EBLOCKS2

Embedded Internet **Comunicaciones**

CP4895

MATRIX

www.matrixtsl.com

CP4895

Internet integrado

Notas del curso

Contenido

1 Primeros pasos

La siguiente información está diseñada para ayudarle a poner en marcha el entrenador de Internet E-blocks2.

1.1 Hardware necesario

Este documento está diseñado para su uso con los kits de formación en Internet E-blocks2 BL0531 o BL0535.

1.2 Pruebe la rutina

Las rutinas de prueba están disponibles para las tarjetas E-blocks2 en la sección de soporte E-blocks2 de la página web de Matrix: www.matrixtsl.com

1.3 Software necesario

Los programas de ejemplo suministrados en el CD de formación Embedded Internet requieren la instalación de Flowcode V8 o posterior en el PC host.

1.4 Software adicional

Se sugiere utilizar un analizador de tráfico de red para ayudar en el aprendizaje de TCP/IP, como Wireshark. La última versión puede descargarse gratuitamente de Internet.

1.5 Documentación

La documentación de las distintas tarjetas E-blocks2 y otros elementos suministrados como parte del kit de formación se encuentra en la sección E-blocks del sitio web de Matrix: www.matrixtsl.com.

1.6 Ejemplos y ejercicios

El CD de formación Embedded Internet contiene una carpeta de programas de ejemplo y ejercicios que pueden utilizarse junto con este documento

1.7 Descargo de responsabilidad

La información aquí contenida es correcta en el momento de su publicación, pero puede ser sustituida o modificada posteriormente. Si esta información se sustituye, se publicará un documento de introducción revisado con el kit de formación de E-blocks2 Embedded Internet.

2 Introducción

El curso de formación sobre Internet integrado está diseñado para introducirle en los conceptos necesarios para comprender los protocolos de comunicación generalmente denominados TCP/IP, incluidos los protocolos Ethernet, TCP, IP, UDP y otros.

Este curso se lleva a cabo utilizando Flowcode V8 o posterior y utiliza el componente TCP/IP que tiene una serie de propiedades y macros para permitir al estudiante construir el sistema que necesite. Esto permite a los estudiantes aprender acerca de TCP / IP sin empantanarse en los problemas de programación en C o un lenguaje de bajo nivel.

2.1 A través de este curso

Este curso recorre las distintas capas de TCP/IP, comenzando por la capa de comunicación física más baja hasta llegar a las comunicaciones de datos de aplicación. El curso cubre las diferentes capas y protocolos TCP/IP y los modos de componentes Flowcode TCP/IP utilizados para implementarlos.

Se incluyen varios ejercicios junto con notas sobre cómo implementarlos en Flowcode. Para los primeros protocolos, las notas son relativamente extensas y constituyen una especie de recorrido del ejercicio. En los ejercicios posteriores, las notas se centran más en explicar los detalles y las diferencias entre el protocolo y los anteriores. A partir de ese momento, el alumno debería tener un conocimiento suficiente de los conceptos básicos para realizar la tarea sin necesidad de apuntes extensos. Se proporcionan ejemplos de soluciones para los ejercicios de demostración, discusión y uso como punto de partida para la programación posterior.

2.2 A quién va dirigido este curso

Este curso está dirigido a dos grupos principales: Técnicos en electrónica que deseen utilizar las comunicaciones TCP/IP y Técnicos en redes que deseen comprender la estructura de datos TCP/IP.

- Los técnicos en electrónica suelen tener cierta experiencia con microcontroladores y se ocupan sobre todo de implementar las comunicaciones TCP/IP desde un punto de vista práctico.
es decir, cómo crear programas y enviar datos. El objetivo general del técnico en electrónica será enviar mensajes.
Los técnicos de redes y los informáticos se preocuparán más por comprender el proceso de comunicación en sí, es decir, los datagramas y las tramas, y cómo se relacionan con los datos que se pueden ver con las herramientas de análisis de redes. El objetivo general del técnico de redes será comprender y depurar los mensajes de la red y solucionar los errores de comunicación.

Se atiende a ambos grupos por igual, cubriendo tanto los aspectos teóricos como los prácticos. Dependerá del profesor implicado poner el énfasis durante la enseñanza.

3 Conceptos básicos de TCP/IP

3.1 ¿Qué es TCP/IP?

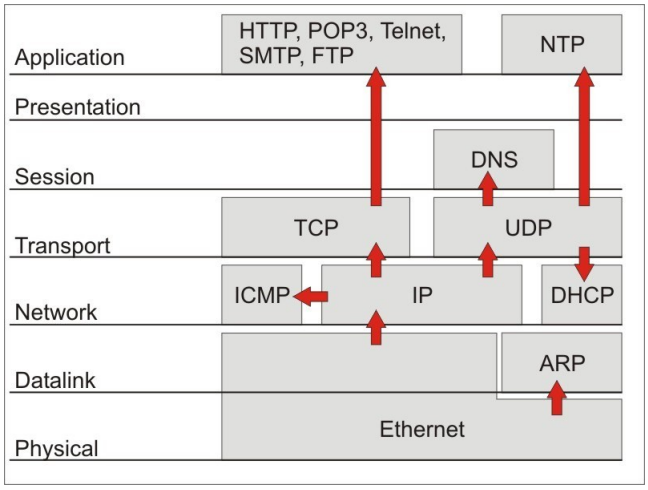
TCP significa Protocolo **de** Control de Transmisión. IP significa Protocolo de Internet. Otros acrónimos comunes que utilizaremos a lo largo del curso son: MAC - Control de Acceso a Medios, UDP - Protocolo **de** Datagramas de Usuario, ICMP - Protocolo de Mensajes de Control de Internet y ARP - Protocolo de Resolución de Direcciones. Se introducirán otros acrónimos a medida que aparezcan, y también se pueden encontrar en el glosario al final.

Podemos referirnos a TCP/IP tanto de forma general como específica. El término general TCP/IP se utiliza aquí para referirse al sistema de mensajería que transfiere datos entre ordenadores a través de Internet. El resultado final puede ser correo electrónico, páginas web o un archivo de datos. El proceso de transferencia del mensaje implicará una serie de protocolos de comunicación en capas. Uno de ellos puede ser TCP. Uno de ellos puede ser IP. Y hay otros: Ethernet, ICMP, UDP, etc. Aquí es donde entra la definición específica de TCP/IP, o más bien la definición específica de TCP y la definición específica de IP. TCP e IP son dos de los protocolos más importantes utilizados en las comunicaciones por Internet, de ahí el término general TCP/IP para describir todo el proceso.

Nota: Aquí utilizaremos *TCP/IP* para referirnos al sistema general, y *TCP* e *IP* para referirnos a los protocolos específicos.

3.2 OSI capas

TCP/IP es un sistema multicapa en el que las capas superiores se abstraen cada vez más de las comunicaciones de datos en serie de la primera capa física. En las capas física y de enlace de datos, lo más importante es enviar y recibir datos entre nodos específicos, normalmente un PC y un concentrador o enrutador. El nivel de red introduce sistemas para especificar el receptor final, lo que permite a la red decidir dónde van los datos y cómo llegan allí. Los protocolos de nivel superior, como TCP y UDP, se encargan de hacer llegar los datos a las aplicaciones informáticas que los utilizan. Aún más arriba en la estructura de niveles están las aplicaciones que pueden manejar los datos por nosotros, o iniciar el procedimiento de enviar o solicitar un mensaje por nosotros.



En las capas superiores se automatiza cada vez más el proceso de transmisión de datos. Por ejemplo, los usuarios no necesitan saber cómo los routers encontrarán el destinatario correcto para una transmisión de datos IP, sólo necesitan saber que los routers pueden hacerlo. Proporcione la dirección IP correcta y los datos llegarán allí, ya sea al otro lado de la habitación o del mundo.

En las capas de nivel superior se tiene cada vez más en cuenta la detección de errores y la secuenciación de la transmisión, y es probable que el sistema receptor espere un diálogo bidireccional con los datos que se están

y los códigos de error se devuelven en momentos determinados. La comprobación de estos códigos y el tratamiento de los posibles errores suponen un aumento correspondiente del procesamiento de datos.

Los protocolos de la capa de aplicación son las partes con las que normalmente asociamos TCP/IP (navegadores web, programas de correo electrónico, etc.). Muchos piensan que son TCP/IP, pero en realidad sólo manipulan y muestran los datos en bruto enviados por TCP/IP. Lo que ves en un programa de correo electrónico no es SMTP (Simple Mail Transfer Protocol), sino una representación visual de los datos enviados a través de TCP/IP en el formato SMTP.

3.3 Componente TCP/IP modos

El componente TCP/IP tiene cuatro modos que pueden utilizarse como puntos de entrada al modelo OSI.

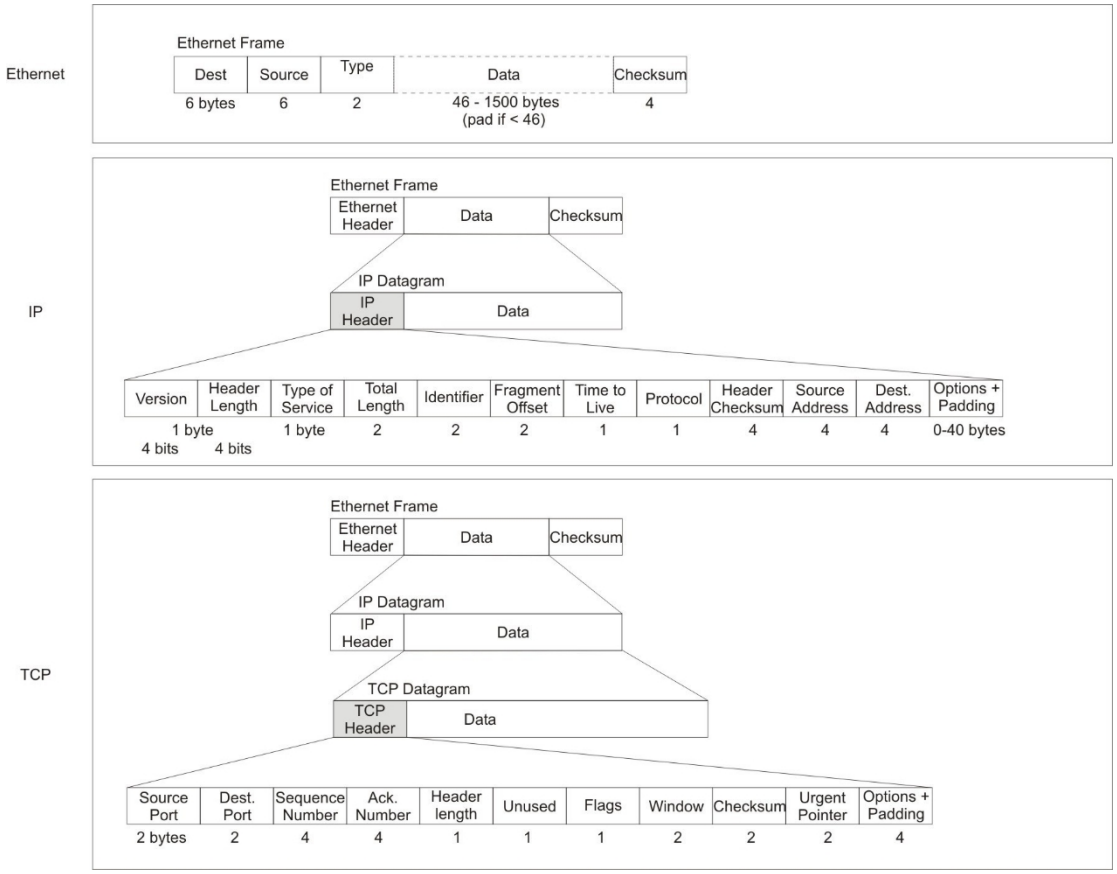
- Modo MAC Introduce al usuario en el nivel físico inferior.
- Modo IP Introduce al usuario en el nivel de Red. Gestiona automáticamente los procedimientos de la capa física.
- Modo UDP Introduce al usuario en el nivel de transporte utilizando el protocolo UDP. Útil para aplicaciones como DNS o DHCP que se basan en UDP.

Modo TCP Introduce al usuario en el nivel de transporte utilizando datagramas TCP. Útil como punto de partida para aplicaciones como SMTP o HTTP que se basan en TCP.

Los modos automatizan los niveles inferiores, por lo que la implementación del modo TCP, por ejemplo, gestionará automáticamente las capas de red (IP) y física (MAC). Las cabeceras también se gestionan automáticamente en todos los modos excepto en MAC, lo que te permite concentrarte más en los procesos de datos.

3.4 Tramas y datagramas - Mensajes dentro de mensajes

El mensaje más básico consiste en un flujo de datos en serie con una sección de cabecera que proporciona información sobre los datos. Como todos los mensajes enviados en TCP/IP necesitan este nivel básico, se necesita un mecanismo para adaptarlo a los muchos propósitos y formatos necesarios para las diferentes capas de protocolo. El mecanismo elegido fue colocar el mensaje destinado a una capa de protocolo superior en la sección de datos de una capa inferior. Un mensaje TCP se coloca en la sección de datos de un mensaje IP, que a su vez se coloca en la sección de datos de una sección Ethernet. A medida que el mensaje asciende por las capas, estas envolturas externas pueden descartarse hasta que el mensaje llega finalmente al nivel que debe manejarlo.



El gráfico anterior muestra el formato de varios protocolos diferentes. El gráfico también muestra cómo los datagramas pueden superponerse a otros datagramas.

La capa física de Ethernet

Con el modo Ethernet se trata de una comunicación directa entre un dispositivo y otro. Utilizando un sistema de correo como metáfora, las comunicaciones Ethernet serían el equivalente a los casilleros de una oficina donde se puede poner el correo para otros y recoger el que le envían a uno. Sin embargo, para muchos sistemas TCP/IP necesitamos poder comunicarnos con cualquier punto de la red sin importar dónde se encuentre. El modo IP entra aquí proporcionando el sobre con la dirección franqueada, que puede ser entregado a la dirección escrita en él, siempre y cuando esa dirección
exista. Los routers son como las oficinas de correos, con sus furgonetas de reparto y sus carteros. Reciben el correo, comprueban la dirección y lo envían a la siguiente parte del sistema, y así sucesivamente, hasta que llega a su destino. Que llegue en tren, en furgoneta o en avión dependerá de la oficina de correos y de cómo crea que debe gestionarse el correo. Del mismo modo, los encaminadores decidirán qué ruta seguir para enviar los mensajes. Puede variar de un mensaje a otro, pero al igual que no importa si una carta ha ido en tren o no, no necesitamos saber cómo han hecho su trabajo los encaminadores, sólo que lo han hecho.

A la red con IP

El datagrama IP (la palabra "datagrama" deriva sin duda de "telegrama", lo que se suma a la metáfora del sistema de correo) se encaja dentro de una trama Ethernet para permitir el envío físico del mensaje a la red. El datagrama IP contiene un archivo de cabecera y un paquete de datos. Al igual que el datagrama IP (el sobre) está metido dentro de una trama Ethernet que lo envuelve, la sección de datos es en sí misma un mensaje (la carta propiamente dicha). Este mensaje puede tener distintos formatos, como TCP, UDP, ICMP, etc.

Un paquete de la vida real contendría detalles como la dirección de entrega, la dirección del remitente, el tipo de franqueo (correo aéreo, clase 1st , etc.), despachos de aduanas, sellos postales, etc. Un datagrama IP contiene el mismo tipo de información en su sección de cabecera: Dirección IP de origen, dirección IP de destino, protocolo, etc.

IP se diferencia de Ethernet en que no se limita a una conexión directa. Ethernet sólo puede pasar el mensaje a otro dispositivo MAC; generalmente será uno de la misma red de área local. IP, sin embargo, va más allá. Se examina la dirección IP y el datagrama se transmite a nivel local o global, dependiendo de dónde se encuentre la dirección IP. Los dispositivos receptores saben cómo descifrar la dirección IP.

y cómo determinar el siguiente destino. Habrá toda una serie de sistemas que se pondrán en marcha para ayudar al paquete en su viaje, pero no los veremos, igual que no vemos a los mozos de equipaje ni a los conductores que llevan el correo por todo el país. Estos detalles de fondo no nos preocupan, sólo el hecho de que el mensaje llegará, a menos que haya algún tipo de problema.

Los servidores pueden no funcionar. Las direcciones pueden ser erróneas. En estos casos, el mensaje no siempre llega a su destino. La dirección del remitente forma parte de los datos enviados, por lo que si el destino está bloqueado o falta, puede enviarse un mensaje de error al remitente para informarle del problema.

3.4.3 Entrega de datos con TCP

En la capa TCP, el contenido del mensaje se entrega finalmente al destinatario; en este caso, una aplicación que puede leer, comprender y responder a los datos enviados. Los datos pueden ser un mensaje de correo electrónico o una página web. Sea cual sea el contenido de los datos, el trabajo de TCP es entregarlos a las manos adecuadas. En niveles superiores como el TCP, esto se hará a menudo en forma de diálogo bidireccional con distintas secuencias de comunicación. Algo así como si un mensajero tuviera que ir a recepción y rellenar un formulario de entrega.

Un punto importante a tener en cuenta aquí es que el contenido del mensaje es irrelevante; lo que cuenta es la comunicación: lo que dice realmente el correo electrónico o la página web no le importa a TCP - lo importante es hacer llegar los datos a la aplicación, y dejar que la aplicación se preocupe por el contenido, las fuentes, el formato y todas las demás cosas que hay que hacer con los datos. El trabajo de TCP es llevar los datos allí, eso es todo.

3.4.4 Aplicaciones y contenidos

La capa de aplicación incluye protocolos como SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) y HTTP (HyperText Transfer Protocol). Son los protocolos que dan formato a los datos brutos que se convierten en correo electrónico y páginas web, etc., en manos de las aplicaciones. Estos protocolos también pueden contener secuencias de transmisión y códigos de error para que el protocolo TCP los siga o actúe en consecuencia. Para terminar nuestra analogía con el correo, este es el punto en el que la carta se abre y se lee, el cerebro recibe y entiende el contenido de esa carta. TCP/IP ha hecho su trabajo y el mensaje ha llegado.

3.5 Otros protocolos

TCP es el principal protocolo utilizado para comunicarse, ya que generalmente contiene datos de aplicación, a diferencia de Ethernet e IP, que suelen ser el papel de envolver, no el mensaje final. Pero hay otros protocolos que podemos utilizar, incluyendo un par que se utilizan con las comunicaciones Ethernet e IP: estos nos permiten demostrar los protocolos de bajo nivel en acción. Estos protocolos de bajo nivel son a menudo protocolos especializados diseñados para ayudar en ciertas tareas comunes de TCP/IP como la verificación de direcciones. Por lo tanto, es útil comprenderlos en general. Los otros protocolos que trataremos en este curso incluyen:

3.5.1 ARP - Protocolo de resolución de direcciones

ARP se utiliza con el modo Ethernet para ayudar a verificar las conexiones MAC. Esto lo convierte en una herramienta útil para la comprobación de errores a nivel físico. Como forma el paquete de datos de una trama Ethernet es un protocolo útil para demostrar el modo Ethernet.

3.5.2 ICMP - Protocolo de mensajes de control de Internet

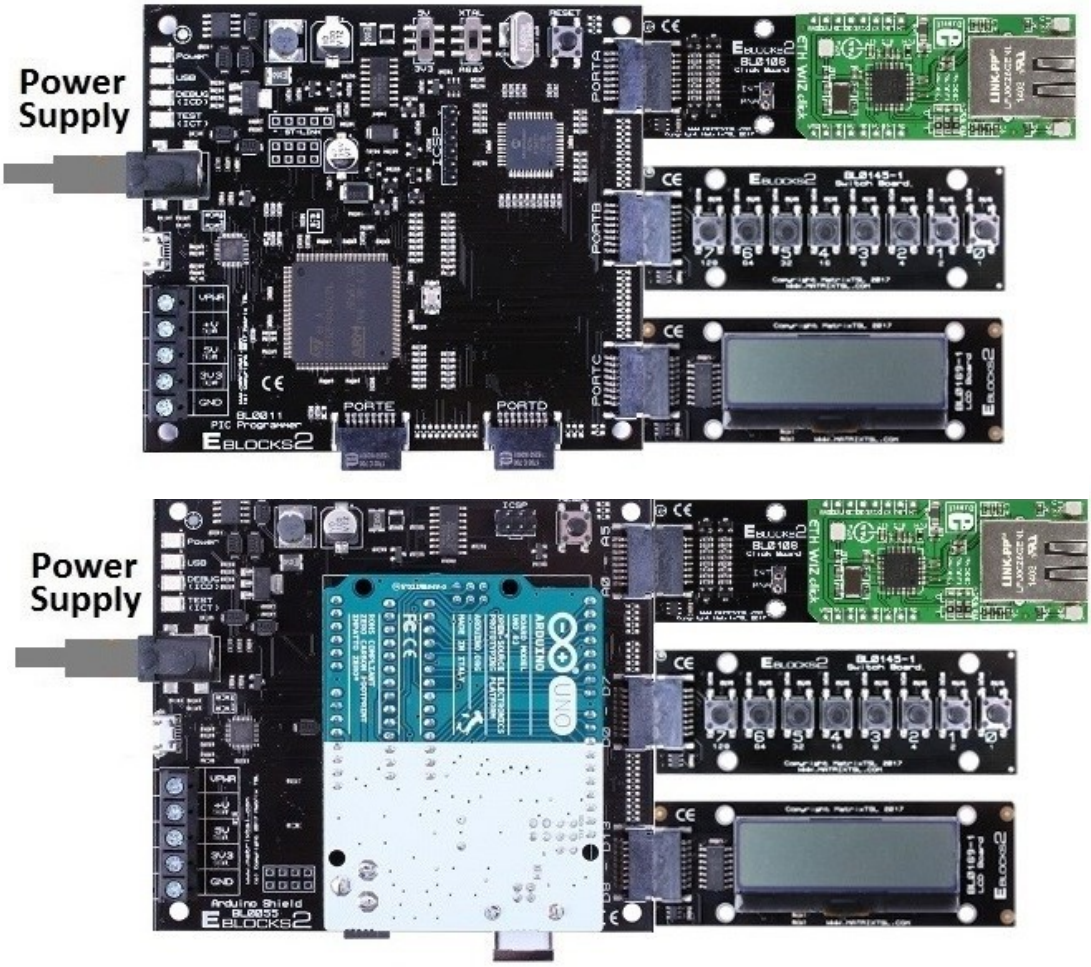
ICMP se utiliza a menudo para verificar direcciones IP, ya que puede contener información de comprobación de errores. El mensaje se envía y, si se recibe, se devuelve como un ping de sonar. Es posible que ya se haya encontrado con una implementación básica del protocolo ICMP en el programa de línea de comandos PING.

3.5.3 UDP - Protocolo de datagramas de usuario

UDP se parece a TCP en que contiene datos de aplicación. Pero carece de los sistemas de diálogo y comprobación de errores que utilizan las aplicaciones TCP. Es una forma de comunicación más directa que envía el mensaje directamente al destinatario. UDP puede ser muy útil para comunicaciones rápidas entre dos sistemas conocidos. Sin embargo, si el destinatario no está allí, o no está escuchando, las transmisiones UDP se perderán.

Configuración de E-Blocks2

A continuación se ilustra la configuración típica del sistema del entrenador de Internet E-Blocks2 para las versiones PIC y Arduino



Conexiones portuarias

BL0011 (PIC) Puerto	Puerto BL0055 (Arduino)	Tablero E-blocks2
A	A0-5	BL0106 Placa Click con ETH WIZ Click
B	D0-7	BL0145 Placa de interruptores
C	D8-13	BL0169 Placa LCD

Limitaciones prácticas del microcontrolador

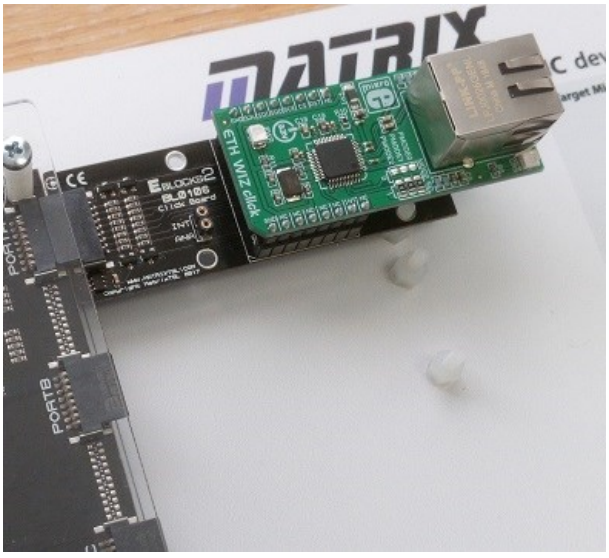
Puede utilizar el microcontrolador para enviar y recibir datos de comunicación TCP/IP, como correo electrónico o una página web. Sin embargo, hay un límite en el tamaño de los datos que se pueden enviar. Los archivos grandes y los adjuntos o imágenes serían simplemente demasiado grandes para ser almacenados en la propia memoria del microcontrolador.
Si bien la adición de dispositivos de memoria proporcionaría algo más de espacio de almacenamiento, debes aceptar las limitaciones de memoria del microcontrolador a la hora de desarrollar tus proyectos. Del mismo modo, tendrás que aceptar las limitaciones de visualización del sistema del microcontrolador. Una pantalla LCD estándar de 20 caracteres y 4 líneas no es tan sofisticada gráficamente como un monitor de PC moderno. Mientras que los mensajes de correo electrónico, por ejemplo, se pueden mostrar, es posible que tengan que ser mostrados línea por línea, y sin ningún tipo de formato.

Además, puede que no sea posible realizar varias funciones y utilizar varios protocolos y niveles de comunicación en el mismo programa debido a las limitaciones de memoria. Los microcontroladores tienen una memoria limitada y necesitan cierta capacidad de reserva para que el programa funcione realmente. El tamaño y la complejidad del código generado para soportar las macros de los componentes TCP/IP pueden ocupar gran parte del espacio de la ROM y la RAM, dejándole con una cantidad de memoria muy reducida en la que desarrollar su programa. Desafortunadamente, puede que sólo se entere de que su programa es demasiado grande para caber en la memoria cuando el ensamblador le diga que se ha quedado sin espacio.

El tablón de internet

El adaptador E-blocks2 Click y la tarjeta de Internet ETH WIZ Click están diseñados para utilizarse en
junto con la gama de placas programadoras Matrix E-blocks2.

El kit de formación en Internet embebido está disponible con las placas de procesador Microchip PIC o Arduino Uno, BL0531 y BL0535 respectivamente.

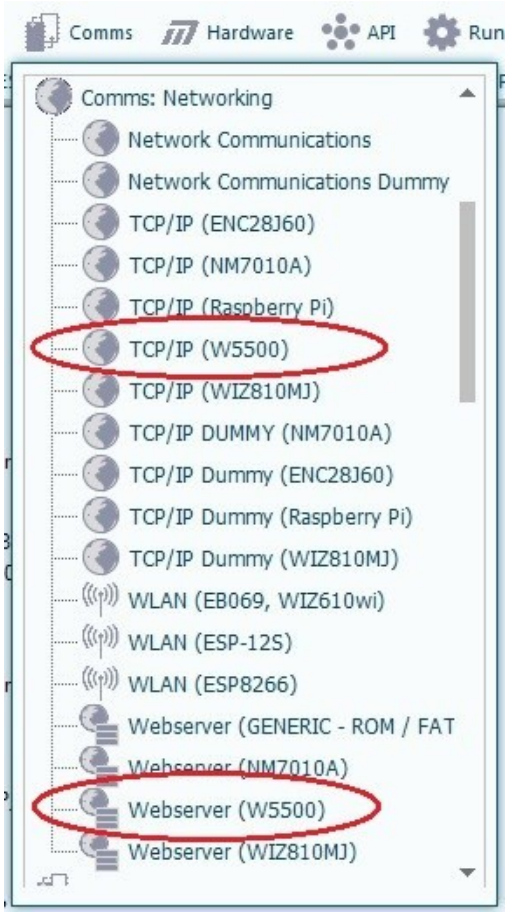


Configuración del tablero de Internet

En la mayoría de los ejemplos y ejercicios de este manual, la tarjeta de interfaz de Internet está conectada al puerto A de la tarjeta procesadora y debe estar conectada a través del cable Ethernet a otro equipo, como un PC o un conmutador de red. El ejemplo del cortafuegos utiliza dos tarjetas de interfaz de Internet.

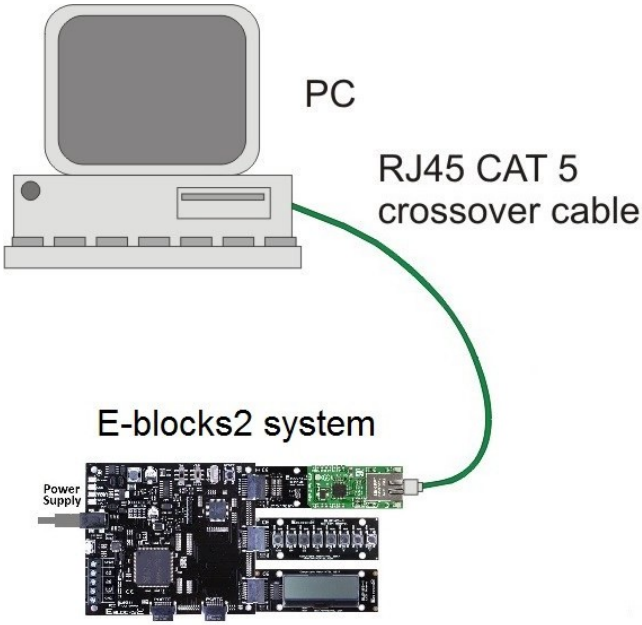
El panel del centro de desarrollo tiene montados pilares de soporte en ambas ubicaciones de los puertos para evitar el movimiento de la placa debido a la conexión de los cables ethernet. Hay que tener cuidado para evitar daños al colocar y retirar los Tablero E-blocks2 Click.

4.5 Flowcode y los componentes TCP/IP y Webserver



Los componentes TCP/IP y Webserver se encuentran en la sección Comms de la barra de herramientas Componentes. Se identifican por el dispositivo W5500 que se utiliza en el módulo ETH WIZ Click.

Se presupone un nivel razonable de competencia en la creación de programas Flowcode. En este documento se asume que el usuario está familiarizado con la adición y edición de macros y con el manejo de las propiedades de los componentes. Si el usuario necesita más formación sobre Flowcode, le recomendamos que revise los archivos del tutorial y que realice un curso gratuito de Flowcode (disponible en www.matrixtsl.com).



4.6 Configuración básica de la conexión directa

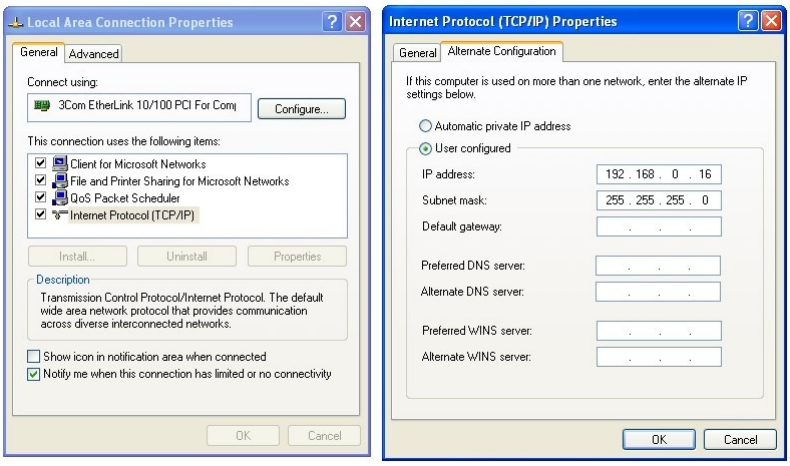
La tarjeta de Internet puede conectarse directamente a un PC mediante un cable cruzado RJ45 CAT 5 estándar que se suministra con la solución de Internet.
.

Es posible que tengas que configurar el PC para que tenga una dirección IP estática y no utilice servidores proxy.

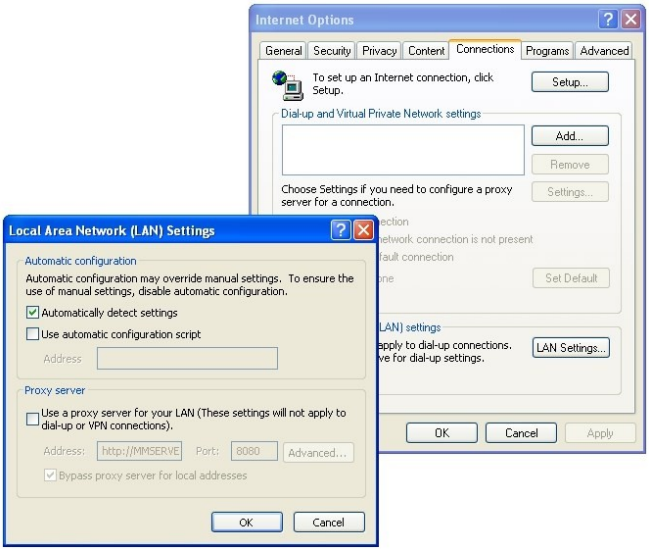
La dirección IP puede configurarse en la pantalla de diálogo Conexiones de red. Generalmente se recomienda una dirección 192.168.0.xxx como 192.168.0.16 para su uso con la placa de Internet. Por favor, anote los detalles de la dirección IP original antes de realizar cualquier cambio. En caso de duda, pida ayuda a un técnico de redes.

Tenga en cuenta que los archivos de ejemplo requerirán la configuración de la dirección IP de destino de los programas a una que haya seleccionado aquí.

Aquí hemos establecido la dirección IP para este PC en 192.168.0.16.

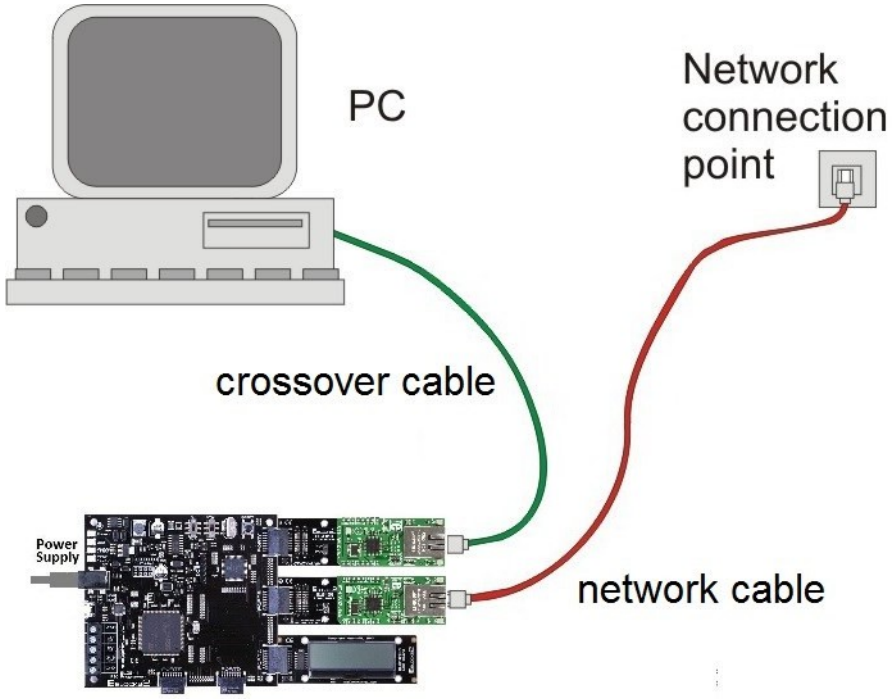


La configuración del servidor proxy LAN se encuentra en el Panel de Control de Opciones de Internet.
Asegúrese de que la opción Utilizar un servidor proxy no está marcada.



Sistema de doble placa Ethernet

Para algunas aplicaciones, como los cortafuegos, puede conectar dos tarjetas de Internet a la tarjeta programadora.

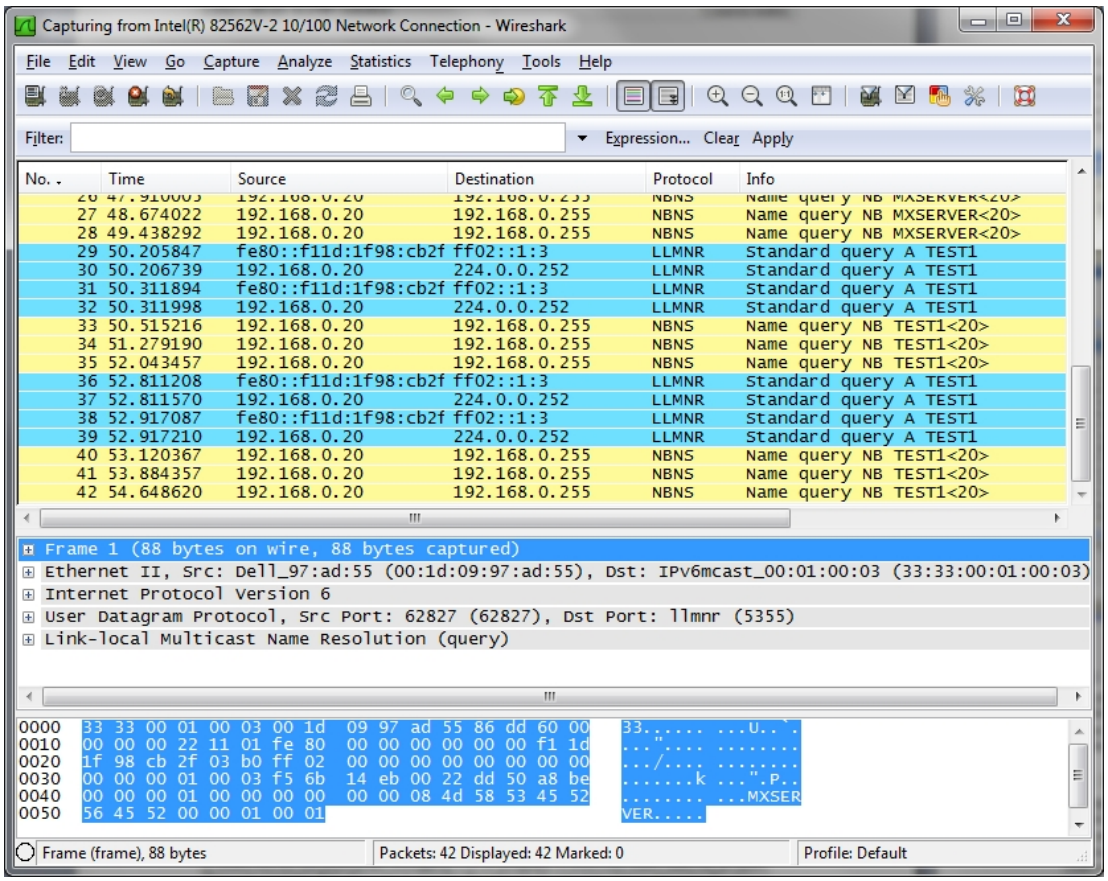


4.8 Software de supervisión de redes

Una herramienta útil para desarrollar y depurar programas TCP/IP es un programa analizador de tráfico de red. Se trata de programas que permiten monitorizar la red en busca de mensajes que se envían utilizando los distintos protocolos TCP/IP, como TCP, IP, UDP, etc. Pueden permitirle comprobar los datos que su programa está enviando o recibiendo para ver si son correctos. También pueden proporcionar información de error útil sobre cualquier mensaje malformado que esté enviando.
Hay una gran variedad de programas disponibles, como Wireshark.

Más adelante hablaremos del uso de Wireshark para ayudar a depurar programas.
La última versión está disponible en el sitio web de Wireshark: http://www.wireshark.org/download.html

La siguiente imagen está tomada de Wireshark. Observe que las distintas secciones le permiten seleccionar
para examinar y ver los distintos protocolos que contiene el mensaje.
Se muestran datos sobre los distintos elementos del protocolo, por ejemplo, "Type: IP (0x0800)" o los detalles de la dirección IP "Src Addr". También se pueden ampliar los distintos niveles y subcategorías de los datos en cuestión. Los datos originales en bruto también están disponibles en la parte inferior, tanto en formato hexadecimal como ASCII.

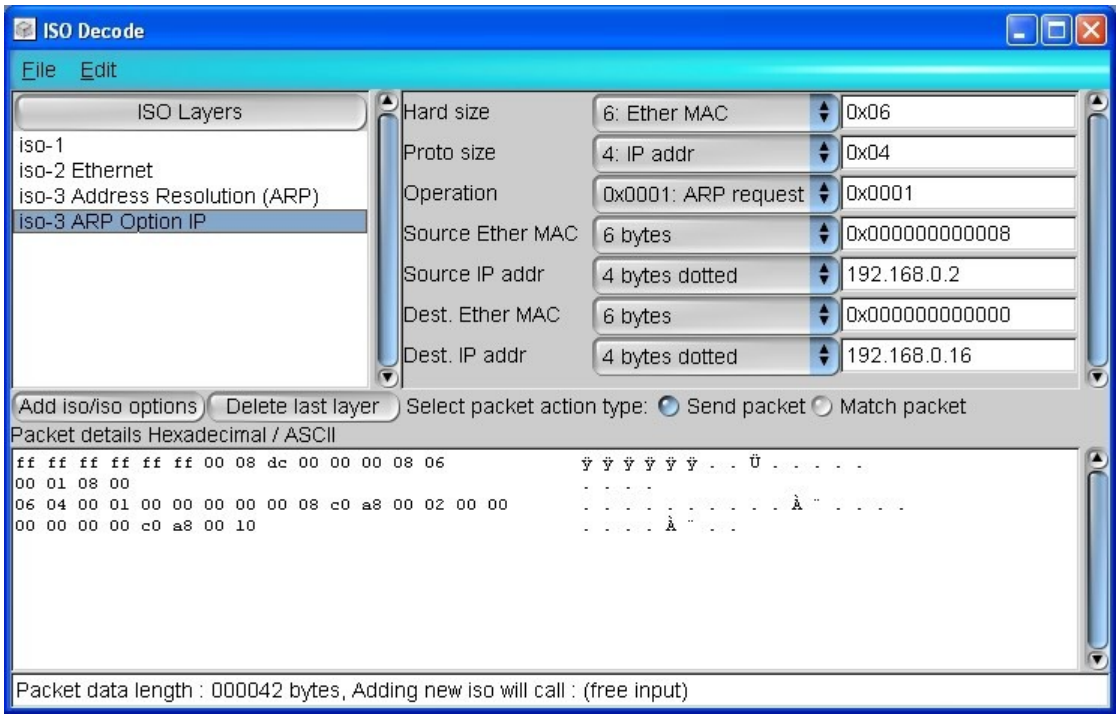


Depuración con herramientas de análisis del tráfico de red

Si el programa funciona a la primera, todos estamos contentos. Pero si se produce un error, sea culpa nuestra o no, la cosa se complica. Estamos bombeando datos a la red (o creemos que lo estamos haciendo) pero no podemos verlo. Aquí es donde entran en juego los analizadores de tráfico de red como Wireshark. Estas herramientas pueden supervisar, registrar y mostrar el tráfico de red.

Inyectores de paquetes

Lo contrario de una herramienta de monitorización de red, estas aplicaciones de software nos permiten crear y enviar paquetes personalizados. Esto nos permite crear mensajes de prueba que nuestros programas pueden recibir y procesar. Esto también puede ayudar con la depuración, ya que podemos modificar los mensajes enviados para ver qué efecto tienen en el programa, útil cuando se trata de averiguar si se trata de datos erróneos, o de un programa defectuoso.
Hay una gran variedad de programas disponibles; en el enlace siguiente se ofrecen diversas herramientas de inyección posibles:
http://wiki.wireshark.org/Tools#Traffic_generators



Depuración con inyectores de paquetes

Los inyectores de paquetes permiten crear un paquete o mensaje que puede enviarse al sistema. Los paquetes pueden guardarse para su uso futuro, y pueden editarse y enviarse de nuevo. Esto permite crear paquetes de prueba que pueden enviarse a programas de prueba. Si sospecha que se ha producido un error, puede modificar el paquete para comprobar el error y probar posibles soluciones.

Los inyectores de paquetes requieren un buen conocimiento de TCP/IP ya que todas las etapas del paquete requieren construcción. Sin embargo, un analizador de tráfico de red como Wireshark se puede utilizar junto con el inyector de paquetes para encontrar gran parte de la información necesaria para la construcción de paquetes.

Configuración del servidor

Si se conecta a su red local y encuentra dificultades, es posible que tenga que configurar la red para que acepte los mensajes producidos. Deberá consultar a su supervisor de red para realizar cualquier cambio de configuración.

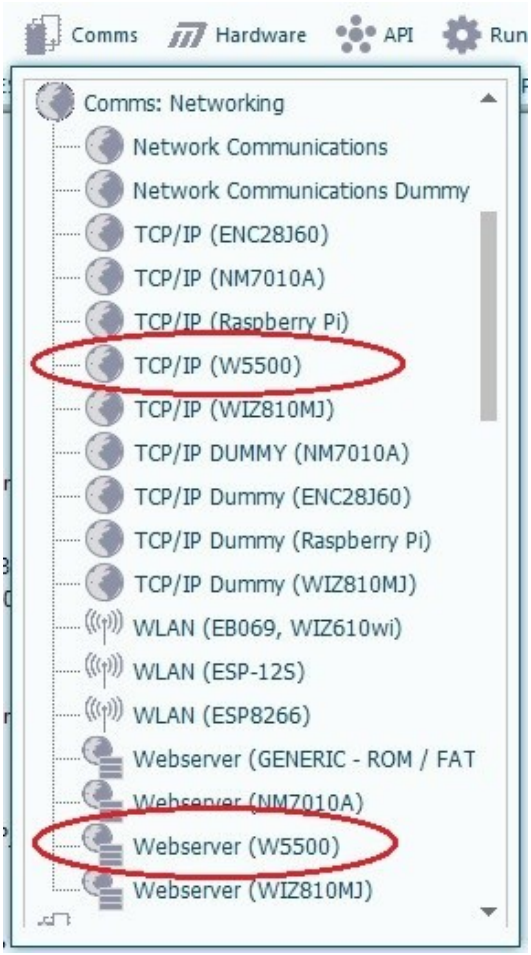
Las áreas que pueden necesitar configuración incluyen:

- Comprobación de la configuración IP de las tarjetas de Internet para evitar conflictos
- Direcciones de correo electrónico de prueba para mensajería SMTP.
- Puede ser necesario comprobar la configuración del cortafuegos si afecta a la mensajería. Esto puede implicar

configuración en el PC, el Servidor y en cualquier cortafuegos de software o hardware.

- Es posible que se requiera autorización para instalar herramientas de analizador de red
- Ajustes de configuración para permitir que las pizarras de Internet (con sus propias direcciones IP) sean visibles para el mundo exterior.

5 El componente TCP/IP



Para añadir un componente TCP/IP o Webserver a un diagrama de flujo, haga clic en el encabezado del menú Comunicaciones de la barra de herramientas Componentes y seleccione el icono correspondiente de la lista desplegable.
Busque las versiones basadas en W5500.

Icono del componente TCP/IP

Icono de componente de servidor web

Los componentes TCP/IP y Webserver tienen propiedades de conexión que deben configurarse en Flowcode y que se enumeran en la página 12 de este documento.

Propiedades de TCP/IP y del servidor web

Los detalles de las propiedades de los componentes TCP/IP y Webserver se tratan en los archivos de ayuda de los componentes, disponibles a través del botón Ayuda de las páginas de propiedades de los componentes.
La mayoría de las propiedades pueden dejarse con los valores por defecto, a menos que éstos entren en conflicto con otros ajustes del hardware.
por ejemplo, un conflicto de dirección IP. En los ejemplos en los que haya más de una placa, la configuración de la dirección MAC y de la dirección IP deberá ser diferente para cada placa.

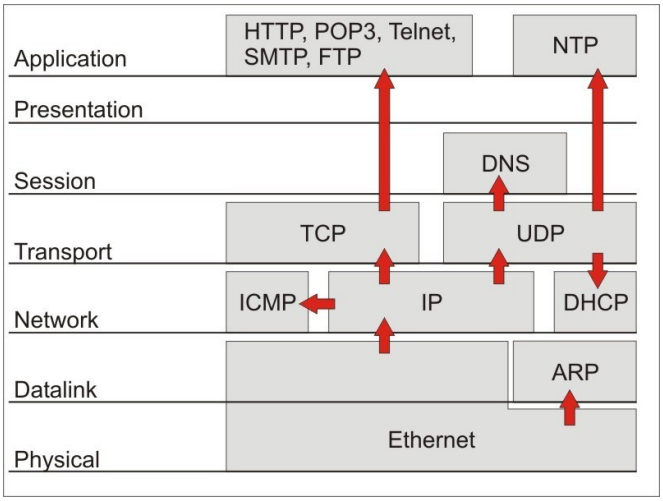
Macros de componentes TCP/IP y Webserver

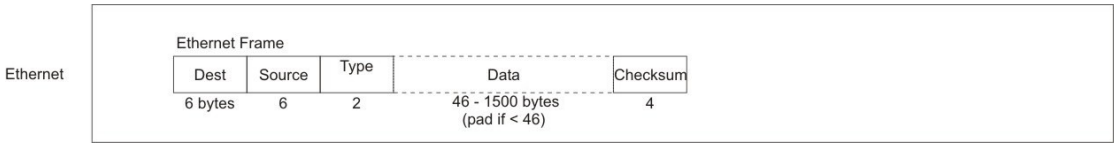
La mayor parte del trabajo se realiza mediante una serie de macros. Las macros, sus parámetros y una descripción de su funcionamiento se encuentran en los archivos de ayuda del componente. Deberá consultar los archivos de ayuda para obtener más información sobre las macros.

6 Ethernet capa

Visión general

La capa Ethernet es la capa física del modelo OSI. Se encarga de las comunicaciones entre el vicio de origen y los demás dispositivos físicos de la red, como concentradores o conmutadores. Todas las comunicaciones en una red basada en Ethernet se realizan como tramas Ethernet. Los protocolos de nivel superior transmiten su mensaje dentro de la sección de datos de una trama Ethernet. El nodo receptor extrae y procesa este mensaje de la trama. Pero el nivel básico fundamental de comunicación es a nivel físico de dispositivo MAC a dispositivo MAC.





La trama Ethernet consta de una sección de encabezado, una sección de datos y una sección de suma de comprobación.

La capa Ethernet también se conoce como capa MAC, por los dispositivos Media Access Controller que gestionan las comunicaciones. Cuando se trabaja con la capa Ethernet, los términos modo MAC y modo Ethernet pueden utilizarse de forma intercambiable. Cada dispositivo MAC tiene un identificador único de 6 bytes conocido como dirección MAC. Esto garantiza que no haya dos dispositivos en la red con la misma dirección MAC. Al pasar las direcciones MAC de origen y destino junto con los datos, se puede comprobar el error del mensaje y responder a él. Los tres primeros bytes se utilizan generalmente como identificador de la empresa, y los tres últimos bytes se utilizan para dar a cada dispositivo de la empresa un identificador único.

Normalmente, el identificador MAC se codifica en un dispositivo en el momento de su fabricación y no se puede cambiar. Sin embargo, las placas de Internet E- Blocks tienen un identificador MAC definible por el usuario. Esto permite configurar identificadores MAC específicos para pruebas y depuración. La dirección MAC predeterminada para la placa de Internet es: 0.8.220.0.0.0, siendo 0.8.220 el identificador de la empresa fabricante del dispositivo concreto utilizado en la placa de Internet.

> Si utiliza más de una tarjeta de Internet, debe asegurarse de que cada tarjeta tiene una dirección MAC única.
> De lo contrario, los protocolos de la capa Ethernet no podrán diferenciar entre placas, lo que provocará problemas.

La trama Ethernet consta de los siguientes elementos:

Marco Ethernet

	Explicación	**Bytes**
Destino	La dirección MAC de destino puede ser la dirección MAC de 6 bytes de un nodo conocido, o un dirección general como la dirección de difusión 255, 255, 255, 255, 255, 255.	6
Fuente	La dirección MAC de origen es la dirección MAC del nodo emisor.	6
Tipo	La sección de tipo puede variar dependiendo del protocolo que se esté utilizando. Por ejemplo el protocolo ARP tiene el tipo data: 8, 6.	2
Datos	La sección de datos puede tener entre 46 y 1500 bytes de longitud. Si la sección de datos es inferior a 46 bytes, deberá rellenarse con bytes adicionales para llegar a 46 bytes.	46 a 1500
Suma de comprobación	La trama Ethernet se redondea con un número de suma de comprobación de 4 bytes. Afortunadamente, esto lo genera el componente TCP/IP, por lo que normalmente no tendrá que ocuparse de ello.	4

7 Protocolo de resolución de direcciones

Con el componente TCP/IP podemos entrar en el modelo OSI en la capa Física utilizando una conexión en modo MAC. La creación de un socket MAC nos permite enviar y recibir tramas Ethernet. Sin embargo, el modo MAC o Ethernet es la envoltura externa en la que se envuelven todos los demás protocolos. Sólo se ocupa del envío físico de datos entre dos dispositivos MAC, no de lo que es el mensaje. Así que para demostrar este sistema de mensajería en acción necesitamos un mensaje que podamos introducir en la trama Ethernet y con el que podamos trabajar directamente a nivel Ethernet. Para ello enviaremos un mensaje de Protocolo de Resolución de Direcciones (ARP).

ARP es lo más cerca que podemos llegar a trabajar con la propia capa Física. ARP es un mensaje que cabe dentro de la sección de datos de la trama Ethernet y sobre el que se puede actuar directamente. A diferencia de los protocolos de nivel superior que pueden consistir en mensajes dentro de mensajes que requieren ser extraídos y procesados antes de actuar sobre ellos. Los niveles superiores, como IP y TCP, utilizan la capa física de Ethernet como la envoltura física externa en la que se envía su propio datagrama. Sin embargo, el componente TCP/IP maneja esta envoltura entre bastidores, por lo que nunca la vemos en acción cuando utilizamos esos modos.

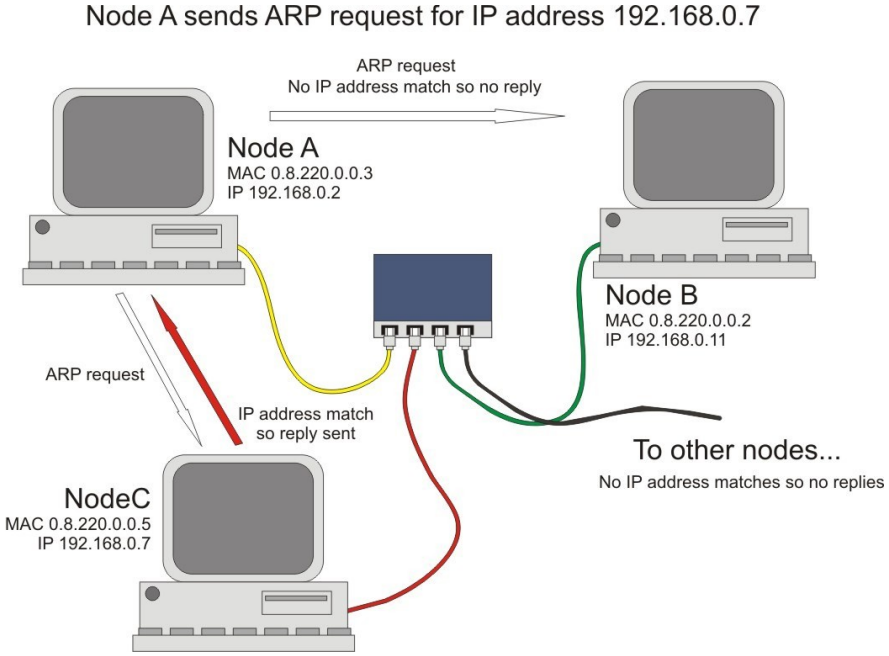
¿Para qué sirve?

- Un programa escáner ARP puede utilizarse como parte de un sistema TCP/IP más amplio para verificar las direcciones MAC antes de enviar más mensajes, es decir, como un servicio de comprobación de errores de punto de contacto inicial.

El escáner también se puede utilizar para identificar posibles conflictos de direcciones IP, ya que diferentes nodos con direcciones IP idénticas tendrán diferentes direcciones MAC.

ARP en acción: encontrar la dirección MAC de un mensaje

Cuando un protocolo envía un mensaje, necesita enviar el mensaje a una dirección IP concreta. Sin embargo, la capa física que realiza el envío se comunica entre dispositivos MAC. Entonces, ¿cómo consigue la dirección MAC que necesita? Aquí es donde entra en juego ARP. ARP puede difundirse a todos los nodos de una red. Si un nodo recibe un mensaje destinado a su dirección IP, puede responder. De lo contrario, puede simplemente ignorar el mensaje. Así que ARP se puede utilizar para pedir la dirección IP.



Por ejemplo:

1. El Nodo A necesita enviar un mensaje a la dirección IP 192.168.0.7 pero no conoce la dirección MAC a la que enviarlo.

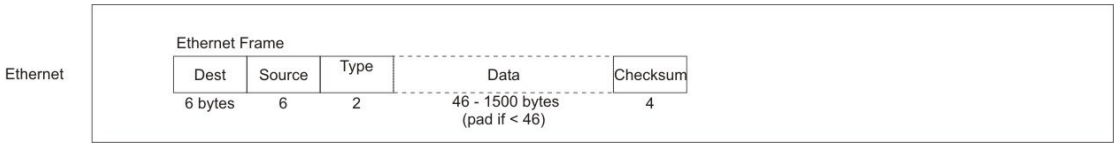
2. El nodo A envía una petición ARP para la dirección IP 192.168.0.7 a todos los nodos de la red.
3. El nodo B (dirección IP 192.168.0.11) recibe el mensaje pero lo ignora.
4. El Nodo C (dirección IP 192.168.0.7) recibe el mensaje y responde, enviando su MAC address con la respuesta.
5. El nodo D (dirección IP 192.168.0.34) recibe el mensaje pero lo ignora.
6. El nodo E (dirección IP 192.168.0.3) recibe el mensaje pero lo ignora.
7. Y así sucesivamente para todos los demás nodos de la red, que ignorarán el mensaje.

8. El Nodo A recibe la respuesta del Nodo C y extrae la dirección MAC de la dirección IP 192.168.0.7.
9. El Nodo A ya está listo para enviar el mensaje.

Además de poder enviar el mensaje, el nodo también puede guardar la combinación de dirección IP y dirección MAC para futuras consultas, de forma que no tenga que saturar la red con difusiones intentando encontrar la misma dirección IP más adelante. En una red de, por ejemplo, 200 nodos, esto puede suponer una reducción significativa del tráfico de red.

Trama Ethernet: La envoltura exterior

La solicitud ARP está contenida en la sección de datos del datagrama Ethernet.
Así que primero hay que crear la cabecera Ethernet, y luego añadir en el datagrama ARP
La cabecera Ethernet consta de la dirección MAC de destino, la dirección MAC de origen y los Datos - en este caso el datagrama ARP.



Se requieren los siguientes datos:

Cabecera Ethernet

	Explicación	Bytes	Notas
Destino	La dirección MAC de destino.	6	Utilizando la dirección MAC 255.255.255.255.255.255 pasará el mensaje a todos los dispositivos MAC de la red de área local.
Fuente	La dirección MAC del dispositivo de envío.	6	Disponible en TCP/IP página de propiedades del componente.
Tipo	El tipo de operación	2	

Datos

La sección de datos Ethernet debe tener 46 bytes o más. Si la longitud de los datos es menor, hay que rellenarlos. Afortunadamente, el componente TCP/IP se encargará de ello automáticamente.

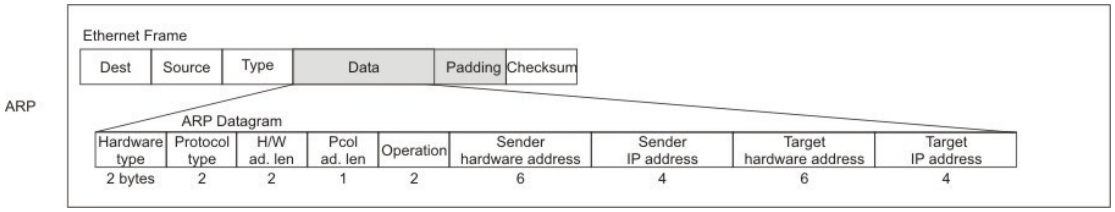
Suma de comprobación

Se necesita una suma de comprobación al final, pero esto también lo gestiona automáticamente el componente TCP/IP.

7 Protocolo de resolución de direcciones

El datagrama ARP

El paquete de datos ARP parece bastante complejo, pero se trata sobre todo de rellenar los espacios en blanco con información conocida. O utilizar códigos estándar para elementos como el tipo de operación y de protocolo.



Datagrama ARP

	Explicación	Bytes	Notas
Tipo de hardware:	El tipo de hardware utilizado. generalmente será una conexión Ethernet, valor 0x0001	2	
Tipo de protocolo	El tipo de protocolo utilizado. IP = 0x0008.	2	
Ajuste H/W longitud	La longitud del hardware es la longitud del identificador de hardware, en este caso 6 para la dirección MAC de 6 bytes.	1	
Longitud de ajuste del protocolo	La longitud del Protocolo es la longitud de el identificador del protocolo, en este caso 4 para la dirección IP de 4 bytes.	1	
Operación	Los valores de las operaciones que utilizaremos son 0,1 para una petición ARP, y 0,2 para una respuesta ARP.	2	
Hardware del emisor dirección	La dirección MAC del hard- emisor ware device.	6	Por defecto es 0.8.220.0.0.0
IP del remitente	La dirección IP del remitente. Disponible en las propiedades del componente página.	4	Por defecto es 192.168.0.2
Hardware de destino dirección	Debe rellenarse, pero es ignorado por el dispositivo receptor.	6	
Dirección IP de destino:	La dirección IP que deseamos comprobar para.	4	

Un punto importante que no debe pasarse por alto es que el datagrama ARP está envuelto en la sección de datos de la trama Ethernet: un mensaje dentro de otro mensaje. Cuando el dispositivo Ethernet en el otro extremo recibe el mensaje Ethernet, el datagrama ARP será dividido y procesado por el nodo. El trabajo de la trama Ethernet es simplemente asegurarse de que el mensaje llega.

8 Implementación del modo Ethernet en Flowcode

El uso del componente TCP/IP en modo Ethernet permite acceder a comunicaciones directas MAC a MAC a nivel físico. Esta sección discute los íconos, configuraciones y macros que se necesitarían para implementar un proyecto en modo Ethernet en Flowcode.

Nota: Esta sección cubre los fundamentos del envío y recepción de datos, y debe leerse antes de pasar a los otros protocolos.

Hay cinco elementos básicos para implantar el modo Ethernet:

- Inicialización de
- Creación de una toma MAC
- Envío de datos
- Recepción de datos

Coincidencia de datos

Inicialización de

Antes de poder hacer nada, el componente TCP/IP necesita inicializarse.
Esta inicialización es requerida por todos los programas que utilizan el programa TCP/IP.
Añada una macro '*Initialize*' al principio de su programa para inicializar el componente TCP/IP.

Creación de una toma MAC

Para enviar un datagrama Ethernet necesitas crear un socket MAC para enviarlo. Un socket es simplemente un término para una conexión a otro sistema con el que te puedes comunicar, igual que necesitas conectar un micrófono al socket para que el sonido llegue a la mesa de mezclas.
Añade una macro *CreateMACSocket* al programa.
CreateMACSocket toma los parámetros *promiscuous, broadcast* y *error,* que se detallan en el archivo de ayuda.

Como *CreateMACSocket* devuelve un valor distinto de cero si la conexión tiene éxito, puede utilizar este valor de retorno para la comprobación de errores para informar al usuario de que ha habido un problema y finalizar el programa.

Envío de datos

Las macros de transmisión toman un parámetro de canal para establecer en qué canal transmitir.
Los paquetes Ethernet deben enviarse por el canal 0, así que asegúrate de que se utiliza para todas estas macros. (Los canales son una característica del Hardware TCP/IP stack IC que se utiliza en la placa de Internet no del sistema TCP/IP en general).

Lo primero que hay que hacer es, por supuesto, calcular o cotejar los datos que hay que enviar.

Una vez que tenga los datos listos, el proceso de envío de un mensaje es el siguiente:

- *TxStart* (canal) Prepara el búfer para empezar a aceptar datos de transmisión.
- *TxSendByte*(channel, byte) Envía el byte de datos al búfer para el canal especificado. Necesita repetirse hasta que todos los datos hayan sido enviados al buffer. También están disponibles otras dos macros útiles que envían datos de propiedad:

✦ *TxSendMyMAC* Envía la dirección MAC de los componentes TCP/IP (6 bytes).

✦ *TxSendMyIP* Envía la dirección IP de los componentes TCP/IP (4 bytes).

- TxEnd (canal) Se utiliza para iniciar la transmisión de datos. A continuación se transmiten los datos enviados al búfer.

Implementación del modo Ethernet en Flowcode

Tenemos las siguientes secciones de datos brutos para enviar:

- Datos de cabecera

Ethernet Datos de
datagrama

Otros elementos necesarios, como el relleno de datos y la suma de comprobación, son gestionados automáticamente por el componente TCP/IP.

Lectura de datos

RxDataAvailable (canal) es la macro clave aquí. Devuelve un valor distinto de cero si hay datos disponibles.
A continuación, puedes revisar y leer los datos.
El componente TCP/IP dispone de varias macros para leer la información.

- *RxReadHeader (channel, idx) RxReadHeader* permite leer los bytes específicos de la sección de cabecera. El tamaño de la cabecera y los datos que contiene varían en función del proto- col que se utilice. Consulte la sección de macros anterior para obtener información detallada sobre los distintos bytes de la cabecera.
- *RxReadByte (channel)* Lee el siguiente byte del buffer. Esta es la macro básica de recuperación de datos que necesitas utilizar.

RxSkipBytes(canal,cuenta)

Una vez que haya recogido todos los datos que necesita del mensaje, tendrá que borrar el búfer de datos para permitir la entrada del siguiente mensaje.

RxFlushData(Channel) borrará el búfer de ese canal.
Si no desea borrar los datos, sino restablecerlos para leerlos de nuevo, por ejemplo, después de comprobar si hay errores, o en una aplicación de cortafuegos, en su lugar llame a la macro *RxDataAvailable,* ya que esto restablece el puntero del búfer de re- cepción al principio del paquete.

Coincidencia de datos

Al analizar los datos, a menudo habrá bits específicos de datos que desee comprobar, es decir, su dirección MAC, su dirección IP o un conjunto específico de bytes.
El componente TCP/IP tiene un lote de macros *RxMatchXXXX* para ayudar con esto.
Todas estas macros funcionan de la misma manera. Comprueban el número requerido de bytes en el búfer y devuelven un valor distinto de cero para una coincidencia, o 0 para ninguna coincidencia.

- *RxMatch_2_Bytes*, *RxMatch_4_Bytes* y *RxMatch_6_Bytes* comprueban los siguientes 2, 4 ó 6 bytes respectivamente para ver si hay una coincidencia.
- *RxMatchMyMAC* y *RxMatchMyIP* comprueban la dirección MAC (6 bytes) y la dirección IP (4 bytes) respectivamente.

Ejercicio 1: Escáner ARP

Instrucciones
Cree un escáner ARP básico que compruebe las direcciones MAC disponibles para el rango de direcciones IP 192.168.0.0 a 192.168.0.255.
Si se recibe una respuesta ARP - indicando una dirección IP válida, muestre la dirección IP y la dirección MAC asociada en la pantalla LCD.

Objetivo del ejercicio:

- Construir un programa que pueda escanear un rango de direcciones IP potenciales en la red Mostrar la dirección IP y la dirección MAC de cualquiera que se encuentre en una pantalla LCD.

Requisitos previos:

- Familiaridad con Flowcode.

Conocimiento del componente de pantalla LCD.

Resultados del aprendizaje:

- Estructura de direcciones MAC e IP
- La estructura del marco Ethernet, que sustenta todos los demás protocolos TCP/IP.
- El principio del datagrama, según el cual los datos dentro de un mensaje pueden ser en sí mismos un mensaje.
- Proceso de envío, recepción y correspondencia de bytes.

10 Ejemplo práctico: Implementación de ARP en Flowcode

Esta sección le llevará a través de un ejemplo trabajado para demostrar las macros básicas involucradas con un poco más de detalle. Utilizaremos el ejemplo de un programa escáner ARP básico como Ejercicio 1 para ilustrar los distintos pasos implicados.

Objetivo del ejercicio 1:
Construir un programa que pueda escanear un rango de direcciones IP potenciales en la red y mostrar la dirección IP y la dirección MAC correspondiente para cualquiera que se encuentre en una pantalla LCD.

10.1 Análisis del tráfico de red

Es muy recomendable utilizar un analizador de tráfico de red como Wireshark, ya que permite examinar los paquetes que se envían. Wireshark es especialmente útil para depurar errores en la formación de paquetes.

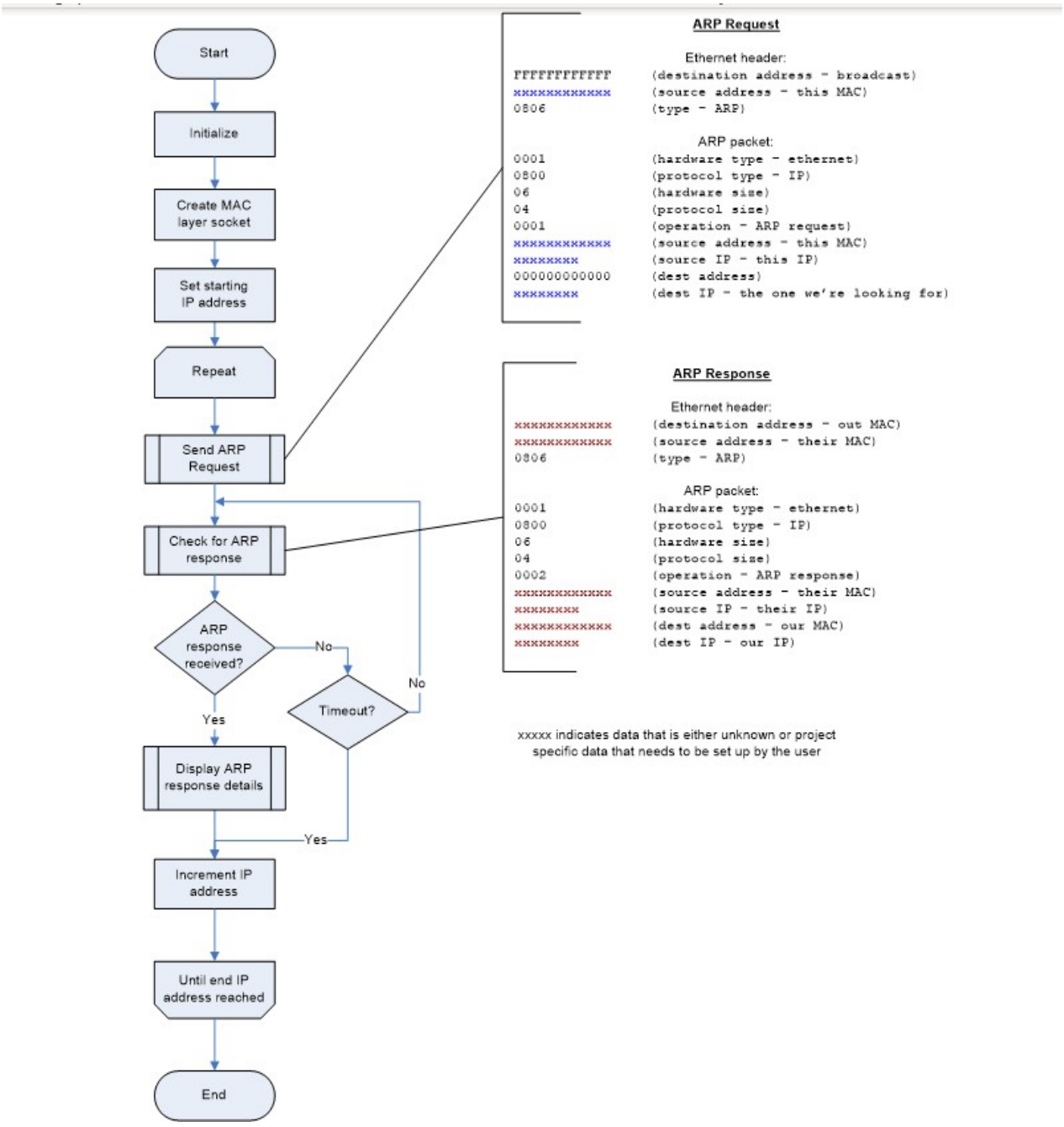
10.2 La estructura básica de

El programa ARP consta de cuatro elementos básicos:

1. Inicialice y configure el componente TCP/IP.
2. Recorrer en bucle el rango de direcciones IP que deseamos buscar y enviarles una consulta.
3. Comprueba si recibimos respuesta a nuestra consulta. Si es así recupera la dirección IP y la dirección MAC. Si no hay respuesta (comprobación de tiempo de espera) entonces pasar a la siguiente dirección IP en el bucle.
Envía los datos recuperados a la pantalla LCD.

Las partes de recepción y visualización del programa estarán contenidas en el bucle principal de envío de mensajes. Para simplificar las cosas buscaremos sólo un cierto rango de direcciones IP 192.168.0.0 a 192.168.0.255. Esto significa que sólo tendremos que preocuparnos del último byte de la dirección IP, lo que simplificará el programa. En realidad hay otro truco aquí también. Este rango de direcciones está tradicionalmente reservado
para sistemas en la misma red local o dominio, lo que nos facilita mucho la vida ya que es mucho más probable que obtengamos una respuesta. También son direcciones que no están en Internet, por lo que no tenemos que preocuparnos de intentar comunicarnos accidentalmente con alguna máquina lejana.

Podemos representar el diseño del programa con el siguiente diagrama de flujo:



10.3 Inicialización de

El primer paso es la etapa estándar de inicialización, en la que se configuran los componentes y las variables.

- Inicializar el componente TCP/IP
- Inicializar la pantalla LCD

Inicializar las variables - p.ej. dirección IP Mín, Máx y valores iniciales

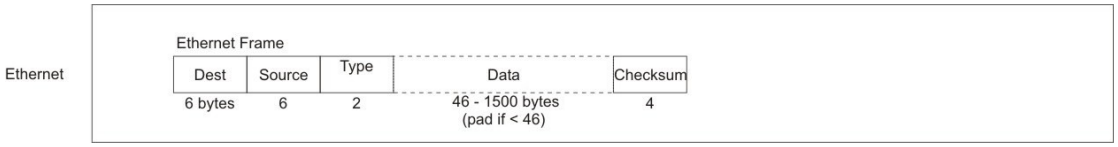
Añade una macro '*Initialize*' al principio de tu programa para inicializar el componente TCP/IP, y una macro Macro '*Start*' para el LCD también.

Para enviar un datagrama Ethernet necesitas crear un socket MAC para enviarlo. Añade una macro *CreateMACSocket* con los parámetros establecidos a *promiscuous = 0, broadcast =0, error = 0. (*No quieres escuchar ningún otro tráfico en la red, sólo mensajes enviados a este nodo).

Establece un valor de retorno como *RETVAL* para comprobar si la conexión ha tenido éxito. Si falla puede entonces atrapar el error e informar al usuario de que ha habido un problema y finalizar el programa.

Envío de la solicitud ARP

La solicitud ARP está contenida en la sección de datos del datagrama Ethernet.
Así que primero hay que crear la cabecera Ethernet, y luego añadir en el datagrama ARP
La cabecera Ethernet consta de la dirección MAC de destino, la dirección MAC de origen y los Datos - en este caso el datagrama ARP.



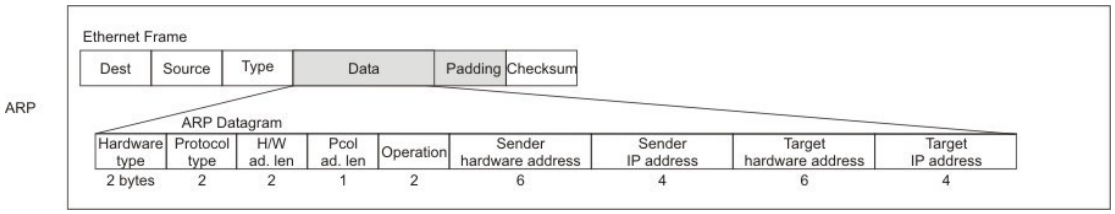
Se requieren los siguientes datos:

Cabecera Ethernet

	Explicación	Bytes	Datos para el ejercicio
Destino	La dirección MAC de destino. Uso de la dirección de difusión MAC 255.255.255.255.255 pasará el mensaje a todos los dispositivos MAC de la red de área local.	6	255.255.255.255.255.255
Fuente	La dirección MAC del dispositivo de envío. Disponible en la página propia del componente TCP/IP.	6	Utilizar TxSendMyMAC
Tipo	El tipo de operación	2	ARP = 8, 6

Datagrama ARP

La sección de datos de este ejemplo es bastante compleja, pero se trata sobre todo de rellenar los espacios en blanco con información conocida. O utilizar códigos estándar para elementos como el funcionamiento y el tipo de protocolo.



Datagrama ARP

	Explicación	Bytes	Datos para el ejercicio
Tipo de hardware:	El tipo de hardware utilizado. generalmente será una conexión Ethernet, valor 0x0001	2	0,1
Tipo de protocolo	El tipo de protocolo utilizado. IP = 0x0008.	2	0, 8
Ajuste H/W longitud	La longitud del hardware es la longitud del identificador de hardware, en este caso 6 para la dirección MAC de 6 bytes.	1	6
Longitud de ajuste del protocolo	La longitud del protocolo es la longitud del identificador del protocolo, en este caso 4 para el identificador de 4 bytes. Dirección IP.	1	4
Operación	Los valores de las operaciones que utilizaremos son 0,1 para una solicitud ARP y 0,2 para una solicitud Respuesta ARP.	2	0, 1 para solicitud

Remitente duro. dirección	La dirección MAC del hardware emisor dispositivo.	6	Utilizar TxSendMyMAC
IP del remitente	La dirección IP del remitente. Disponible en la página de propiedades del componente.	4	Utilizar TxSendMyIP
Hardware de destino dirección	Debe rellenarse, pero es ignorado por el re-dispositivo receptor.	6	0.0.0.0.0.0
Dirección IP de destino:	La dirección IP que deseamos comprobar.	4	Se introducen manualmente o se definen como variables, por ejemplo, una matriz de 4 bytes si la opción pueden cambiar.

10.4.2 Envío del mensaje

Las macros de transmisión toman un parámetro de canal para establecer en qué canal transmitir. Los paquetes Ethernet deben enviarse por el canal 0, así que asegúrate de que se utiliza para todas estas macros.

Una vez que tenemos los datos calculados podemos enviar el mensaje. El proceso de envío de un mensaje es el siguiente:

- *TxStart* (canal)
- TxSendByte(channel, byte) Necesita repetirse hasta que se envíen todos los datos. Tenga en cuenta que TxSendMyMAC y TxSendMyIP se pueden utilizar para enviar la dirección MAC y la dirección IP.

TxEnd (canal) Se utiliza para finalizar la transmisión de datos.

Es necesario enviar las siguientes secciones:

- Datos de cabecera Ethernet

Datos del datagrama ARP

El relleno de datos y la suma de comprobación son gestionados automáticamente por el componente TCP/IP, por lo que pueden ignorarse.

10.5 Obtener una respuesta

Cuando un nodo recibe la solicitud ARP que le corresponde (coincide con la dirección IP), envía una respuesta al remitente utilizando la dirección MAC y la dirección IP obtenidas de los datos del mensaje. En una única conexión, como la utilizada con la solución de formación por Internet, sólo deberíamos recibir una única respuesta del PC, ya que sólo hay una dirección IP a la que enviar. Sin embargo, en una red completa podemos recibir múltiples respuestas, una de cada dirección IP presente en la red a la que se haya enviado una solicitud.

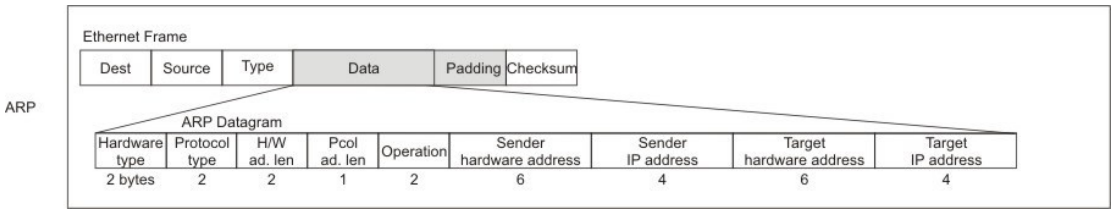
Dependiendo de cuestiones como el retardo de la red o la velocidad a la que un nodo solicite la respons-es posible que no lleguen en el orden en que se enviaron las solicitudes.

Si el nodo está intentando enviar un mensaje propio, también puede contactar con nuestro nodo con una petición ARP propia. Algo que tendremos que comprobar.

El primer paso es comprobar si hay datos entrantes utilizando la macro *RxDataAvailable*. Recuerde que todas las comunicaciones Ethernet son en el canal 0.

El siguiente paso es determinar si se trata de un mensaje ARP. Podemos hacerlo comprobando la cabecera Ethernet para ver si el tipo coincide con un tipo de mensaje ARP. Podemos saltarnos cualquier dato precedente hasta llegar a los bytes que queremos comparar. Las macros *RxSkipBytes* y *RxMatch_2_Bytes*, *RxMatch_4_Bytes* y *RxMatch_6_Bytes* pueden ser usadas para moverse y coincidir con los datos requeridos.

Ejemplo práctico: Implementación de ARP en Flowcode



Cabecera Ethernet

	Bytes	Omitir lectura o coincidencia	Notas
Destino	6	omitir	
Fuente	6	omitir	
Tipo	2	Partido por ARP - 8, 6	

Una vez que hemos determinado que se trata de un mensaje ARP podemos recorrerlo hasta llegar a los bytes de operación que podemos probar para ver si el mensaje es una respuesta ARP. Si es así, podemos recorrer el resto del mensaje y extraer la dirección MAC y la dirección IP. Una vez que tenemos los datos que necesitamos podemos parar en ese punto. No necesitamos leer todo el mensaje.

Datagrama ARP

	Bytes	Omitir lectura o coincidencia	Notas
Tipo de hardware	2	Saltar	
Tipo de protocolo	2	Saltar	
Ajuste H/W longitud	1	Saltar	
Ajuste del protocolo. longitud del ment	1	Saltar	
Operación	2	Coincidencia de respuesta ARP - 0, 2	
Hardware del emisor dirección	6	Lectura y visualización	La dirección MAC que están buscando
Dirección IP del remitente	4	Lectura y visualización	La dirección IP que envió la respuesta
Hardware de destino dirección	6	Skip - En realidad podemos parar aquí ya que no necesitan más datos.	
Dirección IP de destino	4	Saltar	

Observa que obtenemos tanto la dirección MAC como la dirección IP. Esto se debe a que no podemos asumir que la respuesta proviene de una dirección IP específica, como la última a la que enviamos un mensaje. Debido al tráfico de red y a las diferentes velocidades de respuesta de los dispositivos, podría ser cualquiera de las direcciones IP a las que enviamos mensajes anteriormente. O podría ser una búsqueda ARP de otro nodo de la red intentando encontrar nuestra dirección MAC. También si dos máquinas tienen la misma dirección IP podríamos recibir dos respuestas de la dirección IP, pero con diferentes direcciones MAC para los diferentes nodos.
Una vez que hayamos recogido todos los datos que necesitamos del mensaje, tendremos que vaciar el búfer de datos para permitir la entrada del siguiente mensaje utilizando *RxFlushData*.

Tiempo de espera

En una red lenta o con mucho tráfico, el programa puede responder mucho más rápido que la red. Así que tenemos que ser capaces de esperar una respuesta.
Además, nuestra solicitud puede ser ignorada o perderse, sobre todo si la dirección a la que la enviamos no existe. Así que tenemos que decidir un plazo de tiempo razonable para esperar antes de decidir que seguir adelante y tratar la siguiente dirección.

Visualización de la respuesta

Una vez recibida la respuesta, tendremos que mostrar la dirección IP.
Se trata de una tarea relativamente sencilla, que le dejamos a usted para que la lleve a cabo.
Ahora que hemos resuelto el envío de la petición ARP y la lectura de la respuesta, sólo nos queda dar los últimos toques al programa. Por ejemplo, el bucle principal necesita ser configurado. Podemos controlar el

estableciendo una dirección inicial y comprobando cuando ha alcanzado una dirección final. Un rango útil para comprobar es de 198.162.0.0 a 198.162.0.255 ya que este rango de IP en particular está reservado para nodos en la misma red, que es lo que principalmente queremos comprobar.

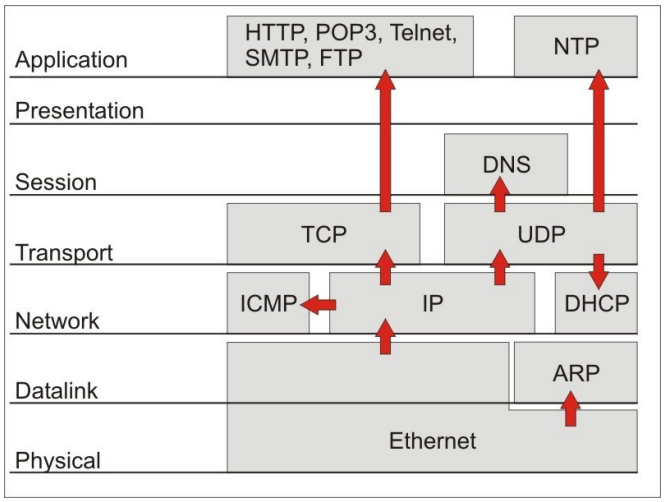
Ejemplo de programa

Puede encontrar una solución de ejemplo en la carpeta "TCP_IP\Examples" de Flowcode, o en la sección Ejemplos del CD, para que pueda consultarla, utilizarla para el código y utilizarla con fines de demostración.

Tenga en cuenta que las propiedades, los parámetros de las macros y los valores de las variables, etc., se configuraron para su uso con nuestra red y puede ser necesario cambiarlos para adaptarlos a su propia red o configuración del sistema.

Capa IP

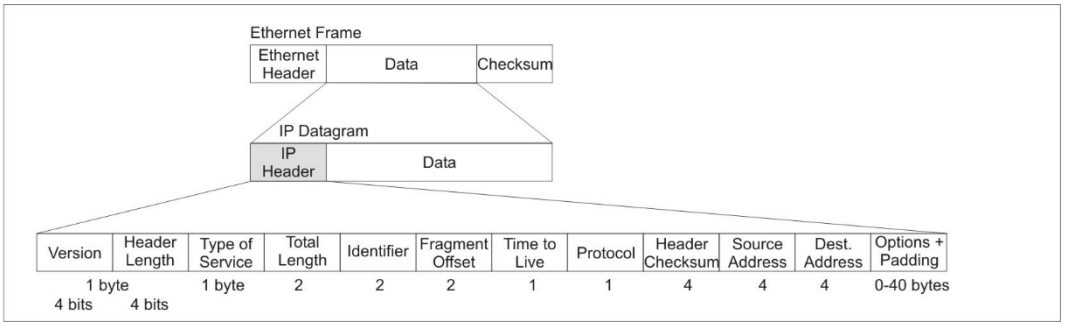
La capa IP se encuentra en el nivel de Red del modelo OSI, donde la comunicación entre nodos es la clave, no la comunicación entre dispositivos físicos. Un mensaje puede tener que enviarse a través de varios dispositivos MAC para llegar a su destino final. La capa IP proporciona el sobre con la dirección estampada para el transporte en red en forma de dirección IP. La dirección IP es normalmente un código de cuatro bytes como 192.168.0.2. La dirección IP proporciona al software y hardware de la capa de red, como los routers, la información que necesitan para enviar el mensaje al nodo destinatario correcto. También se transmite otra información que el software receptor puede utilizar para determinar de dónde procede el mensaje y qué tipo de mensaje es.



Las direcciones de sitios web que utilizamos habitualmente, como www.matrixltd.com, son en realidad direcciones IP disfrazadas. Los protocolos TCP/IP, como el DNS, se utilizan para recuperar la dirección IP numérica a partir de los valores no numéricos que les damos. Esto se debe a que la mente humana es más capaz de recordar palabras que números.

El datagrama IP (derivado sin duda de Telegram, lo que aumenta las metáforas del sistema de correo) se encaja dentro de una trama Ethernet para permitir el envío físico del mensaje a la red. El datagrama IP contiene un archivo de cabecera y un paquete de datos. Al igual que el datagrama IP (el sobre) está metido dentro de una trama Ethernet que lo envuelve, la sección de datos es en sí misma un mensaje (la carta propiamente dicha). Este mensaje puede tener distintos formatos, como TCP, UDP, ICMP, etc.

IP



Cabecera IP

	Descripción	bytes
Versión y longitud de la cabecera	Dos valores de 4 bits combinados en un byte Para este curso utilizaremos la versión 4 de IP. La longitud de la cabecera se mide en palabras de 32 bits. Normalmente son 5 palabras. Por tanto, el valor normal sería 0x45.	1
Tipo de servicio	Establece la prioridad del datagrama. Normalmente puede ignorarse y establecerse en 0.	1

Longitud total	**Longitud total del datagrama, incluida la cabecera y los datos.**	**2**
Identificador de fragmentación	Se utiliza para ayudar a la fragmentación (es decir, dividir los mensajes demasiado grandes para un solo dato). gramo). No vamos a cubrir la fragmentación aquí, por lo que se puede establecer en 0.	2
Desplazamiento de fragmentación	Se utiliza para ayudar a la fragmentación (es decir, dividir los mensajes demasiado grandes para un solo gramo de datos). No vamos a cubrir la fragmentación aquí, por lo que se puede establecer en 0.	2
Tiempo de vida (segundos)	Los mensajes sólo pueden sobrevivir un cierto tiempo en la red. Este valor disminuye durante el tránsito por los encaminadores hasta que el mensaje es finalmente descartado. Un buen valor por defecto es 100 segundos.	1
Protocolo	Identifica el protocolo del mensaje contenido en la sección de datos. Los posibles valores utilizados para indicar qué protocolo se está utilizando incluyen: • ICMP = 1 • TCP = 6 • UDP = 17	1
Suma de comprobación	Un valor de suma de comprobación para la cabecera IP. Normalmente necesitará calcular esto, ya que todo es manejado por el componente TCP/IP, sin embargo si necesita generar la suma de comprobación manualmente puede usar el siguiente proceso. Intercambio de bytes en la cabecera Rellenar con 0 si la longitud es impar Borrar el valor de la suma de comprobación Suma de las palabras de cabecera de 16 bits Poner el complemento de uno en la suma de comprobación	4
Dirección IP de origen	La dirección IP del remitente.	4
Dirección de destino	La dirección IP del nodo de destino.	4
Opciones	Existen varias opciones para la cabecera IP. No utilizaremos ninguna op- de este curso, por lo que esta sección puede ignorarse por ahora.	0-40

Utilización de la capa IP: ICMP y Ping

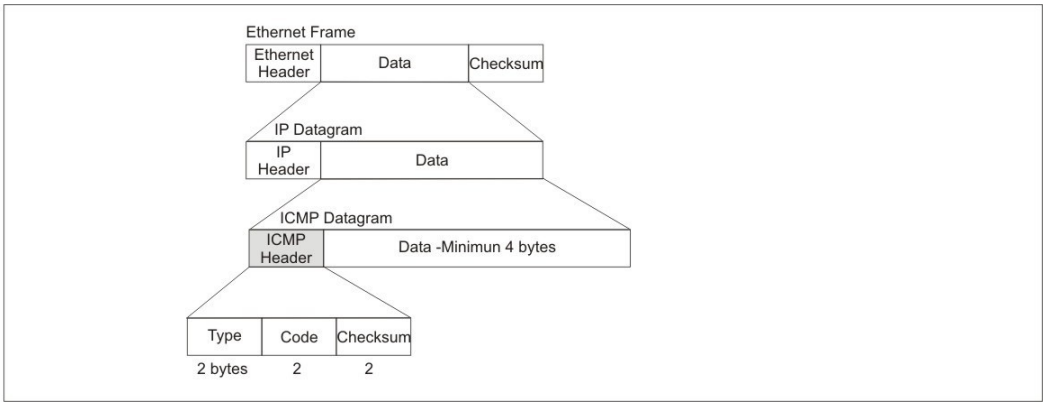
No se puede tener un ejemplo de IP independiente, como tampoco se puede tener un ejemplo de Ethernet independiente. Son portadores de carga, no los datos finales. Normalmente, IP se utiliza para transmitir mensajes de protocolos de capa superior, como TCP o UDP. Lo que podemos hacer es poner un protocolo de mensaje simple en un datagrama IP que podamos usar directamente sin tener que añadir más capas de protocolo encima. Uno bueno y conveniente es ICMP - Internet Control Message Protocol.

ICMP se utiliza habitualmente para comprobar conexiones con otros ordenadores. Esto se conoce como "ping", ya que funciona del mismo modo que el "ping" de sonar que envían los submarinos. Si alcanza un objetivo, se produce un eco que puede ser captado y examinado.

Capa IP

El datagrama ICMP

ICMP



Lo básico es que un mensaje ICMP recibirá una respuesta ICMP que puede ser examinada para ver si hubo algún error, y si es así, cuáles.
Por ejemplo, enviar un mensaje ICMP a un nodo con un puerto inalcanzable generaría una respuesta con Tipo = 3 (Destino inalcanzable) y Código = 4 (Puerto inalcanzable). Como puedes adivinar, obtener este tipo de información es inestimable para la depuración de la red.

	Descripción	**bytes**
Tipo	Detalla el tipo de respuesta o solicitud que se está realizando. Utiliza los siguientes tipos: • 0 Eco respuesta • 3 Destino inalcanzable (código de error) 8 Petición de eco	1
Código	Da detalles sobre cualquier error encontrado. Utiliza los siguientes códigos de error: • 0 Red inalcanzable • 1 Host inalcanzable • 2 Protocolo inalcanzable • 4 Puerto inalcanzable • 5 Fragmentación necesaria pero no permitida • 6 Red de destino desconocida • 7 Host de destino desconocido	1
Suma de comprobación	La suma de comprobación utilizada es la misma que para la cabecera IP • Intercambio de bytes en la cabecera • Rellenar con 0 si la longitud es impar • Borrar el valor de la suma de comprobación • Suma de las palabras de cabecera de 16 bits • Poner el complemento de uno en la suma de comprobación	2
Datos	La longitud del mensaje debe ser de 8 bytes como mínimo (incluida la cabecera), aunque los 4 bytes finales no se utilicen. Si el mensaje tiene menos de 8 bytes, rellene los datos hasta que alcancen los 8 bytes de longitud. Los 4 bits de datos son un número identificador de 2 bytes y un número de secuencia de 2 bytes.	4+

Datos de ping ICMP

Las solicitudes Ping ICMP estándar envían el carácter ASCII a-z y luego a-j como datos. Se sugiere que haga lo mismo ya que las aplicaciones receptoras pueden estar esperando esos datos.
Para facilitar la implementación, podemos recorrer los valores ASCII: del 97 al 120 para a-z y del 97 al 106 para la segunda sección a-j.

Implementación del modo IP en Flowcode

El modo IP le permite entrar en el modelo OSI a nivel de red. En este nivel, el componente TCP/IP manejará las tramas Ethernet de nivel inferior por usted.

¿Dónde están las cabeceras Ethernet e IP?

El cambio más importante entre el modo Ethernet y el modo IP es que las cabeceras son gestionadas por el componente TCP/IP. En el modo Ethernet era necesario añadir toda la información de cabecera al mensaje, así como los datos. Pero con IP no es necesario. Crear un socket IP se encarga de todas las cabeceras por ti, todo lo que necesitas proporcionar son los datos para el datagrama IP.

Tenga en cuenta que si el datagrama que se envía tiene su propia cabecera, como con ICMP, tendrá que proporcionar la información de la cabecera usted mismo, ya que forma parte de los datos que se envían a través del modo IP.

Configuración de la conexión IP

La primera parte que tenemos que ver es la configuración de la conexión IP.

- Inicializar el componente
- Crear una toma IP

Establezca la dirección y el puerto de destino

Primero necesitamos inicializar el componente TCP/IP con una macro

Initialize. A continuación se añade una macro *CreateIPSocket(Channel,*

Protocol, Broadcast).
Para este ejemplo enviaremos un datagrama ICMP (valor de *protocolo* 1) por *el canal* 0, y deseamos enviar y recibir, por lo que activaremos la *difusión* (valor 1).

Añade una macro *SetDestination(Channel, Dst_IP0, Dst_IP1, Dst_IP2, Dst_IP3, Dst_Port_Hi, Dst_Port_Lo)*.

- El canal puede ajustarse de 0 a 3.
- dst_ip0 a dst_ip3 son los cuatro bytes de dirección IP.

Los valores deben ser dados para los números de Puerto aunque el socket IP no los use ya que todos los parámetros son requeridos para ser llenados por Flowcode. Son necesarios para otro modo que comparte la misma macro (modo UDP que veremos más adelante). Cero o un valor aleatorio será suficiente.

Envío del datagrama

Aparte de los datos enviados, el procedimiento es el mismo que el descrito anteriormente para los mensajes Ethernet. Necesitaremos una macro *TxStart(Channel)*, un lote de macros *TxSendByte(Channel, Data)* para añadir los datos del mensaje, y una macro *TxEnd(Channel)* para finalizarlo.

Recibir una respuesta

Una vez más, esto sigue el patrón familiar que utilizamos con los paquetes Ethernet.

- *RxDataAvailable(Channel)* nos permite comprobar si ha llegado algún dato.
- Las macros *RxReadHeader*, *RxReadByte*, *RxSkipBytes* y *RxMatchXXXX* se utilizan para leer y comprobar los datos.

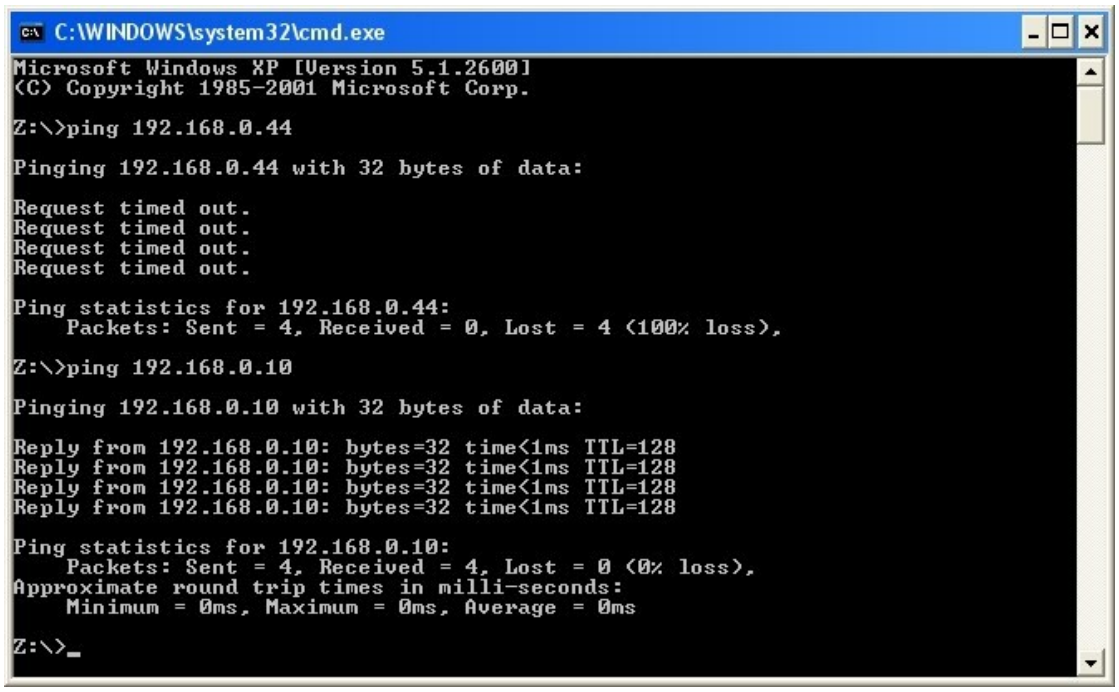
RxFlushData(Channel) se utiliza para borrar el búfer una vez que hemos terminado de leer.

El programa Ping de Windows

Windows viene con su propio programa Ping que podemos usar para probar que nuestro programa Ping puede enviar re sponses. Es posible que ya se haya encontrado con él y lo haya utilizado para comprobar conexiones y velocidades de conexión.

Podemos hacer ping al tablón de internet desde el PC para ver si nos responde.
Ping es fácil de usar. Basta con abrir un símbolo del sistema (Ejecutar CMD) y escriba Ping <dirección IP a ping>.
p.ej. Ping 192.168.0.2

Se enviará un ping a la dirección y se mostrarán los datos de temporización resultantes.



Podemos utilizar Ping para probar nuestro programa. Simplemente haga ping a la dirección IP que configuró para el componente TCP/IP
(192.168.0.2 por defecto). Si todo va bien debería obtener un tiempo de respuesta muy rápido.

Ejercicio 2: Programa Ping

Instrucciones
Cree un programa Ping que envíe un ping - una petición ICMP, a una dirección IP.
Si se recibe una respuesta - indicando una dirección IP válida, muestre la dirección IP y la dirección MAC asociada en la pantalla LCD.

Objetivo del ejercicio:
Para hacer ping a otro nodo y ver si obtenemos respuesta, de ahí que podamos encontrarlo en la red.

Requisitos previos:
Conocimiento de la capa MAC/Ethernet

Resultados del aprendizaje:
- Estructura de un datagrama IP.
- Envío de mensajes IP.
- Recepción de respuestas IP.
- Lectura de datos del mensaje IP.

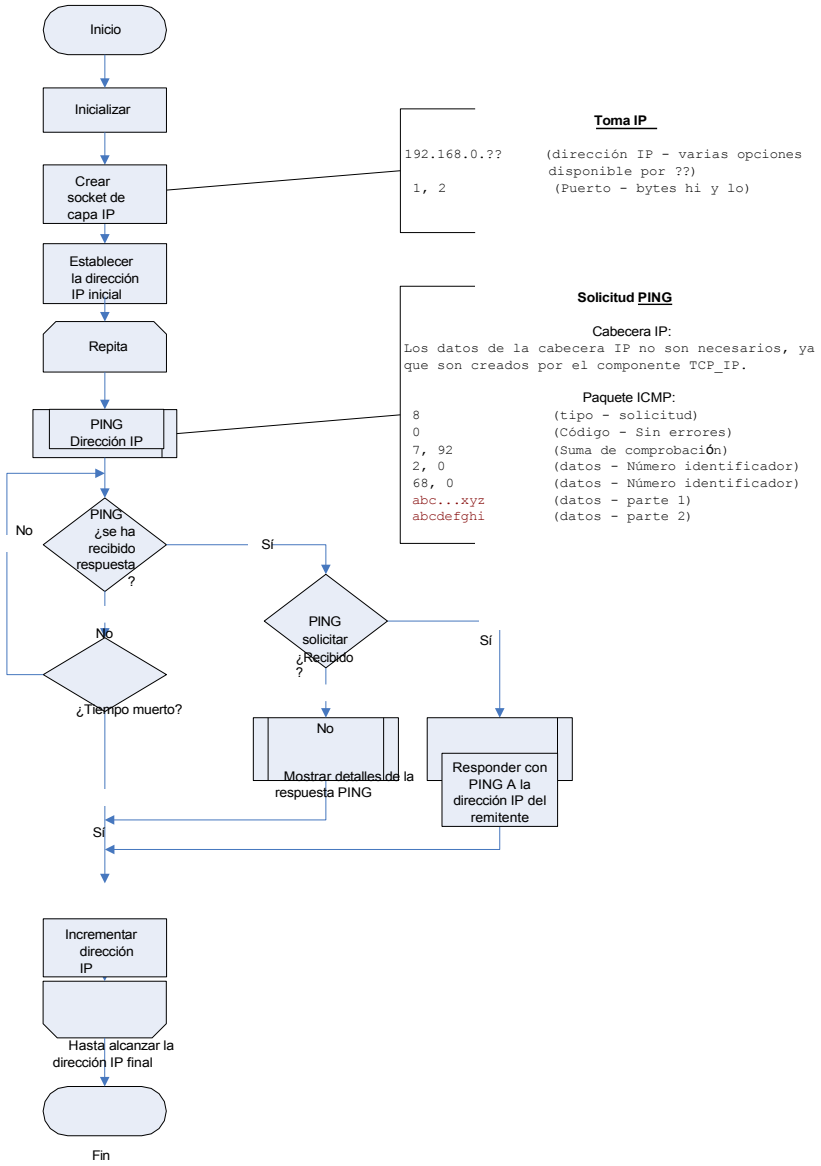
Ejemplo práctico: Ping

Este programa Ping básico es el establecido como Ejercicio 2, por lo que esta sección puede ser usada como base para un conjunto de notas para usar ese ejercicio.

Objetivo del ejercicio 2:
Para hacer ping a otro nodo y ver si obtenemos respuesta, de ahí que podamos encontrarlo en la red.

Ping

Ping envía una señal a una dirección IP determinada, es decir, a un posible nodo de la red.
Si el nodo existe y funciona correctamente, responderá con un eco de los datos enviados.
La dirección IP y la dirección MAC pueden ser eliminadas de la respuesta y mostradas. Otros nodos también pueden hacer Ping
por lo que tendremos que responder a una solicitud Ping con una respuesta propia.



Configuración de la conexión IP

Primero necesitamos inicializar el componente TCP/IP, por lo que se añade una macro *Initialize* al principio del programa.

A continuación se añade una macro *CreateIPSocket(Channel, Protocol, Broadcast)*.
Para este ejemplo enviaremos un datagrama ICMP (valor de protocolo 1) por el canal 0, y deseamos enviar y recibir, por lo que activaremos la difusión (valor 1).

Añade una macro *SetDestination(Channel, Dst_IP0, Dst_IP1, Dst_IP2, Dst_IP3, Dst_Port_Hi, Dst_Port_Lo)* y configura los parámetros para que sean 0, 192, 168, 0, IP, 1, 2
.

Los valores de la cabecera ICMP

Queremos enviar un Ping por lo que Type = 8, Echo Request. No tenemos ningún error que reportar, así que código = 0. Los 4 bits de datos son un número de Identificador de 2 bytes y un número de Secuencia de 2 bytes.
Cuando responda, tendrá que devolverlos junto con los datos enviados, por lo que son importantes para comprobar y responder. En nuestro ejemplo los pondremos a 2, 0 y 68, 0 respectivamente. Todo lo que queda para la cabecera es rellenar la suma de comprobación.

Datagrama ICMP

	Descripción	**bytes**	**Valor a utilizar en ejercicio**
Tipo	Detalla el tipo de respuesta o solicitud que se está realizando. Utiliza los siguientes tipos: • 0 Eco respuesta • 3 Destino inalcanzable (código de error) 8 Petición de eco	1	8
Código	Da detalles sobre cualquier error encontrado. Utiliza los siguientes códigos de error: • 0 Red inalcanzable • 1 Host inalcanzable • 2 Protocolo inalcanzable • 4 Puerto inalcanzable • 5 Fragmentación necesaria pero no permitida • 6 Red de destino desconocida • 7 Host de destino desconocido	1	0
Suma de comprobación	La suma de comprobación utilizada es la misma que para la cabecera IP • Intercambio de bytes en la cabecera • Rellenar con 0 si la longitud es impar • Borrar el valor de la suma de comprobación • Suma de las palabras de cabecera de 16 bits • Poner el complemento de uno en la suma de comprobación	2	7, 92
Datos	La longitud del mensaje debe ser de 8 bytes como mínimo (incluida la cabecera), aunque los 4 bytes finales no se utilicen. Si el mensaje tiene menos de 8 bytes, rellene los datos hasta que alcancen los 8 bytes de longitud. Los 4 bits de datos son un número identificador de 2 bytes y un número de secuencia de 2 bytes.	4+	2 ,0, 68, 0 (Identificador = 2, 0 Secuencia = 68, 0)

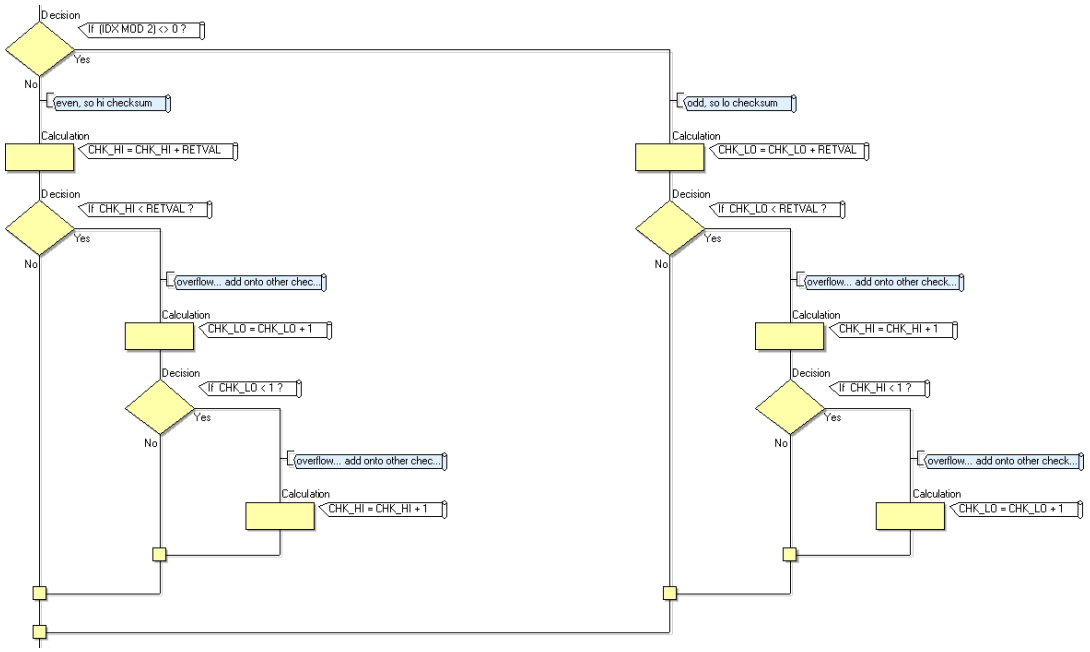
La suma de comprobación

Podemos calcular los valores de la suma de comprobación sobre la marcha o, para los mensajes establecidos, calcularlos de antemano y utilizar esos valores.
En este caso hemos precalculado la suma de comprobación para que sea 7, 92, así que podemos añadir esos dos bytes a continuación.

Sin embargo, cuando responda a una solicitud Ping no tendrá tanta suerte y necesitará crear sumas de comprobación para sus respuestas. El código de la suma de comprobación se ha puesto en una macro conveniente para usted, que se puede encontrar en la carpeta "Flowcode\TCP_IP\Examples". De lo contrario, tendrá que crear su propia implementación de la generación de la suma de comprobación.

Intercambio de bytes en la cabecera
Rellenar con 0 si la longitud es impar

Borrar el valor de la suma de comprobación
Suma de las palabras de cabecera de 16 bits
Poner el complemento de uno en la suma de comprobación



Parte de una rutina de suma de comprobación.

Los datos

Una vez configurados los datos de cabecera, podemos añadir los datos.
Lo hacemos en dos bucles, uno desde 97-120 para añadir ASCII a-z y el segundo desde 97-106 para añadir ASCII a-j.

Envío del datagrama ICMP

Aparte de los datos enviados, el procedimiento es el mismo que el descrito anteriormente para los mensajes Ethernet. Necesitaremos una macro *TxStart(Channel)*, un lote de macros *TxSendByte(Channel, Data)* para añadir los datos del mensaje, y una macro *TxEnd (Channel)* para finalizarlo.

Recibir una respuesta

Estamos enviando una petición de ping, es decir, un mensaje para que el otro nodo nos responda y así poder cronometrar el tiempo de respuesta. Así que tenemos que escuchar la respuesta y reaccionar en consecuencia. Para ello tenemos que hacer una de varias cosas.

Esperar un tiempo determinado y abandonar (timeout) si no hemos recibido respuesta. Procesar y mostrar un resultado si obtenemos una respuesta.
Si recibimos una petición ping necesitamos recuperar los datos de la IP y responder a la petición.

Una vez más, esto sigue el patrón familiar que utilizamos con los paquetes Ethernet.

- *RxDataAvailable(Channel)* nos permite comprobar si ha llegado algún dato.
- Las macros *RxReadHeader*, *RxReadByte*, *RxSkipBytes* y *RxMatchXXXX* se utilizan para leer y comprobar los datos.

RxFlushData(Channel) se utiliza para borrar el búfer una vez que hemos terminado de leer.

Los tipos de respuesta que estamos buscando son una respuesta Ping (tipo = 0), o una solicitud Ping (tipo = 8). La tercera opción es Red inalcanzable (tipo = 3), en caso de que podamos mirar el valor del código de error para ver qué tipo de error se generó. Afortunadamente este es el primer byte en el buffer, así que podemos revisarlo inmediatamente.
Utilice RxReadByte para obtener el valor del tipo. También puede utilizar un segundo RxReadByte para obtener el código de error si es necesario.

Ping respuesta

Si es una respuesta Ping podemos mostrar los detalles del ping en la pantalla LCD.
Los primeros 8 bytes de datos son información de cabecera. Después empiezan los datos. Podemos saltarnos los 7 bytes siguientes (ya hemos leído el byte de tipo 1st) para llegar a la sección de datos. Entonces podemos leer los datos e imprimirlos en la pantalla LCD. Como puede haber una gran cantidad de datos es posible que tenga que elegir algunos bits para imprimir, en lugar de tratar de imprimir todo.

Solicitud de ping

Si es una solicitud ping, necesitamos responder. Es habitual que los programas de línea de comandos como Ping envíen hasta cuatro solicitudes de ping para que pueda obtener datos de sincronización. Así que puede recibir más de una solicitud de ping.

Enviaremos un Ping echo sin código de error, los dos primeros bits serán 0, 0.
Los dos bytes siguientes serán los dos bytes de suma de comprobación, que tendremos que calcular.

El resto del mensaje es un eco del mensaje ping que hemos recibido, incluyendo los números de identificador y secuencia y los datos. Podemos reiniciar el búfer, saltarnos los cuatro primeros bytes (tipo, código y los dos bytes de suma de comprobación) y seguir leyendo bytes y enviándolos de vuelta. Para poder hacer esto, necesitamos saber la longitud de los datos. Afortunadamente la cabecera IP almacena la longitud de los datos por nosotros. Podemos usar *RxRead- Header(0,1)* (canal 0, índice 1) para recuperar la longitud de los datos. No olvides que ya tenemos los primeros cuatro bytes, así que la cifra final de comprobación debe ser 4 menos. (Hay un segundo byte hi para la cantidad de datos, pero los datos Ping son generalmente bastante pequeños así que no deberíamos preocuparnos por ello).

Ampliación del programa Ping

Incluso el programa básico constituye una útil herramienta de diagnóstico para redes, y puede ampliarse fácilmente para proporcionar funciones adicionales de notificación de errores.

Con Ping puedes verificar direcciones IP, útil para multitud de tareas.
Con un poco de código de temporización puedes cronometrar o controlar las conexiones, ya que la latencia es un gran problema en el trabajo en red por Internet.
Hay una serie de códigos de error que se pueden devolver con ICMP. Éstos podrían constituir la base de un proyecto ampliado de comprobación de errores.

Ejemplo de programa

Encontrará una solución de ejemplo en la carpeta "Flowcode\TCP_IP\Examples" de Flowcode, o en la sección Exam- ples del CD, para que pueda consultarla, utilizarla para el código y utilizarla con fines de demostración.

Tenga en cuenta que las propiedades, los parámetros de las macros y los valores de las variables, etc., se configuraron para su uso con nuestra red y puede ser necesario cambiarlos para adaptarlos a su propia red o configuración del sistema.

UDP

UDP (User Datagram Protocol) es un método para enviar datos directamente a un socket específico (ver más abajo). UDP puede utilizarse como protocolo de comunicación directa, para enviar mensajes directamente a una aplicación concreta de un sistema determinado.

Tomas de corriente

Un socket es una combinación de una dirección IP y un puerto. Un puerto es un casillero electrónico que las distintas aplicaciones del sistema pueden explorar en busca de mensajes. Cuando un mensaje llega a un puerto, es responsabilidad de las aplicaciones darse cuenta de su llegada y responder al mensaje. Si ninguna aplicación está monitorizando el puerto cuando llega un mensaje, éste será ignorado.

Las aplicaciones también pueden monitorizar puertos permitiéndote comunicarte con ellas directamente. Dado un número de puerto específico y una dirección IP específica puedes comunicarte con una aplicación específica en un nodo específico. Esto es útil para aplicaciones personalizadas que necesitan comunicarse a través de Internet, como juegos online, o programas de control y monitorización externos.

En UDP enviamos una señal a un puerto específico y nos sentamos a esperar. Si todo ha ido bien y la aplicación que monitoriza el puerto está configurada para responder a los mensajes entrantes entonces el puerto generará una respuesta que recibiremos, si no entonces no recibiremos nada - ningún mensaje de error ni nada. A diferencia de ARP e ICMP que generan respuestas UDP no lo hace - a menos que escribas una aplicación que envíe una. Esto ha llevado a que UDP sea conocido como comunicaciones de "enviar y rezar".

Una cosa para la que UDP es útil es para enviar mensajes de datos personalizados directamente de un sistema a otro. Los datos pasados serán datos sin procesar, por lo que tendrás que implementar tu propio sistema para tratar los datos.

Respuestas predefinidas y puertos reservados

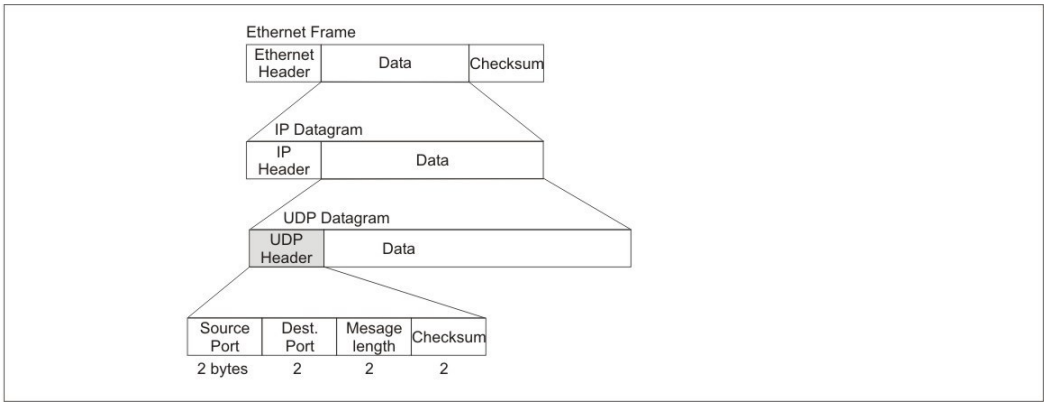
Algunos puertos tienen respuestas predefinidas. Por ejemplo, el puerto 7 se hace eco de los datos que se le envían. El puerto 13 devuelve una cadena de fecha/hora, etc. UDP puede utilizarse para activar sockets que den una respuesta conocida, como devolver una cadena de tiempo. Si necesitas comprobar la hora en otro PC con fines de sincronización, puedes utilizar UDP para enviar un mensaje al puerto 13, que responderá enviando una cadena de fecha/hora que podrás procesar.

Algunos puertos suelen estar reservados para protocolos específicos, como SMTP (puerto 25) o HTTP (puerto80).
Cuando se trata de protocolos conocidos, como SMTP, esto simplifica las cosas, ya que podemos conectarnos a un puerto reservado conocido en lugar de tener que averiguar en el sistema a qué puerto tenemos que conectarnos.
Los puertos que se utilizan pueden cambiarse, pero esto sería inusual y se haría por razones específicas, como cuestiones de seguridad. En tales casos, es posible que tenga que hablar con el administrador de la red para averiguar qué puertos se están utilizando.

Puertos y cortafuegos

En la era actual de virus y troyanos, un puerto abierto puede infectarse rápidamente. La primera línea de defensa contra este tipo de ataques suele ser una aplicación cortafuegos. Los cortafuegos supervisan las comunicaciones y pueden bloquear el acceso tanto de mensajes entrantes como salientes en cualquiera de los puertos. Una táctica estándar para minimizar los ataques potenciales es bloquear los puertos que no se necesitan activamente. Esto tiene una implicación importante para UDP, ya que los cortafuegos pueden bloquear el acceso a un puerto o impedir que las aplicaciones en ese puerto respondan. Las aplicaciones de cortafuegos normalmente tienen propiedades o ajustes que pueden configurarse para permitir el acceso a puertos específicos. Ten cuidado al abrir puertos o desactivar la protección del cortafuegos, ya que los nodos a los que se pueda acceder desde Internet serán vulnerables a los ataques.

El datagrama UDP



Datagrama UDP

	Descripción	bytes
Puerto de origen	La dirección del puerto del remitente.	2
Puerto de destino	El puerto que se utilizará en el destino. Tenga en cuenta que si no hay ninguna aplicación escuchando en ese puerto, el mensaje puede perderse.	1
Longitud del mensaje	La longitud en bytes del paquete de datos	2
Suma de comprobación	Suma de comprobación del paquete UDP. Gestionado automáticamente por el componente TCP/IP en modo UDP.	4+
Datos	Los datos del mensaje UDP. Los datos se envían sin procesar. Es responsabilidad de los programas emisor y receptor procesar estos datos.	0+

Tenga en cuenta que, al igual que con IP, normalmente no necesitará crear la cabecera, ya que el componente TCP/IP la gestiona automáticamente en modo UDP.

Tratamiento de datos UDP

UDP envía bytes de datos sin procesar. Depende totalmente de ti crear e interpretar los datos del mensaje. Esto es a la vez una bendición y una maldición.

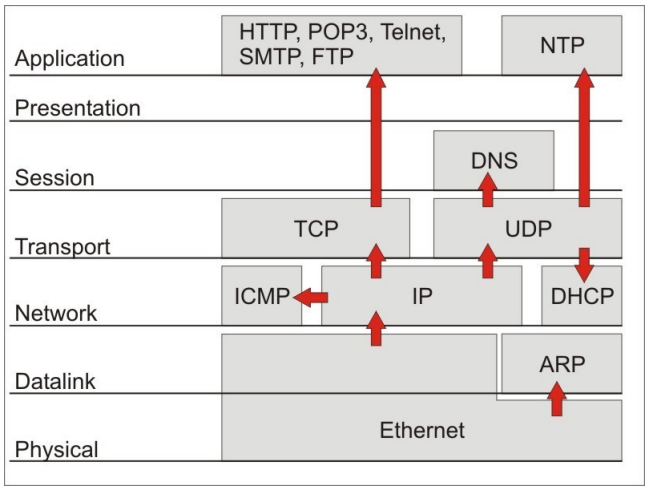
Es una bendición, ya que puedes decidir qué enviar: texto ASCII, valores enteros, bytes de datos para la salida del microcontrolador, etc., etc. Cuántos datos y en qué orden depende de usted también. Básicamente envías lo que quieres. Si desea enviar mensajes personalizados UDP es la forma de hacerlo.

Es una maldición en el sentido de que hay que saber qué esperar cuando se lee un mensaje. Esperar e intentar leer 20 bytes enteros cuando en su lugar se obtiene el texto ASCII de 5 bytes "Hola" podría causar errores.
UDP no tiene información de tipo ni ningún otro tipo de comprobación de errores integrada. Cualquier comprobación de errores debe ser realizada por la aplicación receptora. Como no tienes el control de quién envía un mensaje al socket que has elegido, debes ser consciente de que un mensaje recibido puede no estar necesariamente en el formato esperado. Podría ser, por ejemplo, de un programa completamente diferente con una estructura de mensajes personalizada completamente diferente.

Algunas tomas predefinidas, como el puerto 13, responden con un mensaje predefinido, por ejemplo, una cadena de fecha/hora.
pero son la excepción, no la regla, e incluso en este caso el formato de los datos puede variar de un sistema a otro

Implementación del modo UDP en Flowcode

El Flowcode UDP se sitúa en el nivel de transporte del modelo OSI. Los protocolos inferiores como la capa de transporte IP y la capa física Ethernet son manejados por el componente TCP/IP. En el modo UDP sólo necesita saber qué socket monitorizar y a qué socket enviar datos.



Bytes de puerto

Hay un gran número de Puertos disponibles, hasta 65535 en algunos sistemas. Sin embargo Flowcode está restringido a variables de 8 bits, es decir 0-255. Con el fin de aumentar la cantidad de puertos que podemos abordar la dirección del puerto se divide en dos bytes, un byte alto (src_port_hi) y un byte bajo (src_port_lo). Ambos bytes deben ser suministrados incluso para los puertos 0-255 que podrían ser direccionados con un solo byte (simplemente use el byte alto '0' en este caso).

Establecer una conexión UDP

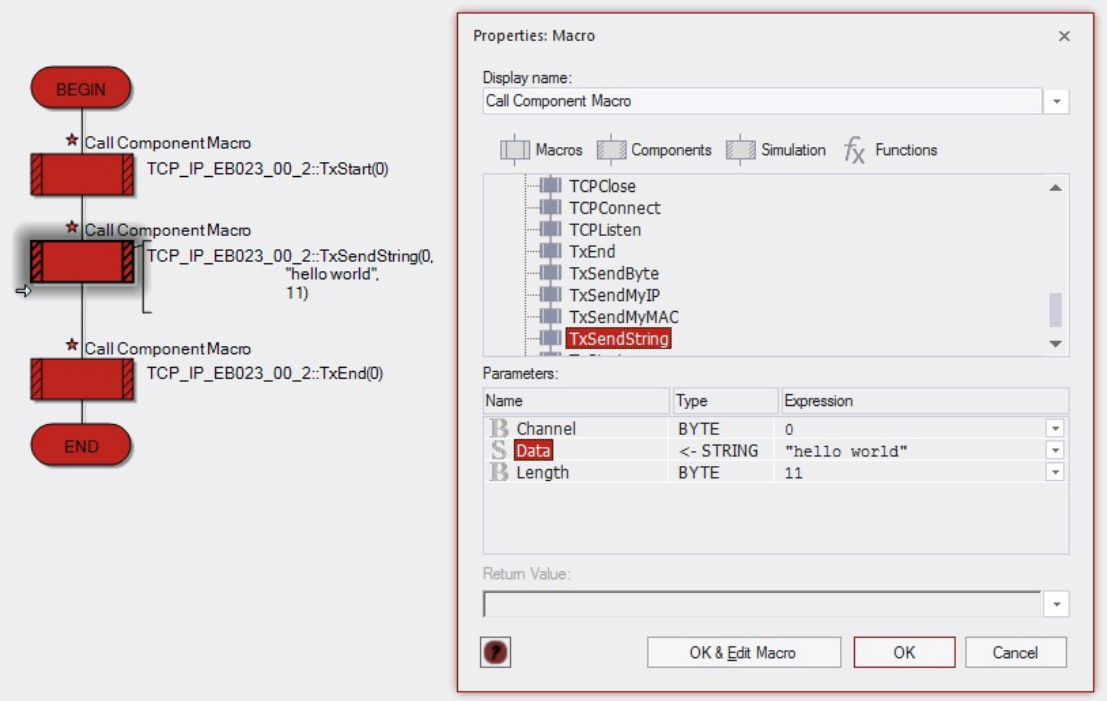
En primer lugar, añada la macro *Initialize* al principio de los programas como de costumbre. UDP necesita una dirección IP y un socket. La dirección IP se configura en la página de propiedades de los componentes TCP/IP, así que lo único que necesitamos configurar es el puerto del socket. La macro *CreateUDPSocket* toma los parámetros *channel*, y los dos bytes de dirección de puerto *src_port_hi* y *src_port_lo*. El puerto que elijas depende de ti. Una vez que el socket ha sido configurado puedes monitorizarlo usando RxDataAvailable.
Esto le permite recibir mensajes aunque no los responda.

Si necesita responder o enviar mensajes UDP necesita establecer el socket de destino al que desea enviar el mensaje. La macro *SetDestination* te permite establecer el canal, el ad- dress IP y los bytes de puerto del nodo de destino. Puede utilizar las macros RxReadByte, RxSkipBytes y RxMatchXXXX para procesar.
Una vez que haya establecido un destino, la macro *TxSendByte* enviará los datos a ese destino. Para cambiar el destino a un nodo diferente, utilice otra macro *SetDestination* con los nuevos datos de destino.

Envío de datos

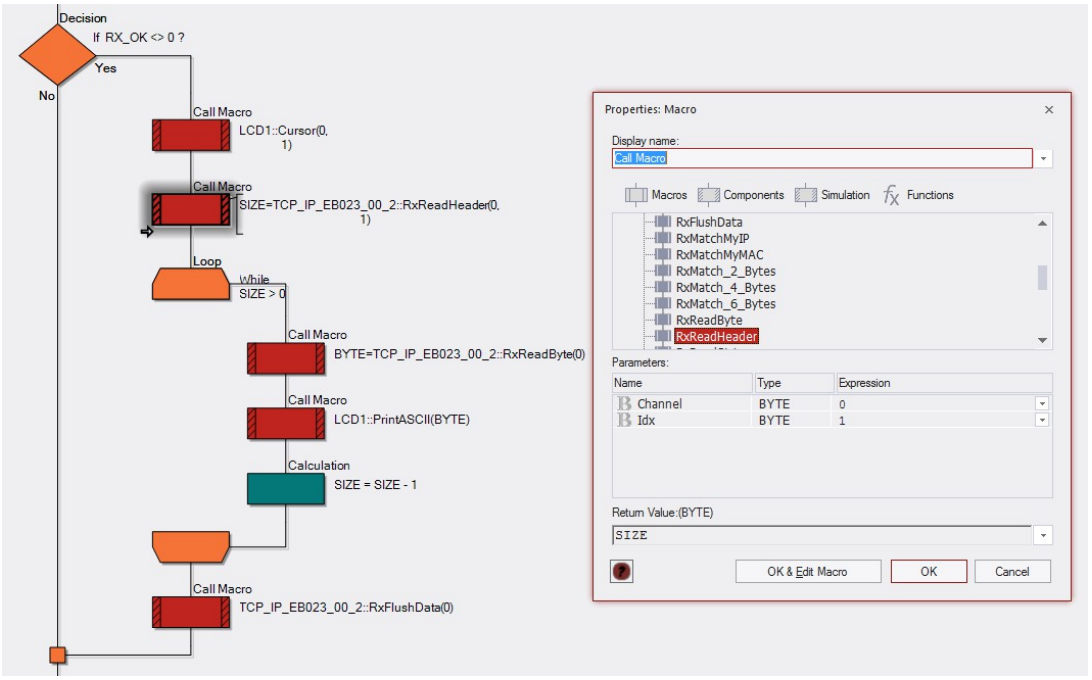
Una vez que hemos conectado el socket UDP, podemos enviar datos directamente.
No necesitamos configurar cabeceras ni ninguna otra información de envoltorio. Simplemente enviamos los datos utilizando las macros TxStart, TxSendByte y TxEnd.

Implementación del modo UDP en Flowcode



Recepción de datos

La recepción también es relativamente sencilla.
Podemos realizar la prueba habitual *RxDataAvailable* para buscar datos nuevos y leerlos utilizando
RxReadByte.
Sin embargo, hay algo que debemos saber. En primer lugar, ¿cuántos datos hay?
Podemos obtener el elemento de cabecera Longitud del mensaje utilizando la macro *RxReadHeader*. Los datos del elemento índice 0 de la cabecera UDP dan el byte alto de la longitud del mensaje, y el elemento índice 1 da el byte bajo de la longitud del mensaje.
Usando *RxReadHeader* podemos poner esto en variables convenientes como SIZE_HI y SIZE_LO. Una vez que tenemos la longitud del mensaje podemos hacer un bucle y *RxReadByte* en listo para ser tratado.



Implementación del modo UDP en Flowcode

Una vez que tenemos los datos, tenemos que tratarlos. Esto, por supuesto, dependerá de lo que los datos son (por ejemplo, el envío de texto ASCII a la pantalla LCD como en el fragmento de código anterior). Como UDP son datos personalizados necesitaremos saber qué datos se supone que son. Y puede que necesitemos comprobar si los datos son los que esperábamos y no un mensaje completamente diferente.

Acuérdate de vaciar el búfer una vez que hayas terminado para que pueda llegar el siguiente mensaje.

Uso de UDP

El punto clave que hay que captar aquí es el hecho de que usted podría ser responsable de ocuparse de todo, incluido el procesamiento de los datos brutos. Si quieres comprobar errores o respuestas, tendrás que hacerlo tú mismo en el programa. Sin embargo, para programas personalizados eso es exactamente lo que quieres - libertad para pasar datos o mensajes como mejor te parezca. Los datos son lo que quieres, no un datagrama preconfigurado como en ARP o ICMP.

Hay algunos protocolos que utilizan UDP como base, como DHCP y DNS. Para comunicarte con estos protocolos y utilizarlos, tendrás que buscar detalles sobre cómo deben estructurarse los datos para ellos. También tendrás que averiguar qué respuestas se envían y cómo se formatean.

Aunque relativamente sencillo, el protocolo UDP permite la comunicación directa entre dos sistemas que sólo necesitan estar conectados a través de una red, o conectados a Internet. La distancia física es irrelevante. Pero esto tiene un coste. Hay que conocer la dirección IP y el puerto correctos para poder enviar el mensaje, o éste desaparecerá. Además, tiene que haber una aplicación en el otro extremo que lea y responda al mensaje. Si no es así, no pasará nada.

Aunque esto está bien para programas personalizados, no es lo suficientemente robusto para estándares de comunicación globales como el correo electrónico SMTP o las páginas web HTTP. Lo que realmente necesitamos es algo que sea un poco más robusto, capaz de comprobar si hay errores, capaz de responder a peticiones o solicitudes, y capaz de manejar mensajes más grandes de forma automática. Aquí es donde entran en juego protocolos como TCP, con secuencias de comunicación definidas.

Ejercicio 3: Hora y fecha utilizando el modo UDP

En este Ejercicio recuperaremos la cadena de Hora y fecha del Socket 13 en un PC objetivo. El Socket 13 responderá a un mensaje UDP devolviendo la Hora y la fecha como una cadena ASCII. Tenga en cuenta que necesitará establecer el Destino a la IP del PC de destino.

Objetivo del programa:

- Utiliza UDP para recuperar la cadena de fecha y hora de un PC. Finaliza si no se recibe respuesta tras un periodo de 'timeout'.

Requisitos previos:

- Conocimiento de la capa MAC/Ethernet

Conocimiento de la capa IP

Información obligatoria

Dirección IP del PC de destino

Objetivos de aprendizaje:

- Estructura de un datagrama UDP
- Envío de mensajes UDP
- Recepción de mensajes UDP
- Respuestas preestablecidas de las tomas

Notas sobre el ejercicio 3: Tiempo y la fecha utilizando el modo UDP

No proporcionaremos un ejemplo práctico para el Ejercicio 3. En su lugar, ofreceremos notas y consejos sobre posibles dificultades. En su lugar, ofreceremos notas y consejos sobre posibles dificultades, así como elementos como los diagramas de flujo del programa, que pueden ser útiles a la hora de compaginar los folletos.

Advertencia del cortafuegos

Este ejercicio requiere acceso al puerto 13 para funcionar. Sin embargo, el puerto 13 puede estar bloqueado por un cortafuegos o no responder automáticamente a los mensajes por diversas razones. Se recomienda comprobar la disponibilidad del puerto. Si el puerto 13 no está disponible puede que tenga que hacer una de las siguientes cosas:

- Póngase en contacto con el administrador del sistema para configurar el acceso al puerto 13
- Utilizar otro puerto disponible

Utilice otra solución de Internet configurada para responder a los mensajes en el puerto 13 para imitar la función Fecha Hora.

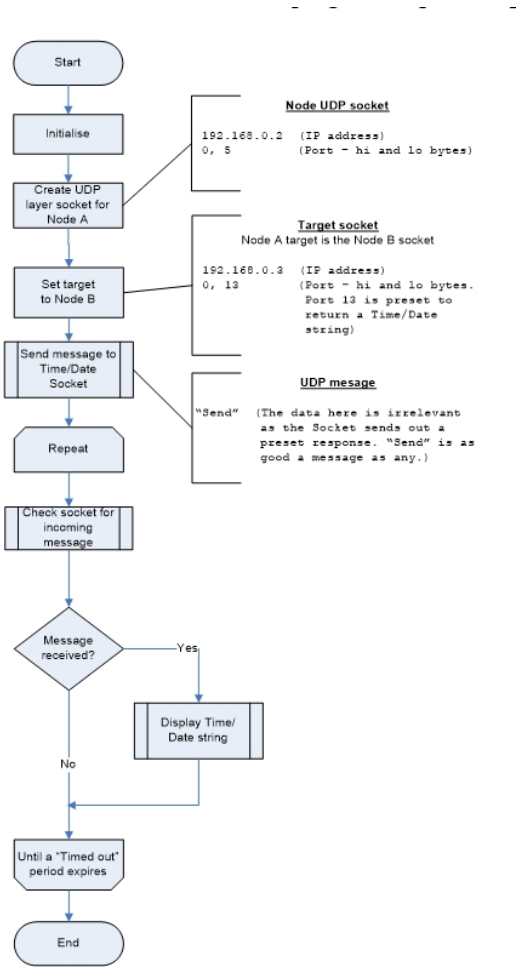
Notas generales

El sistema de destino deberá ser capaz de responder a una petición en el socket 13, como un PC estándar. En caso de duda sobre si un socket está produciendo una respuesta, se puede utilizar un inyector de paquetes, como Excalibur, y analizadores de tráfico de red, como Wireshark, para probar el socket.

Se necesita un socket para la conexión UDP. Puede estar en cualquier puerto, pero hay que tener en cuenta lo siguiente
y evitar el uso de sockets preestablecidos como el puerto 25, el puerto SMTP.

Diagrama de flujo del programa

La estructura básica de este programa es bastante sencilla, como se muestra en el diagrama de flujo siguiente.



Notas sobre el ejercicio 3: Tiempo y la fecha utilizando el modo UDP

Ejemplo de programa

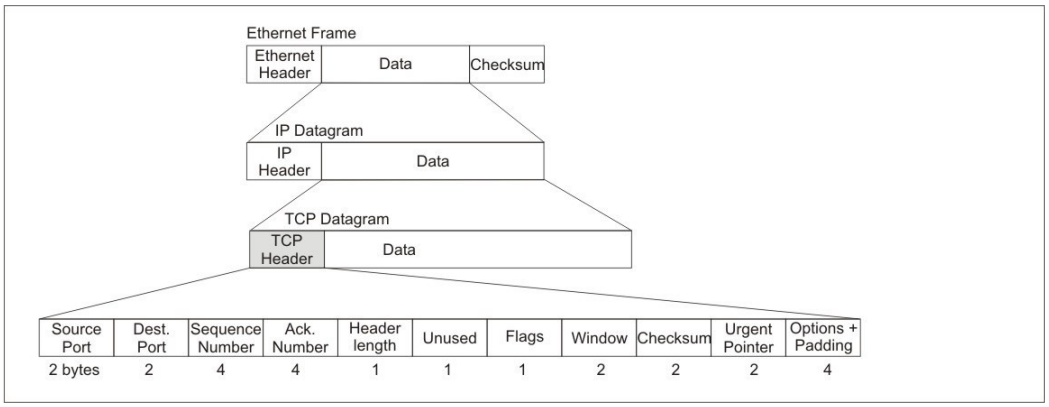
Encontrará un ejemplo de solución en la carpeta "TCP_IP\Examples" de Flowcode, o en la sección Ejemplos del CD, para que pueda consultarlo, utilizarlo como código y utilizarlo con fines de demostración.

Tenga en cuenta que las propiedades, los parámetros de las macros y los valores de las variables, etc., se configuraron para su uso con nuestra red y puede ser necesario cambiarlos para adaptarlos a su propia red o configuración del sistema.

TCP

Para el usuario general, TCP es lo que trae el mensaje de correo electrónico o la página HTML. A menudo se piensa que TCP es el código que utilizan estos programas. Que en cierto sentido lo es (ya que son los datos que transporta), pero en otro no. TCP es el proceso de control de la transmisión de esos datos. Para TCP los datos son irrelevantes; lo que cuenta es el envío. Son las aplicaciones que recogen los datos de TCP las que se preocupan de si es correo electrónico o HTML.

Los sockets predefinidos se utilizan mucho en TCP. Las aplicaciones de correo electrónico escucharán en un puerto, los programas HTML en otro. FTP y otros programas escanearán otros puertos. Si lo deseas, puedes incluso crear un programa personalizado para escanear un puerto en particular y enviarle datos personalizados utilizando TCP.



Cabecera IP

	Descripción	**bytes**	**Notas**
Puerto de origen	El puerto en el que se encuentra el socket TCP de envío conectado a.	2	
Puerto de destino	El puerto de destino al que va dirigido el mensaje se enviará.	2	
Número de secuencia	Se utiliza con la fragmentación para ayudar al reensamblaje del datagrama.	4	
Agradecimiento número	Se utiliza con la fragmentación para informar del envío. er de recepción satisfactoria del datagrama.	4	
Longitud de la cabecera	Longitud de la cabecera en bytes.	1	
No utilizado	Sin utilizar. Reservado para ampliación.	1	
Banderas	Se utiliza para almacenar banderas de bits para FIN, ACK, SYN, Reinicia y empuja.	1	
Ventana	Tamaño del búfer para los mensajes entrantes. es.	2	
Suma de comprobación	Un valor de suma de comprobación para la cabecera TCP.	2	
Puntero urgente	La dirección IP del nodo de destino.	2	
Opciones y acolchado	Hay varias opciones disponibles para la cabecera TCP. No vamos a utilizar ninguna opción en este curso, por lo que esta sección puede ser con seguridad ignorado por ahora.	4	

Conexiones

Las conexiones son la clave para entender TCP. El uso de TCP gira en torno al control y la coordinación de la conexión. Se trata de un proceso bidireccional, en el que las dos partesentablan un diálogo en lugar de un sistema de enviar, responder, responder a la respuesta, etc.

Agradecimientos

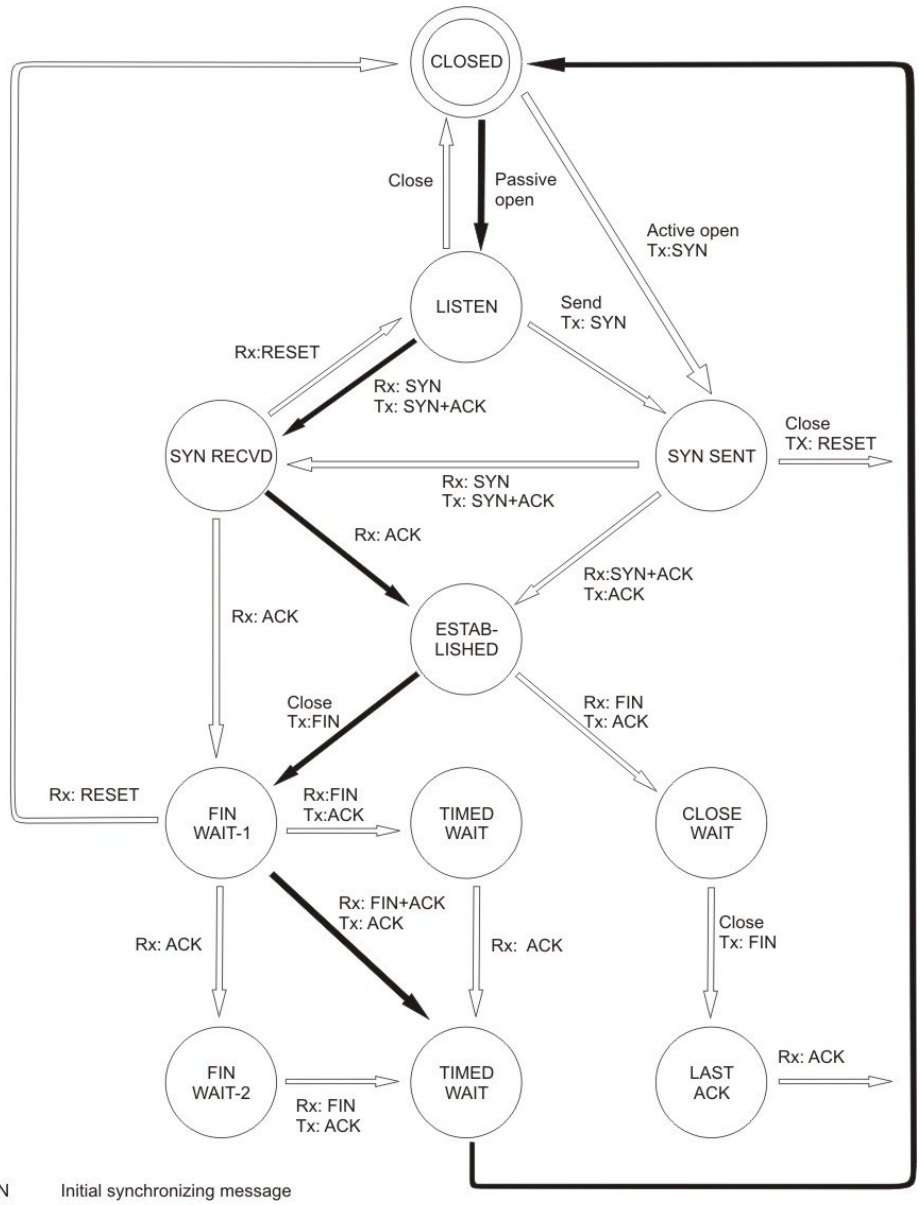
TCP es útil porque acusa recibo del mensaje a medida que llega. De hecho, se puede comprobar, enviar, comprobar, enviar, etc. en TCP para obtener los datos enviados y comprobar al mismo tiempo que se han recibido.

Algunos sistemas, como SMTP, requieren un diálogo activo con datos enviados en distintos pasos, con acuses de recibo enviados en etapas específicas, y varias etiquetas o marcadores enviados para distinguir partes de los datos que se envían.

Fragmentación

TCP permite enviar fácilmente grandes cantidades de datos. TCP permite dividir los mensajes en fragmentos que pueden enviarse como datagramas IP. A continuación, TCP puede enviar los datagramas IP de uno en uno (método lockstep), en bloques numerados (método de secuenciación de bloques) o por recuento de bytes (secuenciación de bytes), en el que ambas partes llevan la cuenta de la cantidad de bytes enviados y recibidos.

Los fragmentos se marcan con un número de secuencia para ayudar a reconstruir el mensaje en el destino. Esto permite que los fragmentos lleguen fuera de secuencia sin dañar los datos del mensaje. También ayuda a comprobar y reenviar los fragmentos perdidos.



SYN	Initial synchronizing message		
ACK	Acknowledgment		
FIN	Final closure message	Rx:	Recieved
RESET	Forced closure signal	Tx:	Tramsmitted

El objetivo de una aplicación es pasar de una conexión CERRADA a una ESTABLECIDA. La ruta depende de si la conexión se ha abierto de forma pasiva o activa. Una vez establecida la conexión, las aplicaciones pueden comunicarse entre sí hasta que una de las partes envíe un mensaje FIN de cierre final. En ese momento se inicia una secuencia de cierre en la que las aplicaciones intentan volver al estado CERRADO. En todas las etapas existe la posibilidad de que se produzca un fallo en las comunicaciones que provoque un RESET de vuelta a CERRADO.

Para establecer una conexión, la parte que inicia la comunicación debe enviar un mensaje SYN. Una vez recibidos un SYN y un ACK, se envía un ACK y se establece la comunicación.

El cierre de una conexión es similar. La parte que desea finalizar la conexión envía un FIN. Como respuesta se espera un FIN y un ACK, y un ACK final lo remata todo.

Aunque el diagrama de estados parece complejo a primera vista, el seguimiento del flujo muestra que los distintos caminos son sólo cuestión de quién inicia el contacto y quién lo termina y en qué orden envían la señal requerida. En las siguientes secciones se muestran algunas de las principales vías a través del diagrama de estado en ac- ción.

Abierto pasivo

Las aplicaciones pueden optar por pasar al estado Listen donde pueden escuchar peticiones SYN o, si lo desean, enviar una petición SYN propia para pasar al estado activo SYN SENT.

Cuando se recibe una petición SYN entrante envían un SYN+ACK para indicar que están listos para comunicarse y pasan al estado SYN RECVD esperando un ACK final antes de pasar a ESTAB- LISHED.

Activo abierto

En apertura activa la aplicación inicia la comunicación con un mensaje SYN pasando de CERRADO a SYN ENVIADO. Si no se recibe respuesta, se envía un RESET y la aplicación vuelve a estar cerrada. Si se recibe un SYN+ACK, es decir, la otra aplicación está lista y esperando, entonces se ha establecido una conexión y el proceso pasa a ESTABLISHED. Una tercera posibilidad es que la aplicación reciba un mensaje SYN, pero no un ACK, en cuyo caso la aplicación necesita transmitir un mensaje SYN+ACK y pasar a SYN RECVD y esperar un ACK antes de pasar a ESTABLISHED.

Cierre de la conexión

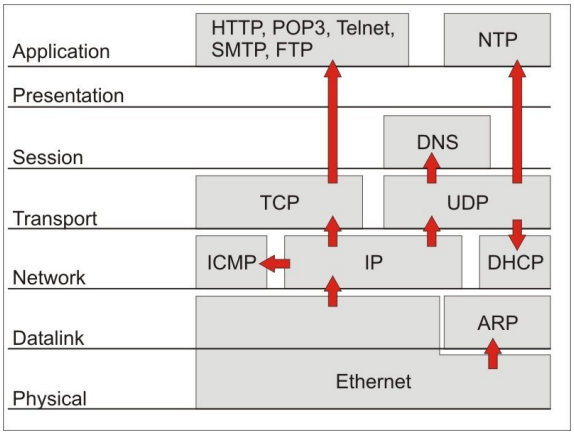
Para cerrar una conexión es necesario enviar un FIN para informar a la otra aplicación de que se desea finalizar la conexión. Una vez que se ha enviado un FIN, la aplicación debe esperar a recibir un FIN y un ACK para indicar que la otra aplicación está lista para cerrar la conexión. El FIN y el ACK pueden llegar en cualquier dirección, por lo que hay varias rutas posibles a través de los distintos estados hasta que la aplicación puede pasar a CERRADO. Si no se recibe ningún mensaje, la aplicación también puede hacer un RESET, ya que lo más probable es que en ese momento la comunicación se haya interrumpido.

Responder a una solicitud de cierre

Cuando una aplicación recibe un mensaje FIN necesita responder con un ACK y un mensaje FIN. Una vez que se han enviado, la aplicación pasa de CLOSE WAIT a LAST ACK y espera este último ACK antes de pasar a CLOSED.

Implementación del modo TCP en Flowcode

TCP te lleva al modelo OSI en el nivel de transporte. Aquí nos ocupamos de hacer llegar los datos a las aplicaciones y de comprobar los errores de ese proceso.



Los fundamentos del modo TCP incluyen:

- Configurar un socket Se conecta a un socket en el que otros nodos pueden comunicarse con el programa.
- Conectarse activamente o escuchar pasivamente

o Si no inicias activamente la comunicación, puedes escuchar pasivamente un mensaje y responder cuando te hablen.

o En el modo activo tendrá que establecer a qué toma desea enviar.

o Envío de datos Las macros *TxStart*, *TxSendByte* y TxEnd se utilizan para enviar la transmisión de datos en el

de la misma manera que con los modos anteriores.

La comprobación y la respuesta a los mensajes se gestionan de la misma forma que en los modos anteriores, con *RxDataAvailable*, *RxReadByte*, *RxMatchXXXX* y *RxFlushData* para comprobar y procesar los datos entrantes.

> Tenga en cuenta que, al igual que con IP y UDP, normalmente no necesitará crear la cabecera, ya que el componente TCP/IP la gestiona automáticamente.

Inicializar una conexión TCP

Puede crear una conexión TCP con la macro *CreateTCPSocket*, especificando qué canal utilizar y los bytes del número de puerto. TCP puede utilizar hasta 4 canales simultáneos, canal 0-3.

Conexión activa con TCPConnect

Si desea iniciar comunicaciones necesitará conectarse al socket apropiado en el otro nodo. Para conectarse para una transmisión activa necesita llamar a la macro *TCPConnect* con los bytes de canal, IP ad- dress y número de puerto. *TCPConnect* devuelve 1 si se puede establecer una conexión y 0 si la conexión no se ha podido establecer. *TCPConnect* gestiona los distintos mensajes SYN y ACK necesarios para establecer la conexión.

Conexiones pasivas con TCPListen

TCPListen te permite monitorizar un puerto en busca de mensajes. Puede permanecer en silencio y pasivo mientras monitoriza este puerto. Una vez que se ha recibido un mensaje, el componente TCP/IP establece automáticamente una conexión y usted puede responder al socket remitente.

Puedes usar la macro *GetSocketStatus para* monitorizar el estado del socket y ver si alguna aplicación externa está intentando conectarse. La sección del archivo de ayuda sobre *GetSocketStatus* enumera los distintos estados y sus valores de retorno.

Observe que muchas de las variables están relacionadas con los distintos estados del diagrama de estados (las otras están relacionadas principalmente con las formas de transferencia de datos). Los tres estados más importantes para la monitorización son SOCK_CLOSED - valor 0, SOCK_ESTABLISHED - valor 6 y SOCK_CLOSE_WAIT - valor 7.

Envío de datos

Las macros *TxStart*, *TxSendByte* y TxEnd se pueden utilizar para realizar la transmisión de datos real una vez que se ha establecido una conexión. Tenga en cuenta que muchas aplicaciones basadas en TCP esperan que los datos se envíen en una secuencia específica.

Comprobación y recepción de datos

Al igual que en los modos anteriores con RxDataAvailable se comprueba si hay datos entrantes. RxReadByte, RxMatchXXXX y RxReadHeader pueden utilizarse para extraer y comprobar los datos. RxFlushData se utiliza para borrar los datos actuales y prepararlos para el siguiente mensaje.

Cerrar una conexión

TCPClose gestiona la secuencia de cierre de forma activa o pasiva dependiendo de si usted o la aplicación receptora ha iniciado el cierre.

Una vez que hayas terminado con la conexión TCP necesitas cerrarla activamente. Llame a la macro *TCPClose* para cerrar la conexión. TCPClose manejará los distintos mensajes FIN y ACK necesarios para cerrar la conexión.

Si la conexión necesita ser cerrada por la aplicación receptora, ésta transmitirá un mensaje FIN que pondrá el socket de tu aplicación en el estado SOCK_CLOSE_WAIT. Puedes monitorizar el estado del socket usando *GetSocketStatus*. Una vez detectado el estado SOCKET_CLOSE_WAIT se debe llamar a la macro *TCPClose* para cerrar la conexión.

<CRLF>, comillas y otros caracteres problemáticos

Mientras que la mayoría de los caracteres se pueden escribir en la macro *TxSendByte* hay algunos extra que necesitamos tener en cuenta. No podemos enviar la tecla Return/Enter para ir a la siguiente línea ya que no entra en la lista de parámetros. Esto se debe a que la tecla Return es un carácter de control no imprimible. Sin embargo, todos los caracteres, incluidos los de control, tienen un valor ASCII asociado que podemos utilizar en su lugar. La tecla Retorno es un poco inusual, ya que en realidad se añade como dos códigos de caracteres en lugar de uno solo - el carácter de retorno de carro (código de carácter 13) y el carácter de avance de línea (código de carácter 13). A menudo se abrevia como CRLF. Todo lo que tenemos que hacer es enviar los códigos de caracteres ASCII en su lugar, en este caso los códigos ASCII 13 y 10 al final de cada línea de código de correo electrónico. (Todo esto es una reminiscencia de los primeros días de la informática, cuando se imprimía en impresoras electrónicas en las que se movía el cabezal de la impresora).

Los mensajes de respuesta SMTP le indican que finalice los datos del correo electrónico con la etiqueta de mensaje <CRLF>.<CRLF>
es decir, un punto y aparte en una línea. Esto tendría que enviarse como caracteres 13, 10, '.', 13, 10.

Otro carácter problemático es el carácter de comilla ". El carácter de comillas es utilizado por Flowcode para distinguir entre cadenas, por ejemplo "Hola mundo", y variables, por ejemplo CONTAR. Por lo tanto, no podemos enviar el carácter " porque confundiría a Flowcode. En su lugar, debemos enviar el código ASCII para ", que es el código de carácter ASCII 34.

Secuencias de comunicación

Las comunicaciones TCP se basan generalmente en secuencias, en las que un mensaje requiere una respuesta, que puede ser tratada o respondida. Entonces se necesitaría otra respuesta y así sucesivamente hasta completar la secuencia.

Para comunicarse con éxito con aplicaciones TCP es necesario saber cuáles son estas secuencias, qué respuestas esperar y qué códigos de error se enviarán. Algunas, como una petición HTTP GET pueden ser una simple respuesta. Otras, como SMTP, pueden requerir una serie de pasos en la secuencia con respuestas específicas que se requieren o se envían en varias etapas.

Es la comprensión de las secuencias de comunicación lo que desbloqueará las aplicaciones TCP. Y son las secuencias las que tendremos que ver en los ejercicios y ejemplos de TCP.

Ejercicio 4: Envío de una página HTML mediante HTTP

Instrucciones

Crear una página web básica que pueda visualizarse en el PC en un navegador web.
Para este ejercicio utilizaremos una única página que se devuelve para todas las peticiones GET, por lo que no necesitamos comprobar qué página se solicita.

Objetivo del programa:
Enviar datos HTML cuando se recibe una petición HTPP GET.

Requisitos previos:

- Conocimiento de la capa MAC/Ethernet
- Conocimiento de la capa IP
- Necesitará un PC con un navegador de Internet conectado a la red para ver la página HTML. Un analizador de tráfico de red también será útil para esta tarea en particular para ayudar en la depuración de los datos que se envían y reciben.

Información obligatoria
Código HTML a enviar. A continuación se muestra un fragmento de HTML que puede enviar:

```
HTTP/1.0 200 OK
Tipo de contenido: text/html

<html>
<head>
<title>Página web de computación TCP/IP</title>
</head>
<body>
<b>Hola Mundo</b>
<p>¿Cómo estás hoy?</p>
</body>
</html>
```

Resultados del aprendizaje:

- Estructura de un datagrama TCP.
- Diagrama de estado TCP
- Crear una conexión TCP de escucha pasiva
- Envío de mensajes TCP.
- Recepción de respuestas TCP.
- Responder a un mensaje.
- La estructura de la solicitud GET
- Código HTML básico

Notas sobre el Ejercicio 4: Envío de un HTML página mediante HTTP

Esta sección contiene notas y consejos sobre áreas de dificultad potencial, y proporciona elementos tales como diagramas de flujo del Programa que pueden ser útiles para cotejar los folletos.

Requisitos previos

Para ver la página HTML necesitará un PC con un navegador de Internet conectado a la red. Para ver el HTML tendrá que dirigir su navegador a la página del nodo. Abra el navegador y vaya a la dirección IP que configuró para el componente TCP/IP (por ejemplo http://192.168.0.2). La página por defecto será enviada para todas las peticiones por lo que no necesitamos preocuparnos por el nombre de la página.

Herramientas útiles

Un analizador de tráfico de red también será útil para esta tarea en particular para ayudar a depurar los datos que se envían y reciben. El analizador de tráfico de red también le permitirá supervisar el estado de TCP y ver los distintos mensajes SYN, ACK y FIN enviados.

Ejemplo de HTML

A continuación se muestra un fragmento de HTML junto con la sección de cabecera correspondiente.

```
HTTP/1.0 200 OK
Tipo de contenido: text/html

<html>
<head>
<title>Página web de computación TCP/IP</title>
</head>
<body>
<b>Hola Mundo</b>
<p>¿Cómo estás hoy?</p>
</body>
</html>
```

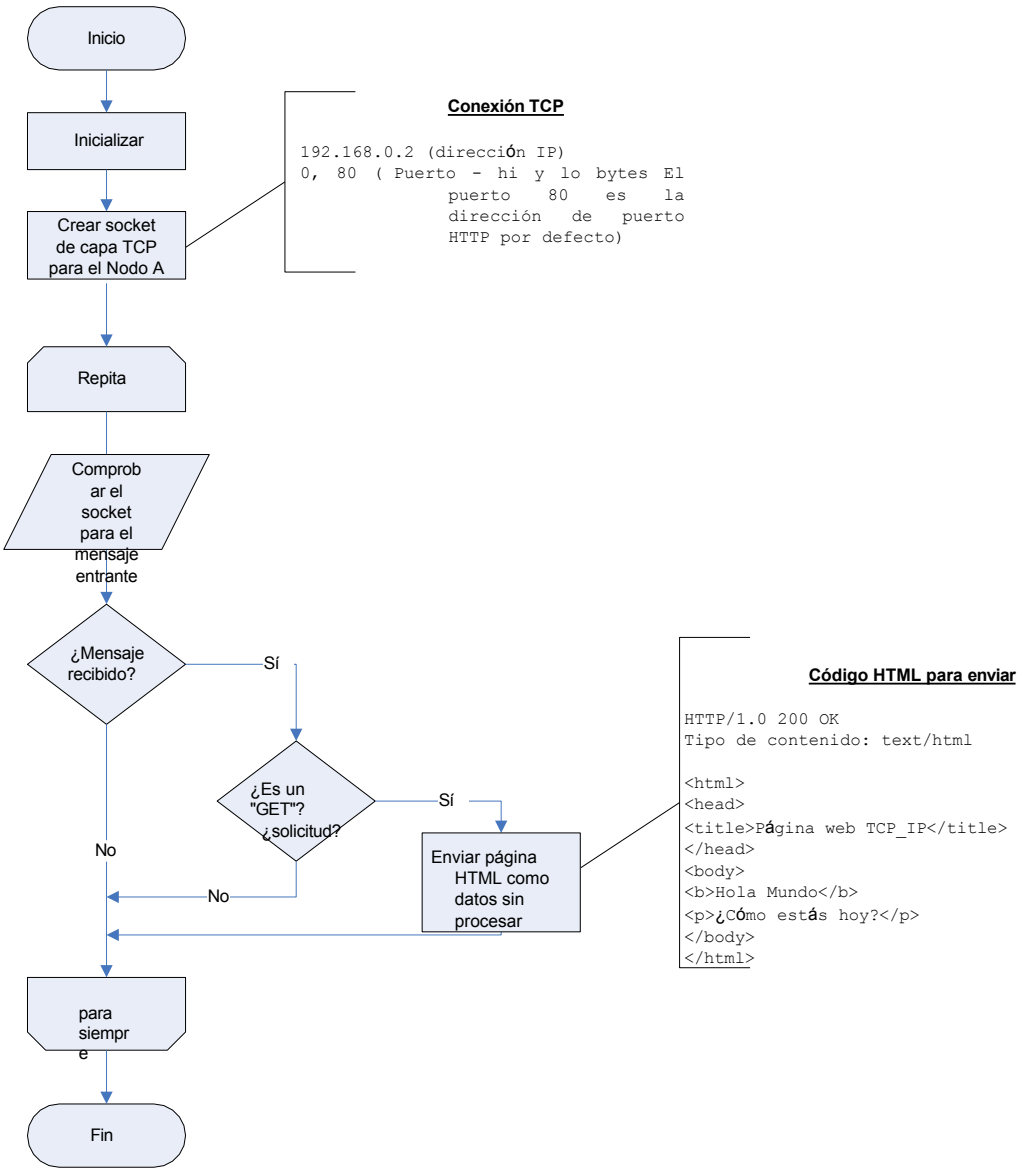
Las dos primeras líneas son una cabecera HTTP para informar a la aplicación del navegador del formato y tipo de mensaje que se envía. El resto de los datos es el código HTML de la página web. (En Internet se puede encontrar información sobre cabeceras HTTP y tutoriales sobre código HTML).

Tenemos que dividir el código HTML para introducirlo en el datagrama TCP. Necesitaremos enviar los códigos de caracteres 13 y 10 al final de cada línea para los valores ASCII de la tecla de retorno CRLF. (Para su propio código HTML también pueden ser necesarios otros códigos de caracteres ASCII, como ASCII 34 para las comillas, dependiendo de su código HTML).

Notas sobre el Ejercicio 4: Envío de un HTML página mediante HTTP

Diagrama de flujo del programa

Los elementos estructurales generales del programa son los siguientes.



Creación de la conexión

El HTML se envía normalmente al puerto 80, por lo que tendremos que configurarlo en la macro *TCPConnect* (*dst_port_hi* = 0,
dst_port_lo = 80).

No estamos iniciando comunicaciones por lo que podemos usar el modo pasivo con *TCPListen*. El socket puede ser comprobado usando *RxDataAvailable* para ver si ha llegado una petición a la que haya que responder.

Comprobación de la solicitud de página web

Una transferencia HTML se inicia con el envío de un comando GET al nodo que contiene la página web.
Por ejemplo GET CARPETA/PAGINA.HTM HTTP 1.0

RxMatch_4_Bytes puede utilizarse para comprobar si se trata efectivamente de un mensaje "GET". (Tenga en cuenta que tenemos que comprobar el
para completar los cuatro caracteres necesarios para la macro. Una cosa pequeña, pero tan fácil de pasar por alto).

Notas sobre el Ejercicio 4: Envío de un HTML página mediante HTTP

Como estamos enviando la misma página para todas las peticiones, podemos ignorar la parte de petición de página de la búsqueda GET para este programa.

Envío de datos

El envío real es fácil, sólo los mismos comandos *TxSendByte* que hemos utilizado anteriormente. No olvides las macros *TxStart* y *TxEnd*.

Cierre de la conexión

Una vez que los datos han sido enviados necesitamos cerrar la conexión TCP para terminar el proceso de transmisión con la macro *TCPClose*. Si la página web es de una sola vez, podemos simplemente cerrar la conexión y terminar el programa. Sin embargo, si se va a acceder a la página web varias veces, debemos restablecer la conexión para la siguiente solicitud. Para ello podemos volver a llamar a las macros *CreateTCPSocket* y *TCPListen*.

Dar sabor a la página

El código HTML de ejemplo es un poco soso, pero puede mejorarse fácilmente.
Cambie <body> por <body bgcolor=0000ff text=ffff00> por ejemplo y la página será un poco más colorida. Tenga en cuenta que el objetivo es enviar datos HTML, no crear una página artística, por lo que el diseño es secundario para que el programa funcione.

Sugerencias para seguir trabajando

Marcadores variables

Con frecuencia, los navegadores almacenan el HTML recibido en una caché que puede utilizarse en su lugar cuando la página es inutilizable. Esto puede causar problemas en las pruebas, sobre todo si hemos cambiado el código HTML. Pulsar refrescar obliga al navegador a obtener una copia nueva de la página y suele solucionar el problema. Sin embargo, tener un marcador de algún tipo en la página que se altere cada vez que se envía es útil para comprobar que la página se recupera correctamente y no sólo se trae de la caché. Para nuestro ejemplo, añadir una variable que se actualice cada vez que se accede a la página.

Varias páginas

El programa básico descrito anteriormente envía una única página en respuesta a una petición GET. Un proyecto interesante podría ser ampliar el sistema para gestionar solicitudes de varias páginas.

Tratamiento de errores con varias páginas

Si tiene un sistema con múltiples páginas web, y el código para comprobar qué página enviar, lo ideal sería crear una página predeterminada 'Archivo no encontrado' para enviar si se solicita una página inexistente. Esta es la infa- mosa página de error 404 con la que los usuarios habituales de la web estarán muy familiarizados.
Por ejemplo:

HTTP/1.0 404 No
encontrado Content-
type: text/plain

Archivo no encontrado.

Imágenes y otros archivos

No hemos cubierto imágenes u otros tipos de archivos permitidos, ya que el pequeño tamaño de la memoria del microcontrolador dificultaría el almacenamiento de datos de imagen. Pero el principio es el mismo. Envíe la cabecera con el tipo de archivo y, a continuación, los datos.
Por ejemplo:

HTTP/1.0 200 OK
Tipo de contenido: image/gif

FF006735BBCC234788.....etc.

Notas sobre el Ejercicio 4: Envío de un HTML página mediante HTTP

Ejemplo de programa

Encontrará un ejemplo de solución en la carpeta "Flowcode\TCP_IP\Examples" de Flowcode, o en la carpeta
"Ejemplos" del CD, para que pueda consultarlos, codificarlos y utilizarlos con fines de demostración.

> Tenga en cuenta que las propiedades, los parámetros de las macros y los valores de las variables, etc., se configuraron para su uso con nuestra red y puede que sea necesario modificarlos para adaptarlos a su propia configuración de red o de sistema.

Ejercicio 5: Recepción de HTML

EBLOCKS2

Instrucciones

Solicitar una página de un servidor web y mostrar una parte del código HTML re-encendido en la pantalla LCD.

Objetivo del programa:

- Recuperar datos HTML con una petición HTPP

GET. Visualizar los datos entrantes en la pantalla LCD.

Requisitos previos:

- Conocimiento de la capa

MAC/Ethernet Conocimiento de la capa IP

Información obligatoria

Dirección IP del servidor que solicita la página

Objetivos de aprendizaje:

- Estructura de un datagrama TCP.
- Diagrama de estado TCP
- Crear una conexión TCP
- Envío de mensajes TCP.
- Recepción de respuestas TCP.
- Responder a un mensaje.
- La estructura de la solicitud GET

Notas sobre el Ejercicio 5: Recepción de HTML

Esta sección contiene notas y consejos sobre áreas de dificultad potencial, y proporciona elementos tales como diagramas de flujo del Programa que pueden ser útiles para cotejar los folletos.

Requisitos previos

Necesitará un servidor capaz de enviar la página solicitada. Es posible que desee probar la página está disponible en un navegador de Internet. Es preferible una página sencilla, ya que una página grande y compleja con múltiples partes e imágenes puede ser demasiado para el programa básico.

Un Sever es simplemente un sistema capaz de "servir una página web HTML" cuando se le solicita.
El ejercicio anterior - Enviar HTML usando HTTP es un ejemplo de un servidor bastante básico. Aunque generalmente los Servidores son grandes y sofisticados sistemas informáticos que realizan también una miríada de otras tareas de red.

Herramientas útiles

Un analizador de tráfico de red también será útil para esta tarea en particular para ayudar a depurar los datos que se envían y reciben. El analizador de tráfico de red también le permitirá supervisar el estado de TCP y ver los distintos mensajes SYN, ACK y FIN enviados.

Recepción del HTML

Recibir el HTML es bastante sencillo; sin embargo, el microcontrolador no tiene mucha memoria ni muchas capacidades gráficas (incluso con una pantalla LCD), lo que limitará el tratamiento de los datos resultantes.

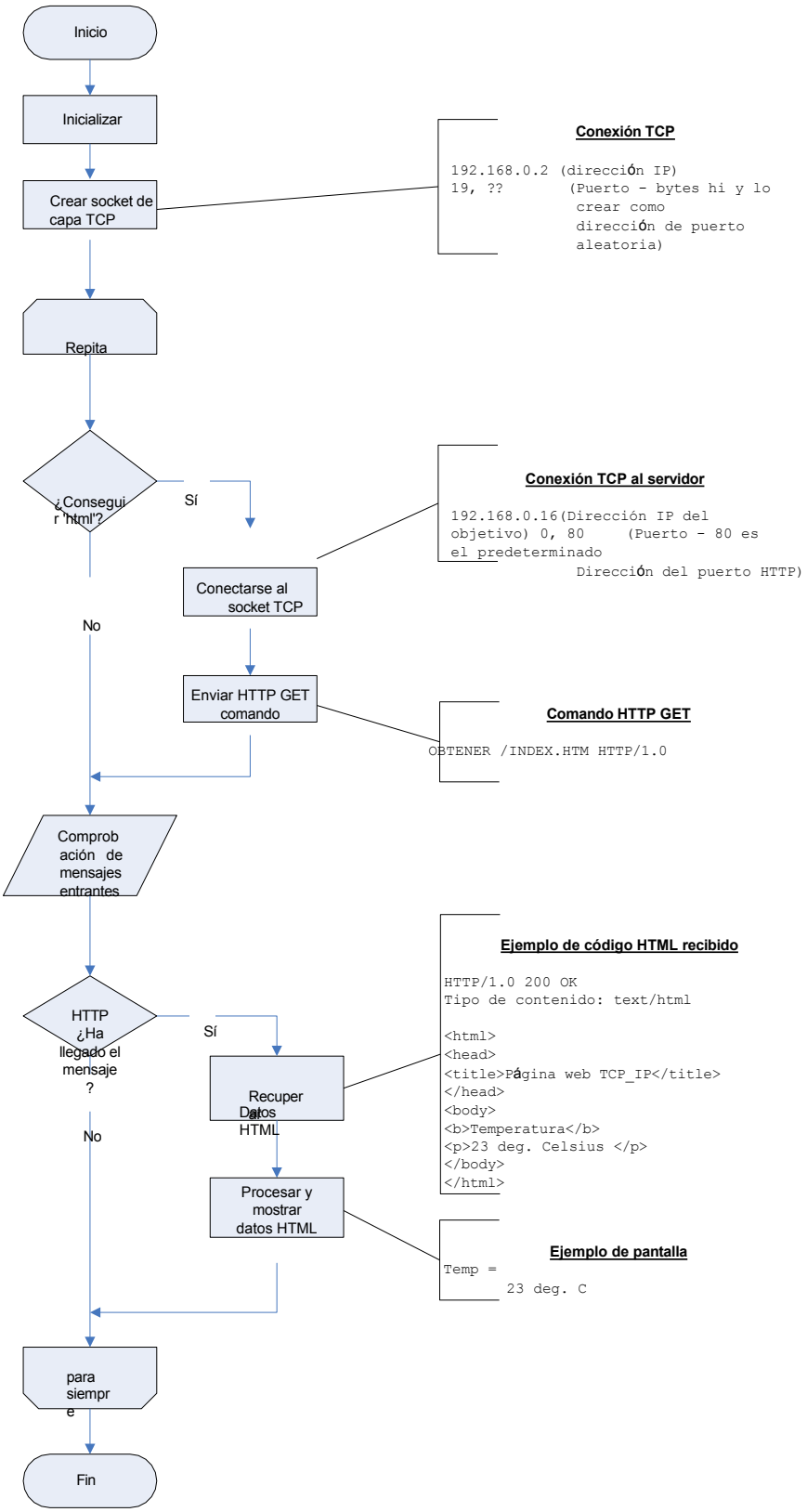
Mostrar los datos es necesario para demostrar que efectivamente hemos recibido la página web. Sin embargo, se puede elegir qué datos mostrar. Por ejemplo:

- Los datos pueden mostrarse tal y como se reciben permitiendo que desborden la pantalla LCD.
- Se puede mostrar una cantidad específica de datos, por ejemplo, suficiente para llenar la pantalla LCD.
- Se puede buscar una etiqueta o un fragmento de texto concretos y extraer del código un fragmento específico de texto HTML.

• Resumen del programa

El proceso básico puede resumirse en el siguiente diagrama de flujo:

Notas sobre el Ejercicio 5: Recepción de HTML



Solicitar la página HTML

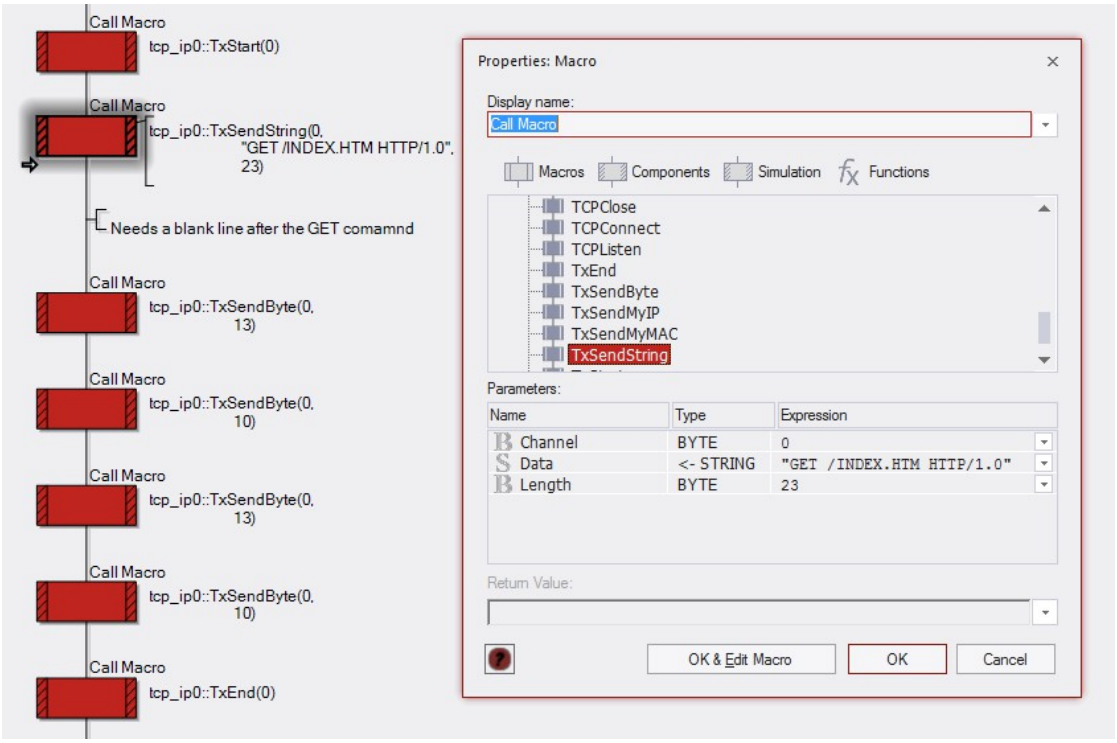
Para obtener una página web es necesario inicializar el componente TCP/IP y luego conectarlo utilizando la macro *TCPConnect* a la dirección IP del servidor que tiene la página web, y al puerto HTML del servidor - normalmente el puerto 80 Por ejemplo, Chan- nel 0, IP 192.168.0.17 Puerto 0, 80.

Para recibir una página de HTML necesita solicitar la página que desea al servidor con una petición GET con el nombre de la página, y la versión HTTP (podemos usar HTTP/1.0 para esto). La petición GET es

seguida de una línea en blanco, por lo que necesitaremos CRLF a la siguiente línea y CRLF de nuevo para producir la línea en blanco.

OBTENER CARPETA/PAGINA.HTM HTTP 1.0

Todo el proceso puede verse en la siguiente imagen.



Una vez que el Servidor reciba el comando GET comenzará a enviar los datos.
En este punto tenemos que empezar a comprobar si hay datos entrantes y leerlos cuando lleguen. Es posible que no podamos almacenar todos los datos debido al tamaño relativamente pequeño de la memoria del microcontrolador, por lo que tendremos que trabajar a través de ella y recuperar los datos allí mismo y mostrarlos, o hacer cualquier proceso que necesitemos hacer con ellos, antes de pasar al siguiente lote de datos entrantes. Si los datos HTML entrantes son particularmente grandes, pueden llegar en más de un mensaje.

Otros trabajos

Ahora que podemos introducir datos HTML, podemos procesarlos. Esto abre todo tipo de posibilidades. Si sabemos que un dato determinado, por ejemplo una lectura de temperatura, estará en los datos en una etiqueta o con una redacción determinada, podemos buscar, recuperar y mostrar esa información específica utilizando las macros *RxMatchXXXX*. Por ejemplo si el HTML enviado en una página contuviera la información "Temperatura = ?? deg C". Podríamos ampliar el programa para buscar el valor de la temperatura y mostrar el valor en la pantalla LCD.

Ejemplo de programa

Encontrará una solución de ejemplo en la carpeta "Flowcode\TCP_IP\Examples" de Flowcode, o en la sección "Examples" del CD, para que pueda consultarla, utilizarla para el código y utilizarla con fines de demostración.

Tenga en cuenta que las propiedades, los parámetros de las macros y los valores de las variables, etc., se configuraron para su uso con nuestra red y puede que sea necesario modificarlos para adaptarlos a su propia configuración de red o de sistema.

Ejercicio 6: Envío de un mensaje de correo electrónico SMTP

Instrucciones

Cree un programa de correo electrónico básico que envíe un mensaje predeterminado a una dirección de correo electrónico preestablecida. (Deseamos probar el lado de comunicación de SMTP así que un simple mensaje predeterminado será suficiente).

Objetivo del programa:

- Crear una conexión TCP.
- Abrir un diálogo estructurado con el servidor de correo. Comprobación de errores en las respuestas del diálogo.

Requisitos previos:

- Conocimiento de la capa MAC/Ethernet
- Conocimiento de la capa IP

Conocimiento de TCP (como se desprende de los ejemplos anteriores de TCP)

Información obligatoria

- Dirección IP del servidor SMTP de correo electrónico

Dirección de correo electrónico utilizable en la red que puede utilizarse para comprobar el mensaje.

Resultados del aprendizaje:

- Conexiones TCP
- Realización de diálogos TCP estructurados
- Respuestas de comprobación de errores
- Formato de datos SMTP

Notas sobre el Ejercicio 6: Envío de un SMTP

Las notas aquí detallan las secuencias involucradas en el envío de un correo electrónico SMTP. Esta sección es bastante extensa en comparación con las notas de las implementaciones TCP anteriores, ya que la secuencia incluye una serie de pasos y requiere relés específicos en etapas concretas. Sin embargo, es una tarea que merece la pena completar, ya que muchas aplicaciones que utilizan comunicaciones TCP requieren diálogos complejos similares.

Correo electrónico SMTP

El correo electrónico es algo con lo que todos estamos familiarizados. Los sistemas de correo electrónico modernos pueden ser muy sofisticados, con gráficos, texto formateado y archivos adjuntos, pero su esencia es el sencillo sistema original basado en texto. Los elementos llamativos no son más que diferentes formas de dar formato a los datos. En su forma más básica, el correo electrónico no es más que un trozo de texto con información preliminar que indica a dónde va el mensaje, quién lo envía, el tipo de mensaje, las opciones, etc. De hecho, se parece mucho a la división de datos y cabecera que hemos visto en los datagramas TCP/IP.
Un correo electrónico se envía como un datagrama TCP al socket SMTP del servidor.
La gran diferencia entre enviar un datagrama TCP al socket de correo SMTP y enviar un datagrama UDP es que el socket SMTP espera que haya un diálogo. Tú dices "Hola", y él te responde para decirte que está listo para ti. Le das la información "Para" y "De" y te pedirá el mensaje de correo electrónico, etc. Incluso se despedirá cuando hayas terminado el mensaje.
El núcleo de esta sección consistirá en explicar el sistema, no en explicar el código. Deberías estar ya en un nivel en el que puedas crear, probar y depurar el propio código.

Mensajes clave de la transmisión

El envío del mensaje consiste en enviar mensajes clave, esperar el acuse de recibo correcto y proceder al paso siguiente. Los siguientes son algunos mensajes clave que alertan al socket SMTP sobre la fase en la que se encuentra la transmisión del mensaje.

- HELO [Nombre del sistema] - un mensaje de introducción para iniciar la secuencia de mensajes. Nota: Es correcto, es HELO y no HELLO. Todos los diálogos de secuencia comienzan con una palabra de cuatro caracteres (excepto '.').
- MAIL FROM:<me@myaddress.com> - la dirección desde la que se envía el correo electrónico.
- RCPT TO:<me@myaddress.com> - la dirección de correo electrónico del destinatario.
- DATOS - un marcador para indicar que está listo para enviar los datos. Esto será seguido por el texto del mensaje en sí.
- . - un solo punto en una línea propia. Se utiliza para indicar que se han enviado los datos del mensaje.

QUIT - un mensaje para terminar el proceso.

Códigos de acuse de recibo SMTP

En varios momentos de la comunicación TCP, el cliente responderá con un mensaje de confirmación. Pueden ser varios mensajes, como una introducción, un eco de los datos enviados o un mensaje de ok de datos. Los mensajes van precedidos de un código de tres dígitos que podemos validar para comprobar que el proceso funciona correctamente. El número de código y los datos de acuse de recibo contienen mucha información útil para la comprobación de errores y la resolución de problemas. Aquí sólo nos centraremos en las respuestas que esperamos. Encontrará más detalles sobre los códigos de respuesta SMTP en las guías y especificaciones SMTP, disponibles en Internet.
Los que utilizaremos en el envío de los datos de correo electrónico son:

- 220 [Detalles del servidor] - el servidor se presenta.
- 250 [Datos del mensaje repetidos] - el servidor acusa recibo de los datos enviados y los devuelve para que puedan ser verificados en caso necesario.
- 354 Ok enviar datos terminados en <CRLF>.<CRLF> - el servidor está listo para recibir los datos del mensaje.
- 221 [Nombre del servidor] cerrando conexión - se ha recibido con éxito un mensaje QUIT.

Notas sobre el Ejercicio 6: Envío de un SMTP

Enviar un mensaje de correo electrónico paso a paso
El siguiente tutorial le llevará a través del envío de un simple mensaje de correo electrónico.
Los datos enviados, y los acuses de recibo del cliente para un típico serán mostrados en secuencia.
El siguiente tutorial le guiará a través del envío de un simple mensaje de correo electrónico. Los datos enviados, y los acuses de recibo del cliente para un típico se mostrarán en secuencia.

- Conexión TCP iniciada.
- Respuesta: 220 [Mensaje de introducción y detalles del servidor]
- HELO MI_NOMBRE
- Respuesta: 250 [Nombre del servidor]
- CORREO DE: <me@my_address.com>
- Respuesta: 250 Remitente <me@my_address.com>
- RCPT PARA: <you@your_address.com>
- Respuesta: 250 Destinatario <usted@su _dirección.com>
- DATOS

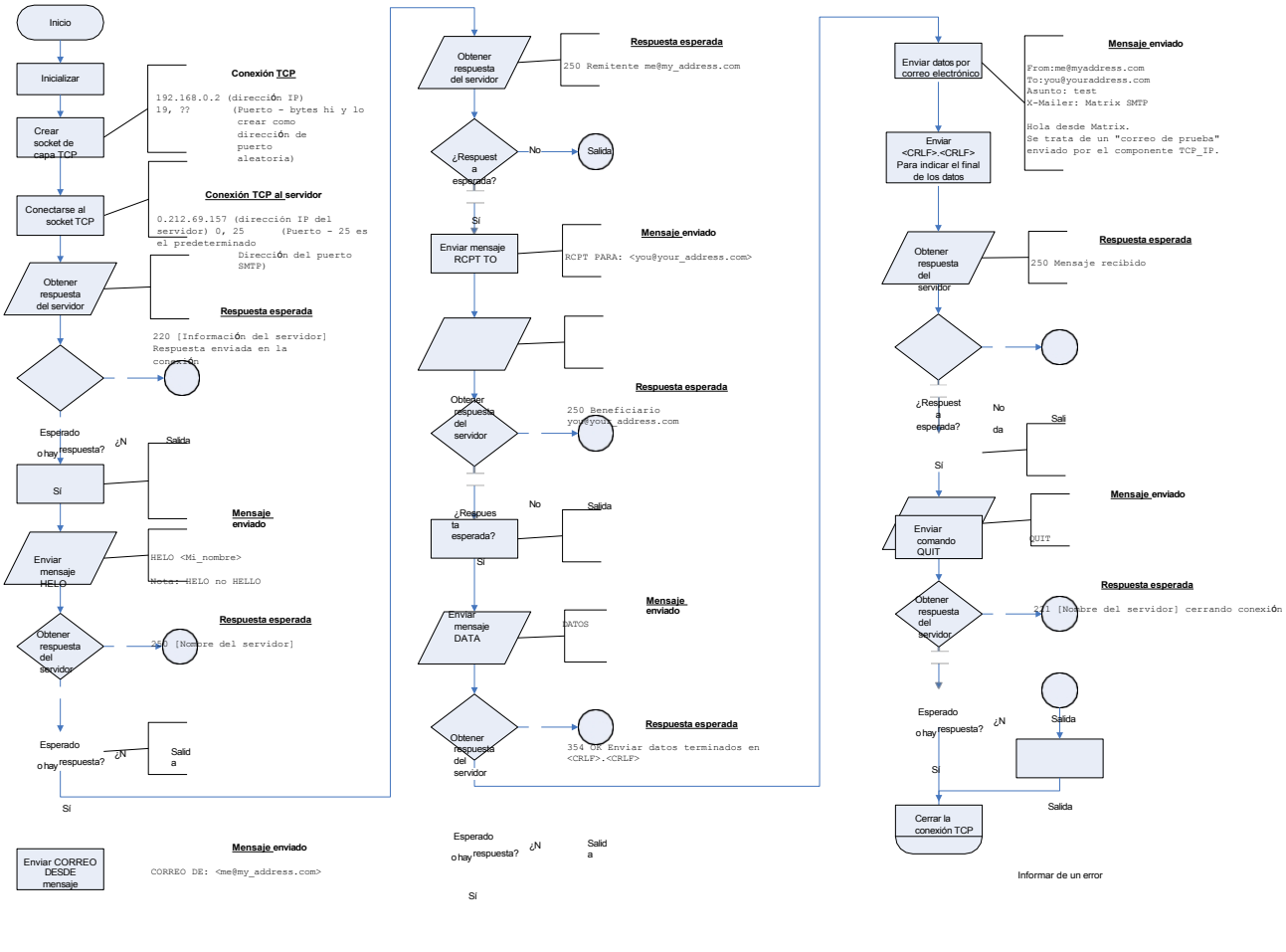
Respuesta: 354 Ok enviar datos terminados en <CRLF>.<CRLF>

.... Se envían los datos del mensaje de correo electrónico

- . (se trata de la etiqueta final <CRLF>.<CRLF> mencionada anteriormente)
- Respuesta: 250 Mensaje recibido .....
- QUIT

Respuesta: 221 [Nombre del servidor] servidor cerrando conexión

A continuación figura un diagrama de flujo que muestra un ejemplo de este proceso en acción:



Notas sobre el Ejercicio 6: Envío de un SMTP

Ejemplo de mensaje de correo electrónico para enviar:

Los correos electrónicos, al igual que el HTML, pueden contener una gran variedad de características y opciones. Sin embargo, para nuestros propósitos (es decir, las comunicaciones TCP) un correo electrónico básico será suficiente.

From:me@myaddress.com
To:you@youraddress.com
Asunto: test
X-Mailer: Matrix SMTP

Hola desde Matrix.
Se trata de un "correo de prueba" enviado por el componente TCP/IP.

Observe cómo las direcciones "*De"* y "*Para"* se añaden aunque se enviaron al servidor como parte del proceso de conexión y acuse de recibo. Hay muchos más elementos de cabecera que se pueden añadir, pero lo anterior debería ser suficiente para los mensajes básicos. Para más detalles sobre otros elementos de la cabecera puede consultar los detalles del protocolo SMTP en la web.

Implementación del programa SMTP en Flowcode

Consideraciones sobre el hardware

Lo primero que hay que tener en cuenta es dónde se encuentra el servidor de correo electrónico en su sistema.
En nuestro sistema de prueba, el servidor de correo electrónico estaba en la dirección IP 212.69.254.157, pero tendrás que averiguar qué dirección IP utilizar para tu red. Pregunte al administrador de la red si no está seguro de cómo encontrarla.
Los mensajes SMTP suelen enviarse al puerto 25, por lo que deberá configurarlo en la macro *TCPConnect* (*dst_port_hi* = 0, *dst_port_lo* = 25).
A continuación, necesitas dos direcciones de correo electrónico válidas: una dirección de correo electrónico "De" y una dirección de correo electrónico "Para". La dirección "Para" debe ser una dirección válida a la que tengas acceso, de lo contrario no podrás recibir los correos para probar y depurar el programa.
La dirección de correo electrónico del remitente puede ser una dirección de prueba falsa o no válida, aunque elementos como los cortafuegos pueden realizar comprobaciones y rechazar este tipo de correos electrónicos. Si es posible, es más fácil configurar una dirección de correo electrónico de prueba.
El único problema en esta fase es conseguir la secuencia correcta. A estas alturas, ya tendrá los conocimientos y la experiencia necesarios para crear el código requerido para el envío y la recepción de mensajes.
Con SMTP ya no nos preocupamos sólo de enviar un mensaje; ahora mantenemos diálogos y negociar las comunicaciones.

<CRLF>, comillas y otros caracteres problemáticos

Mientras que la mayoría de los caracteres se pueden escribir en la macro *TxSendByte* hay algunos extra que necesitamos tener en cuenta. No podemos enviar la tecla Return/Enter para ir a la siguiente línea porque no entra en la lista de parámetros. Esto se debe a que la tecla Return es un carácter de control no imprimible. Sin embargo, todos los caracteres, incluidos los de control, tienen un valor ASCII asociado que podemos utilizar en su lugar. La tecla Retorno es un poco inusual, ya que en realidad se añade como dos códigos de caracteres en lugar de uno solo: el carácter de retorno de carro (código de carácter 13) y el carácter de avance de línea (código de carácter 13). A menudo se abrevia como CRLF. Todo lo que tenemos que hacer es enviar los códigos de caracteres ASCII en su lugar, en este caso los códigos ASCII 13 y 10 al final de cada línea de código de correo electrónico. (Todo esto es una reminiscencia de los primeros días de la informática, cuando la impresión se realizaba en impresoras electrónicas en las que se movía el cabezal de la impresora).
Los mensajes de respuesta SMTP le indican que finalice los datos del correo electrónico con la etiqueta de mensaje <CRLF>.<CRLF> es decir, un punto y aparte en una línea. Esto tendría que enviarse como caracteres 13, 10, '.', 13, 10.
Otro carácter problemático es la comilla. Flowcode utiliza las comillas para distinguir entre cadenas, por ejemplo "Hola mundo", y variables, por ejemplo CONTAR. Por lo tanto, no podemos enviar el carácter " porque confundiría a Flowcode. En su lugar, debemos enviar el código ASCII para ", que es el código de carácter ASCII 34.

Notas sobre el Ejercicio 6: Envío de un SMTP

Ejemplo de programa

Encontrará un ejemplo de solución en la carpeta "Flowcode\TCP_IP\Examples" de Flowcode, o en la carpeta
"Ejemplos" del CD, para que pueda consultarlos, codificarlos y utilizarlos con fines de demostración.

> Tenga en cuenta que las propiedades, los parámetros de las macros y los valores de las variables, etc., se configuraron para su uso con nuestra red y puede ser necesario cambiarlos para adaptarlos a su propia red o configuración del sistema.

Ejercicio avanzado 1: Personalizado mensajería mediante UDP

Se trata de un ejercicio avanzado, ya que requiere dos tarjetas de Internet totalmente configuradas y un concentrador o dos enchufes disponibles en la red.

Objetivo del programa:

- Crea dos Nodos que responderán a la comunicación enviando al comunicador un mensaje de re- puesta.
- Nodo A
  - Puede iniciarse al pulsar el botón "Enviar".
  - Responderá a una señal enviando una respuesta
  - 192.168.0.2
  - Puerto 5
  - Mensaje: "Hola mundo"
  - Nodo B
  - Puede bloquear el contacto cuando se pulsa el "botón de apagado".
  - Responderá a una señal enviando una respuesta
  - 192.168.0.3
  - Puerto 10

Mensaje "Hola globo"

Requisitos previos:

- Conocimiento de la capa MAC/Ethernet Conocimiento de la capa IP

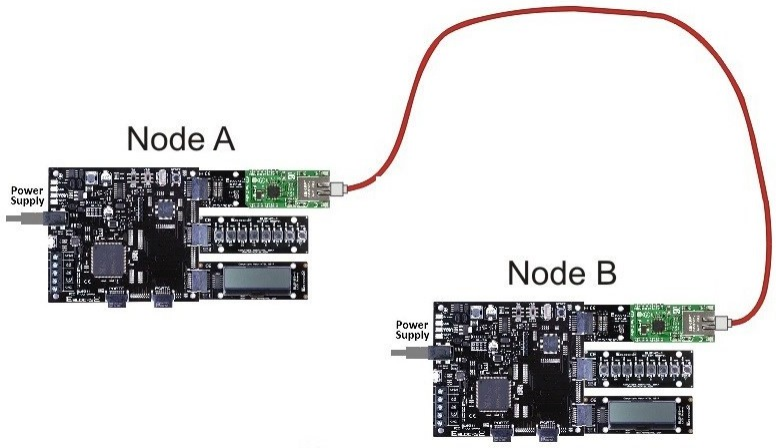
Objetivos de aprendizaje:

- Estructura de un datagrama UDP
- Envío de mensajes UDP
- Recepción de mensajes UDP
- Responder a un mensaje

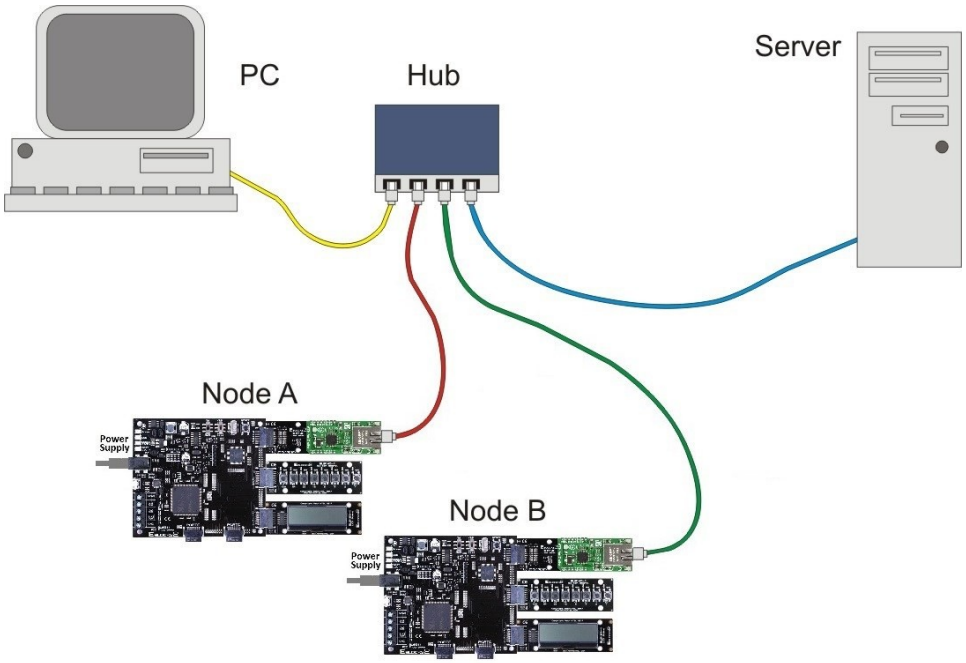
Notas sobre el Ejercicio Avanzado 1: Mensajería personalizada utilizando UDP

Configuración del hardware

Necesitarás dos placas de Internet, ya que cada programa es independiente. También tendrás que habilitar las dos tarjetas para que se comuniquen entre sí. Las dos tarjetas pueden conectarse directamente entre sí mediante un cable cruzado.



Si desea utilizar un software de supervisión del tráfico de red, necesitará un concentrador o dos puntos de conexión a la red.



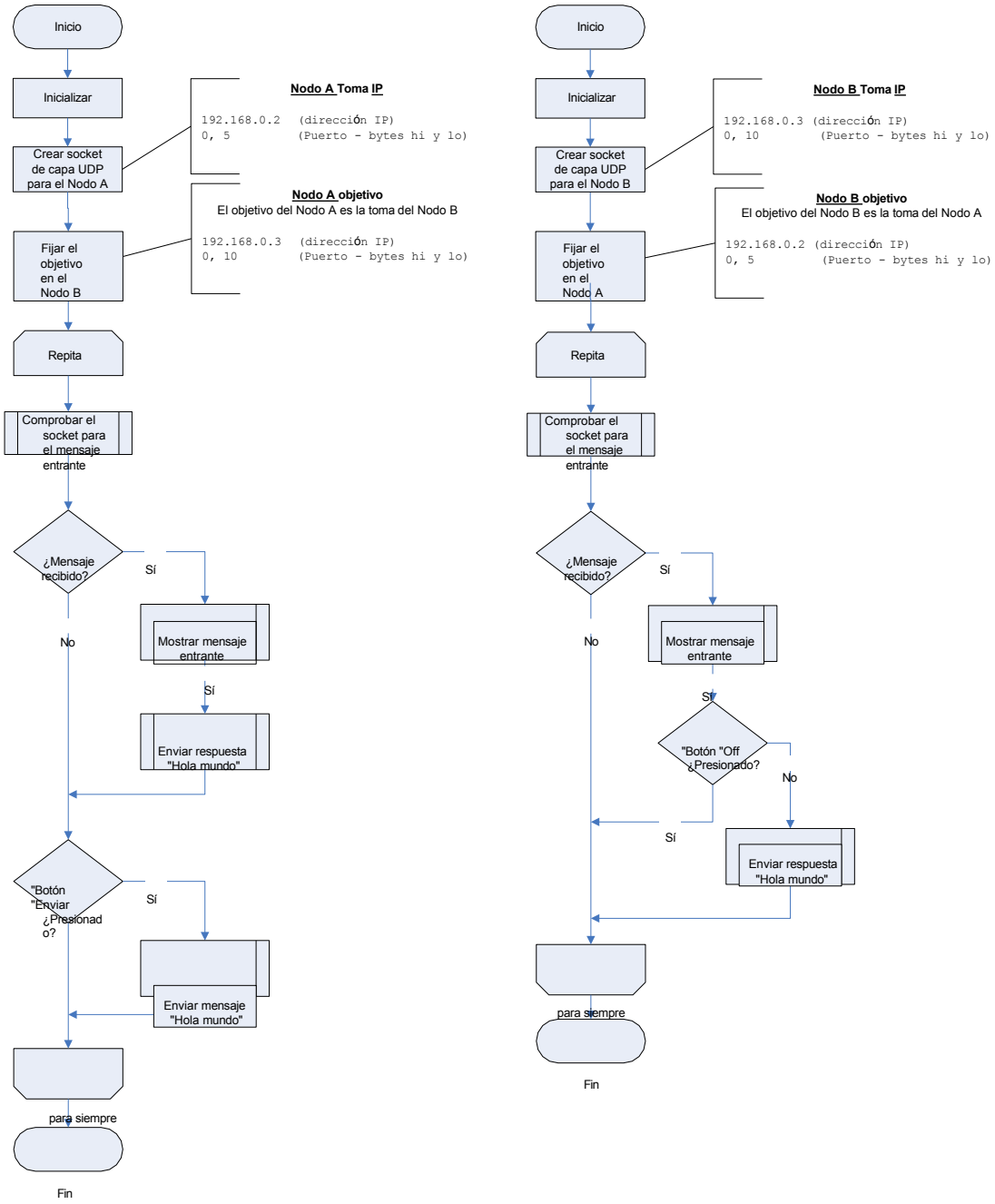
Notas sobre los programas

Necesitarás dos programas, uno para el Nodo A y otro para el Nodo B. La principal diferencia entre los dos programas es que una señal se puede utilizar para iniciar un envío en el Nodo A para poner en marcha el proceso, y una señal se utiliza con el Nodo B para evitar que una respuesta detenga el proceso.
Desde un punto de vista práctico, sería conveniente añadir un retardo antes de responder a un mensaje para evitar que una avalancha de mensajes atasque la red.

Notas sobre el Ejercicio Avanzado 1: Mensajería personalizada utilizando UDP

Resumen del programa

Los programas básicos son bastante sencillos y se describen en el diagrama de flujo que figura a continuación.



Otros trabajos

Los mensajes recibidos fueron aceptados y respondidos. ¿Qué tendríamos que hacer para introducir un elemento de comprobación de errores en el sistema de mensajería?

- ¿Cómo afrontaríamos un mensaje completamente vacío?
- ¿Cómo sabríamos que está vacía?

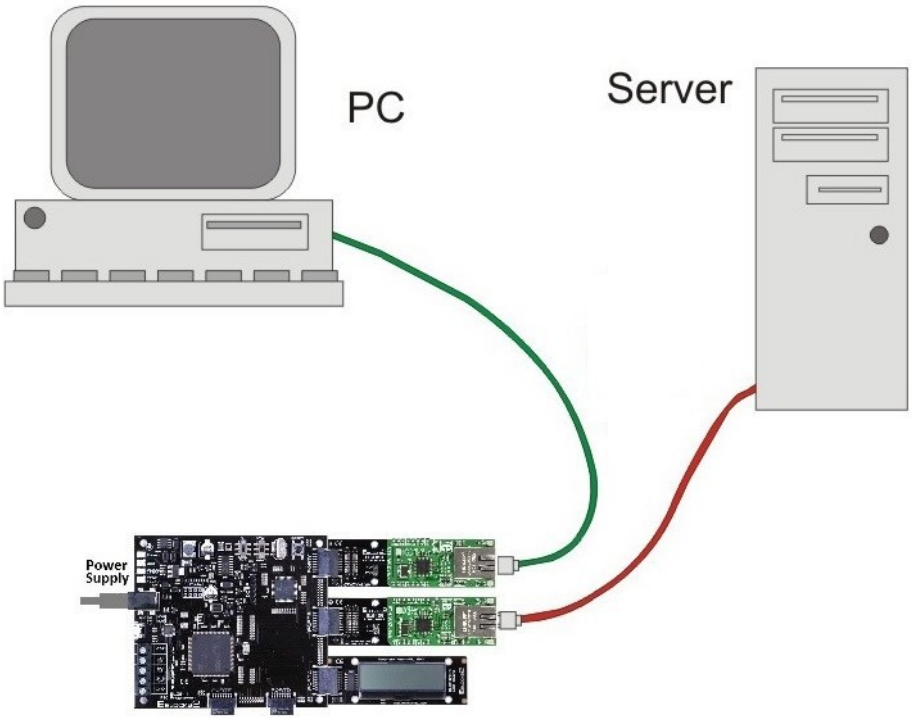
Ejercicio avanzado 2: Diseño de aplicaciones cortafuegos

Un cortafuegos se sitúa delante de una aplicación o aplicaciones y supervisa cualquier comunicación que se dirija a esa aplicación. El cortafuegos está diseñado para detectar y bloquear cualquier acceso no autorizado a la aplicación.
Debido a la propagación de virus y programas de hackers, los cortafuegos se han convertido en una parte importante del arsenal TCP/IP contra intrusiones no deseadas.

Nuestro proyecto consiste en diseñar e implementar una aplicación cortafuegos. Tendrás que crear un programa que actúe como intermediario entre dos placas Ethernet, decidiendo si se debe permitir o no el paso de un mensaje.

Consideraciones sobre el hardware

La solución más sencilla es conectar dos tarjetas de Internet a la misma tarjeta programadora.



Una placa conectada a la red y la otra conectada al PC protegido. Entonces puedes crear una aplicación que monitorice ambas y decida si pasa los datos a la otra placa o no.

Criterios de seguridad

¿Qué desea bloquear? ¿Qué necesita el usuario final? ¿Qué puede bloquear?

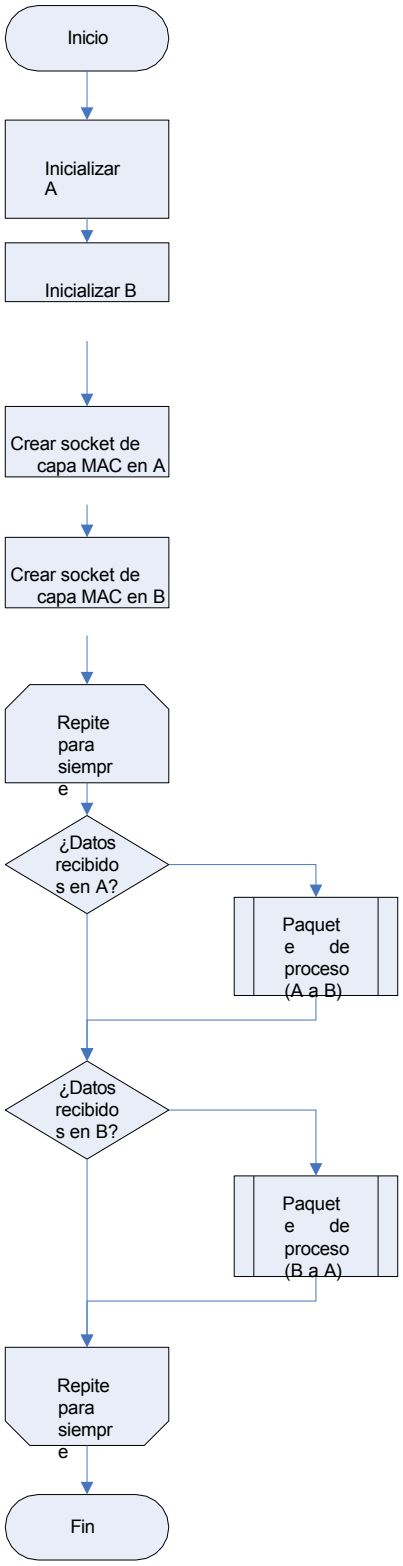
Dejaremos que usted decida qué bloquear: direcciones IP específicas, solicitudes ARP, correos electrónicos, etc.
Ten en cuenta que el microcontrolador dispone de una memoria limitada, por lo que puede que sea mejor simplificar al principio y ampliar la aplicación más adelante, una vez que funcione.

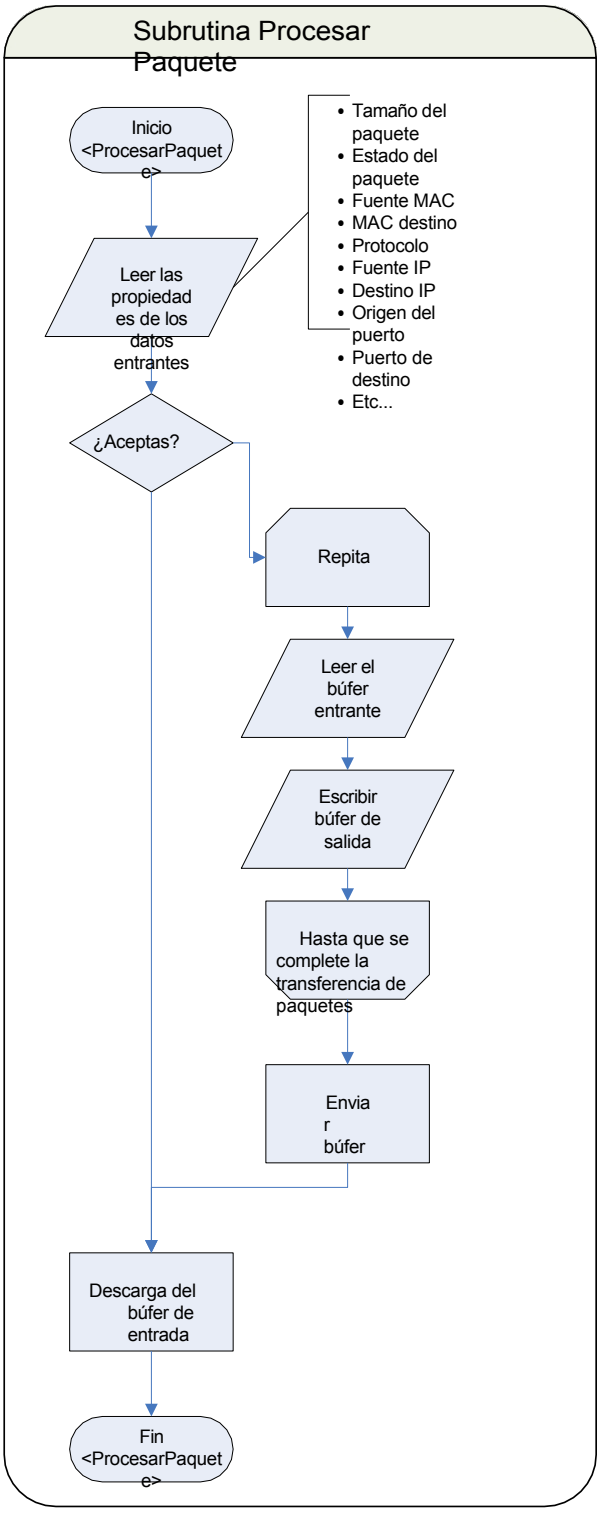
Diseño del programa

Esto también depende de usted. A estas alturas ya tendrás la experiencia con TCP/IP necesaria para poder enviar, recibir y examinar los distintos datagramas y tramas Ethernet utilizados en TCP/IP.

Para ayudarle a empezar, hemos incluido un diagrama de flujo de ejemplo para un sistema de cortafuegos básico que pasa los mensajes Ethernet entre dos sistemas.

Ejercicio avanzado 2: Diseño de aplicaciones cortafuegos





Hay mucho más que aprender sobre TCP/IP, tanto sobre el proceso de envío y recepción de mensajes, como sobre los mensajes de datos que se pueden enviar.

Fragmentación

Existen problemas de fragmentación, ya que los mensajes deben dividirse en partes más pequeñas, o fragmentos, y también deben volver a ensamblarse en el otro extremo. Esto es en gran medida irrelevante para el microcontrolador, ya que la capacidad de memoria relativamente pequeña será probablemente un factor más inmediato. Los problemas básicos de fragmentación también son gestionados automáticamente por la placa de Internet. Sin embargo, el uso de chips de memoria, o el uso del microcontrolador como un conducto de transferencia, puede crear archivos lo suficientemente grandes como para que la fragmentación sea un problema.

30.2 Cabecera opciones

Otras áreas en las que se puede seguir trabajando son las opciones de cabecera y el establecimiento de prioridades. Hay una serie de opciones y ajustes de cabecera que sólo hemos tratado brevemente en este curso. Los usuarios avanzados pueden querer echarles un vistazo para ver qué son y cómo pueden utilizarse para ayudar a la comunicación.

30.3 Error comprobación

Existen varios procedimientos TCP/IP para la comprobación y notificación de errores. Desde la simple comprobación de que el datagrama Ethernet contiene los valores de tipo de mensaje esperados, hasta los códigos de error de diálogo bidireccional de SMTP.

30.4 Mensaje data

En cuanto a los datos, hay mucho que aprender sobre el formato de los mensajes TCP, como HTML y el correo electrónico SMTP. Hay muchos datos disponibles sobre estos formatos de mensaje en la web.

Si desea examinar otros formatos de mensajes TCP, como POP3, FTP o TELNET, tendrá que buscar detalles sobre cómo estructurar los datos y qué diálogos de solicitud/respuesta son necesarios.

30.5 Otros protocolos

Hay una serie de protocolos que no hemos visto aquí, incluyendo SLIP, DHCP y NTP. Los detalles de estos protocolos se pueden encontrar en varios libros y en la web.

31 Enlaces y recursos

Documento SMTP
http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

Documento HTTP
http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Números de asignación de puertos
http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

32.1 Acrónimos

ARP - Protocolo de resolución de direcciones
DHCP - Protocolo de configuración dinámica de host
DNS - Servidores de nombres de dominio
FTP - Protocolo de transferencia de archivos
HTML - Lenguaje de marcado de hipertexto
HTTP - Protocolo de transferencia de hipertexto **ICMP** - Protocolo de mensajes de control de Internet **IP** - Protocolo de Internet
MAC - Control de acceso a los medios
POP3 - Protocolo de oficina de correos 3
SMTP - Protocolo simple de transferencia de correo **TCP** - Protocolo de control de transmisión **UDP** - Protocolo de datagramas de usuario
URL - Localizador Universal de Recursos

32.2 Glosario

Datagrama: paquete de datos (telegrama de datos) contenido dentro de otro mensaje. Los datagramas también se pueden anidar con un datagrama metido dentro de otro datagrama.

Cortafuegos: sistema de control de seguridad que ayuda a evitar que mensajes no autorizados lleguen al nodo. Normalmente se basa en software, pero también puede incluir cortafuegos basados en hardware.

Concentrador - Punto de conexión a la red. Los mensajes recibidos se envían a todas las demás conexiones del concentrador. Los concentradores no son inteligentes y se limitan a pasar los mensajes entrantes sin filtrarlos.

Nodo - Un sistema en la red con una dirección IP y una dirección MAC, ya sea un PC, un servidor Unix, o una solución de Internet E-blocks, etc.

Enrutador - Dispositivo que puede aceptar mensajes TCP/IP y pasarlos al siguiente dispositivo en la ruta entre los nodos emisor y receptor. Los routers son capaces de evaluar de forma inteligente los problemas y bloqueos de la red y redirigir los mensajes en función de las condiciones de la red.

Conmutador - Similar a un concentrador pero con filtrado inteligente que impide el envío de mensajes a conexiones para las que el mensaje no es apropiado. No se recomienda su uso con analizadores de tráfico de red, ya que impiden que los mensajes de tráfico de red general lleguen al nodo analizador.