DATA no utiliza ningún argumento al ser enviado y a menos que exista algún error en la información de destinatario y remitente, el servidor retorna el código de estado 354. Este indica que el servidor está listo para recibir el resto de la información asociada con el comando que es el mensaje que se va a enviar. La primera parte son los encabezados que describen al mensaje incluyendo el campo de formato, el de la fecha de envío y el del asunto del mensaje. Luego, separado por una línea vacía sigue el cuerpo del mensaje que es finalizado por una nueva línea que contiene un solo punto (CRLF.CRLF) que significa el final del mensaje e indica al servidor que la toda la información para el comando ha sido enviada. Entonces, si no hay errores, el servidor retorna un código 250 indicando el éxito del comando. En este momento, el mensaje ha sido enviado.

Para terminar la sesión se utiliza el comando QUIT que informa al servidor que ya no se desea enviar más comandos y se cierra la conexión TCP.

Comandos SMTP utilizados	
HELO	Inicia la sesión e identifica al sistema
MAIL	Envía la dirección del remitente del mensaje
RCPT	Envía la dirección del destinatario del mensaje
DATA	Inicia el envío del mensaje
QUIT	Termina la sesión

Cuadro 3: Comandos SMTP

Para prestar el servicio de envío de correos electrónicos, el programa del proyecto utiliza tres funciones. La primera, SMTP_reset, es la encargada de inicializar la memoria para guardar el correo y la información necesaria para enviarlo. La función recibe tres argumentos, el destinatario, el asunto y el formato con el que se va a enviar el mensaje, HTML o texto simple. Los datos son guardados en memoria y se inicializa el buffer donde se va a guardar el mensaje y se escribe en él las cabeceras del mensaje, incluyendo fecha y asunto, y el inicio del mensaje que de

ser un mensaje de tipo HTML consiste de las etiquetas HTML que preceden el cuerpo del mensaje.

El cuerpo del mensaje se va formando con llamadas externas a las funciones SMTP_add o SMTP_addln. Estas funciones, que reciben una cadena de caracteres como argumento, se encargan de agregar el texto recibido al buffer y avanzar el puntero que indica la posición actual. La diferencia entre las dos funciones es que SMTP_addln agrega un juego de caracteres CRLF al final del texto que recibe.

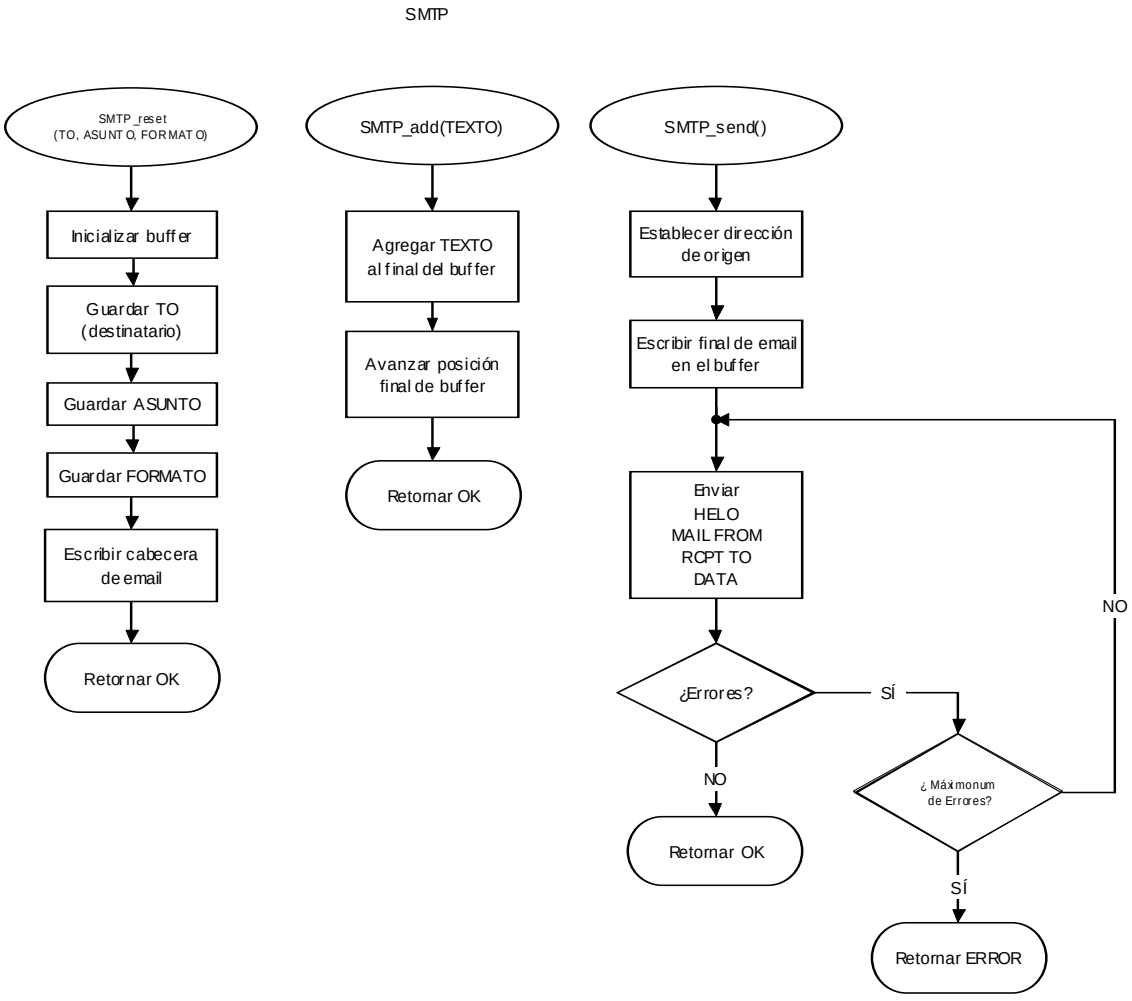
Una vez inicializado el correo y habiendo sido agregada al cuerpo toda la información deseada, el mensaje es enviado con la función SMTP_send. Esta función establece la dirección del remitente a partir de los valores de usuario y servidor SMTP y escribe el final del email al final del buffer que, de ser un mensaje HTML consiste del cierre de las etiquetas HTML abiertas al inicio y que encierran el cuerpo del mensaje. Entonces, a través de una llamada a la función mail del stack TCP/IP del Microcontrolador, se envían los comandos SMTP y el correo al servidor. En el caso de haber errores en la transmisión y que el email no haya podido ser enviado, se vuelve a intentar el envío hasta que el email haya podido ser enviado o se haya alcanzado el número máximo de intentos de envío.

La Ilustración 2 muestra el diagrama de flujo de las funciones de envío de mensajes.

C. ENVÍO Y RECEPCIÓN DE COMANDOS X10

Como ya se explicó antes, para la comunicación con la red X10 se utiliza un módulo *PowerLinc II* que manda y recibe comandos X10 por la línea de alimentación X10 y se comunica con el microprocesador a través de un puerto serial UART.

Entonces, para poder enviar y recibir comandos X10, el programa del procesador tiene que poder enviar y recibir por su puerto UART las secuencias de bytes que el *PowerLinc II* comprende. Este módulo utiliza un protocolo bastante similar al protocolo X10 en su puerto serial. La mayor diferencia radica en los bytes que utiliza para identificar las direcciones y comandos. El cuadro 4 contiene los códigos tanto del protocolo X10 como los utilizados por el *PowerLinc II*.



lustración 2: Diagrama de flujo de SMTP

Códigos	X10	PowerLinc	Códigos	X10	PowerLinc
Casa A	0x06	0x46	Unidad 9	0x0E	0x4E
Casa B	0x0E	0x4E	Unidad 10	0x1E	0x5E
Casa C	0x02	0x42	Unidad 11	0x06	0x46
Casa D	0x0A	0x4A	Unidad 12	0x16	0x56
Casa E	0x01	0x41	Unidad 13	0x00	0x40
Casa F	0x09	0x49	Unidad 14	0x10	0x50
Casa G	0x05	0x45	Unidad 15	0x08	0x48
Casa H	0x0D	0x4D	Unidad 16	0x18	0x58
Casa I	0x07	0x47	Apagar todas las unidades	0x01	0x41
Casa J	0x0F	0x4F	Encender todas las luces	0x03	0x43
Casa K	0x03	0x43	Encender	0x05	0x45
Casa L	0x0B	0x4B	Apagar	0x07	0x47
Casa M	0x00	0x40	Oscurecer	0x09	0x49
Casa N	0x08	0x48	Aclarar	0x0B	0x4B
Casa O	0x04	0x44	Apagar todas las luces	0x0D	0x4D
Casa P	0x0C	0x4C	Código extendido	0x0F	0x4F
Unidad 1	0x0C	0x4C	Petición de saludo	0x11	0x51
Unidad 2	0x1C	0x5C	Prefijar intensidad alta	0x17	0x57
Unidad 3	0x04	0x44	Prefijar intensidad baja	0x15	0x55
Unidad 4	0x14	0x54	Datos extendidos	0x19	0x59
Unidad 5	0x02	0x42	Estado = encendido	0x1B	0x5B
Unidad 6	0x12	0x52	Estado = apagado	0x1D	0x5D
Unidad 7	0x0A	0x4A	Petición de estado	0x1F	0x5F
Unidad 8	0x1A	0x5A			

Cuadro 4: Códigos del PowerLinc II

Toda comunicación al módulo desde el procesador debe comenzar con el envío de un byte con valor 0x02 que es el código de inicio y prepara al módulo para recibir instrucciones. Tras haber enviado este byte se envía un 0x36 que indica al módulo que se desea enviar un comando X10. Tras estos dos bytes se envía el código de casa, el de unidad y el del comando que se desean. Para terminar la orden, se envía un byte indicando cuántas veces se desea que sea enviado el mensaje sobre la red X10. La idea de enviar un comando más de una vez es que si el comando se recibe varias veces no existe ningún efecto negativo y se disminuye la posibilidad de que el mensaje se pierda en la red. Si un módulo recibe varias veces el comando de encender, por ejemplo, sin importar cuantas veces ejecute la orden al final siempre va a estar encendido y si por alguna razón se perdió una de las transmisiones en la línea las otras llevan el mensaje.

La recepción es similar, pero se apega más al protocolo X10. Los mensajes no llegan en la forma casa-unidad-comando como cuando se envían, sino que llega primero el mensaje casa-unidad y luego el casa-comando. Cuando el *PowerLinc II* recibe un mensaje por la red X10, envía dos mensajes por el puerto serial. Los dos mensajes son precedidos por un byte de inicio (0x58) para indicar al procesador que se va a enviar información y terminados por un byte indicando la cantidad de veces que el mensaje fue recibido y un retorno de carro (0x13). El cuerpo de ambos mensajes está compuesto por dos bytes y en ambos caso el primer byte indica la casa. El segundo byte es el que indica la unidad en el primer mensaje y el comando en el segundo.

En el programa del procesador utiliza dos rutinas para la comunicación X10, una de recepción y una de envío. La de envío es la más sencilla de las dos y lo único que hace es comprobar que la dirección a la que se desea enviar el comando sea válida, luego espera dos segundos antes de enviar el byte de inicio para darle tiempo al módulo a terminar si hay transmisiones en progreso y luego le envía los códigos necesarios para enviar la orden.

La rutina de recepción, al ser llamada, espera hasta recibir un byte de inicio de transmisión y lee el mensaje del puerto serial. Dependiendo si el mensaje contenía un código de unidad o de comando además del de casa, se guarda el comando o unidad en la variable que se pasó como parámetro a la función. Esta rutina se repite hasta que se reciba un mensaje que contenga un comando y entonces finaliza retornando sin error.

En la ilustración 3 se muestran los diagramas de estas rutinas.

D. LISTAS DE DATOS

Para almacenar las variables e información necesarias para cumplir con las funciones del programa como los estados de los módulos en la red X10 o las órdenes programadas para realizarse a cierta hora se necesita ubicar estos datos en memoria. Asignar un lugar de memoria fijo para guardarlos implica varios inconvenientes.

Primero, el espacio de memoria entero no está disponible para el uso de ninguna otra parte del programa incluso cuando no haya datos que guardar en el espació en ese momento. Por ejemplo, si se apartó espacio para guardar el estado de los 256 módulos posibles y la red sólo cuenta con 6 módulos conectados, el espacio para guardar la información de los otros 250 está desperdiciado. Segundo, el número máximo de datos está limitado desde el inicio y no puede variar. Por ejemplo, si sólo se dejó espacio para guardar un número fijo de órdenes programadas,

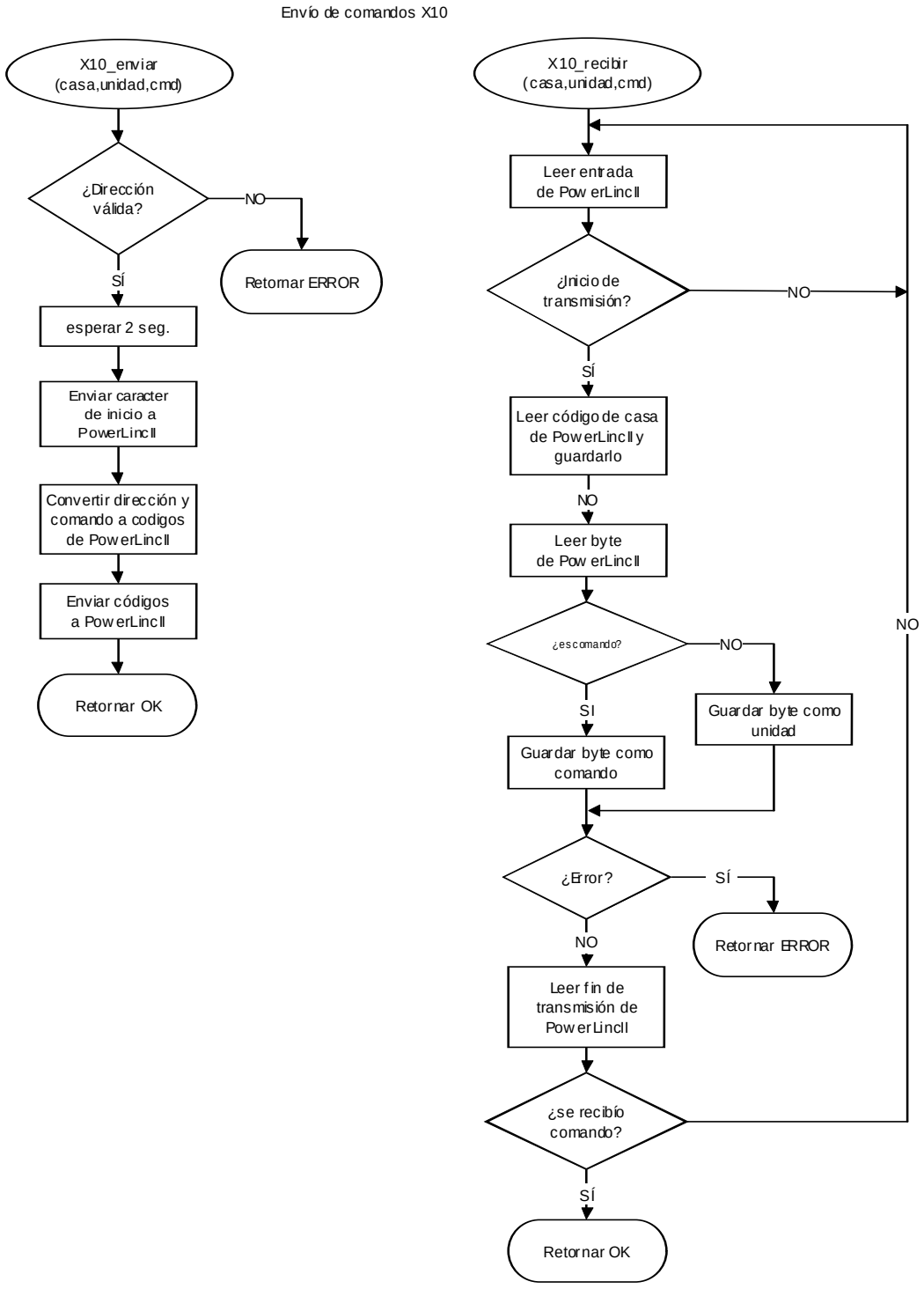


Ilustración 3: Diagramas de flujo de X10

el usuario no puede agregar más, incluso aunque haya memoria disponible. Y por último, si el tamaño de los datos está definido desde el principio se pueden dar dos problemas. Que una buena parte del espacio reservado para el dato esté siendo desperdiciado porque el tamaño del dato es menor que el espacio reservado o que el dato no quepa en el espacio que se había escogido como máximo. Por ejemplo si se quiere tener un espacio fijo en memoria para guardar las direcciones de correo de los usuarios autorizados, se debe definir el tamaño máximo que la dirección puede tener. Si este valor se escoge muy alto, se desperdicia mucha memoria con direcciones pequeñas y si se escoge muy pequeño, las direcciones muy grandes no cabrán en el espacio.

Para evitar todos estos problemas de espacio de memoria desperdiciado y de límites fijos de tamaños se creo una librería para manejar la memoria a través de listas dinámicas de enlace simple. El elemento principal de estas listas es el nodo que consta de un puntero a una cadena de caracteres llamado el puntero de datos y de un puntero a otro nodo, llamado puntero al siguiente.

Cada lista está definida por la dirección de un nodo que es la cabeza de la lista. Al estar vacía la lista, esta cabeza tiene ambos punteros apuntando a NULL. Cuando se introduce el primer dato en la lista, se reserva espacio en memoria para albergar exactamente la cantidad de información que se desea guardar y se hace que el puntero de datos apunte a esta memoria.

Para agregar el segundo elemento de información a la lista, se crea un nuevo nodo en una posición libre de memoria y se hace que el puntero al siguiente de la cabeza apunte a la dirección de este nuevo nodo y que el puntero al siguiente del nuevo nodo apunte a NULL. Nuevamente se vuelve a reservar memoria para la información y se hace que el puntero a datos del nuevo nodo apunte a esta información.

Para seguir agregando a la lista se vuelve a crear un nuevo nodo y espacio en memoria para los datos y se hace apuntar el puntero de datos del nodo a los datos y el puntero al siguiente del último nodo de la lista al nuevo nodo. El puntero al siguiente del nodo recién creado apunta a NULL.

La lista puede recorrerse entonces en su totalidad partiendo desde la cabeza y siguiendo los nodos apuntados por los punteros al siguiente de cada nodo hasta llegar al último nodo que apunta a NULL.

Para eliminar un dato de la lista, se recorre la lista hasta encontrar el dato. Entonces se

hace que el puntero al siguiente del nodo anterior apunte al nodo al que apunta el nodo que se quiere eliminar y se borra y libera el espacio en memoria reservado para la información y para el nodo que la contiene.

Además, para evitar datos duplicados, al agregar nuevos, la rutina revisa que no estén presentes antes de agregarlos.

En la Ilustración 4 se describe| el funcionamiento de las rutinas que manejan las listas.

E. BASE DE DATOS X10

Una de las partes del programa en las que se utilizan las listas es en el control de estados de módulos X10.

A pesar de que X10 es un protocolo de comunicación en doble vía que permite a cualquier modulo tanto enviar comandos como recibirlos e incluso pedir información, la realidad para los módulos comerciales es otra.

Cualquier módulo X10 es capaz de recibir comandos, pero son muy pocos los que están equipados para enviar comandos o respuestas. Esto se debe a que un módulo de doble vía es mucho más complejo y su costo es mucho más elevado. Al ser el bajo costo de automatización uno de los mayores atractivos de X10, los módulos de doble vía que son mucho más caros, son desplazados casi en su totalidad por los módulos unidireccionales. Esto lleva a que la parte del protocolo dedicada a petición de estados a los módulos pierda eficiencia ya que son muy pocos los módulos diseñados para responder a estas peticiones de información. Para poder vencer la barrera de la comunicación de doble vía y poder informar al usuario del estado de los módulos, el microcontrolador tiene que llevar él mismo el control del estado de cada módulo en la red. Para esto lo que hace es escuchar toda comunicación por la red X10 y llevar el control de cuales módulos han sido encendidos y cuales apagados.

Obviamente, el sistema sólo puede conocer el estado de los módulos que hayan sido encendidos o apagados mientras él ha estado escuchando en la red X10. El estado de los módulos que no hayan cambiado sigue siendo desconocido, al igual que su presencia en la red.

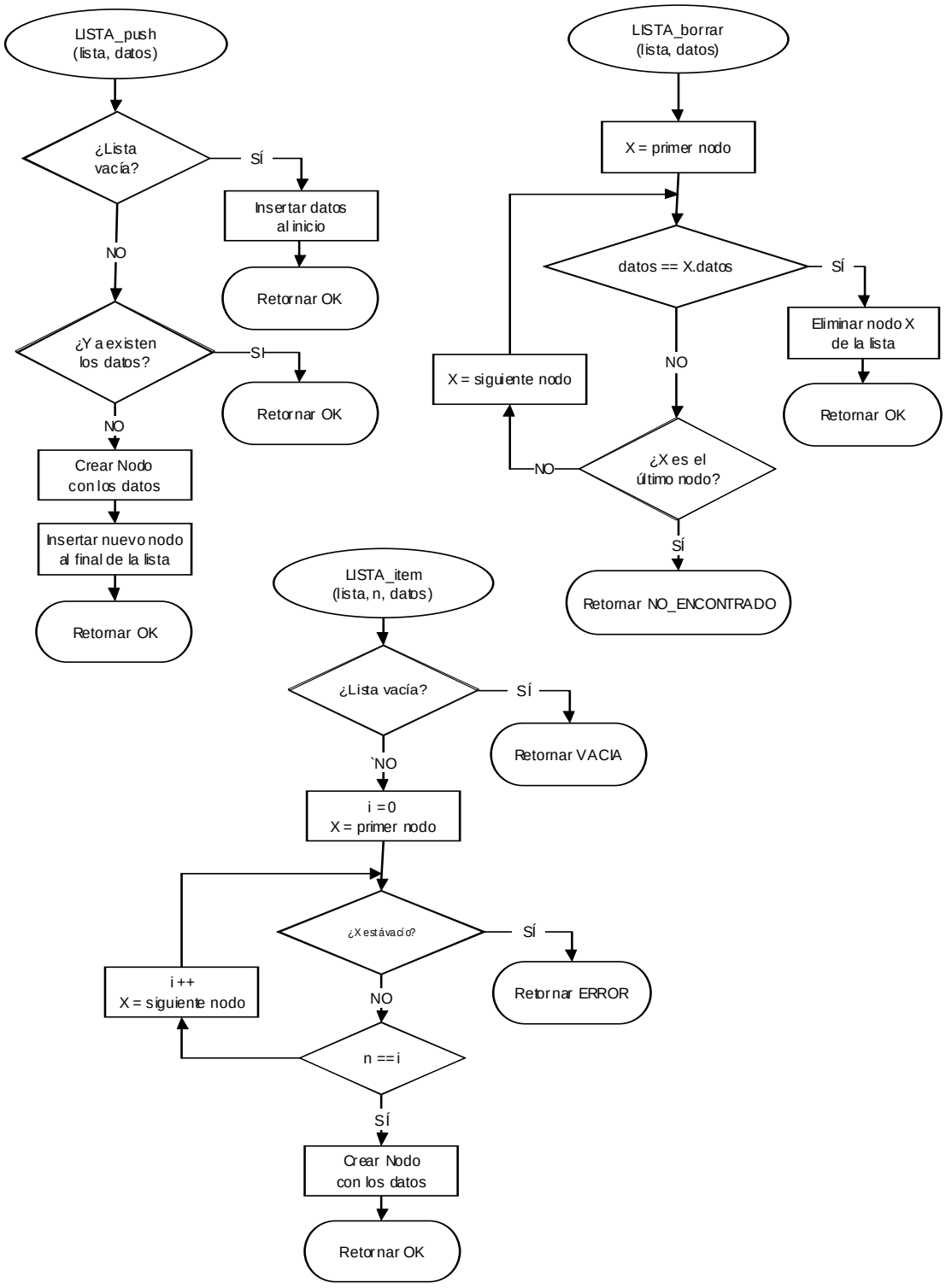


Ilustración 4: Diagramas de flujo de LISTAS

Para llevar este control sin necesidad de mantener una base de datos con información sobre los 256 módulos posibles, se utiliza una lista que contiene información de dirección y estado. Esta lista es actualizada cada vez que se recibe un comando X10.

Como hay comandos que pueden dirigirse a todas las unidades de una casa, cuando se recibe uno de estos comandos se busca en la lista todas las direcciones que pertenezcan a esta casa y se actualizan. A continuación se muestra el diagrama de flujo de la rutina de actualización de la base de datos.

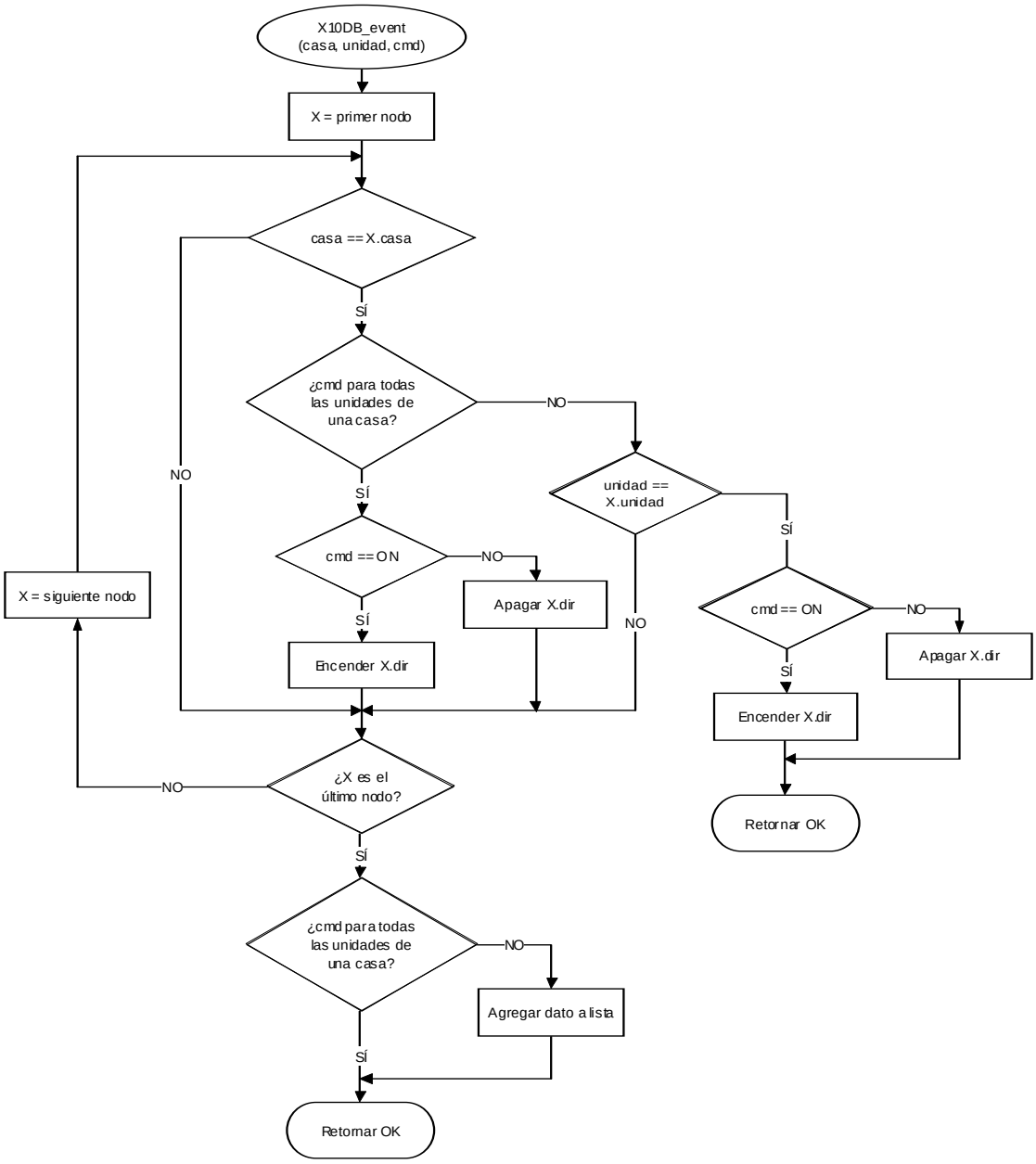


Ilustración 5: Diagrama de flujo de X10DB

F. NOTIFICACIÓN DE EVENTOS

Una parte importante de la funcionalidad del proyecto es poder informar de los sucesos dentro de la casa automatizada. Para esto, el sistema es capaz de enviar un correo electrónico, ya sea en HTML o en formato de texto indicando del cambio de estado de uno de los módulos X10.

Esta parte del proyecto es otra en la que se utiliza una lista dinámica. La utiliza para guardar la información de los eventos que deben ser notificados al usuario.

Un usuario X10 puede querer ser notificado cuando se encienda o se apague un módulo X10, pero no cuando cambie estado otro. Por ejemplo, recibir una notificación cada vez que se encienda una luz de mucho uso puede saturar al usuario reinformación que no necesita. Información del encendido o apagado de una luz con sensor de luz que se encienda todas las noches y se apague todas las mañanas es información probablemente innecesaria para el usuario, pero si se activa un sensor de gas indicando una fuga lo más probable es que éste si desee saberlo para poder actuar rápidamente e impedir cualquier accidente.

Para poder informar de los eventos importantes dentro de la red X10 sin notificar los sucesos que no interese, el sistema mantiene una lista creada y actualizada por el usuario de los eventos que a éste le interesan.

Cada elemento de la lista contiene información de la dirección del módulo del que se desea información, el evento, ya sea encendido o apagado, que se desea conocer y la dirección del correo electrónico del usuario que desea ser notificado. Al estar incluida esta información dentro de la lista se hace posible que cada uno de los usuarios del sistema pueda especificar de que sucesos le interesa ser avisado

La lista es recorrida cada vez que el sistema recibe un comando X10 y de coincidir el comando con alguno de los datos de la lista se envía un correo al usuario que lo solicitó.

A continuación los diagramas de las rutinas con las que se manipula la lista de avisos.

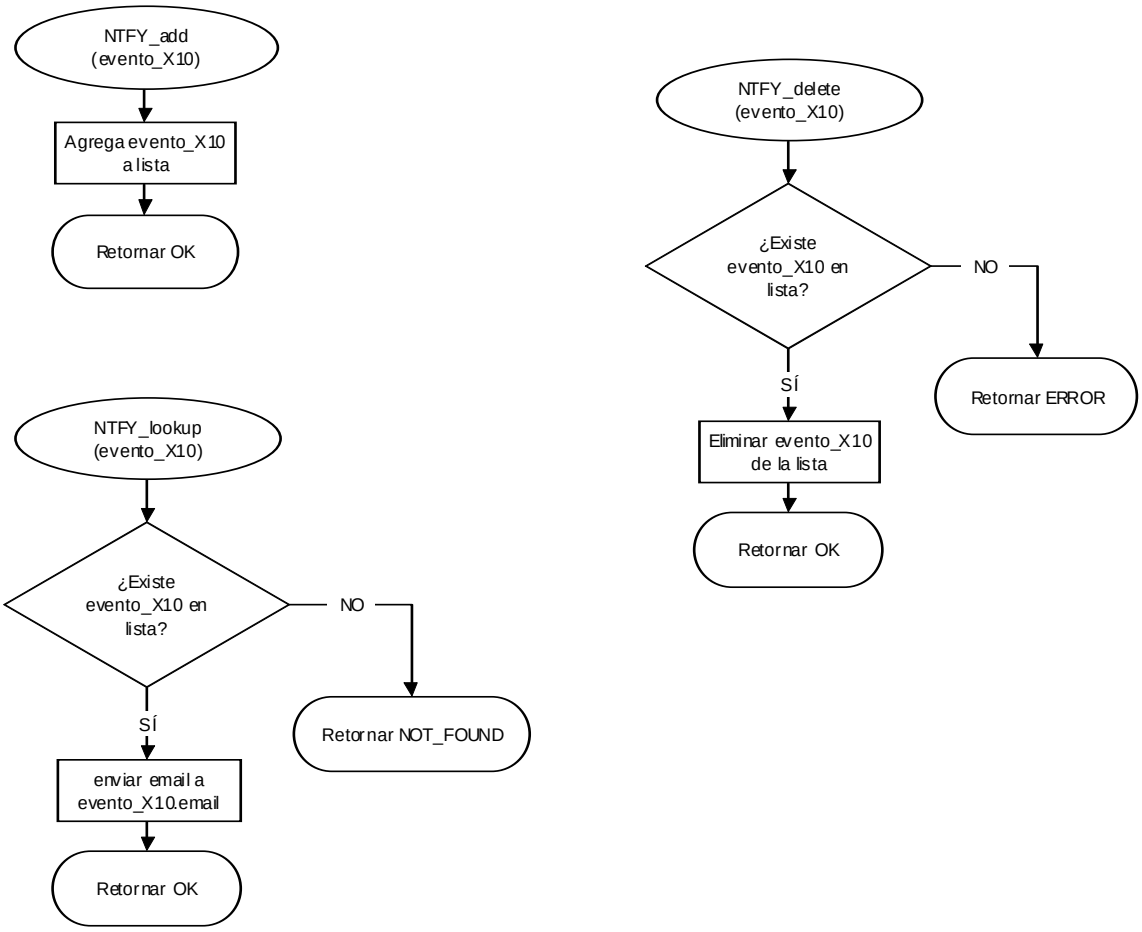


Ilustración 6: Diagrama de flujo de notificación de eventos

G. PROGRAMACIÓN DE COMANDOS

Para poder tener un buen control de la red de automatización tiene que ser posible actuar sobre los componentes de la red a través de órdenes, ser informado de los cambios que se dan y poder mantener un control del estado de todos los módulos integrantes de la red X10. Además, para poder incrementar el grado de control y automatización es también necesario poder programar órdenes para ser ejecutadas más adelante al tiempo deseado sin necesidad de que el usuario esté pendiente del sistema para que se realicen.

El programa del proyecto también es capaz de realizar esta función y para ello hace uso de otra lista dinámica de datos. Esta lista se utiliza para mantener un registro de acciones programadas por el usuario.

Cada elemento de la lista contiene información del tiempo en hora y minutos en el que debe ser efectuada la orden, el día de la semana en el que está programada, la dirección del módulo al que se debe enviar, el comando que se desea que se realice y si la orden debe ser repetida.

La combinación de todas estas variables permite mucha flexibilidad y funcionalidad en las órdenes programadas. Con la variable del día de la semana se puede programar no solo las órdenes para todo el día sino para toda la semana. Además, la variable que informa si una orden debe repetirse o no permite poder programar tanto acciones que se deseen por alguna razón aislada un día en particular como acciones repetitivas que se desea se efectúen diaria o semanalmente a una misma hora, como por ejemplo abrir la puerta del jardín para dejar salir al perro todas las mañanas.

Esta lista de datos es revisada cada minuto por el sistema en busca de órdenes programadas para ese momento y de ser encontradas se envían todos los mensajes necesarios por la red X10 para llevarlas a cabo.

Los diagramas de la ilustración 7 representan las rutinas encargadas de agregar y eliminar comandos de la lista y la encargada de revisar cada minuto la existencia de órdenes que necesiten efectuarse.

H. RELOJ DEL SISTEMA

Para poder efectuar las órdenes programadas en el momento indicado y para poder incluir en las notificaciones y mensajes al usuario el tiempo en el que fueron enviados es necesario mantener el reloj del sistema en tiempo.

La conexión a Internet del microprocesador brinda los medios necesarios para mantener la hora siempre actualizada incluso si hay fallas de alimentación y la batería del reloj de tiempo real se ha agotado. Existen distintos protocolos de la capa de aplicación diseñados para recuperar la hora actual de un servidor de tiempo para sincronizar relojes a través de la red entre los que se encuentran el Protocolo de Tiempo de Red (Network Time Protocol) NTP, el Protocolo Simple de Tiempo de Red (Simple Network Time Protocol) SNTP y el Protocolo de Tiempo (Time Protocol) TP. Los primeros dos protocolos son más modernos, pero más complejos. Requieren que el sistema solicitante envíe el tiempo que cree correcto y la respuesta incluye información del servidor y datos para eliminar errores de sincronización a causa del tiempo de viaje de paquetes.

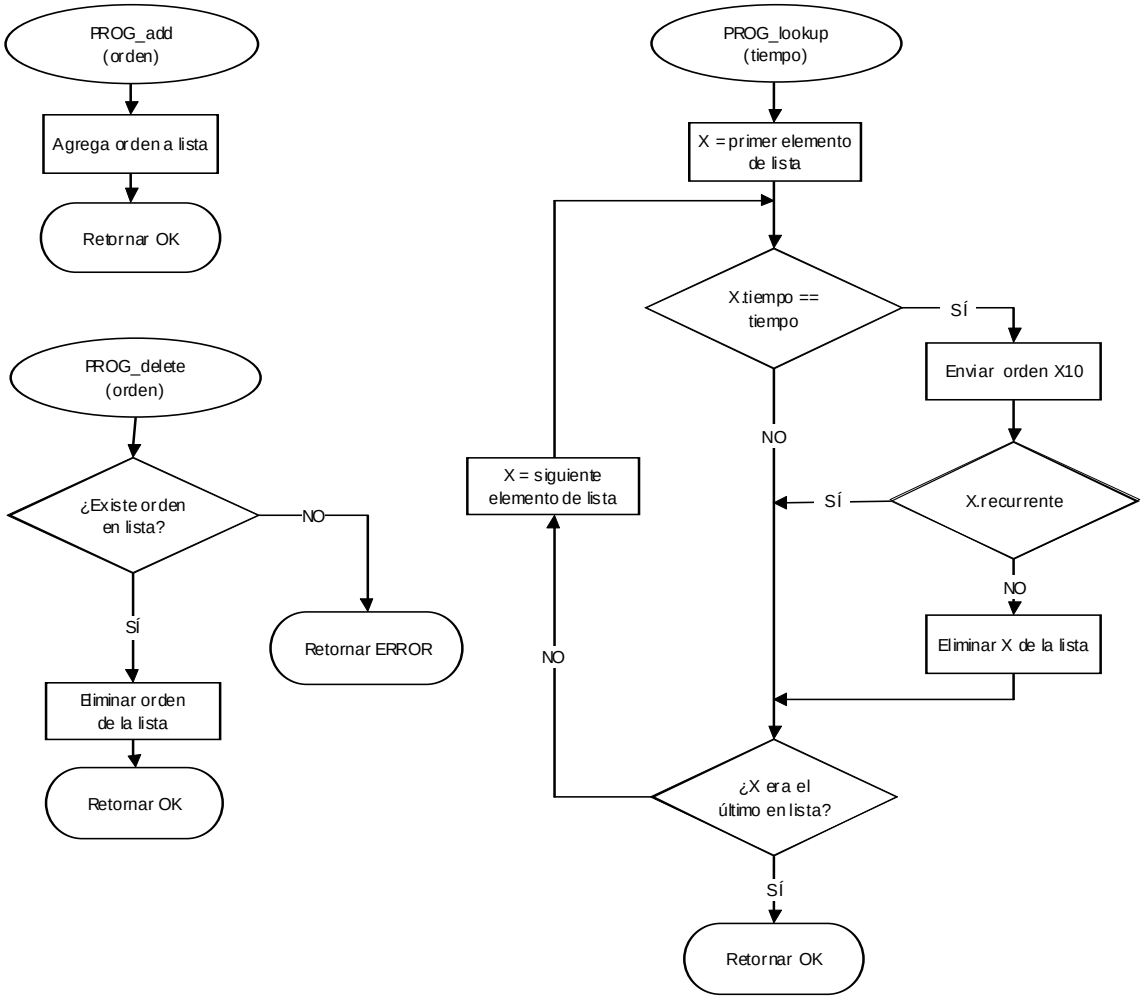


Ilustración 7: Diagramas de flujo de programar órdenes

Como para el sistema una exactitud mucho mayor a un minuto no es necesaria, las complejidades que presentan estos dos protocolos sobre el TP no son necesarias y se prefiere la simplicidad de éste último para la aplicación. Su funcionamiento es muy sencillo. El sistema solicitante envía un paquete vacío ya sea por TCP o por UDP al servidor de tiempo y este contesta con un paquete conteniendo solo 32 bits de datos y luego cierra la conexión. Este número de 32 bits es el número de segundos transcurridos desde medianoche del 1 de enero de 1900. La simplicidad de este protocolo permite que el tiempo pueda ser obtenido incluso en redes con cortafuegos altamente restringidos, además de no usar mucho tiempo de procesador en la rutina que lo implementa. El tiempo que se recibe es el del meridiano de Greenwich así que se le debe agregar la diferencia de zona hora en la que se encuentra el sistema y convertir luego a año, mes, día, hora y minutos.

La rutina que implementa este protocolo, que se corre a intervalos definidos de tiempo, es bastante sencilla. Lo único que hace es enviar el mensaje vacío y esperar la respuesta para convertirla y guardarla en el reloj de tiempo real si es una fecha válida. En la ilustración 8 se encuentra el diagrama.

I. LOCALIZACIÓN DENTRO DE LA LAN

Como ya se comentó antes, para poder localizar el módulo dentro de una red LAN de direcciones dinámicas asignadas por DHCP el proyecto utiliza el servicio de nombres de NetBIOS.

El servicio de nombres de NetBIOS es un protocolo bastante completo. Permite crear grupos de trabajo a través de nombres de grupo y además registrar uno o más nombres únicos a cada sistema para poder ser identificados en la red sin necesidad de direcciones numéricas difíciles de recordar.

El protocolo cumple dentro de una LAN más o menos la misma función que DNS en el Internet e incluso está diseñado para poder funcionar conjuntamente. La diferencia radica en que el servicio de nombres de NetBIOS (NBNS) no necesita un servidor centralizado sino que cada sistema es responsable de mantener una base de datos de los nombres activos en la red.

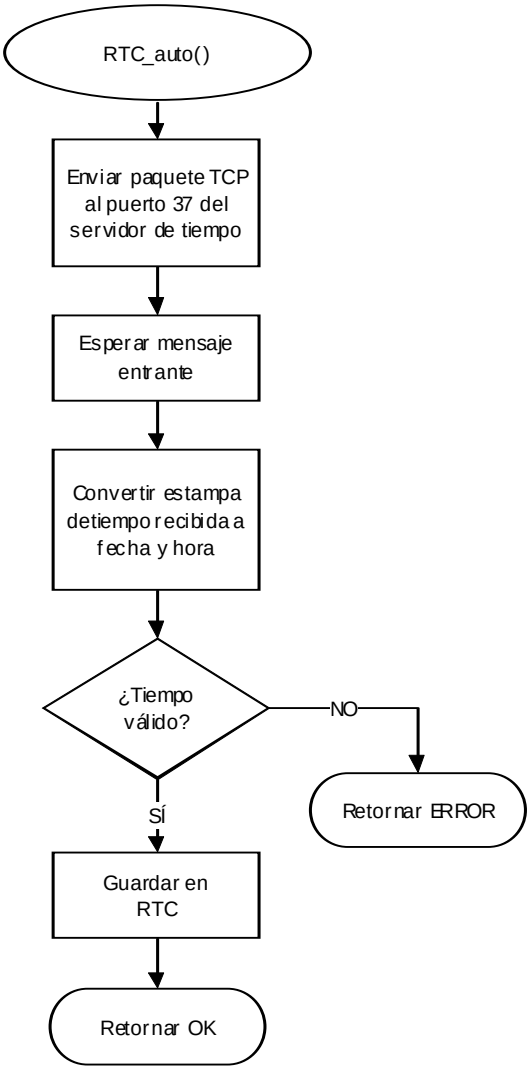


Ilustración 8: Diagrama de flujo del reloj del sistema

Para el proyecto no es necesaria ni deseada toda la funcionalidad del protocolo. No interesa por ejemplo mantener la base de datos de los sistemas de la red ya que en ningún momento el módulo necesita establecer comunicación con ninguno a menos que sea emitiendo respuestas. Además registrar el nombre del módulo (X10MAIL) dentro de la red tampoco es lo que se quiere ya que lo que interesa es que el módulo pueda ser encontrado por quien lo busca sabiendo que está ahí y no que cualquiera en la red pueda encontrarlo tan fácilmente.

La única parte del protocolo que interesa para los fines del proyecto es entonces la respuesta a una búsqueda de nombre emitida por otro sistema. Implementando este servicio, cuando un usuario, utilizando una computadora dentro de la misma LAN ingrese en el navegador el nombre del módulo, la computadora enviará un paquete de NetBIOS por la dirección de difusión de la LAN al puerto 137 preguntando de quien es el nombre. En ese momento, el módulo identifica que el nombre que se pregunta es el suyo y envía un paquete NetBIOS de respuesta a la dirección que lo solicito con información de la dirección IP del módulo y el tiempo de validez del nombre.

El diagrama de la ilustración 9 describe el funcionamiento de la rutina que maneja la respuesta de nombres de NetBIOS.

J. LENGUAJE DE USUARIO

El lenguaje de usuario es una de las partes más importante del sistema. Es a través de éste que los usuarios pueden comunicarse con la red de automatización para interactuar con ella a distancia.

Como se mencionó anteriormente en la sección de métodos de trabajo, una de las principales características que debe tener el lenguaje es que debe ser lo más parecido al lenguaje que utiliza normalmente el usuario con el resto del mundo que sea posible. Para lograrlo se debe ser muy flexible cuando se interpreten comandos tanto con las palabras que pueden utilizarse como en la formulación de la frase en las que se usan.

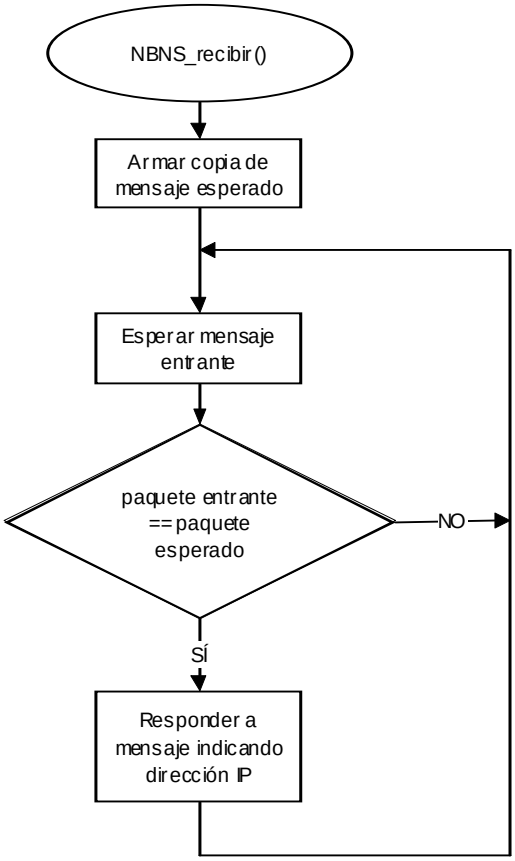


Ilustración 9: Diagrama de flujo de NetBIOS

La flexibilidad al interpretar palabras se logra a través de listas de sinónimos dentro del programa. Para cada palabra clave existe una lista que contiene todas las palabras que pueden ser utilizadas por el usuario para referirse a un concepto en particular. Por ejemplo, para encender un módulo el usuario puede utilizar cualquiera de las siguientes palabras: encender, enciende, encende, enciendelo, enciendela, encendelo, encendela, encienda, abrir, abre, abri, abrelo, abrela, abrilo, abrila, abra, activar, activa, activalo, activala, active más cualquiera de estas palabras con alguna tilde. Con esta variedad de palabras aceptadas se logra que el usuario pueda dirigirse al sistema de la forma en la que mejor se sienta ya sea tratándolo formalmente de usted, más familiarmente de tú o el vos tradicional guatemalteco o en tercera persona. Por ejemplo para encender una luz un usuario puede decir *encienda la luz*, otro puede decir *enciende la luz* o *encendé la luz* y en todos los casos el sistema es capaz de identificar el concepto de *encender*. Además al aceptar también otros verbos (abrir y activar) el usuario se puede dirigir a otro tipo de módulos de manera más natural. Por ejemplo, pedir que se abra una cortina diciendo *encender cortina* sería poco lógico e implicaría modificar la forma en la que se expresa el usuario, pero el programa reconoce la frase *abrir la cortina* que es la que utilizaría normalmente una persona. Con los verbos abrir y activar, se logra reconocer las expresiones habituales para referirse a la activación de módulos de puertas, ventanas, cortinas y similares y cualquier otro tipo de módulo que pueda *activarse* como sensores de temperatura, gas, humedad, nivel u otros.

En el archivo *sinonimos.c* en los apéndices se encuentran las listas de sinónimos para las palabras que forman las órdenes reconocidas por el sistema.

Otro aspecto importante para mantener la flexibilidad es lograr reconocer frases con el mismo significado pero formuladas de diferente manera y por esto el sistema no espera una secuencia fija en las palabras para cada orden reconocida. Por ejemplo, es reconocida de igual manera la frase *encender la luz a las 12:00PM* que la frase *a las 12:00PM, encender la luz*. Para lograr esto el programa interpreta una línea de texto a la vez y revisa cada una de las palabras dentro de la línea. Por cada palabra reconocida dentro de las listas de sinónimos o datos en un formato específico (hora o dirección X10) se enciende la bandera adecuada. Al finalizar de leerse la línea, se utiliza la combinación de banderas encendidas para identificar el comando al que se refería el usuario y entonces se llevan a cabo las acciones necesarias para que la orden se lleve a cabo.

Por supuesto, no toda combinación de banderas es válida. Por ejemplo, si al final de una línea está encendida tanto la bandera de *encender* como la de *apagar*, la frase no tiene sentido para el programa y dentro del correo de confirmación que se envía al usuario tras interpretar todo el

correo entrante se incluye una línea informando que se recibió una orden no reconocida.

Otro aspecto que brinda flexibilidad al lenguaje de usuario del sistema es que solo las palabras clave de la oración entran en la interpretación, que son los verbos y adjetivos reconocidos por las listas de sinónimos y los datos que tienen sentido para el sistema. De esta manera se interpreta con el mismo significado una frase sencilla como *encender la luz* y frases más elaboradas como *Quiero que se encienda la luz ahora* o *Por favor encender la luz*.

Los tipos de datos que puede reconocer el sistema, aparte de los que se encuentran en las listas de sinónimos son direcciones de la red X10 y la hora. Las direcciones están dadas igual que se conocen en X10 por un código de casa y uno de unidad. Ejemplos de direcciones reconocidas son A1, b2 M10 y P16 en donde no importa si el código de casa está en mayúscula o minúscula. El formato de hora que se reconoce es *H:M* en donde *H* es un número de uno o dos dígitos entre 0 y 23 que representa la hora y *M* es un número que indica los minutos. Además, la hora puede ir seguida por un *am* o *pm* que indica si las horas bajo 12 son antes o después de mediodía. Si ninguno de los dos está presente se asume que la hora está en formato de 24 horas y cualquier hora menor de 12 se interpreta como antes de mediodía.

A pesar de que las direcciones X10 son normalmente también las que se utilizan para que los usuarios de la red se refieran a los módulos dentro de ella, tener que referirse a la lámpara de comedor como A3 es muy poco intuitivo e implica que el usuario se aprenda las direcciones de cada módulo que desee programar. Para eliminar este inconveniente, el proyecto utiliza una lista de alias o sobrenombres en la que se guardan nombres asociados a una dirección. Los nombres son cadenas de caracteres alfanuméricos (incluyendo el guión bajo _) con un largo mínimo de cuatro caracteres. Así se puede asociar la palabra *comedor* con la dirección A3 y utilizarla en las órdenes en lugar de a dirección. Una vez registrado el sobrenombre es posible por ejemplo enviar la frase *encender la luz del comedor* para encender el módulo A3.

El comando para registrar sobrenombres es de los comandos menos flexibles. Su forma debe ser más o menos *[dirección X10] se llama [sobrenombre]*, *El alias de [dirección X10] es [sobrenombre]* o *El nombre de [dirección X10] es [sobrenombre]*. Puede contener más palabras dentro de la frase, pero se debe tomar en cuenta que la palabra que se tomará como sobrenombre es la última palabra dentro de la oración que sea mayor de tres caracteres que no sea sinónimo válido ni esté dentro de la lista de palabras ignoradas o sea un alias ya existente. Para asignar un alias ya existente la única forma de hacerlo es borrando primero el sobrenombre y luego asignándoselo a la nueva dirección. Para eliminar el alias se pueden enviar comandos como *Eliminar el alias*

[sobrenombre], *[sobrenombre] ya no se llama así* o *Quitar el nombre de [dirección X10]*.

Los tipos de comandos que se pueden enviar por correo electrónico, además de los de sobrenombres y encendido son de apagado, para pedir que se avise de un evento o que ya no se avise, para programar acciones más adelante o eliminarlas, para preguntar el estado de los módulos dentro de la red y para cambiar el formato de respuesta para los siguientes correos electrónicos de confirmación y avisos que envíe el sistema. La siguiente tabla presenta estos comandos, con un ejemplo de una de las posibles formas de expresarlo asumiendo *comedor* como el sobrenombre del módulo.

Orden	Ejemplo de frase que la expresa
Agregar sobrenombre	A1 se llama comedor.
Eliminar sobrenombre	El comedor ya no se llama así.
Encender un módulo	Encender la luz de comedor.
Apagar un módulo	Quiero que se apague la luz del comedor.
Pedir notificación de evento	Avisame cuando se apague la luz del comedor.
Cancelar la notificación de eventos	Ya avisar cuando se apague el comedor.
Programar una acción	Encender el comedor a las 5:00 PM el lunes.
Eliminar una programación	No encender el comedor a las 5:00 PM el lunes.
Programar una acción recurrente	Apagar la luz del comedor los lunes a las 6:00 PM. Apagar la luz del comedor siempre a las 18:00.
Eliminar una acción recurrente	No apagar la luz del comedor los lunes a las 18:00.
Preguntar el estado de los módulos	Qué módulo está encendido?
Pedir formato HTML para correos siguientes	Usar formato HTML.
Pedir formato de texto para correos siguientes	Quiero recibir correos en formato de texto.

Cuadro 5: Ejemplos de órdenes vía correo electrónico

El diagrama de la ilustración 10 describe las rutinas utilizadas para interpretar los correos electrónicos

K. INTERFASE HTML

La interfase HTML es la que utiliza el usuario para configurar los parámetros necesarios para el funcionamiento del módulo como dirección IP, servidores de correo entrante y saliente, lista de usuarios permitidos y servidor de tiempo para el ajuste de la fecha y hora.

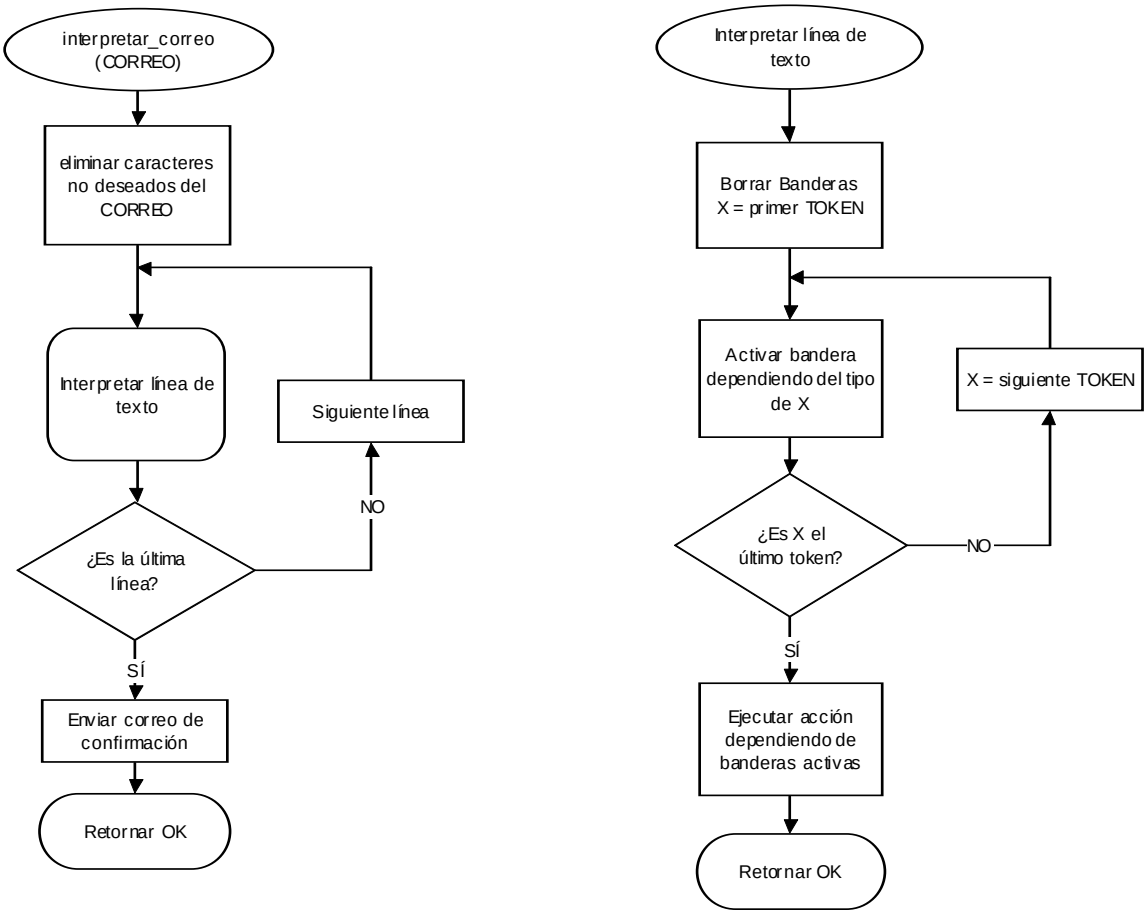


Ilustración 10: Diagramas de flujo de intérprete de correos

Las páginas deben ser sobre todo funcionales, pero también es importante una presentación agradable para el usuario ya que junto a los correos de confirmación y notificación, estas páginas serán la imagen que el usuario tendrá del sistema y es tan importante como su funcionamiento.

El sistema, a pesar de tener suficiente memoria para el resto de partes el proyecto, tiene un espacio sumamente limitado si se pretende utilizarlo para el almacenamiento de imágenes, por lo que el número y tamaño de éstas se mantuvo al límite durante el diseño de las páginas.

Para poder presentar una imagen agradable sin la utilización de muchas imágenes, que suele ser la estrategia más utilizada, se recurrió a Hojas de Estilo en Cascada (Cascading Style Sheets) CSS. Estas permiten definir el estilo de párrafos, fuentes y demás parámetros de la apariencia y posicionamiento de los elementos de todo el sitio desde un solo archivo al que se pueden vincular todas las páginas, reduciendo así el espacio necesario para formato de páginas y dando una apariencia uniforme a todas. Además JavaScript fue también utilizado en el desarrollo de las páginas para mejorar el aspecto y funcionamiento.

La página consiste principalmente de un menú con el que se accede a diferentes páginas cada cual para cambiar un grupo de parámetros del sistema. Estas páginas son formas HTML creadas por CGI dentro del módulo del sistema para presentar la información actual de los parámetros y se reciben a través del método POST de HTTP con los datos editados para poder ser verificados y modificados.

L. ORGANIZACIÓN DEL SISTEMA

En la organización del sistema es donde se aprecian realmente las ventajas del sistema operativo *XINU*. Al ser un sistema multitarea, todas las distintas tareas pueden ser ejecutadas prácticamente simultáneamente sin interferir entre sí. Además, el uso de semáforos permite tanto sincronizar distintos procesos entre sí como administrar el acceso a variables utilizadas por más de un proceso. Por último, los puertos de mensajes o tuberías (pipes) facilitan la comunicación entre distintos procesos.

1. HILOS DE EJECUCIÓN

Todas las distintas tareas del programa están organizadas en hilos de ejecución que el OS se encarga de controlar. La ejecución se administra asignando pequeños tiempos de ejecución

al hilo de turno y después de ese tiempo se cede el procesador al hilo que esté listo para ejecutarse que tenga la mayor prioridad. Si existen varios hilos con el mismo nivel de prioridad esperando a ser ejecutados y no hay ningún hilo de mayor prioridad esperando, los hilos van alternando entre sí el uso del procesador para ejecutarse paralelamente. Si sólo existe un hilo esperando en el nivel de prioridad más alto, al final de su tiempo de ejecución volverá a ser seleccionado y será ejecutado hasta que ya no necesite el procesador o un proceso de mayor prioridad solicite tiempo de ejecución. Los hilos de ejecución dejan de necesitar tiempo de procesador cuando hayan sido bloqueados por la llamada a una función o cuando sean suspendidos por otro proceso o por si mismos.

El bloqueo de procesos se da por lo general cuando se llama a funciones de entrada que necesitan esperar datos de otra fuente para poder retornarlos al proceso solicitante. Por ejemplo, una llamada de lectura del puerto serial retorna inmediatamente si hay datos esperando en el buffer de entrada, pero si está vacío el proceso que llamó la función queda suspendido, sin acceso al procesador, hasta que haya datos disponibles. También se puede bloquear un proceso al leer puertos de mensajes vacíos o llamando a la función *sleep(segundos)* que suspende el proceso el número de segundos indicados al llamar la función. Además los procesos pueden suspenderse y despertarse entre si.

La función de la rutina principal del programa es el de inicializar todas las variables e iniciar los hilos de ejecución. El programa cuenta con 7 procesos principales que ejecutan cada uno una función determinada, READ_MAIL, INTERPRETE, X10_INPUT, X10_OUTPUT, RTC, NBNS y PROG, de los cuales todos tienen sus funciones contenidas dentro de ciclos infinitos y todos menos INERPRETE son iniciados al principio.

El proceso READ_MAIL se encarga de revisar cada 10 segundos si hay correos electrónicos válidos nuevos en el servidor y de guardarlos en memoria para su interpretación. Si hay un correo nuevo, READ_MAIL despierta a INTERPRETE y se duerme a sí mismo. INTERPRETE es entonces ejecutado y lleva a cabo todas las órdenes indicadas por el usuario en el correo además de enviar una confirmación indicando que el correo se recibió y que acciones se reconocieron dentro de él. Una vez que INERPRETE ha terminado de ejecutar sus funciones, despierta a READ_MAIL y se duerme a sí mismo. El resultado es prácticamente el mismo que si se hubiera llamado una función dentro del proceso READ_MAIL con la diferencia que de esta manera INTERPRETE corre en su propio espacio de memoria lo que lo hace más eficiente ya que es el proceso más complejo de todo el sistema.

X10_INPUT tiene un funcionamiento sumamente sencillo. Al inicio hace una llamada a la función que lee la entrada de la red X10 y se bloquea hasta que se haya recibido un comando. En este momento actualiza la base de datos de estados y manda una notificación a los usuarios que hayan solicitado ser informados de ese evento. Una vez terminado retorna al principio y vuelve a llamar a la función de entrada X10 donde se bloquea hasta el siguiente comando.

X10_OUTPUT funciona de manera similar, pero se bloquea con un llamado a un puerto de mensajes que se explicará más adelante.

RTC y PROG son procesos que tras ejecutar sus funciones (actualizar el reloj y revisar si hay acciones programadas para ese momento respectivamente) se bloquean a sí mismas durante un tiempo antes de regresar al inicio y repetir el proceso resultando en funciones de ejecución periódica.

Por último, NBNS se bloquea esperando un paquete entrante de NetBIOS y tras procesarlo vuelve al inicio para esperar el siguiente proceso.

2. SEMÁFOROS

Los semáforos son variables y el sistema los utiliza como contadores para permitir la sincronía de pedazos de código. Básicamente un semáforo es un entero cuyo valor se incrementa con la llamada a la función *signal (semáforo)* y disminuye al llamar a la función *wait (semáforo)*. Cada vez que se llama a la función *wait* se disminuye en uno el valor del semáforo y si el valor resultante es mayor de cero; la función retorna y el proceso continúa su operación. Si el valor resultante es menor de cero; sin embargo, la función se bloquea y reanuda su ejecución hasta que se hayan emitido tantas llamadas a la función *signal* como el valor absoluto del número negativo resultante de la disminución. De esta manera se pueden implementar un tipo de colas de acceso entre procesos ya que si varios procesos se bloquean llamando a *wait* cada uno será reiniciado en el orden en el que fueron bloqueados.

El programa utiliza un semáforo para la sincronización de envíos de emails. En el caso en el que suceda un evento X10 que debe ser notificado mientras se está preparando un correo de confirmación, el email de notificación debe esperar ya que debido al tamaño sólo se desea un espacio en memora para envío de correos. Como se logra la sincronización es al crear un semáforo que permita solamente una llamada sin bloqueo y se llama a la función *wait* al inicio de SMTP_reset que es la que se encarga de inicializar la memoria para el envío y es la primera que debe llamarse para enviar un correo. El semáforo no se libera hasta que el email ha sido enviado y

así, si la memoria está en uso por otra parte del programa cuando se desea mandar un email, el proceso se bloquea hasta que la memoria pueda ser usada.

3. PUERTOS DE MENSAJES

Otro elemento del sistema operativo utilizado para la sincronización de procesos son las tuberías o puertos de mensajes que son colas de datos administradas por el sistema en las que el primer dato que es escrito al puerto es el primero que es leído. El tamaño de datos que utiliza XINU para los puertos es de 24 bits que es lo que ocupa un puntero.

Al crearse un puerto se especifica el tamaño de la cola. Siempre y cuando el número de mensajes enviados al puerto que no hayan sido leídos todavía sea menor que el tamaño de la cola la llamada de escritura al puerto no bloqueará la función. Pero si la cola está llena, una llamada de escritura bloqueará el proceso que la hizo.

En el programa se utiliza un puerto de mensajes para el envío de mensajes X10. Los mensajes X10 pueden ser enviados de distintas partes del proyecto, como son el intérprete de correos o el proceso que controla los comandos programados. A pesar de que el problema de acceso simultáneo al puerto de salida X10 podría ser solucionado con otro semáforo, al requerir la transmisión de los comandos mucho tiempo en relación a la velocidad de ejecución desprogramadle programa, el tiempo requerido para interpretar un correo que solicite el envío de varios comandos X10 se vería incrementado considerablemente. Al utilizar un puerto de mensajes para recibir los comandos que quieran ser enviados se evita este problema. Con el puerto de mensaje cualquier parte del programa puede enviar los comandos y siempre y cuando el tráfico no sea excesivo estos comandos quedarán en cola esperando a ser enviados sin necesidad de que el proceso que lo envió se detenga a esperar a que sea enviado.

4. DIAGRAMAS DE FLUJO

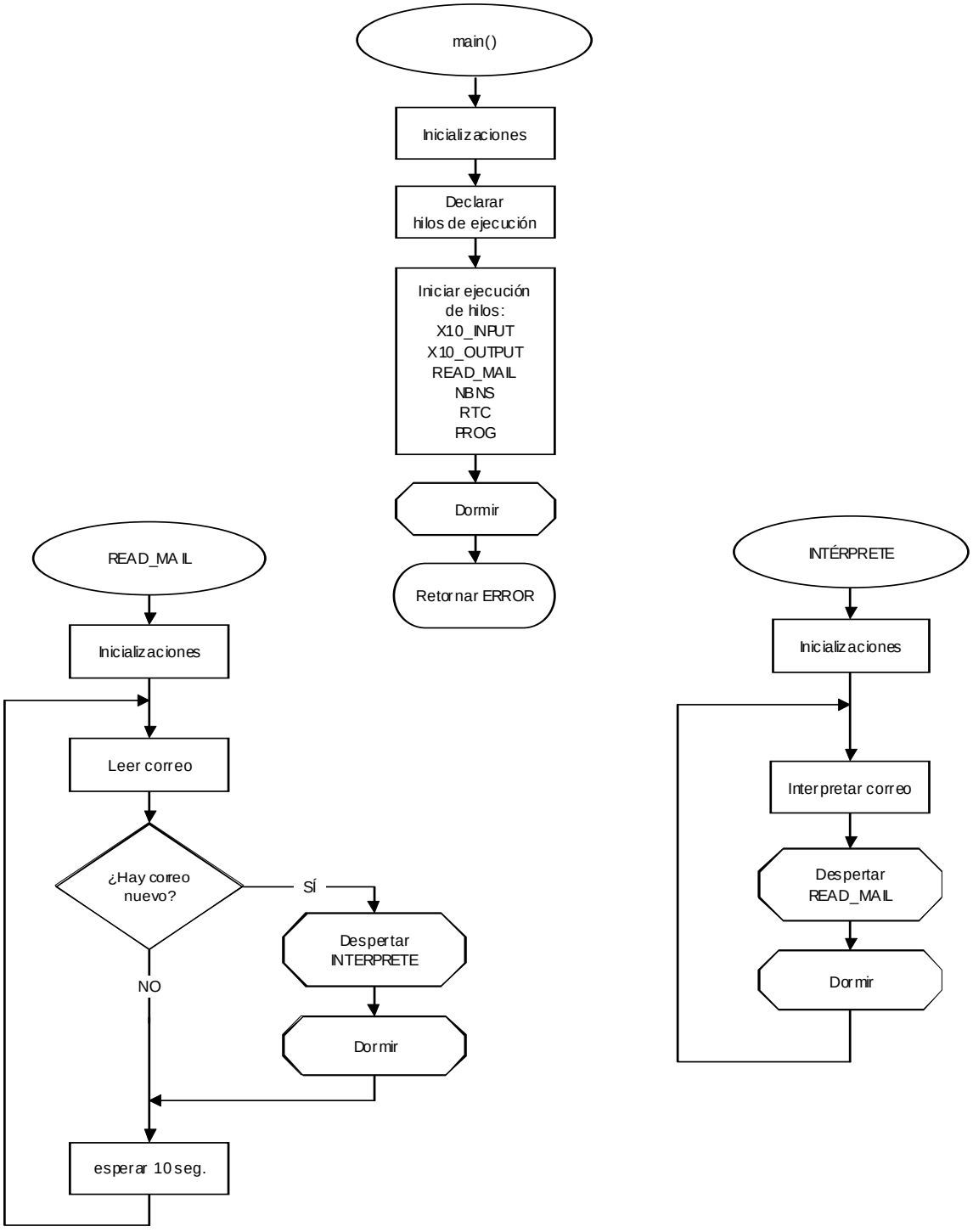


Ilustración 11: Diagrama de flujo de hilos de ejecución (1)

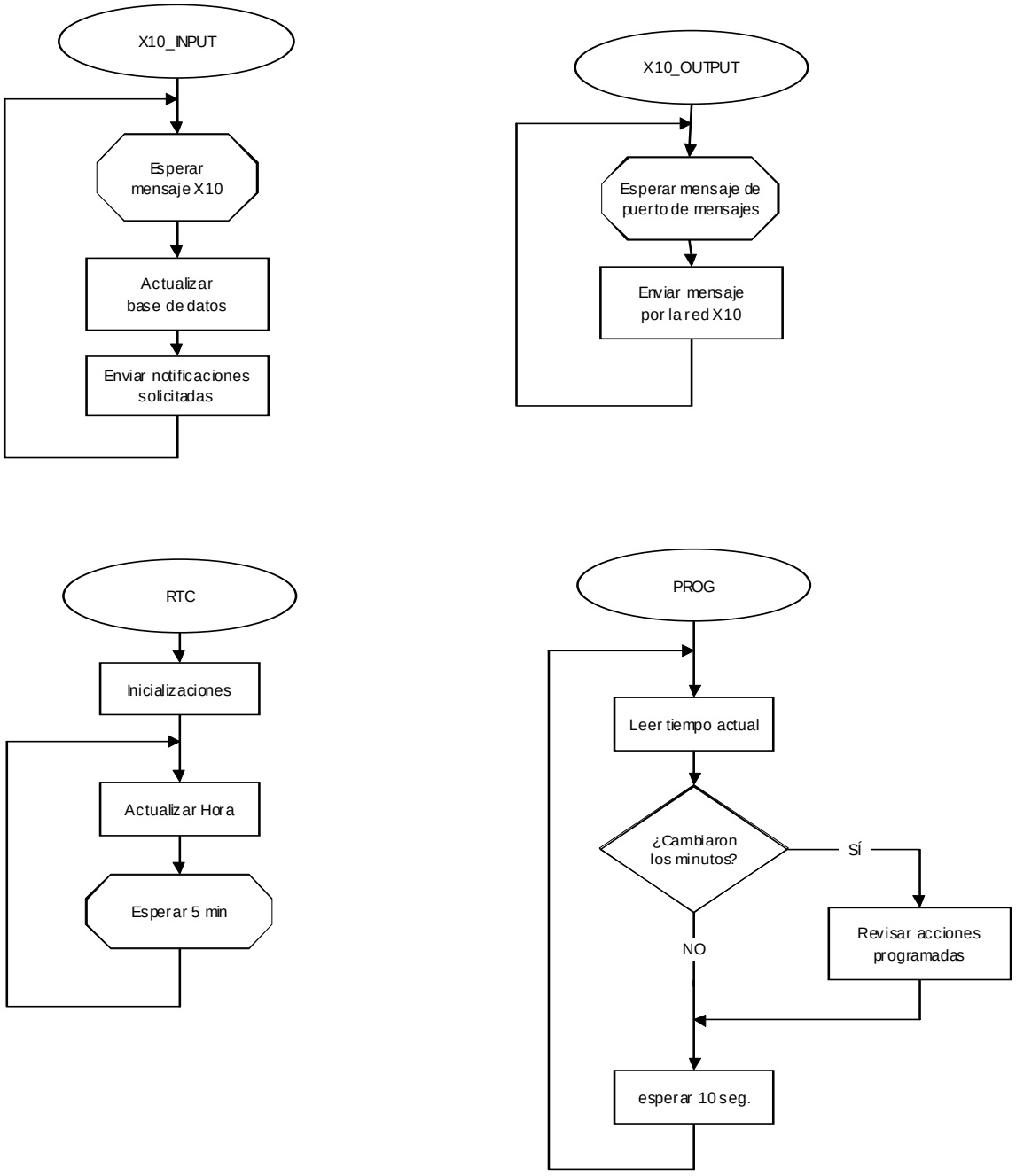


Ilustración 12: Diagrama de flujo de hilos de ejecución (2)

M. CIRCUITO

El circuito del sistema es sumamente sencillo. Su función es la de crear una interfase de acople entre el puerto serial TTL del microcontrolador y el RS232 del *PowerLinc II*. Además es el encargado de regular la alimentación de 3.3 voltios para el microcontrolador y la de 5 voltios para el circuito de acople del puerto serial.

Para la parte del la interfase entre los puertos seriales se utilizó un MAX232 que a través de bombas de carga con capacitores logra generar los voltajes necesarios para el puerto RS232 a partir de un voltaje de 5 voltios. El MAX232 tiene cuatro acoples de voltaje, dos en una dirección y dos en otra. Como solo se utiliza un puerto serial sin líneas de control de flujo, sólo dos de esos acoples son necesarios, uno en cada dirección, para la línea de transmisión y para la de recepción.

La parte de la alimentación de voltaje se llevó a cabo en base a reguladores de voltaje: un regulador variable LM317 ajustado para la salida de 3.3 V y un 7805 para la fuente de 5 V, capaces cada uno de entregar 1 A son más que suficientes para cubrir la demanda de corriente del microcontrolador y del MAX232 que no sube de 300 mA en ninguno de los dos casos. La alimentación de todo el circuito se toma de una fuente de voltaje de 12 voltios no regulados ubicada dentro del *PoweLinc II* cuya salida está incluso disponible en su conector RJ-45.

En los anexos se encuentra el diagrama del circuito con los reguladores de voltaje, el MAX232, un diodo emisor de luz indicador de encendido y los conectores de interfase del módulo del microcontrolador.

Para contener el circuito se diseño una tarjeta de cobre de tamaño similar al del módulo del microprocesador que queda directamente debajo de éste y utiliza un conector RJ-11 con cuatro contactos para la conexión con el puerto serial y la fuente de alimentación del *PowerLinc II*.

En la sección de anexos se encuentra una imagen ampliada de la tarjeta impresa cuyo tamaño real es de 7 cm de ancho por 9 cm de largo.

V. RESULTADOS OBTENIDOS

Una vez terminado el trabajo se cumplieron todos los resultados esperados y se obtuvo un módulo capaz de controlar la red de automatización a distancia como se pretendía.

El módulo es compacto, como se pretendía, con unas dimensiones de 7 cm de ancho por 10 cm de largo y una altura total de 3.5 cm aproximadamente.

Pudo conectarse a Internet por medio del controlador MAC del módulo del *eZ80F91* y la pila TCP/IP del OS XINU.

Es completamente independiente de otros sistemas, aparte de los servidores de correo, para su funcionamiento y solo es necesario un cliente HTTP en algún ordenador de la LAN para configurarlo como era el objetivo.

A través de la red puede enviar y recibir los comandos necesarios POP3 para la descarga de correos electrónicos y puede decidir si acepta el correo como válido o no de acuerdo a la dirección del remitente.

Además es capaz de interpretar los comandos recibidos a través del correo electrónico para ejecutarlos y reaccionar a ellos. Para ello utiliza un lenguaje que fue creado pensando no sólo en la funcionalidad que brindaría sino en la facilidad de uso del usuario, llegando al final a un lenguaje que es prácticamente igual al lenguaje natural que una persona utilizaría para emitir las órdenes a otra persona disminuyendo así al mínimo el esfuerzo del usuario a la hora de aprender a usar el sistema.

También se logró establecer la conexión con la red X10 como se quería y controlarla incluso a un grado mayor al que se pretendía puesto que las órdenes pueden ser programadas para ejecutarse más tarde además de en el momento en el que se reciben. También se logró mantener un registro del estado de la red para mantener informado al usuario de ésto además de los eventos que suceden en la red.

La comunicación por correo electrónico es de dos vías como se pretendía gracias al cliente SMTP del sistema que se utiliza para notificaciones de eventos y confirmación de comandos recibidos.

Se consiguió también crear las páginas HTML y servirlas a través de HTTP para la

configuración del módulo.

Como funcionalidad extra se logró que el sistema mantenga una hora actualizada en todo momento sin la intervención o ajustes periódicos del usuario a través del Protocolo de Tiempo TP y que no sea necesario para el usuario conocer la dirección IP del módulo dentro de una red de direcciones dinámicas DHCP al implementar parte del servicio de nombres de NetBIOS para poder referirse al módulo a través de un nombre en lugar de una dirección.

El proyecto logró entonces llevarse al punto deseado y cumplir con todos los requisitos que se pretendían con funcionalidad más que satisfactoria.

VI. CONCLUSIONES

- Se logro construir el módulo compacto deseado con dimensiones de 7 x 10 x 3.5 centímetros.
- El módulo resultante depende de una computadora en la red de área local solamente para su configuración a través de http.
- Es capaz de conectarse a Internet para recibir órdenes y enviar información.
- El lenguaje comprendido por el sistema en los correos es prácticamente igual al lenguaje natural del usuario.
- Como era la intención principal del proyecto, permite el control remoto de una red de automatización sin necesidad de una dirección IP pública.

VII. RECOMENDACIONES

- Como mejora del proyecto podría agregarse alguna forma de autenticación extra al recibir correos para hacer el sistema más seguro contra acceso no autorizado.

- También debería agregarse mayor seguridad para el acceso al módulo a través de la red de área local.

- El proyecto también podría mejorar el tiempo de respuesta a los comandos del usuario agregando como medio de comunicación, IRC, ICQ o algún otro protocolo de mensajería instantánea. Este nuevo medio no debería sustituir al correo electrónico sino complementarlo ya que la mensajería instantánea necesita de un cliente instalado en la computadora desde donde se quiere comunicarse con el módulo. Esto, además de que la mensajería instantánea está bastante restringida en muchas redes públicas, hace que la mejora del tiempo de respuesta tenga el costo de no ser accesible desde cualquier computadora con acceso a Internet como el correo electrónico. Al coexistir los dos medios de comunicación se conseguiría tanto acceso desde cualquier computadora como velocidad de respuesta.

VIII. BIBLIOGRAFÍA

- Comer, Douglas E.; David L. Stevens. 1995. *Internetworking with TCP/IP: volume II, design implementation and internals.* 3 ed. New Jersey, Prentice may. 660 págs.

- Loshin, Pete. *Essential Email Standards*. 2000. New Cork, Wiley Computer Publishing. 339 págs.

- Postel, Jonathan B. *RFC821 Simple Mail Transfer Protocol.* 1982. California, Information Sciences Institute of the University of Southern California. 68 págs.

- *PS019306-0803 eZ80F91 Module Product Specification*. 2003. ZiLOG. San José, California. 33 págs.

- *PS019208-1003 eZ80F91 Product Specification*. 2003. ZiLOG. San José, California. 390 págs.

- *RM000802-1203 ZiLOG TCP/IP Software Suite Programmer's Guide Reference Manual*. 2003. ZiLOG. San José, California. 412 págs.

- *RFC1001 Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods.* 1987. Network Working Group. 68 págs.

- *RFC1002 Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specification.* 1987. Network Working Group. 85 págs.

- *RFC1939 Post Office Protocol - Version 3.* 1992. Network Working Group. 23 págs.

- *RFC868 Time Protocol.* 1983. Network Working Group. 2 págs.

- Rosenfeld, Louis; Peter Morville. 2000. *Arquitectura de la información para el WWW.* México, D.F., Mc Graw Hill. 204 págs.

- Schank, Jeffrey D. 1994. *Novell's Guide to Client-Server Applications and Architecture.* Canadá, Novell Press. 454 págs.

IX. APÉNDICES

A. DIAGRAMA DEL CIRCUITO

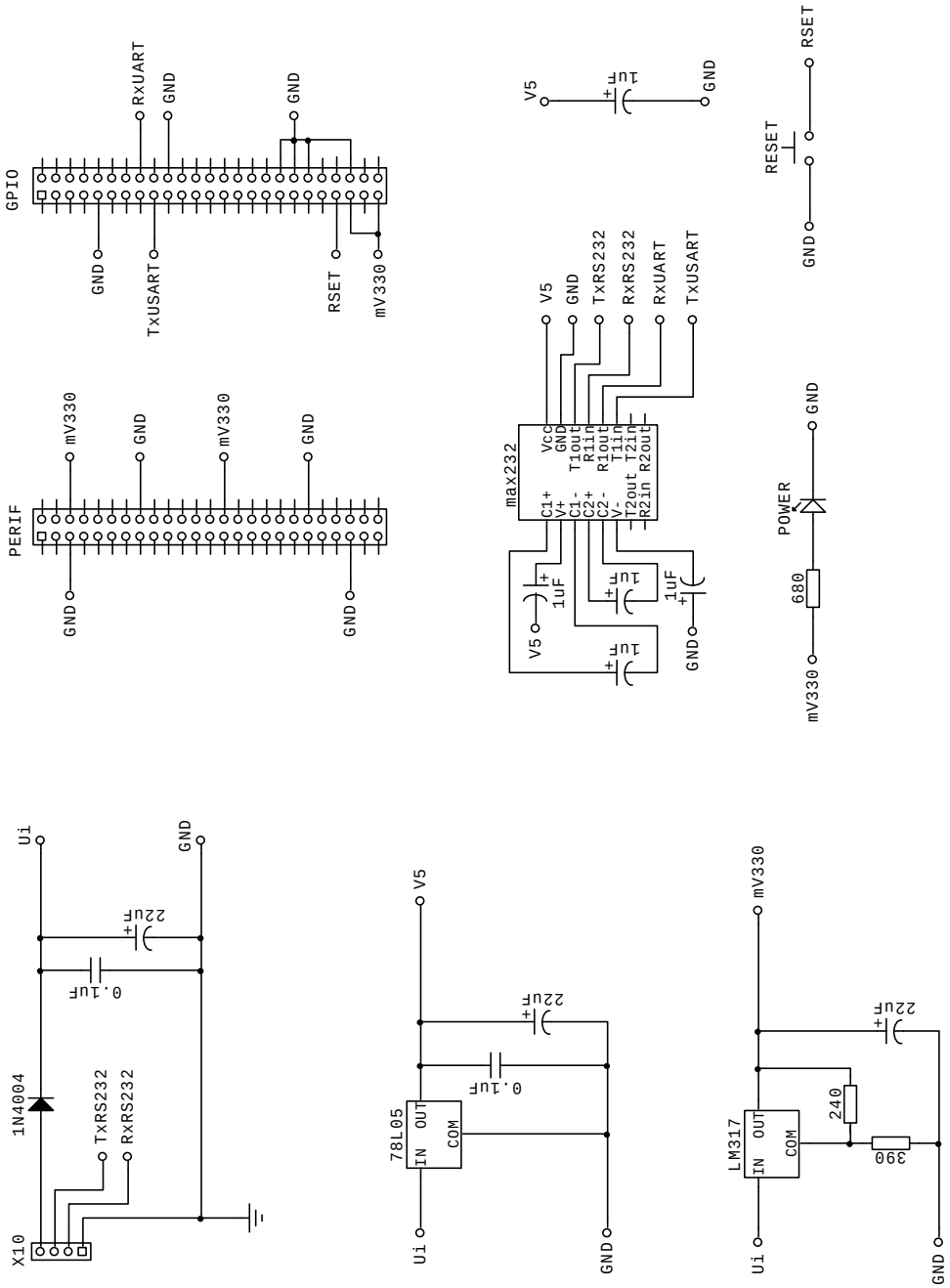


Ilustración 13: Diagrama del circuito

B. TARJETA IMPRESA

Tamaño real 7x9 cm.

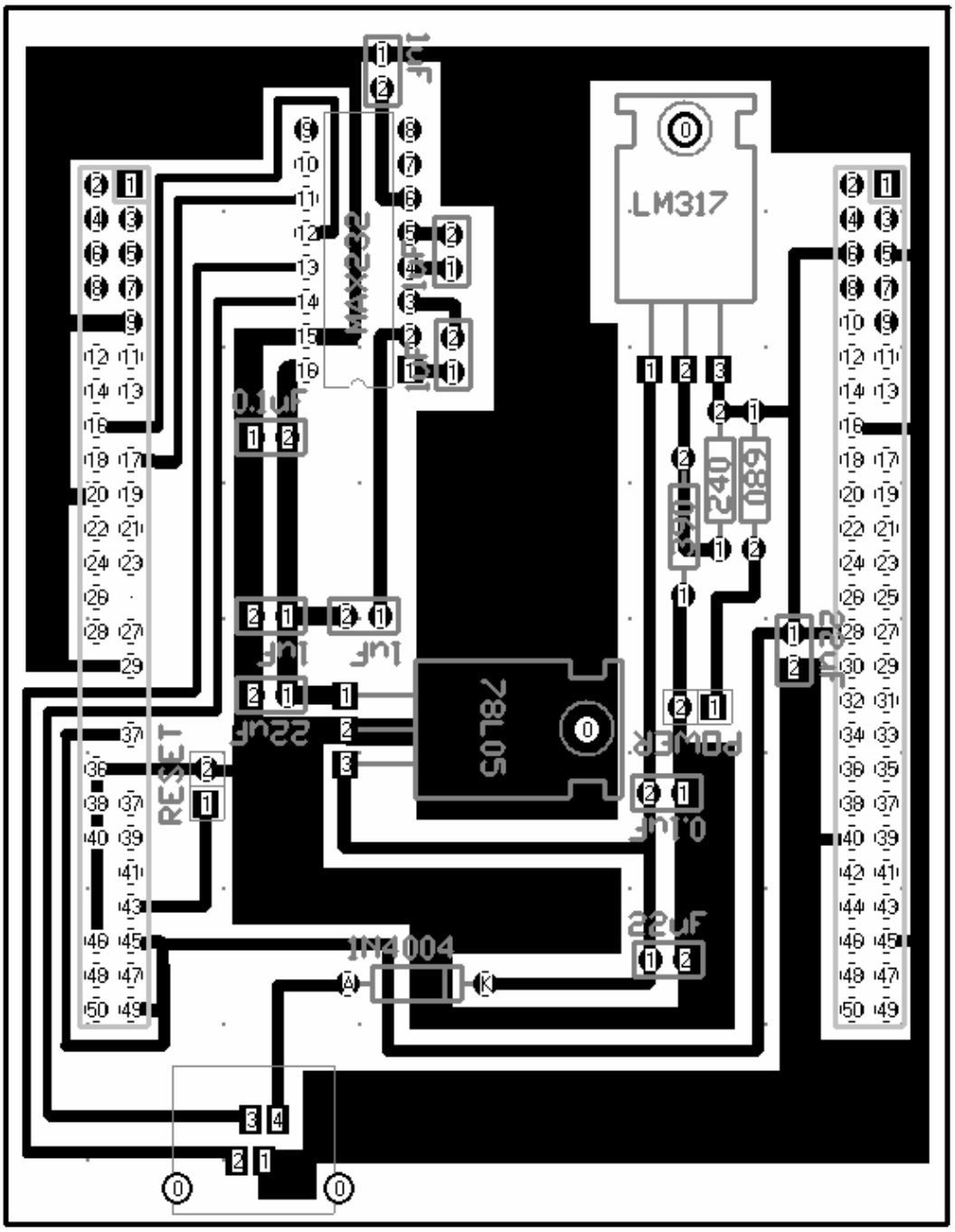


Ilustración 14: Tarjeta impresa

C. CÓDIGO FUENTE

1. POP3

a. POP3.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>
#include "listas.h"
#include "abcmisc.h"

#ifndef _POP3_H_
        #define _POP3_H_

// definición de tipos
        typedef int             POP3_STATUS;

// codigos de estado (POP3_STATUS)
        #define  POP3_DATA                       2
        #define  POP3_NO_DATA                    1
        #define  POP3_OK                         0
        #define  POP3_ERROR                     -1
        #define  POP3_DNS_ERROR                 -2
        #define  POP3_SERVER_ERROR              -3
        #define  POP3_LOGIN_ERROR               -4
        #define POP3_USER_ERROR                 -5
        #define POP3_INFO_ERROR                 -6
        #define POP3_DATA_ERROR                 -7
        #define POP3_TOO_LONG                   -8


// variables para contadores de timeout
        #define  POP3_DNS_TIMEOUT                3
        #define  POP3_SERVER_TIMEOUT             2

// tamaño máximo de mensaje
        #define  POP3_MAIL_SIZE                  4096

// prototipo de funciones de interfaz. ("public:")
        POP3_STATUS     POP3_init(void);                // inicializa el semáforo.
        POP3_STATUS     POP3_check();                   // lee email num y lo coloca en *mail


// prototipo de funciones de librería. ("private:")
        POP3_STATUS     POP3_conectar(void);            // conecta al servidor
        POP3_STATUS     POP3_login(void);               // nogocia sesión
        POP3_STATUS     POP3_nuevos(void);              // devuelve num. de emails nuevos
        int             POP3_info(int num);             // extrae información de los
                                                        // headers del mensaje
        POP3_STATUS     POP3_leer(int num,int desc);    // Lee el mensaje indicado
                                                        // y descarta desc bytes del
                                                        // principio.
        POP3_STATUS     POP3_borrar(int num);           // borra el mensaje indicado
        POP3_STATUS     POP3_desconectar(void);         // desconectar el servidor

#endif
```

b. POP3.c

```
#include "pop3.h"

// Identificadores de dispositivos
DID             POP3;

// buffers de entrada
char            correo[POP3_MAIL_SIZE];
char            POP3_FROM[128];
char            tempc[128];

// Variables POP3
        char    POP3_SERVER[]           = "kirika.uvg.edu.gt";
        char    POP3_USER[]             = "casax10";
        char    POP3_PASS[]             = "passsecreto";
        const char      FIN[5]          = {0x0D,0x0A,'.',0x0D,0x0A};

        LID             POP3_VALID;

// === Función de Inicialización ==============================================================
POP3_STATUS     POP3_init(void){
        LISTA_init(&POP3_VALID);
        LISTA_push(POP3_VALID,"usuariox10@intelnett.com");
        LISTA_push(POP3_VALID,"abalbas@amigo.net.gt");
        LISTA_push(POP3_VALID,"alejandrobalbas@intelnett.com");
        LISTA_push(POP3_VALID,"bal99030@uvg.edu.gt");
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Función principal POP3 (revisa los correos) ============================================
POP3_STATUS POP3_check(){
        int     i, j, n;
        char    de[128];

        correo[0] = 0;

        // establecer conexión
        i =  POP3_conectar();
        if(i != POP3_OK){
        // Error
                POP3_desconectar();
                return i;
        }

        // Autenticación
        i = POP3_login();
        if(i != POP3_OK){
        // Error
                POP3_desconectar();
                return i;
        }

        // Mensajes nuevos
        i = POP3_nuevos();
        if(i < 0){
                // Error
                POP3_desconectar();
                return i;
        }
        if(i == 0){
                // No hay mensajes
                POP3_desconectar();
                return POP3_NO_DATA;
        }

        // información de headers
        n = POP3_info(1);
```

```
        if( (n < 0) || (POP3_FROM[0] == 0) ){            // Error
                POP3_borrar(1);
                POP3_desconectar();
                return n;//POP3_INFO_ERROR;
        }

        // comprovación de dirección de correo autorizada
        j = LISTA_size(POP3_VALID);
        for(i = 1; i <= j; i++ ){
                LISTA_item(POP3_VALID,i,de);
                if( strlen(de) == strlen(POP3_FROM) )
                        if( blkequ(de,POP3_FROM,strlen(de)) )
                                i = j + 1;
                if( i == j  ){          // Error
                        POP3_borrar(1);
                        POP3_desconectar();
                        return  POP3_USER_ERROR;
                }
        }

        // Leer el mensaje
        i = POP3_leer(1,n);
        if( ( i != POP3_OK ) || ( correo[0] == 0 ) ){
        // Error
                POP3_desconectar();
                return POP3_DATA_ERROR;
        }

        // Borrar mensaje leido
        POP3_borrar(1);

        // finalizar conexión
        POP3_desconectar();
        return POP3_DATA;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Conexión al servidor ===================================================================
POP3_STATUS POP3_conectar(void){
        IPaddr          dirIP;
        char    direc[25];
        int             TIMEOUT_TMR;
        char    *temp;

// Obtener Dirección IP
        TIMEOUT_TMR = 0;
        do{
                if(TIMEOUT_TMR++ >= POP3_DNS_TIMEOUT) return POP3_DNS_ERROR;
                dirIP = name2ip(POP3_SERVER);
        }while(dirIP == SYSERR);
        xc_sprintf(direc,"%u.%u.%u.%u:110"       , (unsigned char)  dirIP
                                                 , (unsigned char) (dirIP >> 8)
                                                 , (unsigned char) (dirIP >> 16)
                                                 , (unsigned char) (dirIP >> 24));
// Establecer conexión al servidor POP3_SERVER
        TIMEOUT_TMR = 0;

        do{
                if(TIMEOUT_TMR++ >= POP3_SERVER_TIMEOUT) return POP3_SERVER_ERROR;

                POP3 = open(TCP,direc,ANYLPORT);
                if( POP3 == TCPE_REFUSED  ) POP3 = SYSERR;
                if( POP3 == TCPE_TIMEDOUT ) POP3 = SYSERR;
        }while(POP3 == SYSERR );
// leer respuesta
        tempc[read(POP3,tempc,sizeof(tempc))] = 0;
        if(tempc[0] != '+') return POP3_ERROR;
// todo bien
        return POP3_OK;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
// === Autenticación de usuario ===========================================================
POP3_STATUS      POP3_login(void){
        char    temp;
// enviar usuario
        xc_fprintf(POP3,"USER %s\r\n",POP3_USER);

// leer respuesta
        tempc[read(POP3,tempc,sizeof(tempc))] = 0;
        if(tempc[0] != '+') return POP3_LOGIN_ERROR;

// enviar contraseña
        xc_fprintf(POP3,"PASS %s\r\n",POP3_PASS);

// leer respuesta
        tempc[read(POP3,tempc,sizeof(tempc))] = 0;
        if(tempc[0] != '+') return POP3_LOGIN_ERROR;

// todo bien
        return POP3_OK;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Correos nuevos =====================================================================
POP3_STATUS POP3_nuevos(void){
        char    *temp;
        // preguntar numero de mensajes nuevos
        xc_fprintf(POP3,"STAT\r\n");
        // leer respuesta
        tempc[read(POP3,tempc,sizeof(tempc))] = 0;
        if(tempc[0] != '+') return POP3_ERROR;

        temp = xc_index(&tempc[4],' ');
        *temp = 0;
        return atoi(&tempc[4]);
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Fin de sesión ======================================================================
POP3_STATUS POP3_desconectar(void){
        write(POP3,"QUIT\r\n",6);
        // leer respuesta
        tempc[read(POP3,tempc,sizeof(tempc))] = 0;
        close(POP3);
        POP3 = -1;

        if(tempc[0] != '+') return POP3_ERROR;
        return  POP3_OK;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Información de headers =============================================================
POP3_STATUS POP3_info(int num){
        int     i=0,j,n=0;
        char    *c,*t;
        char    *temp;

        xc_fprintf(POP3,"TOP %d 0\r\n",num);
        POP3_FROM[0] = 0;
        temp = NULL;
        // leer respuesta
        n = 0;
        do{
                n += read(POP3,&correo[n],sizeof(correo)- n -200);
                if( n >= (POP3_MAIL_SIZE - 200))
                        return POP3_TOO_LONG;
        }while(!blkequ(&correo[n-5],FIN,sizeof(FIN)));

        if (correo[0] != '+') return POP3_INFO_ERROR;
        correo[n] = 0;
        for(j=0;correo[j] != 0x0A;j++);
```

```
        j = n - j - 3;                          // Los 3 caracteres son: .\r\n
        c = correo;
        do{
                t = abc_index(c,'\r', &correo[n] - c);
                if (t == NULL) return POP3_INFO_ERROR;
                if ( iguales(t,FIN,sizeof(FIN))) return j;
                t[0] = 0;
                if ( iguales(c,"from:",5) ){
                        temp = &c[6];
                        t = xc_index(temp,'<');
                        if( t != NULL ){
                                temp = &t[1];
                                t = xc_index(c,'>');
                                if (t != NULL) t[0] = 0;
                                for(i=0,t=temp;t[i]!=0x20)&&(t[i]!=0x0A)&&(t[i]!=0x0D)&&(t[i]!=0);
                                        i++);
                                t[i] = 0;
                                strcpy(POP3_FROM,temp);
                        }
                }
                if( t != NULL) c = &t[2];
        }while(c <= &correo[n]);
        return POP3_ERROR;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Datos del email ========================================================================
POP3_STATUS     POP3_leer(int num, int desc){
        int r = POP3_OK,j,n;

        xc_fprintf(POP3,"RETR %d\r\n",num);
        correo[0] = tempc[0] = 0;
        //leer respuesta
        read(POP3,tempc,1);
        if(tempc[0] != '+') r = POP3_ERROR;
        do{
                read(POP3,correo,1);
        }while(correo[0] != 0x0A);
        // descartar bytes
        n = 1;
        do{
                n += read(POP3,correo,desc-n);
        }while(n != desc);

        // leer correo
        n = 0;
        do{
                n += read(POP3,&correo[n],sizeof(correo)-n-200);
                if(n >= (POP3_MAIL_SIZE-200))
                        return POP3_TOO_LONG;
        }while(!iguales(&correo[n-5],FIN,sizeof(FIN)));

        correo[n] = 0;

        xc_fprintf(POP3,"RSET\r\n",num);
        // leer respuesta
        read(POP3,tempc,sizeof(tempc));

        return r;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// === Borrar mensaje del servidor ============================================================
POP3_STATUS     POP3_borrar(int num){
        xc_fprintf(POP3,"DELE %d\r\n",num);
        // leer respuesta
        tempc[read(POP3,tempc,sizeof(tempc))] = 0;
        if(tempc[0] != '+') return POP3_ERROR;
        return POP3_OK;
}
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

2. SMTP

a. smtp_abc.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>
#include <smtp.h>
#include <dgram.h>
#include <tcb.h>
#include "abcmisc.h"
#include "rtc.h"

#ifndef _SMTP_H_
        #define _SMTP_H_

// codigos de estado
        #define         SMTP_OK                  0
        #define         SMTP_ERROR              -1
        #define         SMTP_DNS_ERROR          -2
        #define         SMTP_SERVER_ERROR       -3

// definiciones

        #define         SMTP_HTML                1
        #define         SMTP_TEXT                0

// variables para contadores de timeout
        #define SMTP_DNS_TIMEOUT                 3
        #define SMTP_SERVER_TIMEOUT              2

// definición de estructuras y tipos
        typedef         BYTE                     SMTP_STATUS;

// prototipo de funciones de interfaz. ("public:")
        SMTP_STATUS     SMTP_init(void);
        SMTP_STATUS     SMTP_reset(char* to, char* asunto, Bool);
        SMTP_STATUS     SMTP_add(char*);
        SMTP_STATUS     SMTP_addln(char*);
        SMTP_STATUS     SMTP_send( void );

// prototipo de funciones de libreria ("private:")
        SMTP_STATUS     SMTP_conectar(void);
        SMTP_STATUS     SMTP_mail(void);
        SMTP_STATUS     SMTP_rcpt(void);
        SMTP_STATUS     SMTP_data(BYTE);
        SMTP_STATUS     SMTP_desconectar(void);
        SMTP_STATUS     SMTP_out(void);


#endif
```

b. smtp_abc.c

```
#include "smtp_abc.h"

// Buffers de correo
#define                 SMTP_BUFFSIZE                   4096
char                    SMTP_BUFF[SMTP_BUFFSIZE];
int                     SMTP_BUFFPOS;
char                    SMTP_TO[128];
char                    SMTP_ASUNTO[64];
char                    SMTP_USER[]                     = "casax10";
char                    SMTP_FROM[128];
char                    SMTP_SERVER[128]                = "kirika.uvg.edu.gt";
int                     SMTP_PUERTO                     = 25;
Bool                    SMTP_TYPE;


extern  char    FIN[];

const   char    SMTP_TOP_HTML[]  =      "<html><head><title>X10Mail</title></head>"      \
                                        "<body bgcolor='white'>"                         \
                                        "<div style='color:skyblue;"                     \
                                        "font-family:Arial;"                             \
                                        "font-size:24pt;"                                \
                                        "font-variant: small-caps;"                      \
                                        "text-align:right'>"                             \
                                        "Sistema <span style='font-weight:bold;"         \
                                        "font-size:180%;"                                \
                                        "font-variant:normal;"                           \
                                        "color:0066CC'>"                                 \
                                        "<i>X10mail</i></span></div><br><br>"            \
                                        "<div style='color:black;"                       \
                                        "font-family:Arial;"                             \
                                        "font-style:italic;"                             \
                                        "font-size:12pt;"                                \
                                        "line-height:23pt'>";


const   char    SMTP_BOTTOM_HTML[]=     "<hr><span style='color:white'></body></html>";
const   char    SMTP_TOP_TEXTO[] =      "X10Mail\r\n";

char            tempch[128];

// Identificador de conexión
DID             SMTP_SOCKET;

// Semáforo para envío de mensaje
SID             SMTP_FREE;

// === Inicializar variables ======================================================================
SMTP_STATUS     SMTP_init(void){
        SMTP_FREE = screate(1);
        return SMTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Inicializar buffers y pedir su control =====================================================
SMTP_STATUS     SMTP_reset(char* to, char* asunto, Bool thtml){
        char    tempdate[128];

        wait(SMTP_FREE);
        SMTP_BUFFPOS            = 0;
        SMTP_BUFF[SMTP_BUFFPOS]  = 0;
        SMTP_TO[0]              = 0;
        SMTP_ASUNTO[0]          = 0;
        SMTP_TYPE               = thtml;
        blkcopy( SMTP_TO, to, strlen(to) + 1);
        blkcopy( SMTP_ASUNTO, asunto, strlen(asunto) + 1);
```

```
        // MIME content header
        SMTP_addln("MIME-Version: 1.0");
        if(SMTP_TYPE)
                SMTP_addln("Content-Type: text/html");
        else
                SMTP_addln("Content-Type: text/plain");

        // Mensaje
        if( SMTP_TYPE )
                SMTP_add(SMTP_TOP_HTML);
        else
                SMTP_add(SMTP_TOP_TEXTO);

        if( SMTP_TYPE )
                RTC_timestamp(&tempdate[0]);
        else{
                RTC_timestamp_short(tempdate);
                blkcopy(&tempdate[4],&tempdate[17],15);
        }
        SMTP_add(tempdate);

        if(SMTP_TYPE){
                SMTP_add("</div>");
        }

        return SMTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Agregar texto al correo ================================================================
SMTP_STATUS     SMTP_add(char* data){
        int             i;

        i = strlen(data);
        if( (SMTP_BUFFPOS + i) > SMTP_BUFFSIZE)
                return SMTP_ERROR;
        blkcopy(&SMTP_BUFF[SMTP_BUFFPOS],data,i);
        SMTP_BUFFPOS                            += i;

        return SMTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Agregar linea de texto al correo ========================================================
SMTP_STATUS     SMTP_addln(char* data){
        int             i;

        i = strlen(data);
        if( (SMTP_BUFFPOS + i) > SMTP_BUFFSIZE)
                return SMTP_ERROR;
        blkcopy(&SMTP_BUFF[SMTP_BUFFPOS],data,i);
        SMTP_BUFFPOS                            += i;
        SMTP_BUFF[SMTP_BUFFPOS++]       = 0x0D;
        SMTP_BUFF[SMTP_BUFFPOS++]       = 0x0A;

        return SMTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Enviar el correo ==========================================================================
SMTP_STATUS SMTP_send( ){
        char*   c;
        int             i;
        char*   email_error[128];
        int             send_timeout = 0;

        // dirección de origen
        i = strlen(SMTP_USER);
        blkcopy( SMTP_FROM, SMTP_USER, i );
        SMTP_FROM[i++] = '@';
        c = abc_index(SMTP_SERVER,'.',strlen(SMTP_SERVER));
        if(c == NULL)
                c = SMTP_SERVER;
        blkcopy(&SMTP_FROM[i], SMTP_SERVER, strlen(SMTP_SERVER) + 1);

        c = abc_index(SMTP_TO,'@', strlen(SMTP_TO));
        if( c == NULL ){
                SMTP_out();
                return SMTP_ERROR;
        }

        if(SMTP_TYPE)
                SMTP_add(SMTP_BOTTOM_HTML);

        SMTP_BUFF[SMTP_BUFFPOS++] = 0;

        do{
                email_error[0] = 0;

        mail(SMTP_SERVER,SMTP_PUERTO,SMTP_ASUNTO,SMTP_TO,SMTP_FROM,SMTP_BUFF,email_error,128);
                if(send_timeout++ >= 5){
                        SMTP_out();
                        return SMTP_ERROR;
                }

        }while(email_error[0] != 0);
        SMTP_out();
        return SMTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Soltar control de Buffers ==================================================================
SMTP_STATUS     SMTP_out(void){
        signal(SMTP_FREE);
        return SMTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

3. COMUNICACIÓN X10

a. X10.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#ifndef _X10_H_
        #define _X10_H_

// Codigos de Error
#define X10_OK                                          0
#define X10_ERROR                                       -1
#define X10_INVALIDO                                    -2
#define X10_CODE_ERROR                                  17

// Definiciones
        //CASAS
        #define         X10_CASA_A                      0x46
        #define         X10_CASA_B                      0x4E
        #define         X10_CASA_C                      0x42
        #define         X10_CASA_D                      0x4A
        #define         X10_CASA_E                      0x41
        #define         X10_CASA_F                      0x49
        #define         X10_CASA_G                      0x45
        #define         X10_CASA_H                      0x4D
        #define         X10_CASA_I                      0x47
        #define         X10_CASA_J                      0x4F
        #define         X10_CASA_K                      0x43
        #define         X10_CASA_L                      0x4B
        #define         X10_CASA_M                      0x40
        #define         X10_CASA_N                      0x48
        #define         X10_CASA_O                      0x44
        #define         X10_CASA_P                      0x4C

        //UNIDADES
        #define         X10_UNIDAD_1                    0x4C
        #define         X10_UNIDAD_2                    0x5C
        #define         X10_UNIDAD_3                    0x44
        #define         X10_UNIDAD_4                    0x54
        #define         X10_UNIDAD_5                    0x42
        #define         X10_UNIDAD_6                    0x52
        #define         X10_UNIDAD_7                    0x4A
        #define         X10_UNIDAD_8                    0x5A
        #define         X10_UNIDAD_9                    0x4E
        #define         X10_UNIDAD_10                   0x5E
        #define         X10_UNIDAD_11                   0x46
        #define         X10_UNIDAD_12                   0x56
        #define         X10_UNIDAD_13                   0x40
        #define         X10_UNIDAD_14                   0x50
        #define         X10_UNIDAD_15                   0x48
        #define         X10_UNIDAD_16                   0x58

        //COMANDOS
        #define         X10_CMD_ALLUNITSOFF             0x41
        #define         X10_CMD_ALLLIGHTSON             0x43
        #define         X10_CMD_ON                      0x45
        #define         X10_CMD_OFF                     0x47
        #define         X10_CMD_DIM                     0x49
        #define         X10_CMD_BRIGHT                  0x4B
        #define         X10_CMD_ALLLIGHTSOFF            0x4D
```

```
// definición de tipos y estructuras
        typedef BYTE                                    X10_STATUS;
        typedef BYTE                                    X10_CODIGO;
        typedef struct {
                BYTE    casa;
                BYTE    unidad;
        }                                               X10_DIR;
        typedef struct {
                X10_DIR dir;
                BYTE    cmd;
        }                                               X10_CMD;

// prototipo de funciones de interfaz. ("public:")
        X10_STATUS      X10_init( void );               // inicializa el puerto serial y el
                                                        // semáforo
        X10_STATUS      X10_enviar( BYTE, BYTE, BYTE );  // Envía un comando X10
        X10_STATUS      X10_recibir( BYTE*,BYTE*,BYTE* );// recibe un comando X10

        char*           X10_cmd2texto(BYTE cmd);        // convierte códigos a texto (debugging)

// prototipo de funciones de libreria ("private:")

        BYTE            X10_casa(X10_CODIGO casa);
        BYTE            X10_unidad(X10_CODIGO unidad);

#endif
```

b. X10.c

```
#include "x10.h"

// puerto serial para comunicación con el PowerLincII
DID     puerto_x10;
// Semáforo para para prevenir que se envien dos comandos simultaneamente
SID     X10_SEMAFORO;

// Tablas de conversión
        static BYTE  X10_CONV[17] = { 13,5,3,11,15,7,1,9,14,6,4,12,16,8,2,10,X10_INVALIDO };

        static X10_CODIGO        X10_CASA_CONV[16] = {    X10_CASA_A,
                                                          X10_CASA_B,
                                                          X10_CASA_C,
                                                          X10_CASA_D,
                                                          X10_CASA_E,
                                                          X10_CASA_F,
                                                          X10_CASA_G,
                                                          X10_CASA_H,
                                                          X10_CASA_I,
                                                          X10_CASA_J,
                                                          X10_CASA_K,
                                                          X10_CASA_L,
                                                          X10_CASA_M,
                                                          X10_CASA_N,
                                                          X10_CASA_O,
                                                          X10_CASA_P
                                                 };
        static X10_CODIGO        X10_UNIDAD_CONV[16] = {  X10_UNIDAD_1,
                                                          X10_UNIDAD_2,
                                                          X10_UNIDAD_3,
                                                          X10_UNIDAD_4,
                                                          X10_UNIDAD_5,
                                                          X10_UNIDAD_6,
                                                          X10_UNIDAD_7,
                                                          X10_UNIDAD_8,
                                                          X10_UNIDAD_9,
                                                          X10_UNIDAD_10,
                                                          X10_UNIDAD_11,
                                                          X10_UNIDAD_12,
                                                          X10_UNIDAD_13,
                                                          X10_UNIDAD_14,
                                                          X10_UNIDAD_15,
                                                          X10_UNIDAD_16
                                                       };
        static char              *X10_CMD_TEXTO[17] = {   "All Units Off",
                                                          "All Lights On",
                                                          "On",
                                                          "Off",
                                                          "Dim",
                                                          "Bright",
                                                          "All Lights Off",
                                                          "Extended Code",
                                                          "Hail Request",
                                                          "Comando no valido",
                                                          "Preset Dim High",
                                                          "Preset Dim Low",
                                                          "Extended Data (analog)",
                                                          "Status = on",
                                                          "Status = off",
                                                          "Status Request"
                                                          "Código Erroneo"
                                                 };
// === Inicializa puerto serial y semáforo ============================================
int X10_init(void){
        puerto_x10      = open(SERIAL1,0,0);
        X10_SEMAFORO    = screate(1);
```

```
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Envía comandos X10 ============================================================================
int X10_enviar(BYTE casa, BYTE unidad, BYTE comando){
        int     i = 0;
        char    resp;
        byte    rcode = X10_ERROR;
        if((casa >= 'a')&&(casa <= 'z')) i = 32;          // casa a minúsculas
        wait(X10_SEMAFORO);                               // si la función es llamada más de una
                                                          // vez se ejecutará solo una vez a la vez
                if( (casa != X10_INVALIDO)&&(comando != X10_INVALIDO)){
                        sleep(2);
                        xc_fprintf(puerto_x10,"%c",0x02);
                        sleep10(1);
                        xc_fprintf(puerto_x10,"%c%c", 0x63, X10_CASA_CONV[casa - 65 - i] );
                        if(unidad != X10_INVALIDO)
                                xc_fprintf(puerto_x10,"%c%c",X10_UNIDAD_CONV[unidad - 1],comando);
                        else
                                xc_fprintf(puerto_x10,"%c%c", comando,0x00);
                        xc_fprintf(puerto_x10,"%c", 0x41);
                        rcode = X10_OK;
                }
        signal(X10_SEMAFORO);
        return  rcode;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Recibe comandos X10 ===========================================================================
int X10_recibir(BYTE *casa, BYTE *unidad, BYTE *comando){
        char    x10_buff[15];
        *unidad = *comando = X10_INVALIDO;
        do{
                do{
                        read(puerto_x10,x10_buff,1);                    // Codigo inicio
                }while((x10_buff[0] != 0x58)&&(x10_buff[0] != 0x78));
                read(puerto_x10,&x10_buff[1],1);                        // Casa
                if( x10_buff[1] == '\r' )
                        return X10_ERROR;
                *casa   = X10_casa(x10_buff[1]);
                read(puerto_x10,&x10_buff[2],1);                        // Unidad o Funcion
                if( x10_buff[2] == '\r' )
                        return X10_ERROR;
                if(x10_buff[2] & 0x01)
                        *comando= x10_buff[2];
                else
                        *unidad = X10_unidad(x10_buff[2]);
                read(puerto_x10,&x10_buff[3],1);                        // Num. de veces
                if( x10_buff[3] == '\r' ) return X10_ERROR;
                read(puerto_x10,&x10_buff[4],1);                        // terminador '\n'
        }while(*comando == X10_INVALIDO);                               // Hasta que la recepción
                                                                        // sea un comando válido
        return X10_OK;

}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Hace la conversión al código de casa =======================================================
BYTE X10_casa(X10_CODIGO casa){
        BYTE    i;
        i = (casa - 0x40);
        if( ( i < 0 ) || ( i > 15))
                return X10_CONV[X10_CODE_ERROR];
        return X10_CONV[i] + 96;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Hace la conversión al código de unidad ========================================================
BYTE X10_unidad(X10_CODIGO unidad){
        BYTE    i;
        i = ((unidad - 0x40) >> 1);
        if( ( i < 0 ) || ( i > 15))
                return X10_CONV[X10_CODE_ERROR];
        return X10_CONV[i];
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Convierte el código de comando a texto (usado durante debugging) ============================
char* X10_cmd2texto(X10_CODIGO cmd){
        BYTE    i;
        i = ((cmd - 0x40) >> 1);
        if( ( i < 0 ) || ( i > 15) )
                return X10_CMD_TEXTO[X10_CODE_ERROR];
        return X10_CMD_TEXTO[i];
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

4. LISTAS

a. listas.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>

#ifndef _LISTAS_H_
        #define _LISTAS_H_


//codigos de estado
        #define  LISTA_OK                         0
        #define  LISTA_ERROR                     -1
        #define LISTA_NO_ENCONTRADO              -2
        #define LISTA_NO_VALIDO                  -3
        #define LISTA_VACIA                      -4

// definición de estructuras y tipos
        struct nodo_lista{
                char                    *data;
                struct nodo_lista       *next;
        };

// definición de tipos
        typedef struct nodo_lista*              LISTA_NODO;
        typedef LISTA_NODO                      LID;
        typedef  BYTE                           LISTA_STATUS;

// Tamaños de estructuras
        #define LISTA_NODO_SIZE  sizeof(struct nodo_lista)

// prototipo de funciones de interfaz. ("public:")
        LISTA_STATUS    LISTA_init( LID *head );                // Crea una nueva lista
        LISTA_STATUS    LISTA_clear( LID head );                // Vacía una lista
        LISTA_STATUS    LISTA_push( LID head, char* data );     // Agrega un elemento a una lista
        LISTA_STATUS    LISTA_borrar( LID head, char* data );   // Borra un elemento de una lista
        int             LISTA_size( LID head );                 // Devuelve el num de elementos
        LISTA_STATUS    LISTA_item( LID head, int n, char* data );// busca los datos de un elemento

// prototipo de funciones de libreria ("private:")
        LISTA_NODO              LISTA_last( LID head );          // devuelve el último elemento


#endif
```

b. listas.c

```
#include "listas.h"

// === Crea una nueva lista ===================================================================
LISTA_STATUS    LISTA_init(LID *head){
        LISTA_NODO      temp;

        // crear Cabeza de lista
        *head           = getmem(LISTA_NODO_SIZE);
        if(head == NULL)
                return LISTA_ERROR;
        temp = *head;
        // Inicialización
        temp->data      = NULL;
        temp->next      = NULL;

        return  LISTA_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Vacía una lista ========================================================================
LISTA_STATUS    LISTA_clear(LID head){
        LISTA_NODO      temp;
        if(head->data == NULL) return LISTA_OK;

        temp = head->next;
        while(temp != NULL){
                head->next = temp->next;
                freemem(temp->data, strlen(temp->data) + 1 );     // borrar datos
                freemem(temp, LISTA_NODO_SIZE);                   // borrar nodo
                temp = head->next;
        }
        freemem( head->data, strlen(head->data) + 1 );
        head->data = NULL;

        return  LISTA_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Insertar elemento en lista =============================================================
LISTA_STATUS    LISTA_push(LID head, char* data){
        LISTA_NODO      nuevo, ultimo;
        int i;

        if(head->data == NULL){
                head->data      = getmem( strlen(data) + 1 );
                blkcopy(head->data, data, strlen(data) + 1 );
                return LISTA_OK;
        }
        nuevo = head;
        do{
                if( strlen(data) == strlen(nuevo->data) )
                        if( blkequ(data, nuevo->data, strlen(data)) )
                                return  LISTA_OK;
                // data repe
                nuevo = nuevo->next;
        }while(nuevo != NULL);

        nuevo = getmem(LISTA_NODO_SIZE);
        nuevo->data = getmem( strlen(data) + 1);
        blkcopy(nuevo->data, data, strlen(data) + 1);
        ultimo = LISTA_last(head);

        ultimo->next    = nuevo;
        nuevo->next     = NULL;

        return LISTA_OK;
```

```
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
// === Borrar un elemento de la lista =============================================================
LISTA_STATUS     LISTA_borrar(LID head, char* data){
        LISTA_NODO      prev, temp;
        int                     i;

        if(head->data == NULL) return LISTA_VACIA;

        if( strlen(data) == strlen(head->data) ){
                if( blkequ(data, head->data, strlen(data)) ){
                        if(head->next == NULL){
                                freemem(head->data, strlen(head->data) + 1 );
                                head->data = NULL;
                                return LISTA_OK;
                        }
                        temp = head->next;
                        freemem(head->data, strlen(head->data) +1);
                        head->data              = temp->data;
                        head->next              = temp->next;
                        freemem(temp,LISTA_NODO_SIZE);
                        return LISTA_OK;
                }
        }
        prev = head;
        temp = head->next;
        do{
                if( strlen(data) == strlen(temp->data) ){
                        if( blkequ(data, temp->data, strlen(data)) ){
                                prev->next = temp->next;
                                freemem(temp->data,strlen(temp->data)+1);
                                freemem(temp,LISTA_NODO_SIZE);
                                return LISTA_OK;
                        }
                }
                prev = temp;
                temp = temp->next;
        }while(temp != NULL);

        return  LISTA_NO_ENCONTRADO;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Devuelve el último nodo de la lista ========================================================
LISTA_NODO              LISTA_last( LID head ){
        LISTA_NODO      temp;

        temp = head;
        if(head->next == NULL) return head;
        do{
                temp = temp->next;
        }while(temp->next != NULL);
        return temp;
}

int     LISTA_size(LID head){
        LISTA_NODO      temp;
        int             i = 1;

        if(temp->data == NULL) return 0;
        temp = head;
        do{
                if(temp->next == NULL) return i;
                temp = temp->next;
                i++;
        }while(temp != NULL);
        return  LISTA_ERROR;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Datos de un elemento de la lista ==========================================================
LISTA_STATUS     LISTA_item(LID head, int n, char* data){
        LISTA_NODO       temp;
        int              i = 1;

        if(head->data == NULL) return LISTA_VACIA;

        if(n < 1) return LISTA_ERROR;

        temp = head;
        while(i != n){
                if(temp->next == NULL) return LISTA_ERROR;
                temp = temp->next;
                i++;
        }

        if(temp->data == NULL) return LISTA_ERROR;

        blkcopy(data, temp->data, strlen(temp->data) + 1 );

        return LISTA_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

c. alias.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include "x10.h"

#ifndef _ALIAS_H_
        #define _ALIAS_H_

//codigos de estado
        #define  ALIAS_OK                                 0
        #define  ALIAS_ERROR                             -1
        #define ALIAS_NO_ENCONTRADO                      -2
        #define ALIAS_NO_VALIDO                          -3
        #define ALIAS_VACIO                              -4

// definición de estructuras ALIAS_NODO/ALIAS_LISTA
        struct nodo_alias{
                char                    *alias;
                int                     casa;
                int                     unidad;
                struct nodo_alias       *next;
        };
// Tamaños de estructuras
        #define ALIAS_NODO_SIZE                          sizeof(struct nodo_alias)

// definición de tipos
        typedef struct nodo_alias*                       ALIAS_NODO;
        typedef ALIAS_NODO                               ALIAS_LISTA;
        typedef  int                                     ALIAS_STATUS;

// prototipo de funciones de interfaz. ("public:")
        ALIAS_STATUS    ALIAS_init( void );                      // inicializa la lista
        ALIAS_STATUS    ALIAS_clear( void );                     // vacía la lista
        ALIAS_STATUS    ALIAS_push( int, int, char* );           // agrega un alias
        char*           ALIAS_get_name( int , int );             // devuelve el nombre de la dir.
        ALIAS_STATUS    ALIAS_get_dir( char*, X10_DIR* );        // devuelve dirección del alias
        ALIAS_STATUS    ALIAS_borrar_dir( int, int );            // borra alias por dirección
        ALIAS_STATUS    ALIAS_borrar( char* );                   // borra alias por nombre
        int             ALIAS_size( void );                      // tamaño de la lista de alias
        ALIAS_STATUS    ALIAS_num( int, char*, X10_DIR* );       // devuelve el alias de una
                                                                 // posición dentro de la lista

// prototipo de funciones de libreria ("private:")
        ALIAS_NODO      ALIAS_last( void );                      // devuelve el último nodo


#endif
```

d. alias.c

```
#include "alias.h"

// variables de librería
ALIAS_LISTA             alias_head;

// === Inicializa la lista de alias ===================================================
ALIAS_STATUS    ALIAS_init(void){
        // crear Cabeza de lista
        alias_head = getmem(ALIAS_NODO_SIZE);

        if(alias_head == NULL) return ALIAS_ERROR;

        // Inicialización
        alias_head->alias       = NULL;
        alias_head->casa = X10_INVALIDO;
        alias_head->unidad      = X10_INVALIDO;
        alias_head->next = NULL;

        return  ALIAS_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Vacía la lista de alias ========================================================
ALIAS_STATUS    ALIAS_clear(void){
        ALIAS_NODO      temp;

        if(alias_head->alias == NULL) return ALIAS_OK;

        temp = alias_head->next;
        while(temp != NULL){
                alias_head->next = temp->next;
                freemem(temp->alias,strlen(temp->alias) + 1);
                freemem(temp,ALIAS_NODO_SIZE);
                temp = alias_head->next;
        }
        freemem(alias_head->alias,strlen(alias_head->alias) + 1);
        alias_head->alias = NULL;

        return  ALIAS_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Agrega alias a la lista ========================================================
ALIAS_STATUS    ALIAS_push(int casa, int unidad, char* alias){
        ALIAS_NODO      nuevo,ultimo;
        int i;

        if(alias_head->alias == NULL){
                alias_head->casa        = casa;
                alias_head->unidad      = unidad;
                alias_head->alias       = getmem(strlen(alias) + 1);
                blkcopy(alias_head->alias,alias,strlen(alias) + 1);
                return ALIAS_OK;
        }

        nuevo = alias_head;
        do{
                if( (nuevo->casa == casa) && (nuevo->unidad == unidad) ){
                        freemem(nuevo->alias,strlen(nuevo->alias) + 1);
                        nuevo->alias = getmem(strlen(alias) + 1);
                        blkcopy(nuevo->alias,alias,strlen(alias) + 1);
                        return  ALIAS_OK;
                }
                if( strlen(alias) == strlen(nuevo->alias))
                        if( blkequ(alias,nuevo->alias,strlen(alias)) ){
                                nuevo->casa = casa;
```

```
                                nuevo->unidad = unidad;
                                return  ALIAS_OK;
                        }
                nuevo = nuevo->next;
        }while(nuevo != NULL);

        nuevo                   = getmem(ALIAS_NODO_SIZE);
        nuevo->alias    = getmem(strlen(alias) + 1);

        blkcopy(nuevo->alias,alias,strlen(alias) + 1);
        nuevo->casa     = casa;
        nuevo->unidad   = unidad;

        ultimo = ALIAS_last();

        ultimo->next    = nuevo;
        nuevo->next     = NULL;

        return ALIAS_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Busca el nombre de una dirección dentro de la lista ========================================
char*   ALIAS_get_name(int casa, int unidad){
        ALIAS_NODO      temp;

        if(alias_head->alias == NULL) return NULL;

        temp = alias_head;
        do{
                if((temp->casa == casa)&&(temp->unidad == unidad)) return temp->alias;
                temp = temp->next;
        }while(temp != NULL);
        return  NULL;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Busca la dirección de un nombre en la lista ================================================
ALIAS_STATUS    ALIAS_get_dir(char* alias, X10_DIR* m){
        ALIAS_NODO      temp;
        int i;

        if(alias_head->alias == NULL) return ALIAS_VACIO;

        temp = alias_head;
        do{
                if( strlen(alias) == strlen(temp->alias) )
                        if(blkequ(alias,temp->alias,strlen(alias))){
                                m->casa = temp->casa;
                                m->unidad       = temp->unidad;
                                return  ALIAS_OK;
                        }
                temp = temp->next;
        }while(temp != NULL);
        return ALIAS_NO_ENCONTRADO;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Borra el alias de la dirección dada ========================================================
ALIAS_STATUS    ALIAS_borrar_dir( int casa, int unidad){
        ALIAS_NODO      prev,temp;

        if(alias_head->alias == NULL) return ALIAS_VACIO;

        if( (alias_head->casa == casa) && (alias_head->unidad == unidad) ){
                if(alias_head->next == NULL){
                        freemem(alias_head->alias, strlen(alias_head->alias)+1 );
                        alias_head->alias == NULL;
                        return ALIAS_OK;
                }
                temp = alias_head->next;
```

```
                alias_head->casa                = temp->casa;
                alias_head->unidad       = temp->unidad;
                freemem(alias_head->alias, strlen(alias_head->alias)+1 );
                alias_head->alias               = temp->alias;
                alias_head->next                = temp->next;
                freemem(temp,ALIAS_NODO_SIZE);
                return ALIAS_OK;
        }
        prev = alias_head;
        temp = alias_head->next;
        do{
                if( (temp->casa == casa) && (temp->unidad == unidad) ){
                        prev->next = temp->next;
                        freemem(temp->alias,strlen(temp->alias)+1);
                        freemem(temp,ALIAS_NODO_SIZE);
                        return ALIAS_OK;
                }
                prev = temp;
                temp = temp->next;
        }while(temp != NULL);
        return  ALIAS_NO_ENCONTRADO;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Borra un alias de la lista ===========================================================
ALIAS_STATUS    ALIAS_borrar(char* alias){
        ALIAS_NODO      prev,temp;
        int                     i;

        if(alias_head->alias == NULL) return ALIAS_VACIO;

        if( strlen(alias) == strlen(alias_head->alias) )
                if( blkequ(alias,alias_head->alias,strlen(alias)) ){
                        if(alias_head->next == NULL){
                                freemem(alias_head->alias, strlen(alias_head->alias) +1);
                                alias_head->alias = NULL;
                                return ALIAS_OK;
                        }
                        temp = alias_head->next;
                        alias_head->casa                = temp->casa;
                        alias_head->unidad       = temp->unidad;
                        freemem(alias_head->alias, strlen(alias_head->alias) +1);
                        alias_head->alias               = temp->alias;
                        alias_head->next                = temp->next;
                        freemem(temp,ALIAS_NODO_SIZE);
                        return ALIAS_OK;
                }

        prev = alias_head;
        temp = alias_head->next;
        do{
                if( strlen(alias) == strlen(temp->alias) )
                        if( blkequ(alias,temp->alias,strlen(alias)) ){
                                prev->next = temp->next;
                                freemem(temp->alias,strlen(temp->alias)+1);
                                freemem(temp,ALIAS_NODO_SIZE);
                                return ALIAS_OK;
                        }
                prev = temp;
                temp = temp->next;
        }while(temp != NULL);
        return  ALIAS_NO_ENCONTRADO;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Devuelve el último nodo de la lista ============================================================
ALIAS_NODO      ALIAS_last(void){
        ALIAS_NODO      temp;

        if(alias_head->next == NULL) return alias_head;

        temp = alias_head;
        do{
                temp = temp->next;
        }while(temp->next != NULL);
        return temp;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Tamaño de la lista =============================================================================
int     ALIAS_size(void){
        ALIAS_NODO      temp;
        int             i = 1;

        if(temp->alias == NULL) return 0;

        temp = alias_head;
        do{
                if(temp->next == NULL) return i;
                temp = temp->next;
                i++;
        }while(temp != NULL);

        return  ALIAS_ERROR;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Devuelve el alias del elemento n de la lista ===================================================
ALIAS_STATUS    ALIAS_num(int n, char* alias, X10_DIR* m){
        ALIAS_NODO      temp;
        int             i = 1;

        if(alias_head->alias == NULL) return ALIAS_VACIO;

        if(n < 1) return ALIAS_ERROR;

        temp = alias_head;
        while(i != n){
                if(temp->next == NULL) return ALIAS_ERROR;
                temp = temp->next;
                i++;
        }

        if(temp->alias == NULL) return ALIAS_ERROR;

        blkcopy(alias, temp->alias, strlen(temp->alias) + 1);
        m->casa  = temp->casa;
        m->unidad       = temp->unidad;
        return ALIAS_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

5. BASE DE DATOS X10

a. x10db.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>

#include "abcmisc.h"
#include "listas.h"
#include "x10.h"
#include "alias.h"

#ifndef _X10DB_H_
        #define _X10DB_H_

// codigos de estado
        #define  X10DB_OK                         0
        #define  X10DB_ERROR                     -1

// definiciones

// definición de estructuras y tipos
        typedef  BYTE                            X10DB_STATUS;
        typedef X10_CMD                          X10DB_DATA;

// prototipo de funciones de interfaz. ("public:")
        X10DB_STATUS    X10DB_init();                    // inicializa la base de datos
        X10DB_STATUS    X10DB_event(X10DB_DATA);         // interpreta nuevo evento X10
        X10_CODIGO      X10DB_lookup(X10_DIR);           // Busca el estado de un módulo
        X10DB_STATUS    X10DB_item(int,X10DB_DATA*);     // devuelve un elemento de la lista

// prototipo de funciones de libreria ("private:")


#endif
```

b. x10db.c

```
#include "x10db.h"

// lista
LID             X10DB_LISTA;

// === Inicializar Lista ==========================================================================
X10DB_STATUS    X10DB_init(){
        LISTA_init(&X10DB_LISTA);
        return  X10DB_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Interpretar nuevo evento ===================================================================
X10DB_STATUS    X10DB_event(X10DB_DATA data){
        int                             n,
                                        i;
        X10DB_DATA              comp;
        char                    c[sizeof(X10DB_DATA) + 1];


        n = LISTA_size(X10DB_LISTA);
        for(i = 1; i <= n; i++){
                if(LISTA_item(X10DB_LISTA, i, c) == LISTA_OK){
                        blkcopy((char*)&comp,c,sizeof(X10DB_DATA));
                        if( comp.dir.casa == data.dir.casa ){
                                if( (data.dir.unidad == X10_INVALIDO)
                                        ||(data.dir.unidad == comp.dir.unidad) ){
                                        // si existe en base de datos
                                        // borrar dato de lista para modificarlo y reagregarlo
                                        LISTA_borrar(X10DB_LISTA,c );

                                        switch(data.cmd){
                                                case X10_CMD_ON:
                                                        comp.cmd = data.cmd;
                                                        break;
                                                case X10_CMD_OFF:
                                                        comp.cmd = data.cmd;
                                                        break;
                                                case X10_CMD_ALLUNITSOFF:
                                                        comp.cmd = X10_CMD_OFF;
                                                        break;
                                                case X10_CMD_ALLLIGHTSON:
                                                        comp.cmd = X10_CMD_ON;
                                                        break;
                                                case X10_CMD_ALLLIGHTSOFF:
                                                        comp.cmd = X10_CMD_OFF;
                                                        break;
                                        }
                                        // vuelta a agregar de dato
                                        blkcopy(c,(char*)&comp,sizeof(X10DB_DATA));
                                        c[sizeof(X10DB_DATA)] = 0;
                                        LISTA_push(X10DB_LISTA,c);
                                        i--;
                                        n--;
                                        // si era dirección única, salir
                                        if(data.dir.unidad != X10_INVALIDO)
                                                return X10DB_OK;
                                }
                        }
                }
        }
        // no existe dirección en base de datos
        if(data.dir.unidad != X10_INVALIDO){
                switch(data.cmd){
                        case X10_CMD_ON:
                                break;
```

```
                        case X10_CMD_OFF:
                                break;
                        case X10_CMD_ALLUNITSOFF:
                                data.cmd = X10_CMD_OFF;
                                break;
                        case X10_CMD_ALLLIGHTSON:
                                data.cmd = X10_CMD_ON;
                                break;
                        case X10_CMD_ALLLIGHTSOFF:
                                data.cmd = X10_CMD_OFF;
                                break;
                        default:
                                return X10DB_ERROR;
                                break;
                }
                blkcopy(c,(char*)&data,sizeof(X10DB_DATA));
                c[sizeof(X10DB_DATA)] = 0;
                LISTA_push(X10DB_LISTA,c);
        }
        return  X10DB_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Buscar el estado de un módulo ===========================================================
X10_CODIGO      X10DB_lookup(X10_DIR modulo){
        int                             n,
                                        i;
        X10DB_DATA              comp;
        char                    c[sizeof(X10DB_DATA) + 1];

        n = LISTA_size(X10DB_LISTA);
        for(i = 1; i <= n; i++)
                if(LISTA_item(X10DB_LISTA, i, c) == LISTA_OK){
                        blkcopy((char*)&comp,c,sizeof(X10DB_DATA));
                        if( blkequ((char*)&modulo,(char*)&comp,sizeof(X10_DIR)) )
                                return comp.cmd;
                }
        return X10_INVALIDO;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Buscar el estado de un elemento de la lista ================================================
X10DB_STATUS    X10DB_item(int item, X10DB_DATA *data){
        int                             n;
        char                    c[sizeof(X10DB_DATA) + 1];

        n = LISTA_size(X10DB_LISTA);
        if(item > n)
                return X10DB_ERROR;
        if(LISTA_item(X10DB_LISTA, item,c) == LISTA_OK){
                blkcopy((char*) data, c, sizeof(X10DB_DATA));
                return X10DB_OK;
        }
        return X10DB_ERROR;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

6. NOTIFICACIÓN DE EVENTOS

a. avisar.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/


#include <kernel.h>
#include <network.h>

#include "abcmisc.h"
#include "listas.h"
#include "x10.h"
#include "smtp_abc.h"
#include "alias.h"
#include "rtc.h"

#ifndef _AVISAR_H_
        #define _AVISAR_H_

// codigos de estado
        #define          NTFY_OK                          0
        #define          NTFY_ERROR                      -1
        #define          NTFY_NOT_FOUND                  -2

// definiciones

// definición de estructuras y tipos
        typedef          BYTE                            NTFY_STATUS;
        typedef          struct  {
                X10_CMD          x10;
                char*            email;
        }                                                NTFY_DATA;

// prototipo de funciones de interfaz. ("public:")
        NTFY_STATUS      NTFY_init();                    // inicializa la lista de avisos
        NTFY_STATUS      NTFY_lookup(X10_CMD);           // revisa si debe darse aviso de evento
        NTFY_STATUS      NTFY_add(NTFY_DATA);            // agrega notificación
        NTFY_STATUS      NTFY_delete(NTFY_DATA);         // elimina notificación

// prototipo de funciones de libreria ("private:")


#endif
```

b. avisar.c

```
#include "avisar.h"

extern Bool      FORMATO_HTML;

const char       NTFY_END_HTML[]          =        "</div><br><br><hr><div style='color:black;"\
                                                   "font-family:Arial;"                        \
                                                   "font-style:italic;"                        \
                                                   "font-size:9pt'>"                           \
                                                   "Si no desea ser notificado envie un”       \
                                                   "comando de este tipo:"                     \
                                                   "<br>     "        \
                                                   "     "            \
                                                   "<span style='font-family:Courier;"         \
                                                   "font-style:normal;"                        \
                                                   "color:darkorange'>"                        \
                                                   "No avisar cuando [modulo] se"              \
                                                   " [encienda|apague]</span>"                 \
                                                   "<br>Si desea agregar notificaciones"       \
                                                   " de otros modulos:<br>"                    \
                                                   "     "            \
                                                   "     "            \
                                                   "<span style='font-family:Courier;"         \
                                                   "font-style:normal;"                        \
                                                   "color:darkorange'>"                        \
                                                   "Avisar cuando [modulo] se "                \
                                                   "[encienda|apague]</span></div>"            ;

const char       NTFY_END_TEXT[]          =        "\r\n\r\n\r\n"                              \
                                                   "Si no desea ser notificado envie "         \
                                                   "un comando de este tipo:"                  \
                                                   "\r\n\tNo avisar cuando [modulo] se "       \
                                                   "[encienda|apague]\r\n"                     \
                                                   "Si desea agregar notificaciones "          \
                                                   "de otros modulos: \r\n"                    \
                                                   "\tAvisar cuando [modulo] se "              \
                                                   "[encienda|apague]\r\n\r\n"                 ;

// Listas
LID              NTFY_LIST;
extern LID       POP3_VALID;

// == Inicializa la lista ===================================================================
NTFY_STATUS      NTFY_init(){
        LISTA_init(&NTFY_LIST);
        return  NTFY_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Revisa si debe darse notificación de un evento y la envía ==================================
NTFY_STATUS      NTFY_lookup(X10_CMD data){
        int                     i,
                                n,
                                m;
        NTFY_DATA       comp;
        char            c[128];
        char*           alias;
        Bool            thtml;

        thtml = FORMATO_HTML;

        n = LISTA_size(NTFY_LIST);
        for(i = 1;i <= n;i++){
                LISTA_item(NTFY_LIST, i, c);
                blkcopy( (char*) &comp, c,sizeof(X10_CMD));
                comp.email = &c[sizeof(X10_CMD)];
```

```
                if(     ( comp.x10.dir.casa == data.dir.casa )
                        &&      (       (comp.x10.dir.unidad == data.dir.unidad)
                                 ||     (data.dir.unidad == X10_INVALIDO) )
                        &&      ( comp.x10.cmd == data.cmd) ) {

                        SMTP_reset(comp.email,"Notificación",thtml);
                        if(thtml)
                                SMTP_addln("<div style='font-weight:bold;font-style:normal'>" \
                                                "     "             \
                                                "     "
                                );
                        if(thtml)
                                SMTP_add("El módulo ");
                        if((thtml)&&(data.cmd == X10_CMD_ON))
                                SMTP_addln("<span style='color:yellowgreen;font-size:150%'><i>");
                        if((thtml)&&(data.cmd == X10_CMD_OFF))
                                SMTP_addln("<span style='color:firebrick;font-size:150%'><i>");
                        alias = ALIAS_get_name(data.dir.casa,data.dir.unidad);
                        if(alias != NULL)
                                SMTP_add(alias);
                        else{
                                xc_sprintf(c,"%c%d",data.dir.casa - 32,data.dir.unidad);
                                SMTP_add(c);
                        }
                        if(thtml)
                                SMTP_addln("</i></span>");
                        if(data.cmd == X10_CMD_ON)
                                SMTP_addln(" se ha encendido/abierto.");
                        if(data.cmd == X10_CMD_OFF)
                                SMTP_addln(" se ha apagado/cerrado.");
                        if(thtml)
                                SMTP_addln(NTFY_END_HTML);
                        else
                                SMTP_addln(NTFY_END_TEXT);

                        SMTP_send();

                        return NTFY_OK;
                }
        }
        return  NTFY_NOT_FOUND;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
// === Agrega una notificación ================================================================
NTFY_STATUS     NTFY_add(NTFY_DATA data){
        char    c[128];
        char*   p;

        blkcopy(c,&data,sizeof(X10_CMD) );
        p = data.email;
        blkcopy( &c[sizeof(X10_CMD)], p, strlen(p) + 1 );
        LISTA_push(NTFY_LIST,c);

        return NTFY_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
// === Elimina una notificación ===============================================================
NTFY_STATUS     NTFY_delete(NTFY_DATA data){
        char    c[128];
        char*   p;

        blkcopy(c,&data,sizeof(X10_CMD) );
        p = data.email;
        blkcopy( &c[sizeof(X10_CMD)], p, strlen(p) + 1 );

        if( LISTA_borrar(NTFY_LIST,c) != LISTA_OK )
                return  NTFY_ERROR;
        return NTFY_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

7. PROGRAMACIÓN DE COMANDOS

a. programar.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>

#include "abcmisc.h"
#include "listas.h"
#include "x10.h"
#include "smtp_abc.h"
#include "alias.h"
#include "rtc.h"

#ifndef _PROGRAMAR_H_
        #define _PROGRAMAR_H_

// codigos de estado
        #define  PROG_OK                          0
        #define  PROG_ERROR                      -1
        #define PROG_NOT_FOUND                   -2

// definiciones
        #define  PROG_ZERO                        0x24

// definición de estructuras y tipos
        typedef BYTE                             PROG_STATUS;
        typedef struct {
                RTC_TIEMPO      tiempo;
                X10_CMD         x10;
                Bool            persiste;
        }                                        PROG_DATA;

// prototipo de funciones de interfaz. ("public:")
        PROG_STATUS     PROG_init(void);                 // Inicializar Lista
        PROG_STATUS     PROG_lookup(RTC_TIEMPO);         // Revisar si existe orden para un tiempo
        PROG_STATUS     PROG_add(PROG_DATA);             // agregar orden
        PROG_STATUS     PROG_delete(PROG_DATA);          // borrar orden

// prototipo de funciones de libreria ("private:")

#endif
```

b. programar.c

```
#include "programar.h"

// Listas
        LID             PROG_LIST;
extern  LID             POP3_VALID;


// Pipe para envio de comandos x10
extern  MPID    PIPE_X10_OUT;

// === Crear e inicializar lista ======================================================================
PROG_STATUS     PROG_init(void){

        LISTA_init(&PROG_LIST);
        return  PROG_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Buscar si existe orden para el “ahora” ==========================================================
PROG_STATUS     PROG_lookup(RTC_TIEMPO ahora){
        int                     i,
                                j,
                                n,
                                m;
        PROG_DATA       prog;
        char            c[128];
        char*           alias;
        HANDLE          orden;

        n = LISTA_size(PROG_LIST);
        for(i = 1;i <= n;i++){
                LISTA_item(PROG_LIST, i, c);
                m = strlen(c);
                for(j = 0;j < m; j++)
                        if(c[j] == PROG_ZERO)
                                c[j] = 0;
                blkcopy( (char*) &prog, c,sizeof(PROG_DATA));
                if( ( prog.tiempo.hora == ahora.hora )&&(prog.tiempo.minutos == ahora.minutos) ){
                        if( (prog.tiempo.dia_semana == ahora.dia_semana)
                                ||(prog.tiempo.dia_semana == -1) )
                        {
                                blkcopy((char*)&orden,(char*) &prog.x10,sizeof(X10_CMD));
                                psend(PIPE_X10_OUT,orden);
                                if( !prog.persiste ){
                                        PROG_delete(prog);
                                        i--;
                                        n--;
                                }
                        }
                }
        }
        return  PROG_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Agregar orden ===============================================================================
PROG_STATUS      PROG_add(PROG_DATA data){
        char            c[128];
        int                     i,
                                n;

        n = sizeof(PROG_DATA);

        blkcopy( c, &data, n );
        for(i = 0;i < n; i++)
                if(c[i] == 0)
                        c[i] = PROG_ZERO;
        c[n] = 0;

        LISTA_push(PROG_LIST,c);

        return PROG_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Borrar orden ================================================================================
PROG_STATUS      PROG_delete(PROG_DATA data){
        char            c[128];
        int                     i,
                                n;

        n = sizeof(PROG_DATA);

        blkcopy( c, &data, n );
        for(i = 0;i < n; i++)
                if(c[i] == 0)
                        c[i] = PROG_ZERO;
        c[n] = 0;

        if( LISTA_borrar(PROG_LIST,c) != LISTA_OK )
                return  PROG_ERROR;
        return PROG_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

8. LENGUAJE DE USUARIO

a. sinonimos.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

// arreglos de sinónimos
#ifndef _SINONIMOS_H_
        #define _SINONIMOS_H_

        // tamaños
        #define                  INTP_NO_SIN                     3
        #define                  INTP_ON_SIN                    21
        #define                  INTP_OFF_SIN                   18
        #define                  INTP_ALIAS_SIN                  5
        #define                  INTP_AVISAR_SIN                 8
        #define                  INTP_HORA_SIN                   2
        #define                  INTP_DIA_SIN                    2
        #define                  INTP_MAIL_SIN                  11
        #define                  INTP_STAT_SIN                  11
        #define                  INTP_FORMATO_SIN                1
        #define                  INTP_HTML_SIN                   1
        #define                  INTP_TEXTO_SIN                  1
        #define                  INTP_IGNORAR_SIN                8
        #define                  INTP_DIAS_SIN                   9

        // Arreglos de sinónimos
        extern  char*            INTP_NO[];
        extern  char*            INTP_ON[];
        extern  char*            INTP_OFF[];
        extern  char*            INTP_ALIAS[];
        extern  char*            INTP_AVISAR[];
        extern  char*            INTP_HORA[];
        extern  char*            INTP_DIA[];
        extern  char*            INTP_MAIL[];
        extern  char*            INTP_STAT[];
        extern  char*            INTP_FORMATO[];
        extern  char*            INTP_HTML[];
        extern  char*            INTP_TEXTO[];
        extern  char*            INTP_IGNORAR[];
        extern  char*            INTP_DIAS[];


#endif
```

b. sinonimos.c

```
#include "sinonimos.h"
// arreglos de sinónimos
        const   char*   INTP_NO[INTP_NO_SIN]     = {     "no",
                                                         "cancelar",
                                                         "quitar"
                                                 };

        const   char*   INTP_ON[INTP_ON_SIN]     = {     "encender",
                                                         "enciende",
                                                         "encende",
                                                         "enciendelo",
                                                         "enciendela"
                                                         "encendelo",
                                                         "encendela",
                                                         "encienda",
                                                         "abrir",
                                                         "abre",
                                                         "abri",
                                                         "abrelo",
                                                         "abrela",
                                                         "abrilo",
                                                         "abrila",
                                                         "abra",
                                                         "activar",
                                                         "activa",
                                                         "activalo",
                                                         "activala",
                                                         "active"
                                                 };

        const   char*   INTP_OFF[INTP_OFF_SIN]   = {     "apagar",
                                                         "apaga",
                                                         "apagalo",
                                                         "apagala",
                                                         "apague",
                                                         "cerrar",
                                                         "cierra",
                                                         "cerra",
                                                         "cierralo",
                                                         "cierrala",
                                                         "cerralo",
                                                         "cerrala",
                                                         "cierre",
                                                         "desactivar",
                                                         "desactiva",
                                                         "descativalo",
                                                         "descativala",
                                                         "desactive"
                                                 };

        const   char*   INTP_ALIAS[INTP_ALIAS_SIN] = {   "alias",
                                                         "nombrar",
                                                         "nombre",
                                                         "llama",
                                                         "ponele"
                                                 };

        const   char*   INTP_AVISAR[INTP_AVISAR_SIN]= {  "avisar",
                                                         "avisarme",
                                                         "avisa",
                                                         "informar",
                                                         "informarme",
                                                         "informa",
                                                         "saber"
                                                 };
```

```
        const   char*   INTP_HORA[INTP_HORA_SIN] = {     "hora",
                                                         "horas"
                                                 };

        const   char*   INTP_DIA[INTP_DIA_SIN]   = {     "dia",
                                                         "semana"
                                                 };


        const   char*   INTP_MAIL[INTP_MAIL_SIN] = {     "correo",
                                                         "correos"
                                                         "email",
                                                         "emails"
                                                         "mail",
                                                         "mails",
                                                         "usuario",
                                                         "usuarios",
                                                         "acceso",
                                                         "permiso",
                                                         "permisos"
                                                 };

        const   char*   INTP_STAT[INTP_STAT_SIN] = {     "estado",
                                                         "encendido",
                                                         "encendida",
                                                         "apagado",
                                                         "apagada",
                                                         "abierto",
                                                         "abierta",
                                                         "cerrado",
                                                         "cerrada",
                                                         "activado",
                                                         "activada"
                                                 };

        const   char*   INTP_FORMATO[INTP_FORMATO_SIN]= {"formato" };

        const   char*   INTP_HTML[INTP_HTML_SIN] = {     "html" };

        const   char*   INTP_TEXTO[INTP_TEXTO_SIN]= {    "texto" };

        const   char*   INTP_IGNORAR[INTP_IGNORAR_SIN]= {"como",
                                                         "cuando",
                                                         "porque",
                                                         "donde",
                                                         "tiene",
                                                         "tener",
                                                         "esta",
                                                         "puse"
                                                 };

        // posibilidades de días
        const   char*   INTP_DIAS[INTP_DIAS_SIN] = {     "lunes",
                                                         "martes",
                                                         "miercoles",
                                                         "jueves",
                                                         "viernes",
                                                         "sabado",
                                                         "domingo",
                                                         "sabados",
                                                         "domingos"
                                                 };
```

c. interprete.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>
#include "x10.h"
#include "alias.h"
#include "rtc.h"
#include "POP3.h"
#include "avisar.h"
#include "programar.h"
#include "sinonimos.h"
#include "smtp.h"
#include "x10db.h"

#ifndef _INTERPRETE_H_
        #define _INTERPRETE_H_

// códigos de estado
        #define  INTP_OK                          0
        #define  INTP_ERROR                      -1

// Tamaño de palabras ignoradas
        #define  INTP_IGNORE_SIZE  3

// tipos de tokens y tipos(INTP_TYPE)
        #define  INTP_TOKEN_ON            0x0001
        #define  INTP_TOKEN_OFF           0x0002
        #define  INTP_TOKEN_AVISAR        0x0004
        #define  INTP_TOKEN_ASK           0x0008
        #define INTP_TOKEN_HELP           0x0010
        #define  INTP_TOKEN_NO            0x0020
        #define  INTP_TOKEN_EMAIL         0x0040
        #define  INTP_TOKEN_FORMATO       0x0080
        #define INTP_TOKEN_HTML           0x0100
        #define INTP_TOKEN_TEXTO          0x0200
        #define  INTP_TOKEN_ALIAS         0x0400
        #define  INTP_TOKEN_STAT          0x0800

        #define INTP_TOKEN_PM             0x7001
        #define INTP_TOKEN_AM             0x7002


        #define  INTP_TOKEN_END           0x0FFF

        #define  INTP_TYPE_DIR            0x1001
        #define  INTP_TYPE_ALIAS          0x1004
        #define  INTP_TYPE_HORA           0x1008
        #define  INTP_TYPE_DIA            0x1010
        #define INTP_TYPE_MAIL            0x1020

        #define  INTP_TYPE_ASK_DIA        0x3001
        #define  INTP_TYPE_ASK_HORA       0x3004
        #define INTP_TYPE_ASK_MAIL        0x3008

// Tipos de preguntas
        #define INTP_ASK_WRONG             0
        #define  INTP_ASK_DIA              1
        #define  INTP_ASK_DIR              2
        #define  INTP_ASK_HORA             3
        #define  INTP_ASK_STAT             4
```

```
        #define INTP_ASK_MAIL            5
        #define INTP_ASK_ALIAS           6

// definición de tipos
        typedef  BYTE                   INTP_STATUS;
        typedef  int                    INTP_TYPE;

// prototipo de funciones de interfaz. ("public:")
        INTP_STATUS     INTP_init(void);                        // inicializa el interprete
        INTP_STATUS     INTP_interpretar_correo(void);          // interpreta un correo

// prototipo de funciones de libreria ("private:")
        INTP_STATUS     INTP_filtrar_letras( char* );           // elimina caracteres no deseados
        INTP_TYPE       INTP_linea( char* );                    // interpreta una línea de texto
        INTP_TYPE       INTP_token( char* );                    // interpreta un token
        BYTE            INTP_sinonimo( char*, int, char*, BYTE ); // comprueba si es sinónimo
        INTP_STATUS     INTP_programar( PROG_DATA );            // programa un evento
        INTP_STATUS     INTP_avisos( int );                     // interpreta avisos
        INTP_STATUS     INTP_error( char* );                    // error de comando
        INTP_STATUS     INTP_pregunta( void );                  // interpreta preguntas
#endif
```

d. interprete.c

```
#include "interprete.h"

extern  MPID    PIPE_X10_OUT;
extern  char    correo[POP3_MAIL_SIZE];
extern  char    POP3_FROM[];
extern  char*   RTC_DIAS[];
extern  Bool    FORMATO_HTML;

extern const char        FIN[5];

// constantes
const   char    INTP_TOP_HTML[]         =       "<style type='text/css'>"                \
                                                ".parrafo{"                              \
                                                "font-weight:bold;"                      \
                                                "font-style:normal;"                     \
                                                "left:10%"                               \
                                                "}"                                      \
                                                ".verde{"                                \
                                                "color:yellowgreen;"                     \
                                                "font-size:130%;"                        \
                                                "font-style:italic"                      \
                                                "}"                                      \
                                                ".rojo{"                                 \
                                                "color:firebrick;"                       \
                                                "font-size:130%;"                        \
                                                "font-style:italic"                      \
                                                "}"                                      \
                                                ".azul{"                                 \
                                                "color:navy;"                            \
                                                "font-size:130%;"                        \
                                                "font-style:italic"                      \
                                                "}"                                      \
                                                "</style>"                               ;

const   char    INTP_PARRAFO_START[]    =       "<hr><div class='parrafo'>";
const   char    INTP_PARRAFO_END[]      =       "</div>";

const   char    INTP_VERDE_START[]      =       "<span class='verde'>";
const   char    INTP_ROJO_START[]       =       "<span class='rojo'>";
const   char    INTP_AZUL_START[]       =       "<span class='azul'>";
const   char    INTP_COLOR_END[]        =       "</span>";

// Variables Globales
PROG_DATA       INTP_prog;
int             pregunta;
char            INTP_alias[128];
Bool            INTP_PM;
Bool            INTP_AM;
Bool            formato;

// FUNCIONES:

// === Inicializar el interprete ==================================================================
INTP_STATUS     INTP_init(void){
        INTP_prog.x10.dir.casa          = X10_INVALIDO;
        INTP_prog.x10.dir.unidad        = X10_INVALIDO;
        INTP_PM                         = FALSE;
        INTP_AM                         = FALSE;
        return INTP_OK;
}

// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Interpretar correo ==========================================================================
INTP_STATUS     INTP_interpretar_correo(void){
        int             line_num        =  1,
                        comando         = -1,
                        i               =  0,
                        j               =  0;

        char*           line_start      = correo;
        char*           next_line;
        unsigned int    orden;
        char            temp[128];
        char*           c;

        NTFY_DATA       aviso;

        INTP_filtrar_letras(line_start);

        formato = FORMATO_HTML;
        SMTP_reset(POP3_FROM,"Confirmación",formato);
        if(formato){
                SMTP_addln(INTP_TOP_HTML);
                SMTP_addln(INTP_AZUL_START);
        }
        SMTP_addln("Email recibido:");
        if(formato){
                SMTP_addln(INTP_COLOR_END);
                SMTP_addln("<br>");
        }
        while( comando != INTP_TOKEN_END){
                comando = 0;
                i = 0;
                while(line_start[i]!= 0x0A) i++;
                        next_line = &line_start[i+1];
                comando = INTP_linea(line_start);
                if(comando){
                        if(formato)
                                SMTP_addln(INTP_PARRAFO_START);
                }
                switch(comando){
                        case INTP_TOKEN_ON              :
                                if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                        if(INTP_prog.tiempo.minutos == -1){
                                                SMTP_add("Encender ");
                                                if(formato)
                                                        SMTP_addln(INTP_VERDE_START);
                                                c = ALIAS_get_name( INTP_prog.x10.dir.casa,
                                                                    INTP_prog.x10.dir.unidad);
                                                if(c != NULL)
                                                        xc_sprintf(temp,"%s",c);
                                                else
                                                        xc_sprintf(temp,"%c%d"
                                                                ,INTP_prog.x10.dir.casa-32
                                                                ,INTP_prog.x10.dir.unidad
                                                          );
                                                SMTP_addln(temp);
                                                if(formato)
                                                        SMTP_addln(INTP_COLOR_END);
                                                blkcopy((char*)&orden,
                                                        (char*) &INTP_prog.x10,
                                                        sizeof(X10_CMD));
                                                psend(PIPE_X10_OUT,(HANDLE)orden);
                                        }else
                                                INTP_programar(INTP_prog);
                                }else
                                        INTP_error("Dirección no válida");
                                break;
                        case INTP_TOKEN_OFF             :
                                if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                        if(INTP_prog.tiempo.minutos == -1){
                                                SMTP_add("Apagar ");
                                                if(formato)
```

```
                                        SMTP_addln(INTP_ROJO_START);
                                c = ALIAS_get_name( INTP_prog.x10.dir.casa,
                                                    INTP_prog.x10.dir.unidad);
                                if(c != NULL)
                                        xc_sprintf(temp,"%s",c);
                                else
                                        xc_sprintf(temp,"%c%d"
                                                ,INTP_prog.x10.dir.casa-32
                                                ,INTP_prog.x10.dir.unidad
                                          );
                                SMTP_addln(temp);
                                if(formato)
                                        SMTP_addln(INTP_COLOR_END);
                                blkcopy((char*)&orden,
                                        (char*) &INTP_prog.x10,
                                        sizeof(X10_CMD));
                                psend(PIPE_X10_OUT,(HANDLE)orden);
                        }else
                                INTP_programar(INTP_prog);
                }else
                        INTP_error("Dirección no válida");
                break;

        case INTP_TOKEN_ALIAS    :
                        if((INTP_prog.x10.dir.casa != X10_INVALIDO)&&
                           (INTP_alias[0] != 0) ){
                                ALIAS_push(      INTP_prog.x10.dir.casa,
                                                 INTP_prog.x10.dir.unidad,
                                                 INTP_alias);
                                SMTP_add("Nuevo alias: ");
                                xc_sprintf(      temp,"%c%d = %s "
                                                 ,INTP_prog.x10.dir.casa -32
                                                 ,INTP_prog.x10.dir.unidad
                                                 ,INTP_alias
                                        );
                                if(formato)
                                        SMTP_addln(INTP_AZUL_START);
                                SMTP_addln(temp);
                                if(formato)
                                        SMTP_addln(INTP_COLOR_END);
                        }else
                                INTP_error("Alias no especificado o ya en uso.");
                break;
        case (INTP_TOKEN_ALIAS | INTP_TOKEN_NO)   :
                        if((INTP_prog.x10.dir.casa != X10_INVALIDO)){
                                c = ALIAS_get_name(INTP_prog.x10.dir.casa,
                                                 INTP_prog.x10.dir.unidad);
                                if(c != NULL){
                                        xc_sprintf(      temp,"Alias eliminado ");

                                        SMTP_add(temp);
                                        if(formato)
                                                SMTP_addln(INTP_AZUL_START);
                                        xc_sprintf(temp,"(%c%d != %s) "
                                                ,INTP_prog.x10.dir.casa -32
                                                ,INTP_prog.x10.dir.unidad,
                                                c
                                        );

                                        SMTP_addln(temp);
                                        if(formato)
                                                SMTP_addln(INTP_COLOR_END);
                                        ALIAS_borrar_dir(INTP_prog.x10.dir.casa,
                                                INTP_prog.x10.dir.unidad);
                                }
                        }else
                                INTP_error("Alias a quitar no valido.");
                break;
        case INTP_TOKEN_ASK              :
                        INTP_pregunta();
                break;
```

```
                        case INTP_TOKEN_ASK      | INTP_TOKEN_STAT        :
                                        pregunta = INTP_ASK_STAT;
                                        INTP_pregunta();
                                break;
                        case INTP_TOKEN_ASK | INTP_TOKEN_ALIAS    :
                                        pregunta = INTP_ASK_ALIAS;
                                        INTP_pregunta();
                                        break;
                        case INTP_TOKEN_ASK      | INTP_TOKEN_ON  :
                                        pregunta = INTP_ASK_STAT;
                                        INTP_pregunta();
                                break;
                        case INTP_TOKEN_ASK      | INTP_TOKEN_OFF:
                                        pregunta = INTP_ASK_STAT;
                                        INTP_pregunta();
                                break;
                        case INTP_TOKEN_ASK
                                | INTP_TOKEN_ON
                                | INTP_TOKEN_OFF :
                                        pregunta = INTP_ASK_STAT;
                                        INTP_pregunta();
                                break;
                        case INTP_TOKEN_AVISAR | INTP_TOKEN_ON:
                                if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                        INTP_avisos(comando);
                                        aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                        aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                        aviso.x10.cmd            = X10_CMD_ON;
                                        aviso.email              = POP3_FROM;
                                        NTFY_add(aviso);
                                }else
                                        INTP_error("Dirección no válida");
                                break;
                        case INTP_TOKEN_AVISAR | INTP_TOKEN_OFF:
                                if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                        INTP_avisos(comando);

                                        aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                        aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                        aviso.x10.cmd            = X10_CMD_OFF;
                                        aviso.email              = POP3_FROM;
                                        NTFY_add(aviso);
                                }else
                                        INTP_error("Dirección no válida");
                                break;
                        case INTP_TOKEN_AVISAR | INTP_TOKEN_OFF | INTP_TOKEN_ON:
                                if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                        INTP_avisos(comando);
                                        aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                        aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                        aviso.x10.cmd            = X10_CMD_ON;
                                        aviso.email              = POP3_FROM;
                                        NTFY_add(aviso);
                                        aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                        aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                        aviso.x10.cmd            = X10_CMD_OFF;
                                        aviso.email              = POP3_FROM;
                                        NTFY_add(aviso);
                                }else
                                        INTP_error("Dirección no válida");
                                break;
                        case INTP_TOKEN_AVISAR | INTP_TOKEN_ON | INTP_TOKEN_NO:
                                if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                        INTP_avisos(comando);
                                        aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                        aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                        aviso.x10.cmd            = X10_CMD_ON;
                                        aviso.email              = POP3_FROM;
                                        NTFY_delete(aviso);
                                }else
                                        INTP_error("Dirección no válida");
```

```
                        break;
                case INTP_TOKEN_AVISAR | INTP_TOKEN_OFF | INTP_TOKEN_NO:
                        if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                INTP_avisos(comando);
                                aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                aviso.x10.cmd            = X10_CMD_OFF;
                                aviso.email              = POP3_FROM;

                                NTFY_delete(aviso);
                        }else
                                INTP_error("Dirección no válida");
                        break;
                case INTP_TOKEN_AVISAR | INTP_TOKEN_OFF | INTP_TOKEN_ON |INTP_TOKEN_NO:
                        if( INTP_prog.x10.dir.casa != X10_INVALIDO){
                                INTP_avisos(comando);
                                aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                aviso.x10.cmd            = X10_CMD_ON;
                                aviso.email              = POP3_FROM;
                                NTFY_delete(aviso);
                                aviso.x10.dir.casa       = INTP_prog.x10.dir.casa;
                                aviso.x10.dir.unidad     = INTP_prog.x10.dir.unidad;
                                aviso.x10.cmd            = X10_CMD_OFF;
                                aviso.email              = POP3_FROM;
                                NTFY_delete(aviso);
                        }else
                                INTP_error("Dirección no válida");
                        break;
                case INTP_TOKEN_FORMATO |INTP_TOKEN_HTML:
                        SMTP_add("Formato ");
                        xc_sprintf(      temp,"HTML");
                        if(formato)
                                SMTP_addln(INTP_AZUL_START);
                        SMTP_add(temp);
                        if(formato)
                                SMTP_addln(INTP_COLOR_END);
                        SMTP_addln(" para nuevos correos.");
                        FORMATO_HTML = TRUE;
                        break;
                case INTP_TOKEN_FORMATO |INTP_TOKEN_TEXTO:
                        SMTP_add("Formato ");
                        xc_sprintf(      temp,"TEXTO");
                        if(formato)
                                SMTP_addln(INTP_AZUL_START);
                        SMTP_add(temp);
                        if(formato)
                                SMTP_addln(INTP_COLOR_END);
                        SMTP_addln(" para nuevos correos.");
                        FORMATO_HTML = FALSE;
                        break;
                case INTP_TOKEN_FORMATO  :
                        INTP_error("Formato no especificado; se seguirá usando el formato
actual.");
                        break;
                case INTP_TOKEN_FORMATO
                        | INTP_TOKEN_HTML
                        | INTP_TOKEN_TEXTO       :
                        INTP_error("Solo puede usarse un formato a la vez.");
                        break;
                case INTP_TOKEN_END              :
                        if(formato)
                                SMTP_addln(INTP_ROJO_START);
                                SMTP_addln("Fin de Email.");
                        if(formato)
                                SMTP_addln(INTP_COLOR_END);
                        break;
                case 0:
                        break;
                default                          :
                        xc_sprintf(      temp,"Comando no reconocido.");
```

```
                                if(formato)
                                        SMTP_addln(INTP_ROJO_START);
                                SMTP_addln(temp);
                                if(formato)
                                        SMTP_addln(INTP_COLOR_END);
                                break;
                }
                if((comando)&&(formato))
                        SMTP_addln(INTP_PARRAFO_END);
                if( comando != INTP_TOKEN_END ){
                        line_start = next_line;
                        line_num++;
                }
        }
        SMTP_send();
        return  INTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Filtrar caracteres no deseados ==========================================================
#define INTP_BUF_OFFSET          200

INTP_STATUS             INTP_filtrar_letras(char* datos){
        int             r = 0;
        int             w = 0;
        BYTE    NUEVALINEA = 1;
        BYTE    ESPACIO = 0;

        // correr entrada para poder insertar caracteres
        r=0;
        while(!blkequ(&datos[r],FIN,sizeof(FIN)))
                r++;
        r += 6;
        for(w = r + INTP_BUF_OFFSET; r >= 0; datos[w--] = datos[r--]);

        // quitar caracteres no deseados
        r = INTP_BUF_OFFSET;
        w =0;
        while( !blkequ(&datos[r],FIN,sizeof(FIN)) ){

                if( (datos[r] >= 'A')&& (datos[r] <= 'Z') )
                                datos[r] += 32;
        // mayúsculas --> minúsculas

                switch(datos[r]){
                        // tildes minúsculas
                        case 'á':       datos[r] = 'a';  break;
                        case 'é':       datos[r] = 'e'; break;
                        case 'í':       datos[r] = 'i';  break;
                        case 'ó':       datos[r] = 'o'; break;
                        case 'ú':       datos[r] = 'u'; break;

                        // tildes mayúsculas (ANSI CP 1252)
                        case 'Á':       datos[r] = 'a';  break;
                        case 'É':       datos[r] = 'e'; break;
                        case 'Í':       datos[r] = 'i';  break;
                        case 'Ó':       datos[r] = 'o'; break;
                        case 'Ú':       datos[r] = 'u'; break;

                        // ñ Ñ
                        case 'ñ':       datos[r] = 'n'; break;
                        case 'Ñ':       datos[r] = 'n'; break;

                        case 0x0A:
                        // aceptar NL
                                datos[w++] = 0x0A;

                                break;
                        case 0x20:
                        // dejar pasar los espacios
                                datos[w++] = 0x20;
```

```
                        break;
                case '.':
                // convertir '.' a cambio de línea
                        datos[w++] = 0x0A;
                        break;
                case '?':
                // dejar pasar signo de pregunta
                        datos[w++] = ' ';
                        datos[w++] = '?';
                        datos[w++] = ' ';
                        break;
                case ':':
                // dos puntos solo entre números
                        if( (datos[r-1] < '0')||(datos[r-1] > '9')
                                ||(datos[r+1] < '0')||(datos[r+1] > '9') )
                                        datos[w++] = ' ';
                        else datos[w++] = ':';
                        break;
                case '_':
                        datos[w++] = '_';                               // aceptar '_'
                        break;
                case '@':
                        datos[w++] = '@';                               // aceptar '@'
                        break;
                case 'p':
        // separar pm de la hora.
                        if((datos[r+1] == 'm')||(datos[r+1] == 'M'))
                                if( (datos[r+2] == 0x0D)
                                        ||(datos[r+2] == ' ')
                                        ||(datos[r+2] == '.')
                                )
                                        if( (datos[r-1] >= '0')
                                                &&(datos[r-1] <= '9')
                                        )
                                                datos[w++] = ' ';
                        break;
                case 'a':
        // separar am de la hora.
                        if((datos[r+1] == 'm')||(datos[r+1] == 'M'))
                                if( (datos[r+2] == 0x0D)
                                        ||(datos[r+2] == ' ')
                                        ||(datos[r+2] == '.')
                                )
                                        if( (datos[r-1] >= '0')
                                                &&(datos[r-1] <= '9')
                                        )
                                                datos[w++] = ' ';
                        break;

        }

        if( (datos[r] >= '0')&&(datos[r] <= '9') )              // aceptar números
                datos[w++] = datos[r];
        if( (datos[r] >= 'a')&&(datos[r] <= 'z') )              // aceptar letras
                datos[w++] = datos[r];

        r++;
        // siguiente caracter de entrada
}
blkcopy(&datos[w],FIN,6);

r = 0;
w = 0;

// quitar espacios de más
while( !blkequ(&datos[r],FIN,sizeof(FIN)) ){
        if( (datos[r] == 0x20) ){
                if(NUEVALINEA)  r++;
                else if(ESPACIO){
                        r++;
                }else{
```

```
                                datos[w++] = datos[r++];
                                ESPACIO = 1;
                        }
                }else{
                        if( (datos[r] == 0x0A) ){
                                while(datos[w-1] == ' ') w--;
                                datos[w++]      = 0x0A;
                                ESPACIO = 0;
                                NUEVALINEA      = 1;
                                r++;
                        }else{
                                        datos[w++] = datos[r++];
                                        ESPACIO = 0;
                                        NUEVALINEA      = 0;
                        }
                }
        }
        datos[w]        = 0x0A;
        datos[w+1]      = '.';
        datos[w+2]      = 0x0A;
        datos[w+3]      = 0;



        return  INTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Interpretar una linea de comando ========================================================
INTP_TYPE INTP_linea(char* linea){
        int                     fin,
                                tipo,
                                comando = 0,
                                read = 0,
                                prev = 0;
        char*           c;

        for(fin = 0;(linea[fin] != 0x0A);fin++);
        fin--;

        INTP_alias[0] = 0;
        INTP_prog.x10.dir.casa          = X10_INVALIDO;
        INTP_prog.x10.dir.unidad = X10_INVALIDO;
        INTP_prog.x10.cmd                       = X10_INVALIDO;
        INTP_prog.tiempo.hora           = -1;
        INTP_prog.tiempo.minutos = -1;
        INTP_prog.tiempo.dia_semana     = -1;
        INTP_prog.persiste                      = FALSE;
        INTP_PM                                         = FALSE;
        INTP_AM                                         = FALSE;

        while(read <= fin){
                // interpretar token
                tipo = INTP_token(&linea[read]);
                // Fin de email?
                if(tipo == INTP_TOKEN_END)
                        return  INTP_TOKEN_END;

                // Banderas de Tokens
                if(!(tipo & 0x1000))
                        comando |= tipo;
                // token es un tipo de datos
                switch (tipo){
                        case INTP_TOKEN_ON              :
                                INTP_prog.x10.cmd       = X10_CMD_ON;
                                break;
                        case INTP_TOKEN_OFF             :
                                INTP_prog.x10.cmd       = X10_CMD_OFF;
                                break;
                        case INTP_TYPE_DIR              :
                                pregunta = INTP_ASK_DIR;
```

```
                                break;
                        case INTP_TYPE_DIA              :
                                if( blkequ(&linea[prev],"los",strlen(&linea[prev])) )
                                        INTP_prog.persiste = TRUE;
                                INTP_prog.tiempo.dia_semana = INTP_sinonimo(&linea[read],
                                                                        sizeof(&linea[read]),
                                                                        INTP_DIAS,INTP_DIAS_SIN);
                                if(INTP_prog.tiempo.dia_semana > 7)
                                        INTP_prog.tiempo.dia_semana -= 2;
                                pregunta = INTP_ASK_DIA;
                                break;
                        case INTP_TYPE_HORA             :
                                pregunta = INTP_ASK_HORA;
                                break;
                        case INTP_TOKEN_ALIAS   :
                                pregunta = INTP_ASK_ALIAS;
                                break;
                        case INTP_TYPE_ALIAS    :
                                blkcopy(INTP_alias, &linea[read], strlen(&linea[read])+1 );
                                break;
                        case INTP_TOKEN_PM              :
                                INTP_PM = TRUE;
                                break;
                        case INTP_TOKEN_AM              :
                                INTP_AM = TRUE;
                                break;
                        case INTP_TYPE_ASK_DIA  :
                                pregunta = INTP_ASK_DIA;
                                break;
                        case INTP_TYPE_ASK_HORA :
                                pregunta = INTP_ASK_HORA;
                                break;
                        case INTP_TOKEN_STAT :
                                pregunta = INTP_ASK_STAT;
                                break;
                        case INTP_TYPE_ASK_MAIL :
                                pregunta = INTP_ASK_MAIL;
                                break;
                }



                // siguiente token
                prev = read;
                for(;(linea[read]!= 0x0A)&&(linea[read] != ' ')&&(linea[read] != 0);read++);
                read++;
        }
        return comando;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Interpretar token ========================================================================
INTP_TYPE INTP_token( char* token ){
        int             i               = 0,
                        largo   = 0;
        char*           c;
        X10_DIR         dir;

        while( (token[largo] != 0x0A)&&(token[largo] != ' ')&&(token[largo] != 0) )
                largo++;
        token[largo] = 0;

        // Es el final del email?
        if(token[0] == '.')
                return INTP_TOKEN_END;

        // Se debe ignorar?
        if(INTP_sinonimo(token,largo,INTP_IGNORAR,INTP_IGNORAR_SIN))
                return  0;

        // Es una pregunta?
```

```
        if(token[0] == '?')
                return INTP_TOKEN_ASK;

        // Es una dirección?
        if( (token[0] >= 'a')&&(token[0] <= 'p') ){
                if( (largo == 2)||(largo == 3) ){
                        i = atoi(&token[1]);
                        if( (i > 0)&&(i < 17) ){
                                INTP_prog.x10.dir.casa   = token[0];
                                INTP_prog.x10.dir.unidad = i;
                                return  INTP_TYPE_DIR;
                        }
                }
        }

        // Es "PM"?
        if(largo == 2)
                if((token[0] == 'p')&&(token[1] == 'm')) return INTP_TOKEN_PM;

        // Es "AM"?
        if(largo == 2)
                if((token[0] == 'a')&&(token[1] == 'm')) return INTP_TOKEN_AM;

        // Sinónimo de no?
        if(INTP_sinonimo(token,largo,INTP_NO,INTP_NO_SIN))
                return INTP_TOKEN_NO;

        // Sinónimo de encender?
        if(INTP_sinonimo(token,largo,INTP_ON,INTP_ON_SIN))
                return INTP_TOKEN_ON;

        // Sinónimo de apagar?
        if(INTP_sinonimo(token,largo,INTP_OFF,INTP_OFF_SIN))
                return INTP_TOKEN_OFF;

        // Sinónimo de alias?
        if(INTP_sinonimo(token,largo,INTP_ALIAS,INTP_ALIAS_SIN))
                return INTP_TOKEN_ALIAS;

        // Sinónimo de avisar?
        if(INTP_sinonimo(token,largo,INTP_AVISAR,INTP_AVISAR_SIN))
                return INTP_TOKEN_AVISAR;

        // Sinónimo de formato?
        if(INTP_sinonimo(token,largo,INTP_FORMATO,INTP_FORMATO_SIN))
                return  INTP_TOKEN_FORMATO;

        // Sinónimo de texto?
        if(INTP_sinonimo(token,largo,INTP_TEXTO,INTP_TEXTO_SIN))
                return  INTP_TOKEN_TEXTO;

        // Sinónimo de html?
        if(INTP_sinonimo(token,largo,INTP_HTML,INTP_HTML_SIN))
                return  INTP_TOKEN_HTML;

        // Es un día?
        if(INTP_sinonimo(token,largo,INTP_DIAS,INTP_DIAS_SIN))
                return  INTP_TYPE_DIA;

        // Sinónimo de Hora?
        if(INTP_sinonimo(token,largo,INTP_HORA,INTP_HORA_SIN))
                return  INTP_TYPE_ASK_HORA;

        // Sinónimo de Día?
        if(INTP_sinonimo(token,largo,INTP_DIA,INTP_DIA_SIN))
                return  INTP_TYPE_ASK_DIA;

        // Sinónimo de Estado?
        if(INTP_sinonimo(token,largo,INTP_STAT,INTP_STAT_SIN))
                return  INTP_TOKEN_STAT;
```

```
        // Sinónimo de email?
        if(INTP_sinonimo(token,largo,INTP_MAIL,INTP_MAIL_SIN))
                return  INTP_TYPE_ASK_MAIL;

        // Es un alias?
        if(ALIAS_get_dir(token,&dir) == ALIAS_OK){
                INTP_prog.x10.dir.casa          = dir.casa;
                INTP_prog.x10.dir.unidad        = dir.unidad;
                return  INTP_TYPE_DIR;
        }
        // es hora?
        if( (largo > 3)&&(largo < 6) ){
                c = xc_index(token,':');
                if (c != NULL){
                        c[0] = 0;
                        c = &c[1];
                        i = atoi(token);
                        INTP_prog.tiempo.hora = i;
                        i = atoi(c);
                        INTP_prog.tiempo.minutos = i;
                        return  INTP_TYPE_HORA;
                }
        }

        // token muy pequeño? (normalmente palabras de relleno (el, la, los, se, etc.))
        if(largo <= INTP_IGNORE_SIZE)
                return 0;

        return INTP_TYPE_ALIAS;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Indica si una palabra es sinónimo de un token ============================================
BYTE INTP_sinonimo(char* token, int token_size, char* sinonimo[],BYTE sin_size){
        int     i,j,s;
        for(i=0;i<sin_size;i++){
                if(token_size > sizeof(sinonimo[i]) ) s = token_size;
                        else s = sizeof(sinonimo[i]);
                if (blkequ(token,sinonimo[i],s)) return i + 1;
        }
        return  0;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Programar evento ========================================================================
INTP_STATUS     INTP_programar(PROG_DATA data){
        char*                   c;
        char                    temp[30];

        if(INTP_PM)
                if(data.tiempo.hora < 12)
                        data.tiempo.hora += 12;
        if(INTP_AM){
                if(data.tiempo.hora == 12)
                        data.tiempo.hora = 0;
                if(data.tiempo.hora >12)
                        data.tiempo.hora -= 12;
        }
        if(data.x10.cmd == X10_CMD_ON){
                SMTP_add("Encender/Abrir ");
                if(formato)
                        SMTP_addln(INTP_VERDE_START);
        }
        else{
                SMTP_add("Apagar/Cerrar ");
                if(formato)
                        SMTP_addln(INTP_ROJO_START);
        }

        c = ALIAS_get_name(INTP_prog.x10.dir.casa,INTP_prog.x10.dir.unidad);
        if(c != NULL)
```

```
                xc_sprintf(temp,"%s ",c);
        else
                xc_sprintf(     temp,"%c%d "
                                        ,INTP_prog.x10.dir.casa-32
                                        ,INTP_prog.x10.dir.unidad
                                  );
        SMTP_add(temp);

        if(formato)
                SMTP_addln(INTP_COLOR_END);
        if(formato)
                SMTP_addln(INTP_AZUL_START);
        if(data.persiste){
                SMTP_add("siempre ");
        }
        if(data.tiempo.dia_semana != -1){
                xc_sprintf(temp,"%s ", RTC_DIAS[data.tiempo.dia_semana] );
                SMTP_add(temp);
        }
        if(formato)
                SMTP_addln(INTP_COLOR_END);
        if(data.tiempo.hora == 1){
                SMTP_add("a la ");
        }else{
                SMTP_add("a las ");
        }
        if(formato)
                SMTP_addln(INTP_AZUL_START);

        xc_sprintf(temp,"%d:",data.tiempo.hora);
        if(INTP_PM){
                if(data.tiempo.hora > 12)
                xc_sprintf(temp,"%d:",data.tiempo.hora -12);
        }
        if(INTP_AM){
                if(data.tiempo.hora == 0)
                        xc_sprintf(temp,"%d:",12);
        }

        SMTP_add(temp);
        if(data.tiempo.minutos < 9){
                SMTP_add("0");
        }
        xc_sprintf(temp,"%d ",data.tiempo.minutos);
        SMTP_add(temp);

        if(INTP_PM)
                SMTP_addln("PM");
        if(INTP_AM)
                SMTP_addln("AM");
        if((!INTP_PM)&&(!INTP_AM))
                SMTP_addln("Hrs.");

        if(formato)
                SMTP_addln(INTP_COLOR_END);

        PROG_add(data);

        return INTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Interpretar avisos ========================================================================

INTP_STATUS     INTP_avisos(int cmd){
        char*                   c;
        char                    temp[30];

        // Avisar o no avisar
        if(formato)
                SMTP_addln(INTP_AZUL_START);
```

```
        if(cmd & INTP_TOKEN_NO)
                SMTP_add("No avisar");
        else
                SMTP_add("Avisar");
        if(formato)
                SMTP_addln(INTP_COLOR_END);
        SMTP_add(" cuando ");
        // Encienda?
        if(cmd & INTP_TOKEN_ON){
                SMTP_add("encienda/abra ");
                if(formato)
                        SMTP_addln(INTP_VERDE_START);
        }
        // Apague y encienda?
        if( cmd & INTP_TOKEN_OFF )
                if( cmd & INTP_TOKEN_ON ){
                        if(formato)
                                SMTP_addln(INTP_COLOR_END);
                        SMTP_add("o ");
        }

        // Apague?
        if(cmd & INTP_TOKEN_OFF){
                SMTP_add("apague/cierre ");
                if(formato)
                        SMTP_addln(INTP_ROJO_START);
        }
        if( cmd & INTP_TOKEN_OFF )
                if( cmd & INTP_TOKEN_ON ){
                        if(formato){
                                SMTP_addln(INTP_COLOR_END);
                                SMTP_addln(INTP_AZUL_START);
                        }
        }

        c = ALIAS_get_name(INTP_prog.x10.dir.casa,INTP_prog.x10.dir.unidad);
        if(c != NULL)
                xc_sprintf(temp,"%s",c);
        else
                xc_sprintf(     temp,"%c%d"
                                        ,INTP_prog.x10.dir.casa-32
                                        ,INTP_prog.x10.dir.unidad
                                );
        SMTP_addln(temp);
        if(formato)
                SMTP_addln(INTP_COLOR_END);
        return INTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Error de comando ========================================================================
INTP_STATUS     INTP_error(char* err){
        if(formato)
                SMTP_addln(INTP_ROJO_START);
        SMTP_addln(err);
        if(formato)
                SMTP_addln(INTP_COLOR_END);
        return INTP_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Interpretar pregunta ====================================================================
INTP_STATUS     INTP_pregunta(){
        X10_DIR                 mod;
        X10DB_DATA              db;
        int                             i;
        char                    c[64];
        char                    temp[30];

                switch(pregunta){
                        case INTP_ASK_ALIAS     :
```

```
				if(formato)
					SMTP_addln("<ul>");
				i = 1;
				while(ALIAS_num(i++,c,&mod) == ALIAS_OK){
					if(formato)
						SMTP_addln("<li>");
					xc_sprintf(temp,"\t%c%d = %s",mod.casa,mod.unidad,c);
					SMTP_addln(temp);
					if(formato)
						SMTP_addln("</li>");
				}
				if(formato)
					SMTP_addln("</ul>");
				break;
			case INTP_ASK_STAT	:
				if(formato)
						SMTP_addln("<ul>");
				i = 1;
				while(X10DB_item(i++,&db) == ALIAS_OK){
					if(formato)
						SMTP_addln("<li>");
					xc_sprintf(temp,"\n\t\t%c%d = ",
						   db.dir.casa -32 ,db.dir.unidad);
					SMTP_add(temp);
					if(db.cmd == X10_CMD_ON){
						if(formato)
							SMTP_addln(INTP_VERDE_START);
						SMTP_addln("ON");
					}
					else{
						if(formato)
							SMTP_addln(INTP_ROJO_START);
						SMTP_addln("OFF");
					}
					if(formato){
						SMTP_addln(INTP_COLOR_END);
						SMTP_addln("</li>");
					}
				}
				if(formato)
					SMTP_addln("</ul>");
				break;
			default:
				INTP_error("Pregunta no reconocida.");
				break;
		}

}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

9. LOCALIZACIÓN DENTRO DE LA LAN

a. NetBIOS.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>


#ifndef _NETBIOS_H_
        #define _NETBIOS_H_

//codigos de estado
        #define  NBNS_OK                                 0
        #define  NBNS_ERROR                             -1

// definición de estructuras

        typedef struct {
                unsigned short int       NAME_TRN_ID;
                unsigned short int       NAME_FLAGS;
                unsigned short int       QDCOUNT;
                unsigned short int       ANCOUNT;
                unsigned short int       NSCOUNT;
                unsigned short int       ARCOUNT;
        }NBNS_HEADER;

        typedef struct   {
                NBNS_HEADER              header;
                char                     data[128];
        }NBNS_QUERY;

        typedef struct   {
                NBNS_HEADER              header;
                char                     nombre[34];
                unsigned short   int     tipo;
                unsigned short   int     clase;
                unsigned long    int     ttl;
                unsigned short   int     nb_dirs;
                unsigned short   int     nb_flags;
                unsigned long    int     nb_address;

        }NBNS_QUERY_RESPONSE;

// definición de tipos
        typedef  BYTE                                    NBNS_STATUS;

// prototipo de funciones de interfaz. ("public:")
        NBNS_STATUS              NBNS_recibir(void);              // espera mensajes NetBIOS

// prototipo de funciones de libreria ("private:")
        NBNS_STATUS              NBNS_name_response(void);        // responde a su nombre
        BYTE                     NBNS_name_match(void);           // compara el nombre
#endif
```

b. NetBIOS.c

```
#include "netbios.h"

// Nombre de HOST
const char       NBNS_NAME[32] = {"FIDBDAENEBEJEMCACACACACACACACAAA"};

// Identificadores de Dispositivos
DID              NBNS_SOCKET;

// Varaibles externas
extern struct BootInfo Bootrecord;

// Variables Globales
struct xgram                     NBNS_datagram;
NBNS_QUERY                               NBNS_q;
NBNS_QUERY_RESPONSE              NBNS_qr;

// === Espera a requests preguntando por su nombre y responde ===================================
NBNS_STATUS      NBNS_recibir(void){
        int      data_size;

        NBNS_qr.header.NAME_FLAGS                = 0x0085;
        NBNS_qr.header.QDCOUNT                   = 0x0000;
        NBNS_qr.header.ANCOUNT                   = 0x0100;
        NBNS_qr.header.NSCOUNT                   = 0x0000;
        NBNS_qr.header.ARCOUNT                   = 0x0000;
        NBNS_qr.nombre[0]                        = 0x20;
        blkcopy(&NBNS_qr.nombre[1],NBNS_NAME,32);
        NBNS_qr.nombre[33]                       = 0x00;
        NBNS_qr.tipo                             = 0x0020;        // NB
        NBNS_qr.clase                            = 0x0001;        // inet
        NBNS_qr.ttl                              = 0x50000000;
        NBNS_qr.nb_dirs                          = 0x0600;
        NBNS_qr.nb_flags                         = 0x0000;

        NBNS_SOCKET = open(UDP,ANYFPORT,(char*) 137 );
        do{
                data_size = read(NBNS_SOCKET,(char*)&NBNS_datagram,U_MAXLEN);
                if(data_size > 20){
                        blkcopy((char*)&NBNS_q, NBNS_datagram.xg_data, data_size);
                        if(NBNS_name_match())
                                NBNS_name_response();
                }
        }while(1);
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
// === Manda el paquete de respuesta ============================================================
NBNS_STATUS      NBNS_name_response(void){
        IPaddr   myIP;
        NBNS_qr.header.NAME_TRN_ID       = NBNS_q.header.NAME_TRN_ID;
        myIP                                        = dot2ip(Bootrecord.myip);
        NBNS_qr.nb_address                       = myIP;
        blkcopy(NBNS_datagram.xg_data, (char*) &NBNS_qr, sizeof(NBNS_QUERY_RESPONSE));
        write(NBNS_SOCKET,(char*) &NBNS_datagram, sizeof(NBNS_QUERY_RESPONSE));
        return   NBNS_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
// === Compara el nombre recibido ===============================================================
BYTE NBNS_name_match(void){
        if(!(NBNS_q.header.NAME_FLAGS & 0x00F8))
                if(NBNS_q.header.QDCOUNT)
                        if(NBNS_q.data[0] == 32)
                                if(blkequ(&NBNS_q.data[1],NBNS_NAME,32))
                                        return 1;
        return 0;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

10. RELOJ DEL SISTEMA

a. rtc.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>
#include <network.h>


#include "abcmisc.h"

#ifndef _RTC_H_
        #define _RTC_H_

        #include <date.h>
// codigos de estado
        #define         RTC_OK                   0
        #define         RTC_ERROR               -1

// definiciones
        #define         RTC_LOCK                 0x00
        #define         RTC_UNLOCK               0x01

// definición de estructuras y tipos
        typedef struct {
                BYTE    minutos;
                BYTE    hora;
                BYTE    dia_semana;
                BYTE    dia;
                BYTE    mes;
                BYTE    ano;
                BYTE    siglo;
        }                                       RTC_TIEMPO;

        typedef enum { LUN = 1,MAR,MIE,JUE,VIE,SAB,DOM } RTC_DIA;

        typedef  enum { ENE = 1,FEB,MRZ,ABR,MAY,JUN,JUL,AGO,SEP,OCT,NOV,DIC } RTC_MES;

        typedef         BYTE                    RTC_STATUS;

// prototipo de funciones de interfaz. ("public:")
        RTC_STATUS      RTC_init( void );               // Inicializa el reloj
        RTC_STATUS      RTC_auto( void );               // Sincroniza la hora por internet
        RTC_STATUS      RTC_set_tiempo( RTC_TIEMPO );   // Fija la hora
        RTC_STATUS      RTC_set_dia(RTC_DIA );          // Fija el día
        RTC_STATUS      RTC_timestamp( char* );         // Devuelve la hora
        RTC_STATUS      RTC_timestamp_short( char* );   // Devuelve la hora (formato corto)

// prototipo de funciones de libreria ("private:")


#endif
```

b. rtc.c

```
#include "rtc.h"


BYTE    msize[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
int             RTC_GMT_OFFSET = -6;

const   char*   RTC_DIAS[8]      = {     "",
                                                 "lunes",
                                         "martes",
                                         "miércoles",
                                         "jueves",
                                         "viernes",
                                         "sábado",
                                         "domingo"
                                 };
const   char*   RTC_DIAS_SHORT[8] = {    "",
                                         "Mon",
                                         "Tue",
                                         "Wed",
                                         "Thu",
                                         "Fri",
                                         "Sat",
                                         "Sun"
                                 };
const   char*   RTC_MESES[13]    = {     "",
                                         "enero",
                                         "febrero",
                                         "marzo",
                                         "abril",
                                         "mayo",
                                         "junio",
                                         "julio",
                                         "agosto",
                                         "septiembre",
                                         "octubre",
                                         "noviembre",
                                         "diciembre"
                                 };
const   char*   RTC_MESES_SHORT[13]={    "",
                                         "Jan",
                                         "Feb",
                                         "Mar",
                                         "Apr",
                                         "May",
                                         "Jun",
                                         "Jul",
                                         "Ago",
                                         "Sep",
                                         "Oct",
                                         "Nov",
                                         "Dic"
                                 };




// === Inicializa el reloj ========================================================================
RTC_STATUS      RTC_init(void){
        RTC_CTL =       0x00;
        return  RTC_OK;

}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === Sincroniza la Hora por Internet ============================================================
RTC_STATUS      RTC_auto(void){
        char                    c[50];
        unsigned long   t2;
        unsigned long   t;
        DID                             RTC_NTP;

        int                             ano                     = 0,
                                        i;
        BYTE                    dsemana = 0,
                                        dia                     = 0,
                                        mes                     = 0,
                                        hora            = 0,
                                        minutos         = 0,
                                        segundos= 0;
        long                    dias,
                                        tmp;
        IPaddr                  dirIP;

        do{
                dirIP = name2ip("time.nist.gov");
        }while(dirIP == SYSERR);
        xc_sprintf(c,"%u.%u.%u.%u:37" , (unsigned char)  dirIP
                                , (unsigned char) (dirIP >> 8)
                                , (unsigned char) (dirIP >> 16)
                                , (unsigned char) (dirIP >> 24));

        RTC_NTP = open(TCP,c,ANYLPORT);
                if(RTC_NTP < 0) return RTC_ERROR;
                i = read(RTC_NTP,(char*) &t2,sizeof(long));
                close(RTC_NTP);
                if( i != 4 )
                        return RTC_ERROR;
                t2      =         (( t2 & 0xFF000000) / 0x01000000)
                                | (( t2 & 0x00FF0000) / 0x00000100)
                                | (( t2 & 0x0000FF00) * 0x00000100)
                                | (( t2 & 0x000000FF) * 0x01000000);

                t2  = (unsigned long) ( t2 + (signed long)( RTC_GMT_OFFSET * SECPERHR));

        t = (unsigned long) t2;

        xc_ascdate(t2,c);

        if(iguales(c,"Mon",3)) dsemana = SAB;
        if(iguales(c,"Tue",3)) dsemana = DOM;
        if(iguales(c,"Wed",3)) dsemana = LUN;
        if(iguales(c,"Thu",3)) dsemana = MAR;
        if(iguales(c,"Fri",3)) dsemana = MIE;
        if(iguales(c,"Sat",3)) dsemana = JUE;
        if(iguales(c,"Sun",3)) dsemana = VIE;


        // ano
        for (ano=1900 ; TRUE ; ano++) {
                dias = isleap(ano) ? 366 : 365;
                tmp = dias * SECPERDY;
                if (tmp > (unsigned long)t)
                        break;
                t -= tmp;
        }

        // mes
        for (mes=0 ; mes<12 ; mes++) {
                tmp = msize[mes] * SECPERDY;
                if (tmp > t)
                        break;
                t -= tmp;
        }
```

```
        // dia
        dia = (int)(( t/SECPERDY ) + 1);
        t %= SECPERDY;

        // hora
        hora = (int) ( t/SECPERHR );
        t %= SECPERHR;

        // minutos
        minutos = (int) (t / SECPERMN);
        t %= SECPERMN;


        if(ano < 2000)
                return RTC_ERROR;

        RTC_CTRL = RTC_UNLOCK;

                RTC_CEN         =  ano / 100;
                RTC_YR          =  ano - (RTC_CEN * 100);
                RTC_MON         =  mes + 1;
                RTC_DOM         =  dia - 1;
                RTC_DOW         =  dsemana;
                RTC_HRS         =  hora;
                RTC_MIN         =  minutos;

        RTC_CTRL = RTC_LOCK;
        return RTC_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Fija la Hora ============================================================================
RTC_STATUS      RTC_set_tiempo(RTC_TIEMPO tiempo){
        if( (tiempo.hora > 23)||(tiempo.hora < 0 ) )
                return RTC_ERROR;
        if( (tiempo.minutos > 59 )||(tiempo.minutos < 0 ) )
                return RTC_ERROR;
        RTC_CTL = RTC_UNLOCK;
                RTC_HRS = tiempo.hora;
                RTC_MIN = tiempo.minutos;
        RTC_CTL = RTC_LOCK;

}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Fija el Día =============================================================================
RTC_STATUS      RTC_set_dia(RTC_DIA     dia){
        if( (dia < LUN )||(dia > DOM ) )
                return RTC_ERROR;
        RTC_CTL = RTC_UNLOCK;
                RTC_DOW = dia;
        RTC_CTL = RTC_LOCK;

        return  RTC_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Devuelve el tiempo ======================================================================
RTC_STATUS      RTC_timestamp(char* ts){
        int i;

        xc_sprintf(&ts[0],"%s, %2u de %s del %2u" , RTC_DIAS[RTC_DOW]

        , RTC_DOM

        , RTC_MESES[RTC_MON]

        , RTC_CEN );
        i = strlen(ts);
        if(RTC_YR < 9)
```

```
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u %2u:", RTC_YR, RTC_HRS);

        i = strlen(ts);
        if(RTC_MIN < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u:", RTC_MIN);

        i = strlen(ts);
        if(RTC_SEC < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u [GMT",RTC_SEC);
        i = strlen(ts);
        if(RTC_GMT_OFFSET >= 0)
                xc_sprintf(&ts[i++],"+");
        xc_sprintf(&ts[i],"%d]", RTC_GMT_OFFSET);
        return  RTC_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Devuelve el tiempo (formato corto) ========================================================
RTC_STATUS      RTC_timestamp_short(char* ts){
        int i;

        xc_sprintf(ts,"%s, ",RTC_DIAS_SHORT[RTC_DOW]);

        i = strlen(ts);
        if(RTC_DOM < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u %s %d", RTC_DOM, RTC_MESES_SHORT[RTC_MON] ,RTC_CEN );

        i = strlen(ts);
        if(RTC_YR < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u ", RTC_YR);

        i = strlen(ts);
        if(RTC_HRS < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u:", RTC_HRS);

        i = strlen(ts);
        if(RTC_MIN < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u:", RTC_MIN);

        i = strlen(ts);
        if(RTC_SEC < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%u ",RTC_SEC);

        i = strlen(ts);
        if(RTC_GMT_OFFSET >= 0)
                xc_sprintf(&ts[i++],"+");
        else
                xc_sprintf(&ts[i++],"-");
        if(abs(RTC_GMT_OFFSET) < 9)
                xc_sprintf(&ts[i++],"0");
        xc_sprintf(&ts[i],"%d00", abs(RTC_GMT_OFFSET));

        return  RTC_OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

11. PROGRAMA PRINCIPAL

a. main.c

```
#include <kernel.h>
#include <bootinfo.h>
#include <network.h>
#include <http.h>
#include <shell.h>
#include <netapp.h>
#include <smtp.h>

#include "pop3.h"
#include "x10.h"
#include "alias.h"
#include "interprete.h"
#include "smtp_abc.h"
#include "netbios.h"
#include "avisar.h"
#include "programar.h"
#include "x10db.h"

// Definiciones
#define         INTERVALO_POP3          10                      // en segundos
#define         INTERVALO_RTC           300                     // en segundos

// Información original para dirección de Red (DHCP está activado al inicio)

 struct BootInfo Bootrecord = {
        "192.168.25.125",        /* Default IP address */
        "192.168.25.1",          /* Default Gateway */
        "192.43.244.18",         /* Default Timer Server */
        "192.168.25.4",          /* Default File Server - Not currently Used */
        "",
        "216.230.128.3",         /* Default Name Server */
        "",
        0xffffff00UL             /* Default Subnet Mask */
        };

// Identificadores de Procesos
PID             PROCESO_READ_MAIL;                      // Lectura POP3 de correos
PID             PROCESO_INTERPRETE;                     // Interpretar email válido
PID             PROCESO_X10_INPUT;                      // Entrada de comandos X10
PID             PROCESO_X10_OUTPUT;                     // Envio de comandos X10
PID             PROCESO_NBNS;                           // NetBIOS Name Service reply
PID             PROCESO_RTC;                            // Sincroniza el reloj del sistema
PID             PROCESO_PROG;                           // Realiza los comandos programados

// Identificador de puertos de mensajes
MPID    PIPE_X10_OUT;

// Variables globales
Bool    FORMATO_HTML;

// Variables externas
extern  char            correo[POP3_MAIL_SIZE];
extern  Webpage         web[];
// === Inerprete de correos ======================================================================
int     interprete(VOID){
        while(1){
                INTP_interpretar_correo();
                resume(PROCESO_READ_MAIL);
                suspend(getpid());
        }
        return OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
// === ordena la lectura de mensajes y su interpretación ========================================
int mensajes(void){
        POP3_init();
        INTP_init();

        while(1){
                if(POP3_check() == POP3_DATA){
                        resume(PROCESO_INTERPRETE);
                        suspend(getpid());
                }
                sleep(INTERVALO_POP3);
        }
        return SYSERR;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Lee los comandos de la red X10 ===========================================================
int     entrada_x10(void){
        X10_CMD         input;

        while(1){
                if( X10_recibir(&input.dir.casa,&input.dir.unidad,&input.cmd) == 0){
                        X10DB_event(input);
                        NTFY_lookup(input);
                }
        }
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === envía los mensajes X10 ===================================================================
int     salida_x10(void){
        unsigned int            mensaje;
        BYTE                    cmd[3];

        while(1){
                mensaje = (unsigned int) preceive(PIPE_X10_OUT);
                cmd[0]  = (mensaje & 0x0000FF);
                cmd[1]  = (mensaje & 0x00FF00) / 0x000100;
                cmd[2]  = (mensaje & 0xFF0000) / 0x010000;
                X10_enviar(cmd[0], cmd[1], cmd[2]);
        }
        return OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// === Sincroniza el reloj del sistema ==========================================================
int update_RTC(void){
        RTC_init();
        while(1){
                RTC_auto();
                sleep(INTERVALO_RTC);
        }
        return SYSERR;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// Revisa si hay acciones programadas ===========================================================
int prog_check(void){
        RTC_TIEMPO      ahora;
        ahora.dia = ahora.mes = ahora.ano = ahora.siglo = 0;

        while(1){
                ahora.dia_semana = RTC_DOW;             // Dia actual
                ahora.hora       = RTC_HRS;             // Hora actual

                if(ahora.minutos != RTC_MIN){           // cambiaron los minutos?
                        ahora.minutos    = RTC_MIN;
                        PROG_lookup(ahora);             // revisar programaciones
                }
                sleep(10);                              // esperar 10 seg.
        }
```

```
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// ### Rutina Principal ###########################################################################
SYSCALL main(void){
        STATWORD estado;

        http_init (http_defmethods, httpdefheaders, web, 80);     // Iniciar servidor Web
        shell_add_commands(netcmds, nnetcmds);

// Inicializaciones
        ALIAS_init();
        X10_init();
        SMTP_init();
        NTFY_init();
        PROG_init();
        X10DB_init();
        FORMATO_HTML     = TRUE;


        // crear Pipe paracomandos X10 con capacidad para 10 comandos antes de bloquear
        PIPE_X10_OUT     = pcreate(20);

// declaración de hilos
        PROCESO_READ_MAIL        = create((procptr)mensajes,      3072,20,"Correo POP3",   0);
        PROCESO_INTERPRETE       = create((procptr)interprete,    5120,20,"Interprete",    0);
        PROCESO_X10_INPUT        = create((procptr)entrada_x10,   1024,20,"Entrada X10",   0);
        PROCESO_X10_OUTPUT       = create((procptr)salida_x10,    1024,20,"Salida X10",    0);
        PROCESO_NBNS             = create((procptr)NBNS_recibir,  1024,20,"NetBIOS name",  0);
        PROCESO_RTC              = create((procptr)update_RTC,    1024,20,"Update RTC",    0);
        PROCESO_PROG             = create((procptr)prog_check,    1024,20,"Programación",  0);
// Iniciar ejecución de hilos
        resume(PROCESO_X10_INPUT);
        resume(PROCESO_X10_OUTPUT);
        resume(PROCESO_READ_MAIL);
        resume(PROCESO_NBNS);
        resume(PROCESO_RTC);
        resume(PROCESO_PROG);
// Suspender la rutina principal
        suspend(getpid());
        while(1);
        return OK;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

12. Otras Funciones

a. abcmisc.h

```
/*
Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Electrónica
Trabajo de Graduación
Alejandro Balbás Contreras       99030
*/

#include <kernel.h>

#ifndef _ABCMISC_H_
        #define _ABCMISC_H_

// definición de tipos


// codigos de estado

        #define IGUALES                          1
        #define NO_IGUALES                       0


// prototipo de funciones de interfaz. ("public:")
        int     iguales(char* a, char* b, int n);         // compara cadenas (no distingue may/min)
        char*   abc_index(char* a, int b, int n);         // busca una cadena dentro de otra

// prototipo de funciones de librería. ("private:")


#endif
```

b. abcmisc.c

```
#include "abcmisc.h"

// === Comparación de cadenas ===================================================================
int iguales(char* a, char* b, int n){
        int i;
        for( i=0;i <n;i++)
                if(     ( a[i] != b[i]) && ( a[i] != (b[i] + 32)) &&( a[i] != (b[i] - 32)) )
                        return NO_IGUALES;
        return IGUALES;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


// === Buscar caracter en cadena ================================================================
char*   abc_index(char* a, int b, int n){
        int i;
        for(i=0;i <= n;i++)
                if(a[i] == b) return &a[i];
        return NULL;
}
// xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```