Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto de investigación

Análisis de una aplicación con interfaz web, implementada en dos sistemas basados en microcontroladores: un Arduino y uno hecho a la medida

Alumno: Manuel Rodríguez Camacho
Matrícula: 209333915

Ing. Ricardo Godínez Bravo
Profesor Asistente
Departamento de Electrónica

Dr. Francisco Javier Zaragoza Martínez
Profesor Titular
Departamento de Sistemas

Trimestre 2014 Otoño

16 de noviembre de 2014

Yo, Ricardo Godínez Bravo, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Ricardo Godínez Bravo

Yo, Francisco Javier Zaragoza Martínez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Francisco Javier Zaragoza Martínez

Yo, Manuel Rodríguez Camacho, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Manuel Rodríguez Camacho

Resumen

En éste proyecto de investigación se ofrece el análisis de una aplicación de control y monitoreo que posee una interfaz Web implementada con microcontroladores. El desarrollo de dicha interfaz en un sistema embebido, es una aplicación lo suficientemente compleja para generar datos e información que pueda ser evaluada. Tal implementación requiere de programación tanto del lado del cliente, como del lado del servidor, necesita el desarrollo de un prototipo electrónico, es necesario el manejo de dispositivos disponibles en el mercado, así como la aplicación de protocolos de red y del uso de otras técnicas que se mencionarán a lo largo del desarrollo del proyecto.

Se aborda el aspecto económico comparando un dispositivo comercial y otro diseñado y construido para el uso específico de la aplicación.

Los recursos de tiempo y complejidad de implementación para cada dispositivo, serán medidos y reportados.

Una vez teniendo los datos del análisis, se podrán tomar decisiones de acuerdo a la necesidad del desarrollo, contemplando los recursos con los que se cuenta.

Tabla de contenidos

Introducción

Los sistemas embebidos son hoy en día muy comunes, se encuentran en muchas aplicaciones cotidianas. Para su desarrollo se cuenta con herramientas tecnológicas tales como: FPGAs[1] (Field Programmable Gates Array) y microcontroladores, siendo éstos últimos la opción más económica.

En el caso de los microcontroladores, existen productos con una gran variedad de prestaciones para el desarrollador: módulos preconstruidos, interfaces de desarrollo, tarjetas de entrenamiento, entre otros; los cuales están listos para ser implementados. Un buen ejemplo es la plataforma de electrónica libre Arduino.

Arduino permite implementar aplicaciones de manera rápida y sencilla, sin embargo, también es posible desarrollar aplicaciones hechas a la medida de la necesidad, para lo cual, es necesario diseñar y construir la placa con sus respectivos componentes, además de poseer ciertos conocimientos en electrónica.

En la actualidad hay una importante tendencia hacia las aplicaciones web, las cuáles ofrecen una mayor comodidad para los usuarios, a diferencia de otros medios tales como el cable serial, por ejemplo.

Por lo tanto, se puede decir que el uso de microcontroladores es la solución más económica y accesible para desarrollar sistemas embebidos, y que el uso de interfaces web es la forma más cómoda de operarlos.

El problema que se presenta es conocer el desempeño de un sistema basado en Arduino y de otro hecho a la medida, bajo una aplicación con una interfaz web; además de los costos que implica su implementación en cada sistema.

Justificación

Conocer el desempeño tanto de un dispositivo hecho a la medida como de uno basado en una plataforma ya desarrollada, permite decidir si conviene realizar un gasto para comprar un dispositivo preconstruido (como Arduino), o si es mejor, invertir tiempo y recursos para construir un dispositivo que satisfaga únicamente los requerimientos de la aplicación. En la presente propuesta se pretende investigar cierta información acerca del desempeño de dichos dispositivos bajo una aplicación con interfaz web.

Antecedentes

Proyectos terminales

1. "Implementación de una VPN[2] con el microcontrolador Rabbit[3] para acceso a una red de cámaras web de la Universidad Autónoma Metropolitana". [1]

   Éste proyecto explica la implementación de un sistema basado en el microcontrolador Rabbit, la cuál permite monitorizar una red de cámaras web. De manera similar a la presente propuesta, se desarrolló un dispositivo, el cuál también fue programado. Sin embargo, no se mide su desempeño ni se busca un método óptimo para su desarrollo.

2. "Control distribuido de los dispositivos eléctricos de un inmueble mediante TCP/IP[4]". [2]

   Dicho proyecto se asemeja a ésta propuesta, al igual que el proyecto anterior, en el uso de un sistema

1 Dispositivos que permiten el diseño de hardware con capacidad de reconfiguración.
2 Red privada virtual (*Virtual Private Network)*, ampliamente utilizada para obtener una extensión de red segura dentro de otra red no controlada.
3 Microcontrolador de la familia Rabbit 6000 de la empresa Digi International, que trabaja con comunicación de red.
4 Es la pila de protocolos de red en la que se basa actualmente la Internet.

empotrado para controlar diversos dispositivos a través de una interfaz web. La diferencia está en la implementación de un microprocesador como núcleo del proceso, además de que, como en el caso anterior, no se miden factores de eficiencia.

3. "Implementación de un Router[5] en un sistema empotrado". [3]

   Sin duda, éste proyecto es el que más se asemeja a la ésta propuesta, ya que durante sus desarrollo se realiza un estudio de factibilidad a varias tarjetas de desarrollo, con lo que se concluyó que la mejor opción era la tarjeta NGW100[6] de Atmel, la cuál utiliza un microcontrolador AVR32[7]. Es distinto porque la finalidad del proyecto no es precisamente encontrar la mejor opción para la implementación de las plataformas basadas en microcontroladores.

Artículos de investigación:

1. "Controlling Physical Objects via the Internet using Arduino Platform over 802.15.4 Networks". [4]

   En ésta publicación se explican los resultados de un estudio de funcionalidad hecho a una plataforma Arduino con un módulo de comunicación inalámbrica. Aporta información importante acerca del desempeño de una placa Arduino con un microcontrolador ATMega328[8], con un módulo de comunicación inalámbrica (“WiFi Shield[9]”) integrada a la aplicación. Sin embargo, en ésta propuesta, se trabaja con comunicación alámbrica, y además, con un dispositivo hecho a la medida.

Tesis:

1. "Desarrollo de un sistema SCADA[10] para casa habitación". [5]
2. En 2009 un grupo de alumnos del Instituto Politécnico Nacional, presentaron ésta tesis en la que se describe el proceso para la implementación de un sistema de control y adquisición de datos a distancia. En tal tesis no se realizó ningún estudio de eficiencia a los dispositivos utilizados.
3. "Móvil para telemetría controlado vía remota". [6]
4. En 2008, otro grupo de alumnos del Instituto Politécnico Nacional, presentaron dicha tesis donde se muestra cómo controlar un móvil vía remota para telemetría. En ésta tesis tampoco se realizaron evaluaciones a los componentes físicos del sistema.

Software:

- No existe como tal, una herramienta comercial que permita comparar la actividad de dos microcontroladores con una interfaz web implementada. Sin embargo, se pueden mencionar algunos desarrollos que implementan interfaces web en microcontroladores, tales como:
  - La biblioteca *Appweb* de licencia abierta.
  - Los microcontroladores *Rabbit*.

5 Dispositivo de red que permite la comunicación entre diversas redes, dividiendo el dominio de difusión.
6 Tarjeta de desarrollo de la empresa Atmel.
7 Familia de microcontroladores desarrollados por la empresa Atmel que trabajan con un conjunto de instrucciones de 32 bits.
8 Microcontrolador de la familia AVR32, utilizado en éste proyecto.
9 Componente de conectividad inalámbrica bajo la arquitectura de electrónica libre Arduino.
10 Software que permite controlar y supervisar procesos industriales a distancia.

Objetivos

Objetivo general

Realizar un análisis a una aplicación con interfaz web implementada en dos dispositivos con microcontroladores, el primero basado en la plataforma de electrónica libre Arduino, y el segundo hecho a la medida.

Objetivos específicos

- Construir un dispositivo con comunicación de red, basado en un microcontrolador,.
- Implementar la comunicación de red, en el dispositivo construido y en una tarjeta de desarrollo Arduino, con el uso de una biblioteca disponible bajo la licencia GPL[11] Versión 1 .
- Desarrollar una aplicación que controle y monitorice el estado de la aplicación para cada uno de los dos dispositivos a través de un navegador web, aprovechando la comunicación de red.
- Con la aplicación funcionando en los dos dispositivos, evaluar los siguientes aspectos técnicos:
  - El tiempo promedio de respuesta a las instrucciones de control y monitoreo.
  - La cantidad de errores de respuesta a las instrucciones de control y monitoreo.
  - La calidad de la conexión.
  - El número de usuarios simultáneos que se pueden atender.
  - Compatibilidad con: Teléfonos celulares y computadoras personales, con sistemas operativos como Linux y Windows, y con navegadores web como Internet Explorer, Firefox y Crhomium.
- Evaluar los costos de implementación para ambos dispositivos:
  - Tiempo de desarrollo.
  - Costo económico.
- Reportar el resultado del análisis.

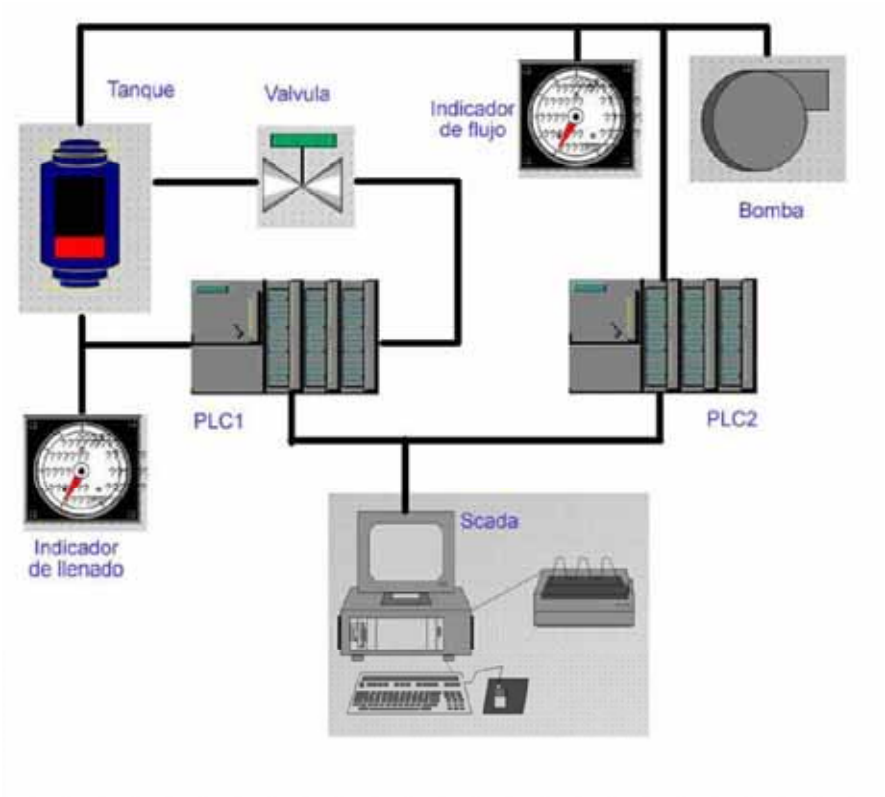
11 Licencia pública general que protege al software del apoderamiento de instituciones o entidades comerciales que priven de las libertades de uso y distribución a sus usuarios.

Marco teórico

La aplicación para controlar ciertos procesos a través de una conexión por red no es algo nuevo, se le conoce desde hace tiempo con el nombre de SCADA (*Supervisory, Control And Data Acquisition*), y se implementa en un sin fin de dispositivos, tales como cámaras IP y Enrutadores de red (*Routers*) por mencionar sólo algunos.

Las aplicaciones que son diseñadas específicamente para el hogar, entran dentro de lo que se le llama Domótica[12], la cuál cada vez tiene mayor presencia en electrodomésticos, sistemas de seguridad, iluminación, videovigilancia y comunicación.

El sistema analizado en éste trabajo, puede ser utilizado tanto en ambientes industriales, así como en el hogar.



1 Figura: Ilustración del modelo SCADA.

Modelos de red

Aunque los protocolos del modelo OSI ya no son tan utilizados, es importante tenerlo en cuenta como una referencia, ya que es un modelo bastante general, lo que le ha permitido estar aún vigente. Por otro lado, el modelo TCP/IP no es muy usado, sin embargo, sus protocolos son la base de la gran mayoría de aplicaciones de Internet[7].

12 Nombre que recibe el conjunto de sistemas capaces de automatizar una vivienda.

Modelo OSI

Su nombre ISO (Interconexión de sistemas abiertos en español) se debe a que tiene que ver con al conexión de sistemas que están abiertos a la comunicación con otros sistemas. Cuenta con siete capas, las cuales fueron definidas bajo los siguientes criterios:

- Una capa se debe crear donde se necesite una abstracción distinta.
- Cada capa debe realizar una función bien definida.
- La función de cada capa se debe elegir con la intención de definir protocolos estandarizados internacionalmente.
- Los límites de las capas se deben elegir a fin de minimizar el flujo de información a través de interfaces.
- La cantidad de capas debe ser lo suficientemente grande para no tener que agrupar funciones distintas en la misma capa y lo bastante pequeña para que la arquitectura no se vuelva inmanejable.

Las capas implementadas en el modelo son los siguientes:

- La capa física

  Es la capa más baja en el nivel de abstracción del modelo, aquí se gestiona la transmisión de los bits de una máquina a otra con la ayuda de interfaces mecánicas , eléctricas y de temporización, además del medio físico utilizado para la transferencia de las señales. Quizás ésta capa es donde la electrónica juega su papel más importante.

- La capa de enlace de datos

  En ésta capa los datos son fragmentados en tramas, las cuales se componen de varios, e incluso centenas de bytes. Es importante que el receptor reciba los datos de manera confiable, para ello se implementan mecanismos de confirmación y revisión de datos, que son precisamente en ésta capa donde se manejan.

  La subcapa de control de acceso al medio tiene como herramientas protocolos y algoritmos que minimizan los errores en la transmisión, podemos mencionar al CSMA/CD[13] que se asegura de que el medio esté disponible al momento de la transmisión y detecta colisiones de tramas.

  El direccionamiento físico a través de direcciones MAC[14], es una tarea que se ejecuta en la capa de enlace.

- La capa de red

  En ésta capa se controlan las operaciones de subred, tales como el direccionamiento lógico o enrutamiento de las tramas a través de la red. El tráfico de red es comparable al tráfico de autos en una ciudad, los datos se encuentran con caminos congestionados o dañado, la solución a tales problemas se logra en ésta capa.

  La misión de la capa de red es hacer que redes heterogéneas se intercomuniquen, esto se logra resolviendo incompatibilidades en los protocolos manejados por las distintas redes.

13 CSMA se refiere al censado que realiza la tarjeta de red al medio físico para verificar que está disponible para poder transmitir; CD es la detección de coliciones.

14 El control de acceso al medio, o MAC, es el conjunto de mecanismos y protocolos utilizados para que diversos dispositivos puedan compartir un medio de transmisión.

- La capa de transporte

  En ésta capa es donde la información se divide en unidades más pequeñas para ser enviadas a la red, además es la encargada de asegurar que los datos lleguen bien al otro extremo de la conexión. Los cambios en la electrónica y las dificultades físicas deben ser transparentes para las capas superiores.

  La capa de transporte también determina los servicios proporcionados a la capa de sesión y por lo tanto, a los usuarios de la red. La conexión en esta capa se realiza de extremo a extremo, usando los que se conoce como puertos para seleccionar el tipo de servicio que será usado para el procesamiento de los datos.

- La capa de sesión

  En la capa de sesión los usuarios pueden establecer sesiones entre ellos, ofreciendo servicios tales como el control de diálogo,administración de token y sincronización; lo que permite dar seguimiento de quién debe transmitir, impedir que dos usuarios realicen una operación crítica simultáneamente, además de agregar puntos de referencia a transmisiones grandes que permiten continuar desde donde se encontraban después de una caída.

- La capa de presentación

  Mientras que en las capas inferiores se trabaja con bits, en ésta se revisa la semántica y la sintaxis de la información transmitida, logrando que las máquinas que tienen representaciones de datos distintas puedan comunicarse.

- La capa de aplicación

  Es la capa donde los usuarios tienen al alcance protocolos que permiten la transferencia de información, por ejemplo, enviar y recibir archivos, correos electrónicos, mensajes a través de clientes de mensajería instantánea, o contenido Web utilizando algún navegador.

  Quizás el protocolo más utilizado es el HTTP, que permite servir y consultar páginas en la red, y que es la base de la *World Wide Web*[15].

El modelo TCP/IP

La antecesora de Internet, la ARPANET[16], implementaba ya este modelo. El departamento de defensa de los Estados Unidos estaba a cargo del desarrollo de ARPANET, la cuál fue una red que conectaba a computadoras de varias universidades e instalaciones gubernamentales. Debido al temor de que algún equipo fallara y afectara a la comunicación en la red, y a la poca eficacia de los protocolos de ese tiempo frente a la aparición de medios de transmisión como la comunicación satelital, se llegó a conocer al modelo de referencia TCP/IP.

El modelo TCP/IP, al igual que OSI, implementa varias capas, en donde se definen las funciones de red que se necesitan para establecer la comunicación:

- La capa de interred

15 La red informática mundial comúnmente conocida como la Web, es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles a través de Internet.

16 La red de computadoras *Advanced Research Projects Agency Network* (ARPANET por sus siglas en inglés) fue creada por encargo del Departamento de Defensa de Estados Unidos como un medio de comunicación para los diferentes organismos del país. El primer nodo se creó en la Universidad de California, Los Ángeles, y fue la espina dorsal de Internet hasta 1990, tras finalizar la transición al protocolo TCP/IP iniciada en 1983.

Ésta capa es la pieza clave que mantiene unida a la arquitectura, permite que las máquinas inyecten paquetes en cualquier red y que éstos viajen de manera independiente hasta llegar a su destino. Para lograrlo, se deben vencer obstáculos tales como las características físicas de las diversas redes así como los protocolos que usan.

EL protocolo de Internet (IP), está definido aquí, el cuál consiste en un formato para los datos que deben ser entregados a un destinatario a través de un proceso de enrutamiento, para evitar congestiones en el medio.

- La capa de transporte

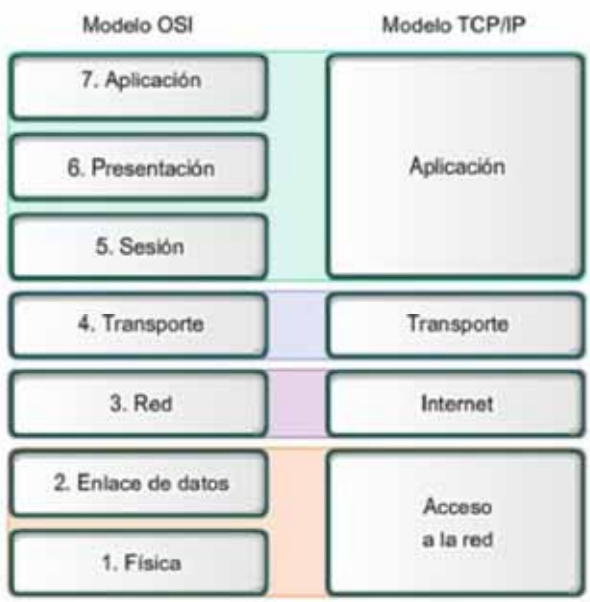
  Tan similar a la capa de transporte del modelo OSI, se encarga de comunicar a dos máquinas de extremo a extremo. El protocolo TCP, es un protocolo confiable que reporta el estado de los datos en el destino. Divide el flujo de bytes que entran en mensajes discretos y los envía a la capa de interred. TCP también cuida que un emisor rápido no sature a un emisor más lento con un exceso de mensajes. El otro protocolo de esta capa es UDP, un protocolo no orientado a conexión, lo que lo hace más vulnerable a errores de transmisión pero con una gran velocidad.

- La capa de aplicación

  Aquí se tiene a los protocolos de más alto nivel: TELNET, FTP, SMTP, como ejemplos. Con el tiempo se agregaron otros protocolos como DNS, NNTP y HTTP.

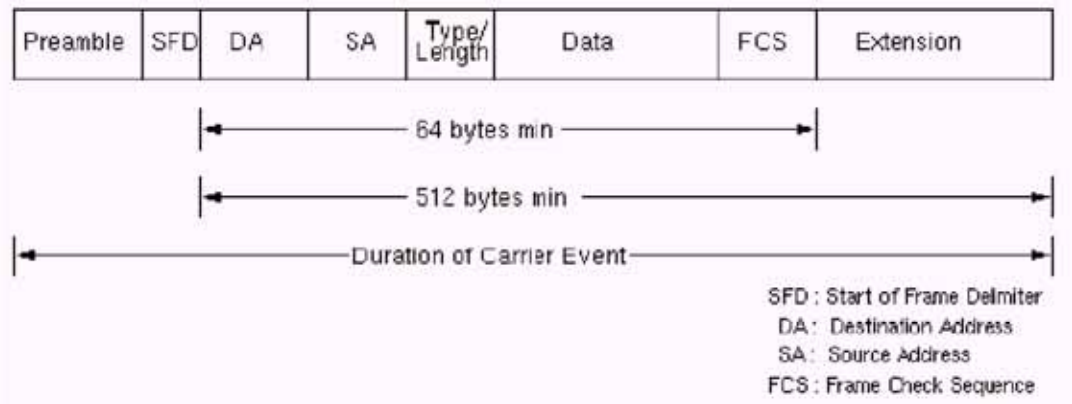
- La capa de *Host* a red

- Ésta capa no habla mucho de cómo se realiza la comunicación de los datos aquí, sólo menciona que la máquina debe conectarse a la red mediante el mismo protocolo para que le puedan enviar paquetes IP.



2 Figura: Comparación del modelo OSI y del modelo TCP/IP.

Ethernet



3 Figura: Elementos de una trama IEEE 802.3.

Se conoce con éste nombre al conjunto de tecnologías, protocolos y estándares que permiten la comunicación en una red de área local. La historia de Ethernet es ampliamente conocida por los especialistas en redes y muchos profesionales de la computación y las telecomunicaciones. Después de sus inicios evolucionó al llamado estándar DIX, quién a su vez con dos cambios menores llegó a convertirse en el estándar IEEE[17] 802.3.

Cableado Ethernet

Por lo general se usan 4 tipos de cableado:

- 10Base5 Coaxial grueso.
- 10Base2 Coaxial delgado.
- 10Base-T Par trenzado.
- 10Base-F Fibra óptica.

El primero resulta ser un medio obsoleto, el segundo no es muy usado en la actualidad y la fibra óptica es utilizada para comunicar equipos que trasmiten grandes cantidades de información a largas distancias. El medio más asequible en una red local es actualmente el 10Base-T (para trenzado).

Con 10Base-T ya no hay cables como en un principio con los coaxiales, sólo se necesita un concentrador del cuál se conecta un cable dedicado para cada máquina. El cambio de equipos resulta mucho más fácil con esta configuración y la detección de errores en el medio es también mas sencilla.

Una limitante importante es el límite de la longitud en el segmento, esto debido al estándar que asegura una correcta detección de colisiones en el medio con un límite de 100 metros y de 200 en cables de calidad alta. Para resolver éste problema, se puede hacer uso de repetidores, que son dispositivos, con limitaciones también, tales como: el número de repetidores entre las estaciones y la distancia entre ellas.

Codificación Manchester

Las señales binarias no pueden transmitirse con los valores convencionales TTL[18], ya que conducen a

17 El Instituto de Ingeniería Eléctrica y Electrónica es una asociación mundial de técnicos e ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas.

18 TTL que significa lógica de transistor a transistor en español, es una una tecnología de construcción de circuitos electrónicos

ambigüedades. Un bit 0, puede ser interpretado de diferentes maneras. La codificación Manchester permite identificar a los bits sin la necesidad de tener un reloj externo. Dicha codificación consiste en dividir a los bits en dos intervalos iguales.

4 Figura: Ejemplo de codificación de la cadena 01111001, la cual se transmite como 0110101010010110.

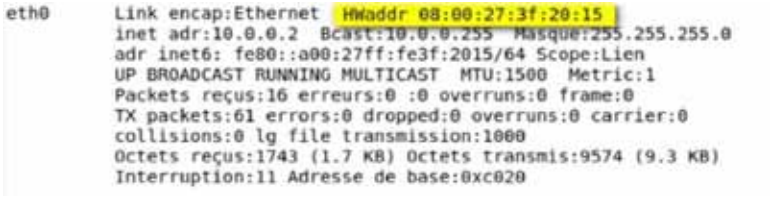
Un bit se transmite teniendo el voltaje alto durante el primer intervalo y un voltaje bajo en el segundo; un cero es precisamente al contrario. Este proceso facilita a las máquinas identificar a los bits, sin embargo requiere de un doble de ancho de banda en la comunicación.

Ethernet utiliza ésta codificación en sus sistemas.

Protocolo de subcapa MAC de Ethernet

La trama de Ethernet está formada de varios campos, inicia con un preámbulo de 7 bytes que permite la sincronización entre las máquinas, después un campo de un byte que indican el inicio del marco (*Start of frame*), una dirección para el destino y otra para el origen de dos o seis bytes, un campo con dos bytes para definir la longitud, un campo de datos de hasta 1500 bytes que incorpora el LLC[19] de 8 bytes, un campo de relleno de hasta 46 bytes, y por último un campo de 4 bytes para la suma de verificación (*Checksum*[20]).

El envío de tramas se puede realizar a un grupo de máquinas (multidifusión o *Multicast*), a todas las máquinas de la red (difusión o *Broadcast*) o a una sola máquina (*Unicast*).



5 Figura: La dirección física identifica de manera única a la tarjeta de red, se dice que es una dirección de capa 2 (debido al que la capa de enlace en el modelo OSI, es la segunda en orden ascendente).

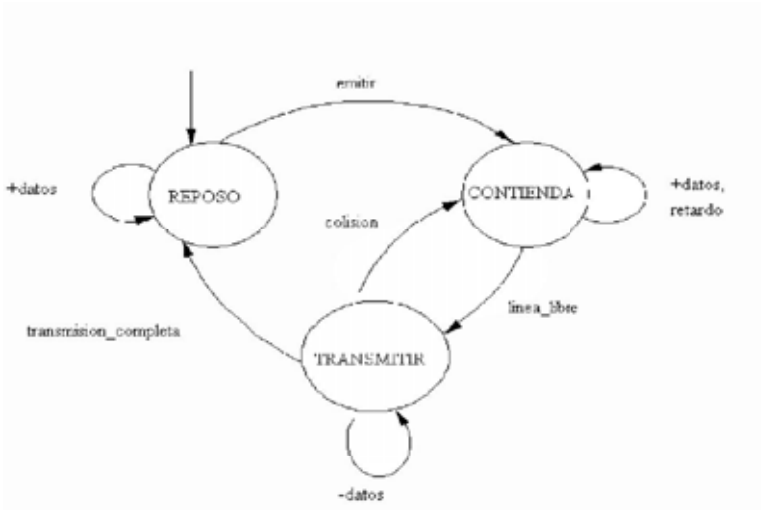
Colisiones

Las colisiones se dan en un segmento de red que está compartido por dos o más máquinas. Cuando dos dispositivos de red transmiten y ocupan el medio al mismo tiempo, las señales se encuentran provocando un choque y una alteración en las señales originales, lo cuál se traduce en pérdida de datos y de hecho, de todas las tramas enviadas. El problema de las colisiones ha sido prácticamente resuelto con la división de los dominios de colisión que se realiza con la ayuda de los dispositivos conmutadores (*Switch*).

digitales. En los componentes fabricados con tecnología TTL los elementos de entrada y salida del dispositivo son transistores bipolares. Su tensión de alimentación se halla comprendida entre los 4,75V y los 5,25V. Normalmente TTL trabaja con 5V.

19 Control de enlace lógico LLC (*Logical Link Control*) define la forma en que los datos son transferidos sobre el medio físico, proporcionando servicio a las capas superiores.

20 Una suma de verificación (*checksum*), es una función que tiene como propósito detectar cambios accidentales en una secuencia de datos para proteger su integridad, verificando que no haya discrepancias entre los valores obtenidos al hacer una comprobación inicial y otra final tras la transmisión. La idea es que se transmita el dato junto con su valor de suma, de esta forma el receptor puede calcular dicho valor y compararlo así con el valor de suma recibido. Si hay una discrepancia se pueden rechazar los datos o pedir una retransmisión.



6 Figura: Máquina de estados que representa la ejecución de CSMA/CD.

Otros métodos surgidos junto con Ethernet son algoritmos implementados en electrónica para tratar las colisiones. Por ejemplo el algoritmo de retroceso exponencial binario, el cual consiste en lo siguiente: tras una colisión, el tiempo se divide en ranuras discretas que tienen una longitud igual al tiempo de propagación de ida y vuelta en el peor de los casos en el cable. De acuerdo a la ruta más larga permitida en Ethernet el tiempo de cada ranura se establece en 512 tiempos de bit, es decir 51.2 microsegundos. Después de la primera colisión las estaciones esperan un tiempo aleatorio, si tras 16 colisiones la transmisión no tiene éxito, el controlador informa a la máquina que no se pudo enviar, y luego entonces, la capa de transmisión se encarga de recupera la transmisión. CSMA/CD no proporciona información que confirme la recepción de la trama, es necesario que la máquina destino confirme la recepción verificando los datos con una suma de verificación.

Ethernet conmutada

Cuantas más estaciones existan en una red, el tráfico aumenta, y con la evolución en la información, no sólo se transmiten archivos de texto plano, sino que también viaja contenido multimedia que aumenta las exigencias de conexión. La solución a la saturación de los medios físicos llegó con los conmutadores, los cuáles son dispositivos que trabajan en la capa 2 del modelo OSI. El conmutador separa el dominio de colisión realizando conexiones entre las máquinas que se conectan a sus puertos, reduciendo así, el número de colisiones y aumentando la eficiencia de la red. Venciendo a las colisiones, el nuevo enemigo ahora resulta ser el dominio de difusión, las “tormentas de *Broadcast*”, entre otras nuevas dificultades.

El control lógico del enlace

A diferencia de las redes CSMA/CD y de la comunicación a través de TCP/IP, donde si los datos se perdían, simplemente se reenviaban; existen sistemas que exigen un control de errores y flujo, implementado con protocolos de enlace de datos. La IEEE definió un protocolo que trabaja con Ethernet y el conjunto de estándares 802, lo llamaron LLC (*Link Logic Control*). El cuál proporciona un formato único y una interfaz con la capa de red, lo cual esconde las diferencias entre los distintos tipos de redes 802. LLC es la parte superior de la capa de enlace de datos, teniendo a MAC como la subcapa inferior a ésta. LLC y MAC son las subcapas que conforman a la capa de enlace.

LLC agrega un encabezado a la trama, ofreciendo tres opciones de servicio: servicio no confiable de datagramas, servicio de datagramas sin confirmación de recepción y servicio confiable orientado a

conexión.

La capa de red de Internet y el protocolo IP

La Internet se encuentra unida gracias al protocolo de capa de red IP (Protocolo de Internet), el cuál fue creado desde el inicio con la finalidad de interconectar diversas redes[8].

Datagramas IP

Los datagramas IP consisten en un encabezado y una parte de texto. El encabezado cuenta con veinte bytes fijos y una parte opcional de tamaño variable. Es transmitido en orden *Big Endian*[21], obligando a las máquinas *Little Endian*[22] a convertir por software los datagramas que son transmitidos, así como los que son recibidos.

Debido a la creciente cantidad de dispositivos que se conectan a Internet, las direcciones IP fueron una preocupación, por lo que se comenzó a trabajar en una transición desde la versión IP V4 a la IP V6, sin embargo, parece ser que es una transición que no termina. El problema se ha aliviado con lo que se conoce como *Subnetting*[23] (Creación de subredes), sin embargo, la transición continúa. La versión se incluye en el encabezado.

7 Figura: Ilustración de la estructura de un datagrama IP.

Ya que el tamaño del encabezado no es fijo, se introduce un campo ILH de 32 bits, para indicar su longitud. El valor mínimo es de 5, lo cual se aplica cuando no hay opciones. El máximo es 15 (4 bits), lo que limita el tamaño del encabezado a 60 bytes, dejando 40 bytes para el campo de opciones.

El campo de tipo de servicio tiene la finalidad de distinguir entre las distintas clases de servicios, donde se tienen diversas combinaciones entre velocidad y confiabilidad.

La longitud total del datagrama incluye al encabezado y sus datos, su límite es de $2^{16}-1$ bytes, es decir

21 Es un formato en el que se designa la forma en la que se almacenan los datos en la memoria de una computadora, en éste caso, *Big Endian* almacena el byte más significativo a la derecha.

22 Al contrario que *Big Endian*, el byte más significativo es trasladado hacia la izquierda, para el caso de éste formato.

23 El *subneting* es dividir las grandes redes en redes mas pequeñas. Fue una solución al problema del agotamiento de direcciones de Internet disponibles.

65535.

El campo de identificación permite a la máquina destino identificar a los datagramas y determinar si pertenece a algún fragmento recién llegado (todos los fragmentos de un datagrama poseen el mismo valor de identificación).

Después, tenemos un bit no utilizados, y luego dos campos de un bit que representan a las banderas DF (*Don't fragment*) y MF (*More fragments*).

El desplazamiento del fragmento indica dónde debe ir el fragmento dentro del datagrama actual. El campo de tiempo de vida cuenta con un contador que limita el tiempo que tiene un datagrama para llegar al destino, su valor máximo está fijado a 225 segundos.

Teniendo el datagrama completamente ensamblado, la acción que se debe tomar, se indica en el campo de protocolo. La enumeración de los protocolos está definida en el RFC[24] 1700. Aunque en la actualidad existe una base de datos que contiene dicha información en el sitio www.iana.org.

La suma de verificación del encabezado, sirve para comprobar la integridad únicamente del encabezado, el algoritmo consiste en sumar todas las medias palabras de 16 bits conforme van llegando usando complemento a uno, y luego obtener el complemento del resultado; si el resultado es cero, el encabezado llegó bien. Ésta suma debe hacerse en cada salto, una tarea compleja para los enrutadores.

Las direcciones de origen y destino, indican el número de red y el número de máquina de origen y destino.

Direcciones IP

Una dirección IP identifica, de manera lógica y jerárquica, a la interfaz de un dispositivo dentro de una red que utiliza el protocolo IP. Las direcciones lógicas trabajan en el nivel de red del Modelo OSI. La dirección IP puede cambiar debido a cambios en la red o por la implementación de servicios tales como los de un servidor DHCP[25]. A esta forma de asignación de dirección IP se denomina también dirección IP dinámica.

Los servidores, por su naturaleza requieren de una dirección IP que no cambie, es decir una IP estática o fija, ya que de lo contrario sería imposible localizarlos en la red, y un servidor siempre debe estar disponible para los clientes.

Las máquinas trabajan bien con éste tipo de direcciones, sin embargo, para la comodidad de las personas se han diseñado aplicaciones de servidor como DNS[26] que traduce esas direcciones en palabras que los humanos pueden manejar con mayor facilidad.

Las direcciones IP en su versión 4, miden 32 bits de longitud, logrando así, tener un rango de 4294967296 direcciones distintas, por lo general representadas por cuatro octetos separados por puntos.

Las direcciones se clasifican en clases de acuerdo a los bits que se asignan para la identificación de la red, las cuáles se denominan como: clase A para redes de 8 bits, clase B para redes de 16 bits, y clase C para redes de 24 bits. Existen más clasificaciones, que tienen como finalidad flexibilizar la administración de las redes, por ejemplo el ya mencionado *Subnetting*.

24 *Request for Comments* son una serie de publicaciones del *Internet Engineering Task Force* (IETF por sus siglas en inglés) que describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como protocolos, procedimientos,. y comentarios e ideas sobre estos, entre otras cosas.

25 El protocolo de configuración dinámica de *host*, es un protocolo de red que permite a los clientes de una red IP obtener sus parámetros de configuración automáticamente.

26 El sistema de nombres de dominio, es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a la red, los nombres de dominio son asignados a cada una de las máquinas. Su función más importante, es traducir direcciones lógicas asociadas con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente con mayor facilidad para las personas.

Como convención se toma la primer dirección como identificador para la red, y la última dirección para la difusión (*Broadcast*).

La máscara de subred se utiliza para obtener tanto el número de máquina como el número de la red, a través de operaciones lógicas con bits.

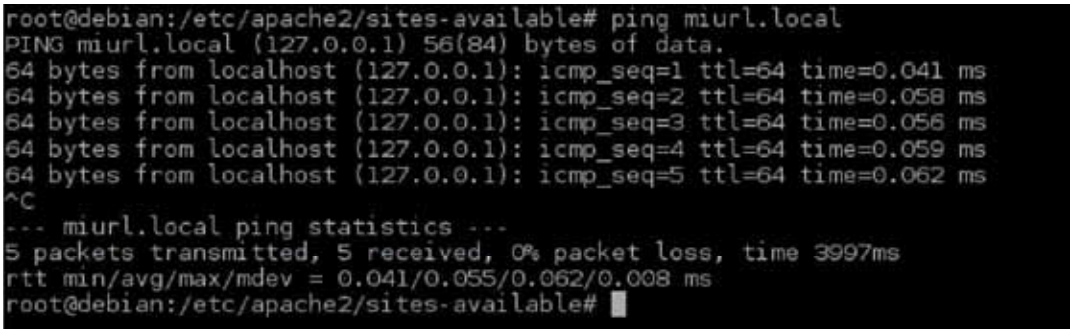
Las direcciones IP en su versión 6, se compone de 128 bits, lo cuál permite tener un mayor rango de direcciones IP disponibles, se desarrolló a partir de la necesidad conectar más dispositivos a la red bajo el Protocolo IP.

Una solución al agotamiento del número de direcciones IP, se dio con la traducción de direcciones de red (NAT[27]), definido en el RFC 3022. La idea de NAT es asignar una sola dirección a cada compañía u organización para que puedan conectarse a Internet, Ya dentro de la red de la organización se asignan direcciones para enrutar el tráfico interno. Cuando los paquetes salen de la organización, sus direcciones son traducidas por el NAT del ISP[28], sin embargo, al regresar, la caja NAT no sabe a qué máquina enviarlo, este es un problema para NAT. Sin embargo el problema se resuelve usando mecanismos como el puerto del servicio.

Protocolo ICMP

El protocolo de mensajes de control de Internet informa de un evento inesperado en la red, que también es usado para probar Internet. Se han definido varios mensajes, algunos son:

- *Destination unreachable*: El paquete no se puede entregar.
- *Time Exceeded*: Campo de tiempo de vida = 0.
- *Parameter problem*: Campo de encabezado no válido.
- *Source quench*: Paquete regulador.
- *Redirect*: Enseña a un enrutador sobre geografía.
- *Echo*: Pregunta a una máquina si está viva.
- *Echo reply*: Responde, sí, estoy viva.
- *Timestamp request*: Igual que Echo, pero con marca de tiempo.
- *Timestamp reply*: Igual que respuesta Echo, pero con marca de tiempo.



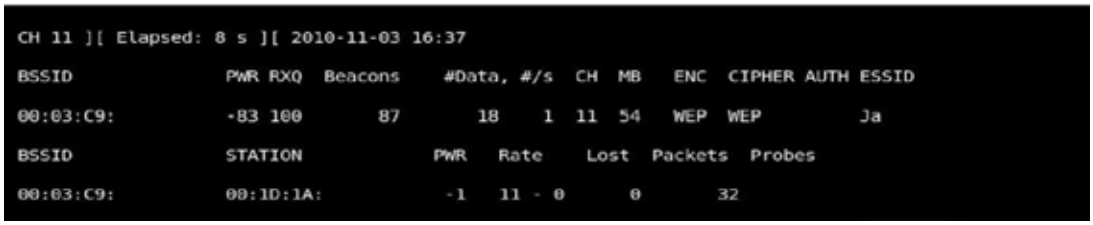
8 Figura: Los mensajes del protocolo ICMP son utilizados en la instrucción Ping.

27 NAT (Traducción de Dirección de Red) es un mecanismo utilizado por enrutadores IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

28 Proveedor de servicios de Internet.

Protocolo ARP

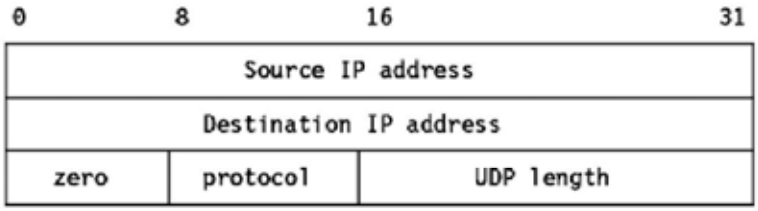
El protocolo de resolución de direcciones obtiene una dirección física a partir de la dirección lógica, ésto debido a que las máquinas en su electrónica de la capa de enlace, no sabe manejar las direcciones IP. La direcció física o MAC, viene escrita en cada tarjeta de red de fábrica. La definición de ARP está en el RFC 826.



9 Figura: Los paquetes ARP son explotados por algunas aplicaciones, tales como la suit de auditoría de redes "Aircrack".

Protocolo UDP

El protocolo de datagramas de usuario es prácticamente un IP con un encabezado corto y que no está orientado a la conexión. Proporciona un forma sencilla para que las aplicaciones envíen datagramas IP encapsulados sin tener que establecer una conexión. Está definido en el RFC 768.



10 Figura: Esquema de la estructura de UDP.

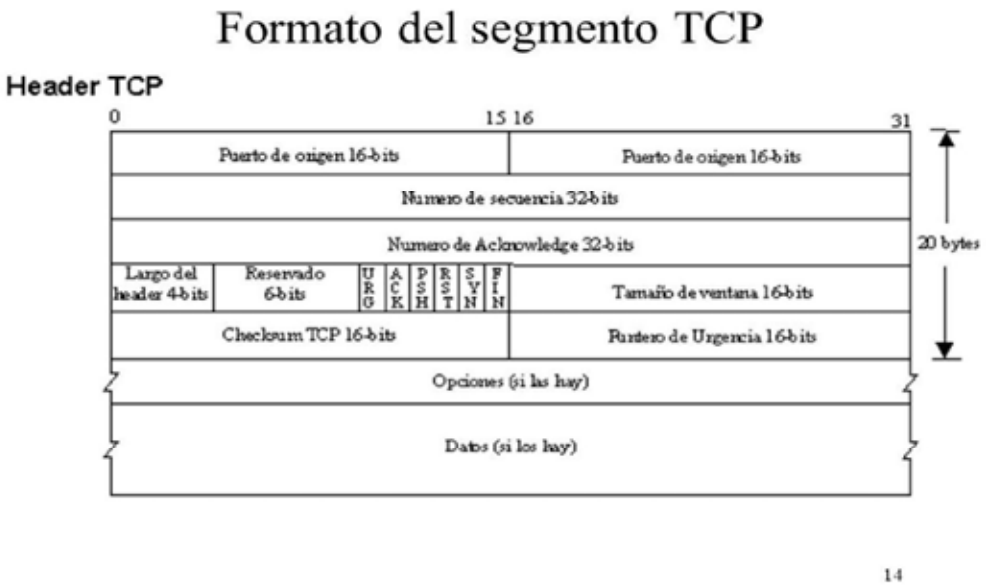
UDP transmite segmentos que son encabezados de 8 bytes de tamaño seguido de la carga útil.

El campo de longitud incluye al encabezado de 8 bytes y los datos. El campo de la sume de verificación, es opcional y se define como 0 cuando no se utiliza.

UDP no realiza control de flujo, control de errores o retransmisión de datos, sin embargo proporciona una interfaz para el protocolo IP con la característica de desmultiplexar[29] varios procesos utilizando puertos.

UDP es ideal para aplicaciones que requieren tener control preciso sobre el flujo de paquetes, control de errores y temporización.

29 La desmultiplexación es el acto inverso a la multiplexación. La multiplexación se refiere a la combinación de fuentes independientes de información, de manera que puedan transmitirse por un sólo canal de comunicación. La multiplexación ocurre tanto en el hardware como en el software .



11 Figura: Ilustración de la estructura de un segmento TCP.

Está diseñado para transferir paquetes de extremo a extremo a través de una interred no confiable, de manera segura. Está definido en el RFC 793, sin embargo en el RFC 1122 se corrigen varios errores y se detallan algunas especificaciones, y en el RFC 1323 se agregan ciertas extensiones.

TCP es un protocolo orientado a conexión, IP no proporciona ninguna garantía de que los paquetes sean entregados de manera exitosa, es TCP quien se encarga de terminar los temporizadores y retransmitir los datagramas que sean necesarios, así como ordenarlos si han llegado en desorden[9].

Modelo de servicio TCP

El servicio TCP se tiene al definir puntos terminales para el servidor y el cliente llamados *sockets*, cada *socket* tiene un número de identificación llamad puerto. Para tener servicio TCP, se debe hacer una llamada a un *socket* para establecer de manera explícita la conexión entre ambas máquinas.

Los puertos inferiores a 1024 se les denomina Puertos bien conocidos y son utilizados por servicios estándar, por ejemplo TELNET que trabaja en el puerto 23, HTTP en el 80 o FTP en el 21.

Todas la conexiones TCP son en modo dúplex y de punto a punto, es decir, que transmiten y reciben de manera simultánea, y no soporta multidifusión o difusión, debe comunicarse necesariamente de una máquina a otra.

La entidad emisora TCP y la receptora transmiten datos en forma de segmentos[30]. El software de TCP es el que decide el tamaño del segmento, hay dos límites que restringen tal tamaño, el primero es el tamaño de la carga útil de IP (65535 bytes), y el segundo es la unidad máxima de transferencia (MTU), y cada segmento debe caber en el MTU. Por lo general el MTU mide 1500 bytes, aunque puede cambiar dependiendo de cada red.

30 Un segmento consiste en un encabezado TCP fijo de 20 bytes mas una parte opcional, seguido de cero o más bytes de datos.

El encabezado del segmento TCP

El encabezado de cada segmento es un bloque fijo de 20 bytes, que puede ir seguido de opciones de encabezado. Después de las opciones, se encuentran el encabezado IP y el encabezado TCP de 20 bytes cada uno, luego puede continuar hasta 65495 bytes de datos. Los segmentos sin datos son válidos y por lo regular se usan para confirmar recepción de datos y mensajes de control.

Los campos de puerto de origen y destino son los puntos terminales locales de la conexión. Los campos de dirección de destino y de origen junto con sus respectivos puertos forman un punto terminal de origen y destino de 48 bits, de manera única a la conexión.

El número de secuencia y el número de confirmación de recepción, se utilizan para organizas a los datagramas transmitidos y para detectar los errores en la conexión, ambos poseen 32 bits.

La longitud del encabezado TCP indica la cantidad de palabras de 32 bits contenidas en el encabezado TCP.

Después viene un campo de 6 bit que no es usado, seguido de seis banderas de un bit cada una:

- URG: el apuntador urgente indica un desplazamiento en bytes a partir del número actual de secuencia en el que se encuentran datos urgentes.
- ACK: indica si el número de confirmación de recepción es válido.
- PSH: indica datos que se deben transmitir de inmediato.
- RST: reinicia una conexión que se ha caído debido a algún problema en la comunicación.
- SYN: se usa para establecer conexiones.
- FIN: es usado para liberar una conexión.

TCP también proporciona una suma de verificación para dar confiabilidad a los datagramas transmitidos.

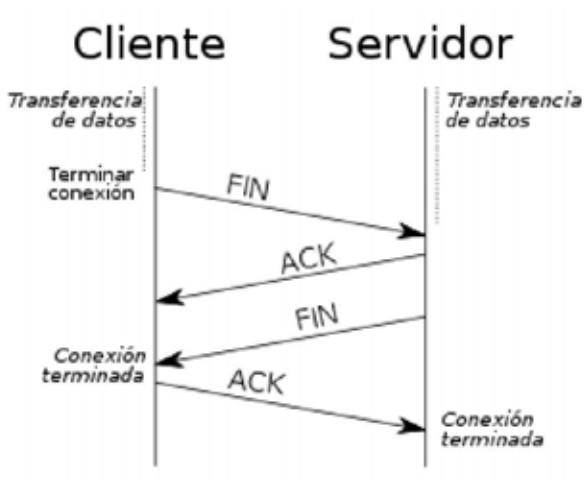
Cómo se establece una conexión TCP

Para establecer una conexión, el servidor tiene disponibles la primitivas *Listen* y *Acept*, mientras que el cliente puede usar la primitiva *Connect*, especificando la dirección y el puerto IP con el que desea comunicarse, así como el tamaño máximo de segmento TCP que puede aceptar y puede que también envíe datos de usuario como contraseñas o nombre del usuario.

El cliente usa la primitiva *Connect*, enviando un segmento TCP con el bit SYN habilitado y el ACK sin activar.

Si llega al servidor y no existe ningún proceso que haya ejecutado *Listen* en el puerto indicado en el campo de destino, el servidor responde con la bandera RST encendida para rechazar la conexión. De lo contrario la conexión se acepta.

Para liberar una conexión cualquiera de las dos partes transmite con la bandera FIN encendida, terminando con un ACK.



12 Figura: Protocolo de conexión TCP entre un cliente y un servidor.

La *World Wide Web*

Las páginas de Internet revolucionó la comunicación en los últimos años. Las páginas construidas a partir de hipertexto permiten que los usuarios no necesiten tener amplios conocimientos técnicos para poder navegar a través de la Web con la ayuda de un navegador. El programa navegador solicita la página a un servidor HTTP y recibe la página para ser interpretada. Las páginas se nombran utilizando localizadores uniformes de recursos (URL), conformado por tres partes: el nombre del protocolo, el nombre DNS de la máquina servidor, y el nombre del archivo.

Cuando el usuario da clic en algún vínculo de la página, el navegador determina el URL y pide al servidor DNS la dirección IP que está asociada con la URL; el DNS le responde con la IP. El navegador realiza una conexión TCP con el puerto 80 y con la dirección obtenida, después envía un mensaje en el que solicita un archivo. El servidor envía el archivo solicitado y se libera la conexión TCP. El navegador despliega el contenido de la página y el usuario la puede ver en su pantalla.

Por otra parte, en el servidor se reciben las solicitudes de conexión de los clientes, se acepta la conexión, se obtiene el nombre del archivo solicitado, lo busca en su disco duro, se lo regresa al cliente que lo solicitó y termina la conexión.

Uno de los problemas en los servidores, es que sus recursos deben ser suficientes para atender al mayor número de clientes de manera simultánea, por ejemplo, el tiempo de acceso a disco, la velocidad de acceso a la memoria, el ancho de banda de su conexión, etcétera.

Los servidores Web modernos no sólo aceptan y responden a las peticiones de los clientes, los siguientes pasos pueden ser un ejemplo del complejo trabajo que se realiza para servir páginas Web:

1. El servidor resuelve el nombre de la página solicitada.
2. Autentica al cliente.
3. Realiza control de acceso al cliente.
4. Realiza control de acceso a la página Web.

5. Verifica el caché[31].
6. Obtiene del disco la página solicitada.
7. Determina el tipo MIME[32] que se incluirá en la respuesta.
8. Se encarga de diversos detalles.
9. Regresa la respuesta al cliente.
10. Realiza una entrada en el registro del servidor.

Los URL

A cada página se le asigna un nombre único de manera internacional, los URL tienen tres partes, el protocolo, el Nombre de la máquina asignado por DNS, y el nombre del archivo.

El protocolo HTTP es el lenguaje que hablan los servidores Web, es su lenguaje nativo. El protocolo FTP permite el intercambio de archivos. El protocolo TELNET permite establecer una conexión en línea con una máquina remota. Todos estos protocolos pueden se incluidos en el URL de una página.

Un problema con URL es la necesidad de acceder a un recurso copiado en otra ubicación para minimizar el tráfico de un sistema, sin embargo, ya se está trabajando en una solución: el sistema URN que puede consderarse como un URL generalizado.

Las cookies

Otro problema de Web es que no tiene capacidad para iniciar una sesión, de manera que el servidor envía la página solicitada y no recuerda al cliente que se la solicitó. El servidor Web, en un principio no puede almacenar datos del cliente. Para resolver este tipo de problemas, el navegador Netscape[33] diseñó una solución: las cookies. Posteriormente se formalizaron en el RFC 2109.

Cuando el cliente solicita información del servidor, le envía también un archivo de a lo más 4KB, que contiene información del cliente. Los clientes almacenas tales archivos en un directorio asignado en su máquina. Las cookies pueden tener hasta cinco campos:

- El dominio indica de dónde viene la cookies (cada dominio puede almacenar hasta 20 cookies).
- La ruta indica que partes del árbol de archivos del servidor podrían utilizar la cookie.
- El contenido toma la forma nombre = valor, que es donde se almacena el contenido de la cookie.
- El campo de expiración indica cuando la cookie deja de ser válida.
- El campo seguro puede establecerse para indicar que el navegador podría regresar simplemente la cookie a un servidor seguro.

31 La caché es la memoria de acceso rápido de una computadora, que guarda temporalmente las últimas informaciones procesadas.

32 *Multipurpose Internet Mail Extensions* (en español "extensiones multipropósito de correo de Internet"), son una serie de especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario.

33 Netscape Navigator fue un navegador web y el primer producto comercial de la compañía Netscape Communications, quién anunció el 13 de octubre de 1994 que lanzaría Navigator disponible de forma gratuita para todos sus usuarios no comerciales, y que las versiones betas de 1.0, 1.1 se podrán descargar en noviembre de 1994 y marzo de 1995. La versión final, 1.0 estuvo disponible en diciembre de 1994.

13 Figura: Ejemplo de un archivo Cookie, con datos del cliente que puede utilizar el servidor.

Las cookies se usan también para que el servidor lleve un control del número de visitantes, por mencionar sólo un ejemplo de uso en el servidor.

Páginas Web estáticas

Una página estática es un archivo simple que es transferido como tal hacia el cliente, es la forma mas sencilla de servir contenido Web.

Una página Web se compone de un encabezado y un cuerpo encerrados entre etiquetas <HTML> </HTML>. Los encabezados se delimitan por las etiquetas <HEAD> </HEAD>, y el cuerpo <BODY> </BODY>. Las etiquetas no distinguen entre mayúsculas y minúsculas, aunque los nuevos estándares indican que se escriban en minúsculas. Los comandos contenidos en las etiquetas, se llaman directivas. Los parámetros nombrados dentro de las etiquetas se llaman atributos.

Los formularios son una parte importante en las páginas Web, pues ofrecen una interacción adicional para los usuarios, ya que pueden proporcionar datos o recibirlos. Aunque en una página dinámica, su capacidad es bastante limitada.

Las páginas estáticas se construyen básicamente con el ya mencionado HTML, pero además se pueden dinamizar con la ayuda de lenguajes como Javascript y CSS, que son enviados al cliente en el mismo documento, o en archivos adjuntos y que son interpretados por el navegador. En los inicios de la Web, éstas herramientas fueron de gran ayuda para los desarrolladores de sitios en Internet.

Técnicas como AJAX permiten realizar tareas bastante complejas en el cliente. Siendo una técnica de desarrollo para crear aplicaciones interactivas, permiten ejecutar *scripts*[34] en el lado del cliente mientras se mantiene una comunicación asíncrona en segundo plano con el servidor, logrando así una comunicación constante, y liberando a la página de los *Postback*[35] indeseados o innecesarios.

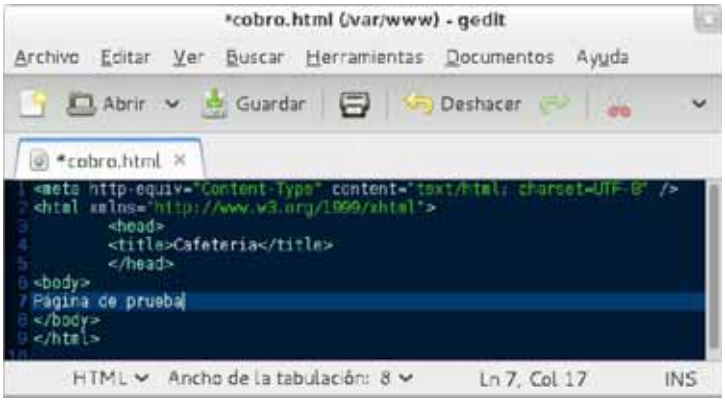
Éstas páginas no permiten modificación entre cada petición al servidor, siempre son enviadas tal cual, sin posibilidad de cambiar su contenido. Esto limita muchas de las actividades que se realizan hoy en día en Internet, tales como la interacción con bases de datos o con aplicaciones dentro o fuera del propio servidor.

34 Un *script* es un programa normalmente simple, que por lo regular se almacena en un archivo de texto plano. Los *script* son casi siempre interpretados, aunque no todo programa interpretado es considerado un *script*. Pr lo general, los *scripts* realizan diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso es frecuente que los *shells* sean a la vez intérpretes de este tipo de programas. Aunque pueden encontrarse en otros ambientes, por ejemplo, una página Web.

35 Un *Postback* es un mensaje POST del protocolo HTTP a la misma página que contiene el formulario. En otras palabras, el contenido del formulario es enviado de nuevo a la misma URL que la del formulario. Los *Postback* son vistos comúnmente en formularios de edición, donde el usuario introduce información en un formulario y pulsa en "guardar" o "enviar", provocando un *Postback*. Entonces, el servidor actualiza la misma página con la información que acaba de recibir.

Debido a éstas necesidades, surgen tecnologías que permiten el desarrollo de sitios dinámicos.

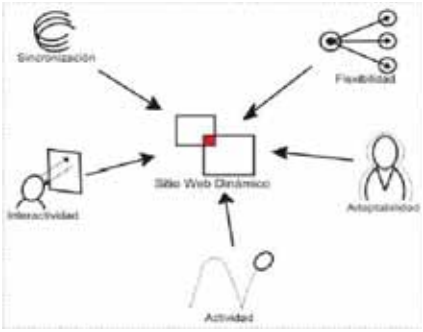
Un ejemplo de éste tipo de páginas sería el siguiente código:



14 *Figura: Código HTML editado con Gedit.*

Páginas Web dinámicas

Para resolver las limitaciones que ofrecen las páginas Web estáticas, se desarrollaron tecnologías que trabajan en el servidor, las cuales permiten procesar los archivos que se encuentran alojados y presentar una página de acuerdo a la petición del usuario o del proceso del servidor. Tecnologías tales como ASP o PHP permiten realizar conexiones con bases de datos, tener acceso dispositivos externos al servidor Web, incluso controlar otros sistemas a través de una interfaz Web.



15 *Figura: Sincronización, flexibilidad, interactividad, actividad y adaptabilidad, son conceptos relacionados al desarrollo de aplicaciones dinámicas con tecnologías del lado del servidor.*

Active Server Pages (ASP) de Microsoft es un entorno de *Scripting*[36] del lado del servidor que se puede utilizar para crear y ejecutar aplicaciones en servidores Web, de manera interactiva y dinámica. Con ASP, se pueden combinar páginas HTML, secuencias de comandos y componentes COM para crear páginas web interactivas y aplicaciones basadas en la Web de gran alcance que son fáciles de desarrollar y modificar .

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. En lugar de usar muchos comandos para mostrar HTML como en lenguajes como C o Perl, las páginas de PHP contienen HTML con código incrustado. El código de PHP está encerrado entre las etiquetas especiales de comienzo y final *<?php* y *?>* que permiten entrar y salir del modo PHP.

36 El *Scripting* es un tipo de lenguaje de programación que es generalmente interpretado. Los programas compilados son convertidos de forma permanente a un código especial antes de que puedan ejecutarse, en cambio los *Scripts* permanecen en su forma original y son interpretados comando por comando cada vez que se ejecutan.

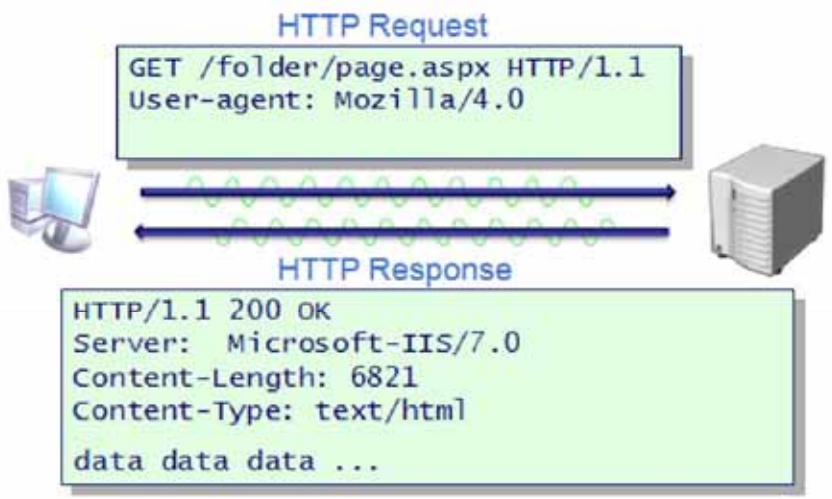
Una página dinámica incorpora todas las facilidades que ofrecen las páginas estáticas, de hecho, son una mejora de éstas.

La forma tradicional de generar contenido dinámico fue CGI[37], la cuál es una interfaz estandarizada que permite que los servidores se comunique con programas que se encuentran del otro lado de la interfaz del usuario, capaces de recibir parámetros y datos de formulario, pudiendo procesarlos y devolver una respuesta en formato HTML. Se utilizan diversos lenguajes para escribir aplicaciones CGI, por lo general se usa Perl o Phyton, sin embargo es posible utilizar cualquier otro lenguaje, incluso lenguaje C.

Protocolo HTTP

Su acrónimo significa Protocolo de Transferencia de Hipertexto, está definido en el RFC 2616, y es el lenguaje nativo de Web e indica qué mensajes pueden ser enviados a los servidores y qué respuestas le envían a los clientes. La interacción consta de una solicitud ASCII[38] seguida por una respuesta de tipo MIME definida en el RFC 822, siendo así tanto para clientes y servidores sin excepción.

Una conexión HTTP



16 Figura: Proceso de petición y respuesta HTTP, entre un cliente y un servidor.

Para establecer la comunicación, el cliente realiza una solicitud TCP con el puerto 80 al servidor, aunque el puerto puede ser redefinido por el desarrollador, siendo utilizados por ejemplo, puertos externos tales como el 8080 y el 8081 entre otros. TCP evita que tanto los servidores como los clientes, no tengan que preocuparse por problemas en la comunicación en los mensajes largos, perdidos o duplicados, ni por confirmar su recepción, ya que como se ha mencionado anteriormente, es un protocolo orientado a la conexión.

La versión 1.0 de HTTP, trabajaba con solicitudes simples de documentos con texto muy ligeros, aceptaba la conexión y respondía con el archivo solicitado, entonces cerraba la conexión. Sin embargo, con la evolución de Internet, el contenido de las páginas se fue enriqueciendo hasta llegar a contener muchas imágenes, videos y demás contenido multimedia. Transferir contenido de este tipo, resulta muy costoso si se

37 CGI es una importante tecnología de la World Wide Web que permite a un navegador web solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa.

38 El código ASCII (*American Standard Code for Information Interchange*) es un estándar para el intercambio de Información que fue creado en 1963 por el Comité Estadounidense de Estándares o *ASA*, este organismo cambio su nombre en 1969 por "Instituto Estadounidense de Estándares Nacionales" o *ANSI* como se lo conoce desde entonces.

realiza con peticiones TCP para cada uno de sus elementos. El problema fue solucionado con la versión 2.0 del protocolo, la cuál soportaba ya conexiones persistentes, es decir, el cliente podía seguir enviando peticiones TCP, sin que el servidor necesariamente hubiese respondido a las respuestas anteriores. Esta nueva característica alivió la sobrecarga del protocolo TCP en cada solicitud.

Métodos POST y GET

El diseño de HTTP permite el desarrollo de aplicaciones orientadas a objetos, por lo que soporta operaciones tales como las llamadas a métodos, las cuales son distintas a las que se realizan al solicitar una página Web.

17 Figura: Los formularios HTML son la herramienta más utilizada por los desarrolladores Web, para enviar o recibir información del servidor.

Las solicitudes consisten en una o más líneas de texto ASCII, y la primer línea es el nombre del método. Los nombres son sensibles a mayúsculas y minúsculas, lo que obliga a ser cuidadosos.

El método GET solicita al servidor una página, la cuál está codificada de manera adecuada en MIME, como ejemplo podemos tener: GET unapagina.html HTTP/1.1.

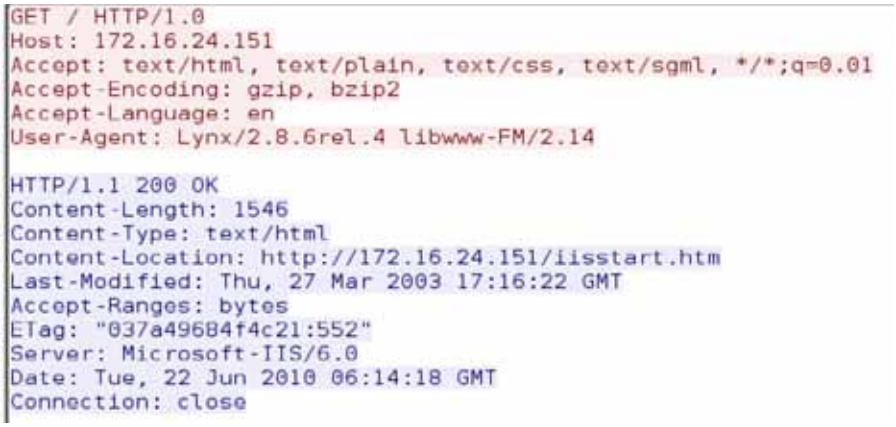
El método POST transporta un URL donde los nuevos datos son insertados en él en un sentido generalizado. En la práctica no es tan utilizado como GET.

Ambos métodos permiten enviar datos desde el cliente al servidor, ya sea a través de un formulario HTML, o de una URL. Los datos son procesados en el servidor y se devuelve una respuesta de acuerdo a la información generada.

Existen otros métodos en HTTP sin embargo, en el diseño de páginas Web no son muy utilizados.

Encabezados de los mensajes

Después de la línea de solicitud, por ejemplo GET, pueden continuar más lineas con información las cuáles reciben el nombre de Encabezados de Solicitud, que pueden compararse con los parámetros en una llamada a una función. Cuando el servidor responde, puede agregar también encabezados a la respuesta, los cuales se denominan Encabezados de respuesta.



18 Figura: Encabezados HTTP tanto del cliente como del servidor.

Algunos encabezados importantes para el desarrollo de aplicaciones que implementen el protocolo HTTP en el lado del cliente, pueden ser los siguientes:

- Encabezado *User-Agent*: informa al servidor acerca de características del cliente, cono lo son: el navegador utilizado para la petición, el sistema operativo, y otras propiedades.
- Cuatro encabezados *Accept*: Le indican al servidor el tipo de datos que el cliente soporta.
  - El primero especifica los tipos MIME, por ejemplo *text/html*.
  - El segundo proporciona el conjunto de caracteres para la codificación, por ejemplo ISO-8859-5.
  - El tercero tiene que ver con métodos de compresión, por ejemplo GZIP.
  - El cuarto indica el idioma natural, por ejemplo: el español.
- El encabezado *Host*: nombra al servidor. Es obligatorio y se toma del URL, y se usa para indicar la máquina a la cuál entregar la petición.
- El encabezado *Authorization*: se utiliza en páginas protegidas, que necesitan de una autenticación.
- El encabezado *Cookie*: es utilizado para enviar el contenido de las cookies almacenadas en el cliente.
- El encabezado *Date*: es bidireccional y contiene la fecha y la hora del mensaje.

Ejemplos de los encabezados utilizados por el servidor para incluirlos en las respuestas a los clientes, son:

- El encabezado Server: el servidor lo utiliza para informar quién es, y de manera opcional, algunas propiedades adicionales.
- Cuatro encabezados *Content-*: Describen el contenido que el servido envía como respuesta.
- El encabezado *Last-Modified*: indica cuándo se modificó la página por última vez, útil para el manejo del caché.
- El encabezado *Location*: se utiliza si la página ha cambiado de ubicación o para permitir que múltiples URLs se dirijan a la misma página.
- El encabezado *Accept-Ranges*: es usado por algunos servidores que aceptan solicitudes en rangos de bytes, esto permite que las páginas obtengan la respuesta en porciones pequeñas.
- El encabezado *Set-Cookie*: el servidor lo utiliza para enviar *cookies* al cliente para que los almacene,

y sean usadas en solicitudes posteriores.

DOM

El modelo de objetos del documento (DOM por sus siglas en inglés) es una conjunto de elementos que permiten desarrollar aplicaciones Web. Es una API[39] bastante poderosa, pero desafortunadamente, las posibilidades teóricas de DOM son mucho más avanzadas de las que se pueden utilizar en la práctica para desarrollar aplicaciones Web. El uso de DOM siempre está limitado por las posibilidades que ofrece cada navegador. Mientras que algunos navegadores como Firefox y Safari implementan DOM de nivel 1 y 2, otros navegadores como Internet Explorer en sus versiones 7 y anteriores, ni siquiera ofrecen una implementación completa de DOM nivel 1. [10]

En páginas HTML, el nodo raíz de todos los demás se define en el objeto *HTMLDocument*. Además, se crean objetos de tipo *HTMLElement* por cada nodo de tipo *Element* del árbol DOM.

A continuación se muestra la página HTML básica que se va a emplear en todos los siguientes ejemplos:

```
<html>
<head>
  <title>Ejemplo de DOM</title>
</head>
<body>
  <p>Ejemplo de DOM</p>
  <p>Un parrafo</p>
  <p>Otro parrafo</p>
</body>
</html>
```

La operación mas básica consiste en obtener el objeto que representa el elemento raíz de la página:

```
var objeto_html = document.documentElement;
```

La instrucción anterior, guarda en la variable *objeto_html* un objeto de tipo *HTMLElement* el cual representa el elemento <html> de la página web. Según el árbol de nodos DOM, desde el nodo <html> derivan dos nodos del mismo nivel jerárquico: <head> y <body>.

Utilizando los métodos proporcionados por DOM, es sencillo obtener los elementos <head> y <body>. En primer lugar, los dos nodos se pueden obtener como el primer y el último nodo hijo del elemento <html>, por ejemplo:

```
var objeto_head = objeto_html.firstChild;
var objeto_body = objeto_html.lastChild;
```

Otra forma directa de obtener los dos nodos consiste en utilizar la propiedad *childNodes* del elemento <html> es la siguiente:

```
var objeto_head = objeto_html.childNodes[0];
var objeto_body = objeto_html.childNodes[1];
```

Si se desconoce el número de nodos hijo que tiene un nodo, se puede usar la propiedad *length* de

39 Una API es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

childNodes:

```
var numeroDescendientes = objeto_html.childNodes.length;
```

Además, el DOM de HTML permite acceder directamente al elemento <body> utilizando el atajo document.body:

```
var objeto_body = document.body;
```

Una operación común en muchas aplicaciones consiste en comprobar el tipo de nodo, que se obtiene de forma directa mediante la propiedad *nodeType*:

```
alert(document.nodeType);
alert(document.documentElement.nodeType);
```

Lo anteroir devuelve un número, el número del tipo de nodo, sin embargo, no es necesario memorizar los valores numéricos de los tipos de nodos, ya que se pueden emplear las constantes predefinidas:

```
alert(document.nodeType == Node.DOCUMENT_NODE);
alert(document.documentElement.nodeType == Node.ELEMENT_NODE);
```

Estas constantes no están soportadas en Internet Explorer 7 y sus versiones anteriores, por lo que si se quieren utilizar es necesario definirlas de forma explícita:

```
if(typeof Node == "undefined") {
  var Node = {
    ELEMENT_NODE: 1,
    ATTRIBUTE_NODE: 2,
    TEXT_NODE: 3,
    CDATA_SECTION_NODE: 4,
    ENTITY_REFERENCE_NODE: 5,
    ENTITY_NODE: 6,
    PROCESSING_INSTRUCTION_NODE: 7,
    COMMENT_NODE: 8,
    DOCUMENT_NODE: 9,
    DOCUMENT_TYPE_NODE: 10,
    DOCUMENT_FRAGMENT_NODE: 11,
    NOTATION_NODE: 12
  };
}
```

El código anterior comprueba si el navegador en el que se está ejecutando tiene definido el objeto Node. Si este objeto no está definido, se trata del navegador Internet Explorer 7 o alguna versión anterior, por lo que se crea un nuevo objeto llamado Node y se le incluyen como propiedades todas las constantes definidas por DOM.

DOM permite el acceso directo a todos los atributos de cada etiqueta. Para ello, los nodos de tipo *Element* contienen la propiedad *attributes*, que permite acceder a todos los atributos de cada elemento. Aunque técnicamente la propiedad *attributes* es de tipo *NamedNodeMap*, sus elementos se pueden acceder como si fuera un arreglo. DOM proporciona diversos métodos para trabajar con los atributos:

```
getNamedItem(nombre);
removeNamedItem(nombre);
setNamedItem(nodo);
item(posicion);
```

Utilizando estos métodos, es posible procesar y modificar fácilmente los atributos de los elementos HTML:

```
<p id="unElemento" style="color: blue">Párrafo de prueba</p>

var p = document.getElementById("unElemento");
var elId = p.attributes.getNamedItem("id").nodeValue;
var elId = p.attributes.item(0).nodeValue;
p.attributes.getNamedItem("id").nodeValue = "otraCosa";

var atributo = document.createAttribute("lang");
atributo.nodeValue = "es";
p.attributes.setNamedItem(atributo);
```

DOM también proporciona otros métodos que permiten el acceso y la modificación de los atributos de forma directa:

```
getAttribute(nombre) === attributes.getNamedItem(nombre).
setAttribute(nombre, valor) === attributes.getNamedItem(nombre).value = valor.
removeAttribute(nombre) === attributes.removeNamedItem(nombre).
```

El ejemplo anterior utilizando los nuevos métodos:

```
<p id="unElemento" style="color: blue">Párrafo de prueba</p>

var p = document.getElementById("unElemento");
var elId = p.getAttribute("id");
p.setAttribute("id", "otraCosa");
```

Con los métodos vistos, sólo se puede encontrar un elemento realizando un recorrido del árbol, sin embargo, en páginas con demasiados elementos, esto resulta una tarea poco eficiente. Para solucionar este problema, DOM proporciona una serie de métodos para acceder de forma directa a los nodos deseados. Los métodos disponibles son *getElementsByTagName()*, *getElementsByName()* y *getElementById()*.

La función *getElementsByTagName()* obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

El siguiente ejemplo muestra cómo obtener todos los párrafos de una página XHTML:

```
var parrafos = document.getElementsByTagName("p");
```

El valor que devuelve la función es arreglo con todos los nodos que cumplen la condición. En realidad, el valor devuelto no es de tipo *array* normal, sino que es un objeto de tipo *NodeList*.

La función *getElementsByTagName()* se puede aplicar de forma recursiva sobre cada uno de los nodos devueltos por la función.

La función *getElementsByName()* obtiene todos los elementos de la página XHTML cuyo atributo *name*

coincida con el parámetro que se le pasa a la función.

En el siguiente ejemplo, se obtiene directamente el único párrafo de la página que tiene el nombre indicado:

```
var parrafoEspecial = document.getElementsByName("especial");

<p name="prueba">...</p>
<p name="especial">...</p>
<p>...</p>
```

Normalmente el atributo *name* es único para los elementos HTML que lo incluyen, aunque no siempre, por lo que puede devolver un arreglo de elementos al igual que el métod *getElementsByTagName()*.

Internet Explorer 7 y sus versiones anteriores no implementan de forma correcta esta función, ya que también devuelven los elementos cuyo atributo id sea igual al parámetro de la función.

La función *getElementById()* es la función más utilizada cuando se desarrollan aplicaciones web dinámicas. Se trata de la función preferida para acceder directamente a un nodo y para leer o modificar sus propiedades. La función *getElementById()* devuelve el elemento XHTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo id debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

```
var cabecera = document.getElementById("cabecera");

<div id="cabecera">
<a href="/" id="logo">...</a>
</div>
```

Codificación Base64

Base64 es un grupo de esquemas de codificación similares al binario, que representan los datos binarios en un formato de cadena ASCII traduciéndolo a una representación radix-64. El término base 64 se origina en una codificación específica de la transferencia de contenido MIME .

Los esquemas de codificación Base64 se utilizan comúnmente cuando hay una necesidad de codificar datos binarios que necesita ser almacenada y transferida a través de los medios de comunicación que están diseñados para tratar con datos de texto. Esto es para asegurar que los datos se mantienen intactos durante el transporte. Base64 se utiliza comúnmente en un número de aplicaciones, incluyendo contenido MIME en correo electrónico, y almacenamiento de datos complejos en XML.

La elección del conjunto de caracteres seleccionado para los 64 caracteres necesarios para la base varía entre implementaciones. La regla general es elegir un conjunto de 64 caracteres que es a la vez parte de un subconjunto común a la mayoría de las codificaciones. Por ejemplo, la implementación de Base64 MIME utiliza los conjuntos de caracteres {A,...,Z}, {a,...,z}, y {0,...,9} durante los primeros 62 valores. Otras variaciones, por lo general derivados de la base 64, comparten esta propiedad, pero difieren en los símbolos elegidos para los dos últimos valores; un ejemplo es UTF-7[40].

Los datos transmitidos o almacenados en Base64, pueden ser incrustados en una página Web por ejemplo.

40 UTF-7 es una codificación de caracteres de longitud variable que fue propuesta para representar texto codificado con Unicode usando un flujo de caracteres ASCII, para ser usado, en mensajes de correo electrónico de Internet y otros medios. UTF-7 no es un formato de transformación y no forma parte del estándar Unicode.

El lenguaje de programación C

El lenguaje C[11], ha mostrado ser una poderosa herramienta para el desarrollo de diversos tipos de aplicaciones, desde programas para escritorio, programación de sistemas operativos, implementación de aplicaciones móviles, software para microcontroladores, e incluso, para desarrollar aplicaciones Web en el lado del servidor con la tecnología CGI.

Programación de Sockets TCP en sistemas Unix

De manera sencilla, se puede decir que un *socket* es una forma de hablar con otra computadora. Para ser más precisos, es una manera de hablar con otras computadoras usando descriptores de ficheros estándar de Unix. En Unix, todas las acciones de entrada y salida son desempeñadas escribiendo o leyendo en uno de estos descriptores de fichero, los cuales son simplemente un número entero, asociado a un fichero abierto que puede ser una conexión de red, un terminal, o cualquier otra cosa, recordemos que los sistemas Unix manejan todo como si de un archivo se tratara.

Sobre los diferentes tipos de *sockets* en Internet, hay muchos tipos pero sólo se describirán dos de ellos *Sockets de Flujo* y *Sockets de Datagramas*.

Los *Sockets de Flujo,* están libres de errores: por ejemplo, si enviáramos por el *socket* de flujo tres objetos "A, B, C", llegarán al destino en el mismo orden: "A, B, C"; debido a que estos *sockets* usan TCP.

Sockets de Datagramas. Éstos usan UDP y no necesitan de una conexión accesible como los *Sockets* de Flujo, en este tipo de *sockets* se construirá un paquete de datos con información sobre su destino y se enviará afuera, sin necesidad de una conexión.

Estructuras de datos para Sockets

Las estructuras utilizadas para la programación de *sockets*, se usan para almacenar información sobre direcciones. La primera de ellas es *struct sockaddr*, la cual contiene información del *socket*.

```
struct sockaddr
{
   unsigned short sa_family;  /* familia de la dirección */
   char sa_data[14];          /* 14 bytes de la dirección del protocolo */
};
```

Struct sockaddr_in, nos ayuda a hacer referencia a los elementos del *socket*.

```
struct sockaddr_in
{
   short int sin_family;        /* Familia de la Dirección             */
   unsigned short int sin_port; /* Puerto                              */
   struct in_addr sin_addr;     /* Dirección de Internet               */
   unsigned char sin_zero[8];   /* Del mismo tamaño que struct sockaddr */
};
```

La siguiente estructura no es muy usada pero está definida como una unión.

```
struct in_addr
{
```

```
   unsigned   long s_addr;
};
```

La siguiente estructura está definida en *netdb.h*:

```
struct hostent
{
   char *h_name;               /* El nombre oficial del nodo.              */
   char **h_aliases;           /* Lista de Alias.                          */
   int h_addrtype;             /* Tipo de dirección del nodo.              */
   int h_length;               /* Longitud de la dirección.                */
   char **h_addr_list;         /* Lista de direcciones del nombre del      */
                               /* servidor.                                */
  #define h_addr  h_addr_list[0]/* Dirección, para la compatibilidad con    */
                               /* anteriores.                              */
};
```

Existen dos tipos de ordenamiento de bytes: bytes más significativos, y bytes menos significativos (*Big endian* y *Little Endian*). Éste es llamado *Ordenación de Bytes para Redes*, algunas máquinas utilizan este tipo de ordenación para guardar sus datos internamente.

Existen dos tipos a los cuales seremos capaces de convertir: *short* y *long*. Imaginemos que se quiere convertir una variable larga de *Ordenación de Bytes para Nodos* a una de *Orden de Bytes para Redes*. ¿Qué haríamos? Existe una función llamada *htonl()* que haría exactamente esta conversión. Las siguientes funciones son análogas a ésta y se encargan de hacer este tipo de conversiones:

- *htons()*: Nodo a variable corta de Red.
- *Htonl()*: Nodo a variable larga de Red.
- *Ntohs()*: Red a variable corta de Nodo.
- *ntohl()*: Red a variable larga de Nodo.

Una cosa importante, es que *sin_addr* y *sin_port*, de la estructura *sockaddr_in*, deben ser del tipo *Ordenación de Bytes para Redes*.

Funciones para *Sockets*

En C, existen algunas funciones que nos ayudan a manipular direcciones Ip, dos de ellas son precisamente *inet_addr()* y *inet_ntoa()*.

Por un lado, la función *inet_addr()* convierte una dirección IP en un entero largo sin signo (*unsigned long int*), por ejemplo:

```
dest.sin_addr.s_addr = inet_addr("195.65.36.12");
```

Por el contrario, *inet_ntoa()* convierte a una cadena que contiene una dirección IP en un entero largo. Por ejemplo:

```
char *ip;
ip=inet_ntoa(dest.sin_addr);
```

```
printf("La dirección es: %s\n",ip);
```

Por el contrario, *inet_ntoa()* convierte a una cadena que contiene una dirección IP en un entero largo. Por ejemplo:

La función *socket()*

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain,int type,int protocol);
```

y sus argumentos:

- *domain:* Se podrá establecer como *AF_INET* (para usar los protocolos ARPA de Internet), o como *AF_UNIX* (si se desea crear sockets para la comunicación interna del sistema). Éstas son las más usadas, pero no las únicas. Existen muchas más, aunque no se nombrarán aquí.
- *type:* Aquí se debe especificar la clase de socket que queremos usar (de Flujos o de Datagramas). Las variables que deben aparecer son *SOCK_STREAM* o *SOCK_DGRAM* según querramos usar sockets de Flujo o de Datagramas, respectivamente.
- *Protocol:* Aquí, simplemente se puede establecer el protocolo a 0.

La función *socket()* nos devuelve un descriptor de *socket*, el cual podremos usar luego para llamadas al sistema. Si nos devuelve -1, se ha producido un error (útil para rutinas de verificación de errores).

La función *bind()*

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int fd, struct sockaddr *my_addr,int addrlen);
```

Sus argumentos:

- *fd*: Es el descriptor de fichero socket devuelto por la llamada a *socket()*.
- *my_addr:* es un puntero a una estructura *sockaddr*
- *addrlen:* contiene la longitud de la estructura *sockaddr* a la cuál apunta el puntero *my_addr*. Se debería establecer como *sizeof(struct sockaddr)*.

La llamada *bind()* se usa cuando usaremos los puertos locales de nuestra máquina, normalmente cuando utilizamos la llamada *listen()*. Su principal función es asociar un *socket* con un puerto de nuestra máquina.

La función *connect()*

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int fd, struct sockaddr *serv_addr, int addrlen);
```

Sus argumentos:

- *fd:* Debería configurarse como el fichero descriptor del *socket*, el cuál fue devuelto por la llamada a *socket()*.
- *serv_addr:* Es un puntero a la estructura *sockaddr* la cuál contiene la dirección IP destino y el puerto.

- *addrlen:* Análogamente de lo que pasaba con *bind()*, este argumento debería establecerse como *sizeof(struct sockaddr)*.

La función *connect()* se usa para conectarse a un puerto definido en una dirección IP. Devolverá -1 si ocurre algún error.

La función *listen()*

```
#include <sys/types.h>
#include <sys/socket.h>
int listen(int fd,int backlog);
```

Los argumentos de *listen()*:

- *fd*. Es el fichero descriptor del *socket*, el cual fue devuelto por la llamada a *socket()*
- *backlog*. Es el número de conexiones permitidas.

La función *listen()* se usa si se están esperando conexiones entran y que así, alguien pueda conectarse a nuestra máquina.

Después de llamar a *listen()*, se debe llamar a *accept()*, para que se acepten las conexiones entrantes. La secuencia de llamadas al sistema es:

1. *socket();*
2. *bind();*
3. *listen();*

Como todas las funciones descritas arriba, *listen()* devolverá -1 en caso de error.

La función *accept()*

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int fd, void *addr, int *addrlen);
```

Los argumentos de la función:

- *fd*. Es el fichero descriptor del *socket*, que fue devuelto por la llamada a *listen()*.
- *addr*. Es un puntero a una estructura *sockaddr_in* en la que se puede determinar qué nodo nos está contactando y desde qué puerto.
- *addrlen*. Es la longitud de la estructura a la que apunta el argumento *addr*, por lo que conviene establecerlo como *sizeof(struct sockaddr_in)*, antes de que su dirección sea pasada a *accept()*.

Cuando alguien intenta conectarse a nuestra computadora, se debe usar *accept()* para conseguir la conexión.

Un pequeño ejemplo del uso de *accept()* para obtener la conexión:

```
(...)
sin_size=sizeof(struct sockaddr_in);
/* En la siguiente línea se llama a accept() */
if ((fd2 = accept(fd,(struct sockaddr *)&client,&sin_size))==-1){
printf("accept() error\n");
```

```
exit(-1);
}
(...)
```

Se usa la variable *fd2* para añadir las llamadas *send()* y *recv()* que se explican en seguida.

Función *send()*

```
#include <sys/types.h>
#include <sys/socket.h>
int send(int fd,const void *msg,int len,int flags);
```

Los argumentos de esta llamada:

- *fd:* Es el fichero descriptor del *socket*, con el cual se desea enviar datos.
- *msg:* Es un puntero apuntando al dato que se quiere enviar.
- *len:* es la longitud del dato que se quiere enviar (en bytes).
- *flags:* deberá ser establecido a 0 .

Esta función envía datos usando *sockets de flujo* o *sockets conectados de datagramas*. Si se desea enviar datos usando *sockets no conectados de datagramas* debe usarse la llamada *sendto()*. La llamada a *send()* devuelve -1 en caso de error, o el número de bytes enviados en caso de éxito.

La función *recv()*

```
#include <sys/types.h>
#include <sys/socket.h>
int recv(int fd, void *buf, int len, unsigned int flags);
```

Veamos los argumentos:

- *fd:* Es el descriptor del *socket* por el cual se leerán datos.
- *buf:* Es el búfer en el cual se guardará la información a recibir.
- *len:* Es la longitud máxima que podrá tener el búffer.
- *flags:* Por ahora, se deberá establecer como 0.

Al igual que *send()*, esta función es usada con datos en *sockets de flujo* o *sockets conectados de datagramas*. Si se deseara enviar, o en este caso, recibir datos usando *sockets desconectados de Datagramas*, se debe usar la llamada *recvfrom()*. Análogamente a *send()*, *recv()* devuelve el número de bytes leídos en el búfer, o -1 si se produjo un error.

La función *recvfrom()*

```
#include <sys/types.h>
#include <sys/socket.h>

int recvfrom(int fd,void *buf, int len, unsigned int flags
struct sockaddr *from, int *fromlen);
```

Los argumentos de la función:

- *fd:* Es el descriptor del *socket* por el cual se leerán datos.
- *buf:* Es el búfer en el cual se guardará la información a recibir.
- *len:* Es la longitud máxima que podrá tener el búfer.
- *flags:* Por ahora, se deberá establecer como 0.
- *from:* Es un puntero a la estructura *sockaddr*.
- *fromlen:* Es un puntero a un entero local que debería ser inicializado a *sizeof(struct sockaddr)*.

Análogamente a lo que pasaba con *recv()*, *recvfrom()* devuelve el número de bytes recibidos, o -1 en caso de error.

Función *close()*

```
#include <unistd.h>
close(fd);
```

La función *close()* es usada para cerrar la conexión de nuestro descriptor de *socket*. Si llamamos a *close()* no se podrá escribir o leer usando ese *socket*, y si alguien trata de hacerlo recibirá un mensaje de error.

La función *shutdown()*

```
#include <sys/socket.h>
int shutdown(int fd, int how);
```

Veamos los argumentos:

- *fd:* Es el fichero descriptor del *socket* al que queremos aplicar esta llamada.
- *how:* Sólo se podrá establecer uno de estos nombres:
    1. *0:* Prohibido recibir.
    2. *1:* Prohibido enviar.
    3. *2:* Prohibido recibir y enviar.

Llamar a *close()* y establecer el argumento *how* con un valor: *2* tiene el mismo efecto. La función *shutdown()* devolverá 0 si todo ocurre bien, o -1 en caso de error.

La función *gethostname()*

```
#include <unistd.h>
int gethostname(char *hostname, size_t size);
```

Sus argumentos:

- *hostname:* Es un puntero a un array que contiene el nombre del nodo actual.
- *size:* La longitud del array que contiene al nombre del nodo (en bytes).

La función *gethostname()* obtiene el nombre de la máquina local.

Ejemplo de un servidor Web en C

Código de un programa en C que acepta peticiones HTTP y responde con un documento incrustado en el propio código.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <err.h>

char response[] = "HTTP/1.1 200 OK\r\n"
"Content-Type: text/html; charset=UTF-8\r\n\r\n"
"<doctype !html><html><head><title>Servidor Web</title></head>"
"<body><h1>Ejemplo de servidor Web escrito en lenguaje C, para el proyecto de
integración</h1></body></html>\r\n";

int main()
{
  int one = 1, client_fd;
  struct sockaddr_in svr_addr, cli_addr;
  socklen_t sin_len = sizeof(cli_addr);

  int sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    err(1, "No se puede abrir el socket");

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(int));

  int port = 8080;
  svr_addr.sin_family = AF_INET;
  svr_addr.sin_addr.s_addr = INADDR_ANY;
  svr_addr.sin_port = htons(port);

  if (bind(sock, (struct sockaddr *) &svr_addr, sizeof(svr_addr)) == -1) {
    close(sock);
    err(1, "No se puede hacer bind");
  }

  listen(sock, 5);
  while (1) {
    client_fd = accept(sock, (struct sockaddr *) &cli_addr, &sin_len);
```

```
    printf("Conexión obtenida\n");

    if (client_fd == -1) {
      perror("No se puede hacer accept");
      continue;
    }

    write(client_fd, response, sizeof(response) - 1); /*-1:'\0'*/
    close(client_fd);
  }
}
```

La plataforma de electrónica libre Arduino

El entorno de *Arduino*[12] ha sido diseñado para ser fácil de usar para los principiantes que no tienen experiencia en programación o la electrónica. Con *Arduino*, se pueden construir objetos que pueden responder y/o controlar luz, sonido, tacto y movimiento. *Arduino* se ha utilizado para crear una increíble variedad de aplicaciones, incluyendo instrumentos musicales, robots, juegos, muebles interactivos, e incluso la ropa interactiva.

Arduino se utiliza en muchos programas educativos en todo el mundo, sobre todo por diseñadores y artistas que quieren crear fácilmente prototipos, pero sin la necesidad de una comprensión profunda de los detalles técnicos que hay detrás de sus creaciones. Debido a que está diseñado para ser utilizado por personas no técnicas, el software incluye una gran cantidad de código de ejemplo para mostrar cómo utilizar diversas instalaciones de la placa *Arduino*.

A pesar de que es fácil de usar, la electrónica subyacente de *Arduino* trabaja en el mismo nivel de sofisticación que los ingenieros emplean para construir dispositivos integrados. La gente que ya ha trabajado con microcontroladores también se sienten atraídos por *Arduino* debido a su desarrollo ágil, capacidades y facilidad para la implementación rápida de sus proyectos.

Arduino es popular por su electrónica, pero también se necesita el software con el que ha de trabajar el hardware. Tanto el hardware como el software se llaman *"Arduino"*. El software es de código abierto y multiplataforma. Las tarjetas se pueden comprar o también se puede construir su propia placa, ya que la electrónica *Arduino* es *Hardware Libre*[41]. Además, hay una comunidad de apoyo muy activa, que es accesible en todo el mundo a través de la los foros de *Arduino* y el *wiki*[42]. Los foros y los ejemplos de desarrollo *wiki* ofrece proyectos y soluciones a los problemas que pueden surgir en durante el desarrollo de los proyectos.

41 Hardwre libre o Hardware de código abierto consta de componentes físicos de la tecnología diseñada y ofrecida por la comunidad del diseño abierto. Tanto el software libre y de código abierto, como el hardware de código abierto son creados por este movimiento de cultura de código abierto. El término por lo general significa que la información sobre el hardware se discierne fácilmente. El diseño electrónico, es decir, dibujos mecánicos, esquemas, listas de materiales, los datos de diseño de la PCB, el código fuente de HDL y los datos de diseño de circuitos integrados, además del software que maneja el hardware, están todos en libertad con el enfoque de software libre.

42 Se le llama *Wiki* a los sitios Web cuyas páginas pueden ser editadas directamente desde el navegador, donde los usuarios crean, modifican o eliminan contenidos que, generalmente, comparten.

El software de Arduino

Los programas de software, llamados *Sketch*[43], se crean en una máquina con el entorno de desarrollo integrado (IDE) de *Arduino*. El IDE permite escribir y editar código y convertir el código en instrucciones comprensibles por hardware Arduino. El IDE También transfiere esas instrucciones a la placa *Arduino* en un proceso llamado *carga*.

19 *Figura: IDE Arduino con un código de ejemplo.*

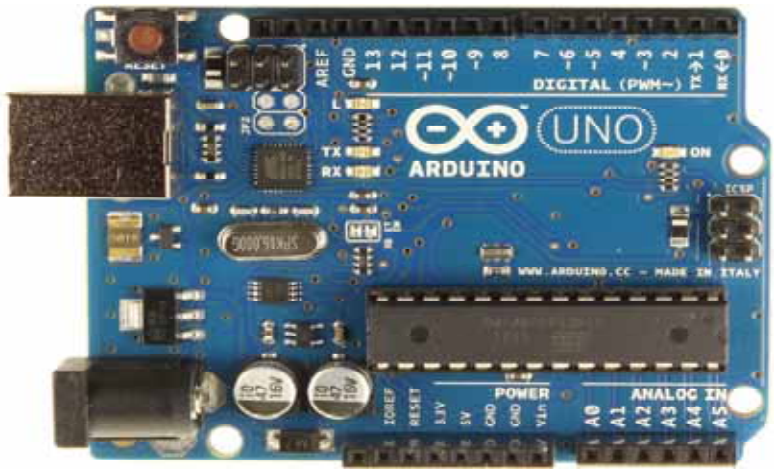
La electrónica de Arduino

La placa *Arduino* es donde se ejecuta el código compilado y cargado desde el IDE. La tarjeta sólo puede controlar y responden a la electricidad, por lo que se requiere de componentes específicos que se conectan a ella para que pueda interactuar con el mundo real. Estos componentes pueden ser sensores, que convierten algunos aspectos del mundo físico a la electricidad para que el sistema pueda sentirlo; o actuadores, que obtienen electricidad a partir de la placa y la convierten en algo que puede ser observado, escuchado o sentido en el mundo físico.

Ejemplos de sensores incluyen interruptores, acelerómetros, y sensores de distancia ultrasónicos. Los actuadores son cosas como las luces, altavoces, motores, y pantallas.

Hay una gran variedad de tarjetas oficiales que se pueden utilizar con el software de *Arduino* y una amplia gama de tarjetas compatibles producidas por los miembros de la comunidad.

43 Los *sketch* son los programas que se escriben en la herramienta de Arduino. Un *sketch*, puede tener varios documentos de código fuente abiertos.



20 Figura: Placa de prototipado Arduino UNO.

Las tarjetas más populares contienen un conector USB[44] que se utiliza para proporcionar energía y conectividad para cargar el código compilado en la placa.

Programación Arduino basada en el Lenguaje C

La estructura básica un programa en Arduino es bastante simple y se compone de al menos dos partes. Estas partes son necesarias y engloban bloques que contienen declaraciones, sentencias o instrucciones.

```
void setup()
{
      sentencias;
}
void loop()
{
      sentencias;
}
```

En donde *setup()* es la parte encargada de establecer la configuración y *loop()* es la que contienen el programa que se ejecutará en un ciclo infinito (similar a la forma en la que trabaja OpenGL[45]). Ambas funciones son necesarias para que el programa funcione.

La función de configuración debe contener la declaración de las variables. Es la primera función a ejecutarse en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar *pinMode()* (modo de trabajo de las entradas y salidas de la tarjeta), configuración de la comunicación en *serial* y otras.

La función *loop()* contiene el código que se ejecutara continuamente (lectura de entradas, activación de salidas, etc) en una especie de *poleo*[46]. Esta función es el núcleo de todos los programas de Arduino y la que

44 La comunicación realizada a través del puerto USB con la tarjeta de *Arduino*, en realidad trabaja como la comunicación *serial* por el puerto DB9. La ventaja es la comodidad y flexibilidad en el cable.

45 OpenGL (*Open Graphics Library)* es una interfaz de programación de aplicaciones multiplataforma (API) para la representación en 2D y 3D de gráficos vectoriales. El API se utiliza típicamente para interactuar con una unidad de procesamiento de gráficos (GPU), para lograr la representación acelerada por hardware. Fue desarrollado por *Silicon Graphics Inc.* en 1991 y liberado en enero de 1992. Se utiliza ampliamente en CAD, realidad virtual, visualización científica, visualización de información, simulación de vuelo, y en juegos de video.

46 Se le llama *poleo* al proceso de tener un ciclo que esté al pendiente de un evento.

realiza la mayor parte del trabajo.

Tanto la declaración y definición de variables, tipos y funciones, estructuras de control, estructuras de datos, aritmética, operaciones lógica, etc; se realizan exactamente igual que en cualquier otro programa escrito con lenguaje C, ya que *Arduino* se basa en éste ocultando detalles como las macros que incluyen bibliotecas estándar y bibliotecas propias de la plataforma

Estructuras estándar de Arduino	Variables de Arduino	Funciones
Estructuras: • *setup()* • *loop()* Estructuras de control: • *if, if...else* • *for* • *switch case* • *while* • *do... while* • *break* • *continue* • *return* • *goto* Sintáxis adicional: • *; (punto y coma)* • *{} (llaves)* • *// (comentar una línea)* • */* */ (comentarios)* • *#define* • *#include* Operadores aritméticos: • *= (asignación)* • *+ (suma)* • *- (resta)* • ** (multiplicación)* • */ (división)*	Constantes: • *HIGH | LOW* • *INPUT | OUTPUT | INPUT_PULLUP* • *LED_BUILTIN* • *true | false* • *Constantes enteras* • *Constantes de punto flotante* Tipos de datos: • *void* • *boolean* • *char* • *unsigned char* • *byte* • *int* • *unsigned int* • *word* • *long* • *unsigned long* • *short* • *float* • *double* • *string - char array* • *String - object* • *array*	Entradas y salidas digitales: • *pinMode()* • *digitalWrite()* • *digitalRead()* Analog I/O: • *analogReference()* • *analogRead()* • *analogWrite() - PWM* Sólo la versión *Due*: • *analogReadResolution()* • *analogWriteResolution()* Entradas y salidas avanzadas: • *tone()* • *noTone()* • *shiftOut()* • *shiftIn()* • *pulseIn()* Funciones de tiempo: • *millis()* • *micros()* • *delay()* • *delayMicroseconds()* Funciones matemáticas: • *min()* • *max()* • *abs()*

- *% (modulo)*

Operadores de comparación:

- *== (igualdad)*
- *!= (desigualdad)*
- *< (menor que)*
- *> (mayor que)*
- *<= (menor o igual que)*
- *>= (mayor o iguel que)*

Operadores binarios:

- *&& (conjunción)*
- *|| (disyunción)*
- *! (negación)*

Operadores de acceso a apuntadores:

- ** operador para deshacer referencias*
- *& operador para referenciar*

Operadores *bit a bit*:

- *& (conjunción bit a bit)*
- *| (disyunción bit a bit)*
- *^ (disyunción exclusiva bit a bit)*
- *~ (negación bit a bit)*
- *<< (corrimiento a la izquierda bit a bit)*
- *>> (corrimiento a la derecha bit a bit)*

Operadores compuestos:

- *++ (incremento)*
- *-- (decremento)*
- *+= (suma compuesta)*
- *-= (resta compuesta)*
- **= (multiplicación compuesta)*
- */= (división compuesta)*

Conversión de tipos:

- *char()*
- *byte()*
- *int()*
- *word()*
- *long()*
- *float()*

Alcance de las variables:

- *variable scope*
- *static*
- *volatile*
- *const*

Utilidades:

- *sizeof()*

- *constrain()*
- *map()*
- *pow()*
- *sqrt()*

Funciones Trigonometricas:

- *sin()*
- *cos()*
- *tan()*

Números aleatorios:

- *randomSeed()*
- *random()*

Funciones con Bits y Bytes:

- *lowByte()*
- *highByte()*
- *bitRead()*
- *bitWrite()*
- *bitSet()*
- *bitClear()*
- *bit()*

Interrupciones externas:

- *attachInterrupt()*
- *detachInterrupt()*

Interrupciones:

- *interrupts()*
- *noInterrupts()*

Funciones de comunicación:

- *Serial*
- *Stream*

USB (sólo las versiones *Leonardo* y *Due*)

- *Keyboard*
- *Mouse*

• *&= (conjunción compuesta bit a bit)* • *|= (disyunción compuesta bit a bit)*		

Bibliotecas de Arduino

El entorno Arduino se puede ampliar a través del uso de bibliotecas, al igual que la mayoría de las plataformas de programación. Las bibliotecas proporcionan funcionalidad adicional para uso en los *sketch*, por ejemplo, trabajar con hardware o la manipulación de datos. Un cierto número de bibliotecas ya vienen instaladas con el IDE, sin embargo se pueden descargar o crear bibliotecas propias.

Bibliotecas estándar

- *EEPROM.h:* Leer y escribir en el almacenamiento "permanente" (memoria *ROM*).
- *Ethernet.h:* Para conectar a Internet utilizando el *Arduino Ethernet Shield*.
- *Firmata.h:* Para la comunicación con las aplicaciones en el ordenador utilizando un protocolo estándar de serie.
- *GSM.h:* Para la conexión a una red GRPS/GSM con el *Shield* GSM.
- *LiquidCrystal.h:* Para el control de pantallas de cristal líquido (LCD).
- *SD.h:* Para la lectura y escritura de tarjetas SD .
- *Servo.h:* Para el control de servomotores.
- *SPI.h:* Para comunicarse con los dispositivos que utilizan la interfaz periférica serial (SPI).
- *SoftwareSerial.h:* Para la comunicación serial en cualquier *pin* digital.
- *Stepper.h:* Para el control de motores paso a paso.
- *TFT.h:* Para dibujar texto, imágenes, y las formas en la pantalla TFT de Arduino.
- *WiFi.h:* Para conectar a Internet utilizando el escudo Arduino WiFi.
- *Wire.h:* Dos cables de interfaz (TWI/I2C) para enviar y recibir datos a través de una red de dispositivos o sensores.

Bibliotecas de la comunidad

Comunicación (creación de redes y protocolos):

- *Messenger:* Para el procesamiento de mensajes de texto desde la computadora.
- *NewSoftSerial:* Una versión mejorada de la biblioteca *SoftwareSerial*.
- *OneWire:* Dispositivos de control que utilizan el protocolo *OneWire*.
- *PS2Keyboard*: Leer caracteres de un teclado PS2[47].

47 El conector PS/2 o puerto PS/2 debe su nombre a la serie de máquinas *IBM Personal System/2* que es creada por IBM en 1987, y empleada para conectar teclados y ratones. Muchos de los adelantos presentados fueron inmediatamente adoptados por el mercado del PC, siendo este conector uno de los primeros.

- *SimpleMessageSystem*: Enviar mensajes entre Arduino y la máquina.
- *SSerial2Mobile*: Enviar mensajes de texto o mensajes de correo electrónico utilizando un teléfono celular.
- *Webduino*: Biblioteca de servidor web extensible (para su uso con el s*hield* Arduino Ethernet).
- *X10*: El envío de señales X10 a través de líneas de alimentación de corriente alterna.
- *XBee*: Para comunicarse con XBees[48] en modo API .
- *SerialControl*: Control a distancia de otros Arduinos a través de una conexión en serie.

Sensores:

- *CapacitiveSensing*: Dos o más patas simultáneas en sensores capacitivos.
- *Debounce*: Para la lectura de las entradas digitales ruidosas, por ejemplo: botones.

Pantallas y LEDs:

- *GFX*: Clase base con rutinas de gráficos estándar.
- *GLCD*: Rutinas para controlar gráficas para LCD basados en el *chipset KS0108* o equivalente.
- *Improved LCD library*: Correcciones para la biblioteca estándar *LCD* de Arduino.
- *LedControl*: Para el control de matrices de LED o pantallas de siete segmentos con un *MAX7221* o *MAX7219*.
- *LedControl*: Una alternativa a la biblioteca *Matrix* para el manejo de múltiples LED con chips Maxim.
- *LedDisplay*: Control de un desplazamiento HCMS-29xx de una pantalla LED.
- *Matrix*: Biblioteca para la manipulación de pantallas de Matriz Básica LED.
- *PCD8544*: Para el controlador de pantallas LCD de Nokia 55100.
- *Sprite*: Biblioteca básica para la manipulación de imagen para usar en animaciones con una matriz de LED.
- *ST7735*: Para el controlador de pantalla de 1.8 pulgadas TFT de 128x160.

Audio y formas de onda:

- *FFT*: Análisis de frecuencia de señales analógicas o de audio.
- *Tone*: Genera ondas cuadradas en segundo plano en cualquier *pin* del microcontrolador.

Motores y PWM:

- *TLC5940*: 16 canales para controlar *PWM*[49] de 12 bits.

Tiempo:

48 XBee es el nombre comercial de *Digi International* para una familia de módulos de radio compatible con el factor de forma . Las primeras radios XBee se introdujeron bajo la marca *MaxStream* en 2005 y se basan en el estándar IEEE 802.15.4-2003 diseñado para comunicaciones punto a punto y en estrellas e inalámbricas, con velocidades de transmisión de 250 Kb/s.

49 La modulación por ancho de pulsos (*pulse width modulation*) de una señal o fuente de energía, es la técnica de modificar el ciclo de trabajo de una señal periódica, ya sea para transmitir datos o para controlar la cantidad de energía que se envía a una carga.

- *DateTime*: Una biblioteca para hacer el seguimiento de la fecha y la hora actuales en el software.
- *Metro*: Ayuda tener acciones en el tiempo a intervalos regulares.
- *MsTimer2*: Utiliza la interrupción del temporizador 2 para desencadenar una acción cada *n* milisegundos.

Utilidades:

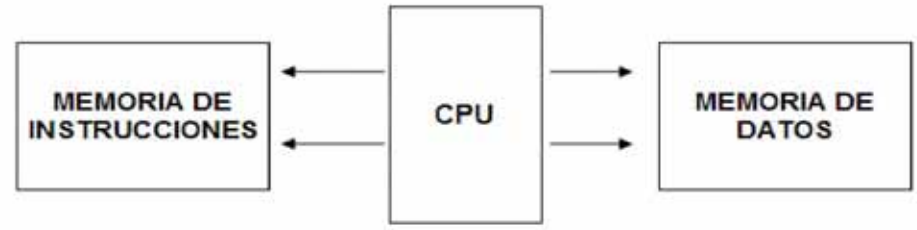
- *PString*: Una clase para la impresión en búfers.
- *Streaming*: Un método para simplificar las sentencias de impresión.

Programación de microcontroladores AVR con lenguaje C

El núcleo AVR[13] es una arquitectura avanzada RISC[50] adecuada para el código C. Cuando se habla de optimización, por lo general se refiere a dos aspectos: el tamaño del código y velocidad de código. Hoy en día, los compiladores de C tienen diferentes opciones de optimización para ayudar a los desarrolladores a obtener un código eficiente a cierto tamaño o velocidad. Sin embargo, la buena codificación C da más oportunidades a los compiladores para optimizar el código como se desee. Y en algunos casos, la optimización de uno de los dos aspectos afecta, o incluso provoca la degradación en la otra, por lo que un desarrollador tiene que equilibrar los dos de acuerdo con sus necesidades. El conocer algunos consejos y trucos sobre la codificación de C para un AVR de 8 bits, ayuda a los desarrolladores a saber como mejorar la eficiencia del código.

Arquitectura AVR

Los AVR utilizan arquitectura Harvard[51], con memoria y buses separados para el programa y los datos.



21 Figura: Esquema de la arquitectura Harvard.

Tiene un archivo de registro de acceso rápido de 32 × 8 bits con registros de propósito general con un tiempo de acceso de un solo ciclo de reloj (debido a su arquitectura *RISC*). Los 32 registros de trabajo son una de las claves para la codificación eficiente en C. Estos registros tienen la misma función que el

50 Cómputo con un conjunto de instrucciones reducido, o RISC, es una arquitectura de la CPU que se basa en la idea de que simplificar un conjunto de instrucciones proporciona un rendimiento superior cuando se combina con una arquitectura de microprocesador capaz de ejecutar dichas instrucciones usando menos ciclos por instrucción del microprocesador. Un equipo basado en esta arquitectura es una máquina con un conjunto de instrucciones reducido, también llamado RISC. La arquitectura de oposición se llama *Cómputo con un conjunto de instrucciones complejo*, es decir CISC .

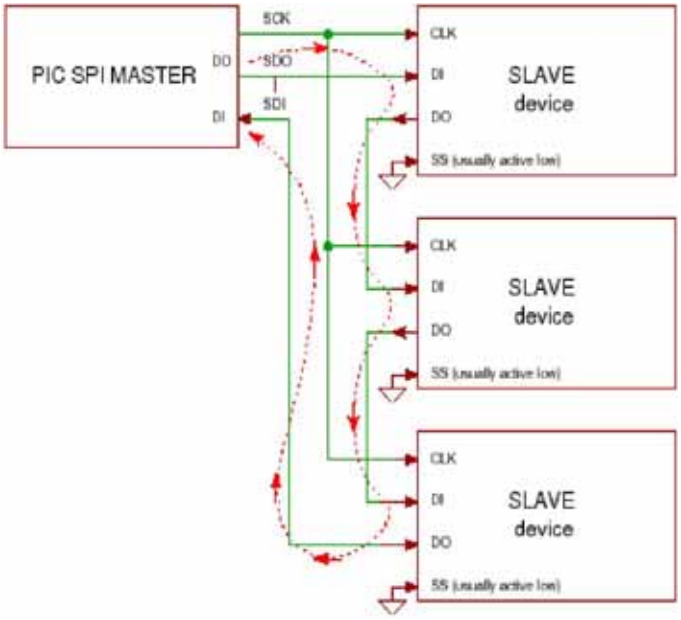
51 La arquitectura Harvard es una arquitectura de computadora con buses de almacenamiento y de señal físicamente separados para instrucciones y datos. El término se originó a partir de la computadora *Harvard Mark I* basada en relés, que almacenaba instrucciones sobre cinta perforada con 24 bits de ancho y los datos de los contadores electromecánicos. Estas primeras máquinas tenían almacenamiento de datos totalmente contenida dentro de la unidad central de procesamiento, y no proporcionan acceso al almacenamiento de instrucciones como datos.

tradicional acumulador *AX*[52], excepto que hay 32 de ellos. Las instrucciones aritméticas y lógicas del AVR trabajan en éstos registros, por lo que, ocupan menos espacio para cada instrucción. En un ciclo de reloj, el AVR puede alimentar a dos registros arbitrarios desde el archivo de registro de la ALU, realizar una operación, y escribir de nuevo el resultado al archivo de registro.

Las instrucciones en la memoria de programa se ejecutan con un solo nivel de *pipelining*[53]. mientras se está ejecutando una instrucción, la instrucción siguiente se comienza a cargar del programa en la memoria, lo que permite que mas instrucciones se ejecuten en cada ciclo de reloj. Las instrucciones AVR tienen un único formato de palabra de 16 bits. Cada dirección de memoria del programa contiene una instrucción de 16 o 32 bits.

El protocolo SPI

SPI es un estándar para comunicaciones utilizado para comunicar periféricos en un sistema electrónico. Se compone de un bus de tres líneas, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas tres líneas porta la información entre los diferentes dispositivos conectados al bus. Cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo, por lo que este tipo de comunicación serial es *full duplex*[54]. Dos de estas líneas transfieren los datos (una en cada dirección) y la tercer línea es la del reloj, usada para la sincronización. Algunos dispositivos solo pueden ser transmisores y otros solo receptores, generalmente un dispositivo que tramite datos también puede recibir.



22 Figura: Protocolo de comunicación entre periféricos SPI.

Un ejemplo podría ser un memoria EEPROM, el cual es un dispositivo que puede transmitir y recibir información. Los dispositivos conectados al bus son definidos como maestros y esclavos. Un maestro es

52 El registro acumulador *AX*, se utiliza en los microprocesadores basados en la arquitectura del 8086 de Intel, y se usa para propósitos generales en su programación.

53 Pipelining o entubado, es un conjunto de elementos de procesamiento de datos conectados en serie, donde la salida de un elemento es la entrada del siguiente. Los elementos de una tubería a menudo se ejecutan en paralelo o en forma de tiempo de rodajas; en ese caso, se inserta a menudo cierta cantidad de almacenamiento intermedio entre los elementos .

54 Cuando los datos circulan en ambas direcciones a la vez,la transmisión se denomina *full-duplex*, a diferencia de *half-duplex*, en donde se turna la comunicación en el medio. A pesar de que los datos circulan en ambas direcciones, el ancho de banda se mide en una sola dirección..

aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y control. Un esclavo es un dispositivo controlado por el maestro. Cada esclavo es controlado sobre el bus a través de una línea selectora llamada *Chip Select* o *Select Slave*, por lo tanto es esclavo es activado solo cuando esta línea es seleccionada. Generalmente una línea de selección es dedicada para cada esclavo. En un tiempo determinado *T1*, solo podrá existir un maestro sobre el bus. Cualquier dispositivo esclavo que no este seleccionado, debe deshabilitarse (ponerlo en alta impedancia) a través de la línea selectora (*chip select*).

Compilación con GCC para AVR

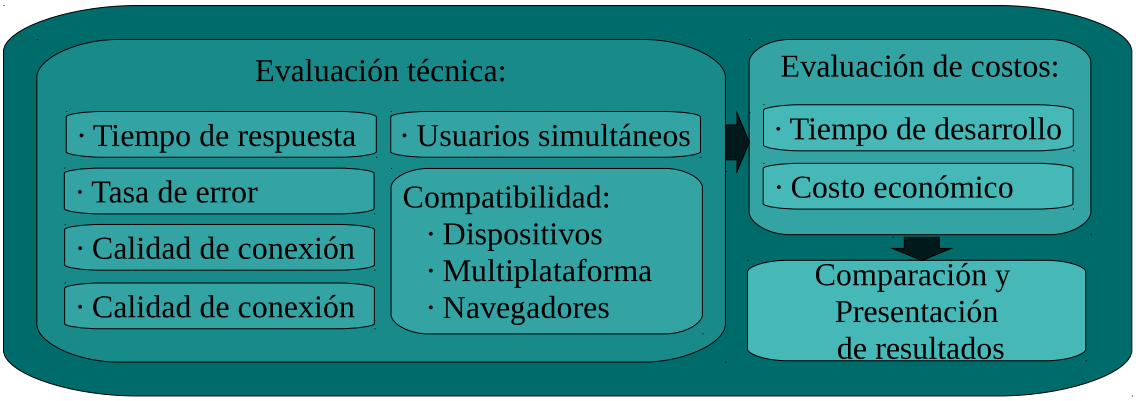
GCC es la colección de compiladores por excelencia en sistemas tipo *UNIX* tales como *Linux*. Cuando GCC se utiliza para el objetivo AVR, se conoce comúnmente como AVR GCC (*avr-gcc*). AVR GCC proporciona varios niveles de optimización: -O0, -O1, -O2, -O3 y -Os. En cada nivel, hay diferentes opciones de optimización habilitadas, a excepción de -O0 que significa que no hay optimización. Además de las opciones habilitadas en niveles de optimización, también puede habilitar opciones de optimización por separado para conseguir una optimización específica.

El programa "avr-gcc" toma muchas otras herramientas de trabajo en conjunto para producir el ejecutable de la aplicación final para el microcontrolador AVR. En este grupo de herramientas AVR, *avr-libc* sirve como una biblioteca importante de C, que ofrece muchas de las mismas funciones que se encuentran en la biblioteca estándar de C, y muchas son específicas de un AVR. El paquete *AVR Libc* proporciona un subconjunto de la biblioteca estándar de C para microcontroladores RISC *Atmel AVR* de 8 bits. Además, la biblioteca proporciona el código de inicio básica necesaria por la mayoría de las aplicaciones.

Descripción técnica

El proceso de análisis se plantea como un flujo que evoluciona a través de dos etapas.

23 *Figura: Etapa de desarrollo.*



24 *Figura: Etapa de evaluación.*

Etapa de desarrollo.

Se debe construir un dispositivo basado en un microcontrolador, que sea capaz de comunicarse a través de la red. Para ello se debe diseñar el circuito con sus respectivos componentes (ver figura 3).

25 *Figura: Esquema del dispositivo hecho a la medida.*

En la figura 4, se ilustra la arquitectura de una placa Arduino con un módulo para comunicación de red, donde todos los componentes están construidos y listos para ser usados.

26 Figura: Esquema de la plataforma Arduino con un módulo de red.

Una vez teniendo las dos plataformas preparadas y funcionando, se hace uso de una biblioteca (que ya ha sido desarrollada por la comunidad del software libre), para lograr la comunicación de los dispositivos por medio de la red.

En ésta etapa se realiza el desarrollo de una aplicación que pueda mostrar y controlar las actividades de los dispositivos a través de una interfaz web. Para ello se escribe un programa en los microcontroladores que aproveche las funciones de la biblioteca utilizada y del hardware disponible. Dicho programa deberá conocer el estado de la actividad que está realizando el sistema para poder controlarlo y reportarlo a través de la red.

Etapa de evaluación.

Para conocer el desempeño de cada uno de los sistemas, se debe realizar una serie de pruebas, las cuales medirán el tiempo en el que tardan en responder a una instrucción o comando del usuario, la cantidad de errores obtenidos en un cierto número de pruebas, la calidad de la conexión, la capacidad multiusuario y la compatibilidad con otros dispositivos, sistemas operativos y navegadores.

Los costos económicos y el tiempo real que llevó implementar cada sistema, también será información a evaluar para cada sistema.

Finalmente, con la información obtenida, se reportan los resultados en tablas comparativas.

Especificación técnica

Etapa de desarrollo

En la etapa de desarrollo, se trabajará con elementos de hardware y software.

Hardware:

- Se usará una tarjeta de desarrollo basada en la plataforma de electrónica abierta Arduino, la cuál posee una interfaz USB (Universal Serial Bus) para su programación y alimentación, y un módulo de comunicación de red (Ethernet Shield[55]).
- El módulo de red de Arduino se basa a su vez, en el circuito integrado Wiznet W5100, el cual provee una pila de red IP con los protocolos TCP y UDP implementados en hardware.
- El dispositivo construido a la medida, trabajará con un circuito integrado ENC28J60, que a diferencia del Wiznet W5100, no implementa la pila de protocolos TCP y UDP dentro de su arquitectura, sin embargo, permite controlar la comunicación por medio del puerto de red (Ethernet) a través de la programación de dicha pila en el microcontrolador.
- Ambos dispositivos trabajarán con el microcontrolador ATMega328, el cuál ofrece un conjunto reducido de instrucciones (RISC) de 8 bits, trabajando a una frecuencia de reloj de 16 MHz.
- En el caso del dispositivo hecho a la medida, será necesario para la comunicación entre el microcontrolador y el circuito integrado ENC28J60, utilizar el protocolo de comunicación entre periféricos SPI (Serial Peripheral Interface).

Software:

- La programación de los dispositivos se llevará a cabo con el lenguaje de programación C.
- En la aplicación de control y monitoreo, se incrustará código HTML, que será enviado a los navegadores para la ejecución de la interfaz web.

La etapa de evaluación:

- Se realizarán pruebas técnicas.[56]
    1) Se ejecutarán 32[57] instrucciones desde una computadora personal, y desde un teléfono celular con capacidad de navegación web, para cada uno de los siguientes puntos:
        a) Para el caso de la computadora personal las pruebas se harán bajo los sistemas operativos:
            i. Linux Debian 6.x, con los navegadores:

55 Componente de comunicación de red alámbrica bajo la arquitectura de electrónica libre Arduino.
56 Para las pruebas de tiempo de respuesta y de errores, se hará uso de Javascript debido a la naturaleza de la interfaz.
57 En estadística se considera que el tamaño de una muestra es lo suficientemente grande cuando es mayor a 30.

- Firefox 27.x.
- Chromuim 18.x.

ii. Windows 8, con los navegadores:

- Internet Explorer 9.x.
- Firefox 27.x.
- Chromuim 18.x.

b) Para el caso del teléfono celular, se harán pruebas en:

i. Un teléfono LG 360i con los navegadores:

- Opera mini.
- Navegador nativo del teléfono.

2) Se ejecutará un comando “Ping” recurrente con la ayuda de la terminal de Linux Debian 6.x, a cada dispositivo durante 1 minuto.

3) Se ejecutará la misma instrucción de manera simultánea desde 32 instancias distintas de la interfaz web, bajo el navegador Chromium 18.x en Linux Debian 6.x.

Nota importante: En caso de que una instrucción o comando no responda, la prueba se considerará como fallida al pasar un minuto desde el inicio de su ejecución[58].

- Y se evaluarán los costos:

1) Se registrarán los gastos totales en pesos mexicanos invertidos para la implementación de cada dispositivo, los cuáles no deberán exceder los $1,000.00[59] pesos mexicanos en ninguno de los dos casos.

2) Se registrará el tiempo real en horas, que se utilizó para la implementación de cada dispositivo.

Finalmente se reportará toda la información organizada en tablas comparativas.

58 La restricción de tiempo acota el alcance de la prueba a un resultado aceptable.
59 Exceder $1,000.00 pesos mexicanos en una aplicación de esta naturaleza, la haría una solución no factible y costosa.

Desarrollo del proyecto

De a cuerdo al programa inicial para este proyecto, el desarrollo se llevó a cabo en dos etapas, la primera consistió en diseñar e implementar la interfaz Web en ambos sistemas, para luego en la etapa de evaluación, obtener datos acerca de la comparación de su desempeño y costos.

Para la etapa de desarrollo, se separó el trabajo en dos partes, la electrónica y la programación.

Etapa de desarrollo

Para el caso del dispositivo basado en Arduino, no se necesita de un gran trabajo de diseño e implementación electrónica, debido a que la plataforma proporciona todas las herramientas necesarias en lo que a la parte física se refiere.

Sin embargo, para el caso del dispositivo hecho a la medida, se necesita realizar un diseño completo y la construcción del mismo.

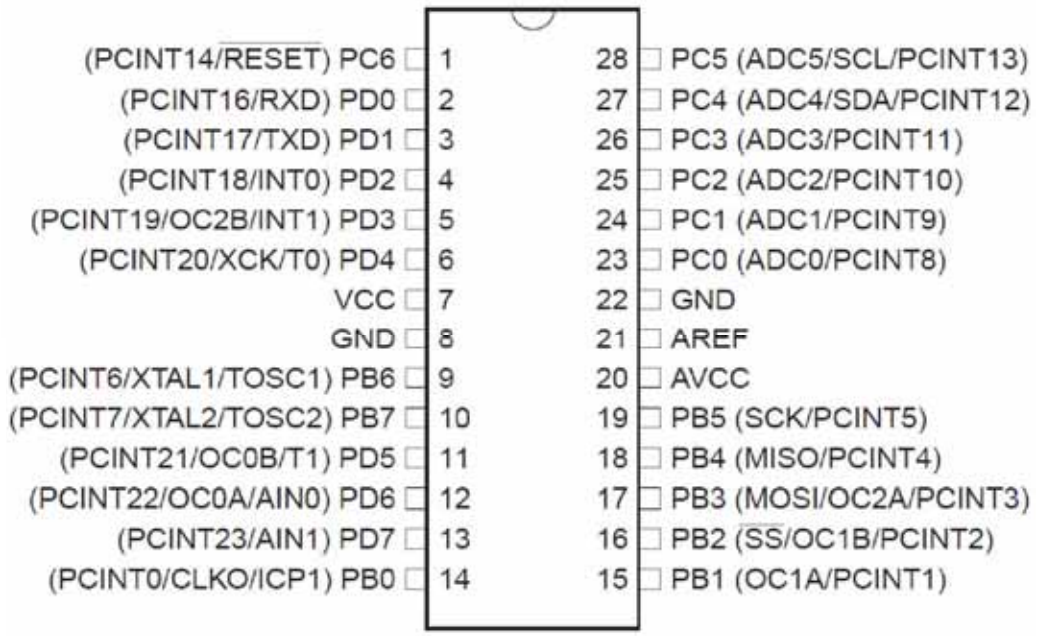
La programación es la misma en ambos casos, los dos sistemas trabajan con el mismo microcontrolador, y deberán servir exactamente la misma interfaz para el navegador.

La parte electrónica

Para el dispositivo basado en Arduino, se tienen dos elementos: La placa Arduino y el *Etherner Shield* para la comunicación por red. Sus conexiones son intuitivas y estándar, esto ayuda a que no exista ningún problema en su ensamble.

Sistema hecho a la medida

Para su construcción se usó el microcontrolador *Amega328-P*, el cual pertenece a la familia de microcontroladores AVR de la empresa Atmel. El *Amega328-P* cuenta con 32 KB de memoria para el programa (ISP Flash), 1 KB de memoria para datos (EEPROM) y 2KB de memoria RAM (SRAM). Cuenta con 32 registros de propósito general. Puede correr a una velocidad de hasta 20 Mhz, con un cristal adecuado. Además cuenta con 14 puertos para uso general, 2 puertos para transmisión *serial*, y 5 puerto para leer o escribir señales analógicas.



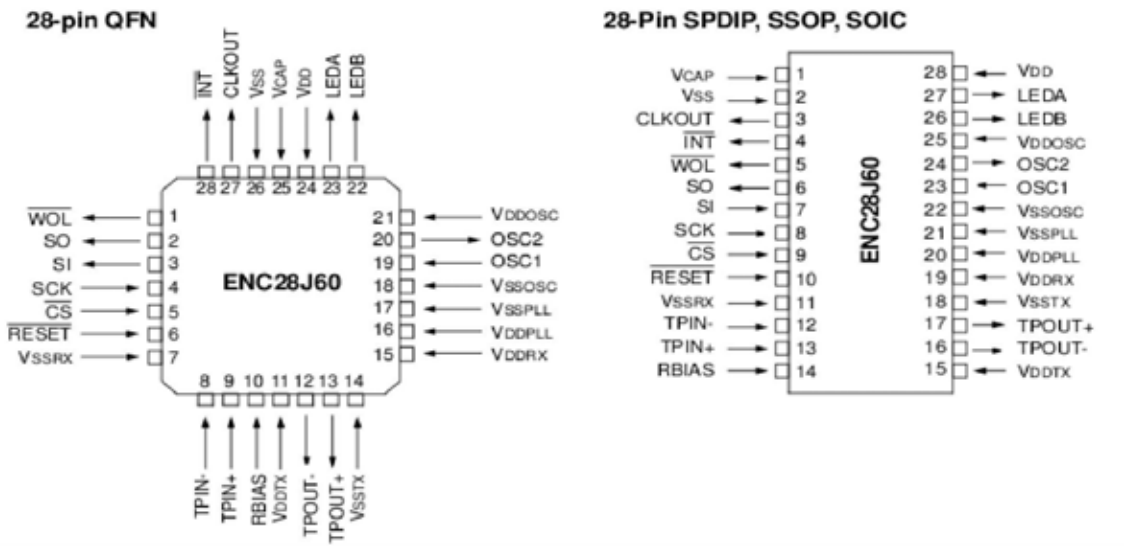
27 Figura: Esquema de pines del microcontrolador Atmega328P en su encapsulado DIP.

De manera interna cuenta con una USART, convertidores Analógico-Digital y Digital-Analógico. Soporta comunicación con otros dispositivos a través del protocolo de comunicación entre periféricos SPI.

Para su ensamble, requiere de un cristal (en éste caso de 16 Mhz) con sus respectivos condensadores de 33 nf aterrizados para eliminar ruido en las señales. Su alimentación, debe ser en este caso de 5 voltios. De esta manera tenemos el sistema mínimo para que pueda funcionar.

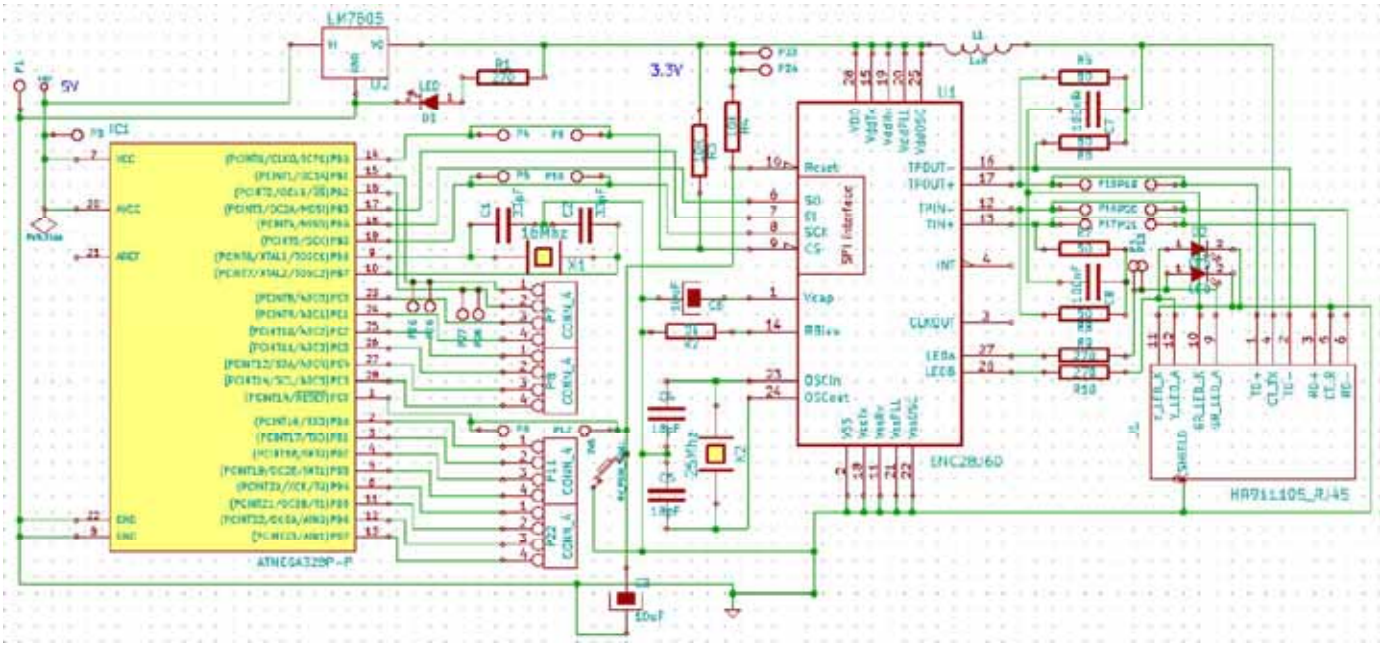
Para la comunicación de red se usó el controlador de red *ENC28J60*, el cuál incorpora de manera electrónica una implementación de la pila de protocolos TCP/IP, la cuál puede ser explotada para desarrollar aplicaciones capaces de comunicarse a través de la red.

El *ENC28J60* necesita para funcionar, un cristal de 25 Mhz con sus respectivos condensadores de 18 nf, su alimentación es de 3.3 voltios. Un condensador de 10 uf, ayudará a regular el voltaje de referencia para las entradas y salidas analógicas. Se recomienda una resistencia de Bias, de 2.8 KOhms. Además de los componentes demás componentes pasivos como resistencias inductores y condensadores para conectar el conector RJ45.



28 *Figura: Esquema de pines del microcontrolador ENC28J60 en sus diferentes encapsulados.*

En la siguiente figura se ilustra el diseño del diagrama eléctrico del prototipo.



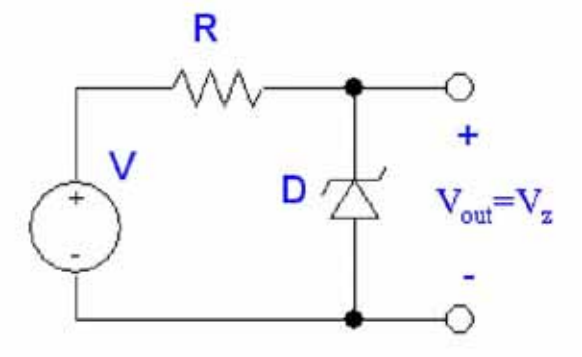
29 *Figura: Diagrama eléctrico del prototipo del dispositivo hecho a la medida.*

En el diagrama se observan las conexiones con otros elementos tales como el conector RJ45, los puertos de entrada y salida, y los indicadores luminosos.

Algo importante que se debe mencionar es que el diseño se ha hecho para que la placa tenga una alimentación de 5 voltios, sin embargo, el controlador de red *ENC22J60* trabaja con 3.3 voltios, para ello se utilizó un regulador de voltaje.

30 Figura: Regulador de voltaje a 3.3 voltios (LM7833).

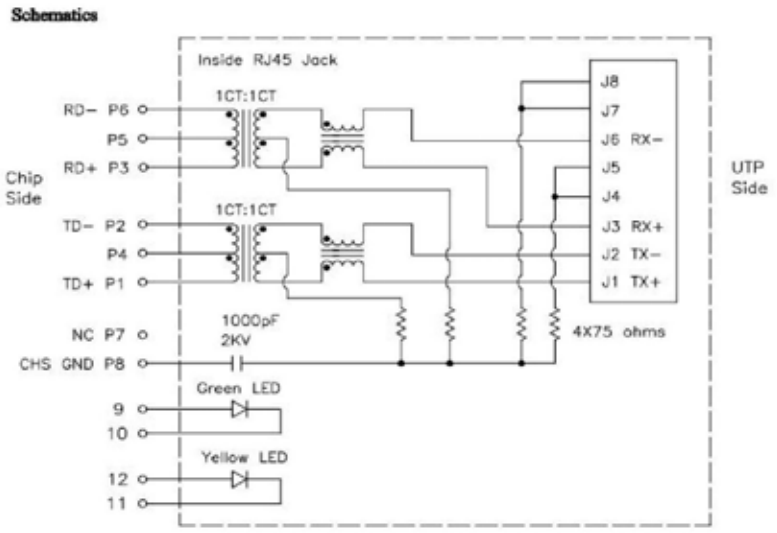
La interfaz entre el *ENC22J60* y el *Atmega328-P* también es un detalle importante que se debe explicar, ya que los niveles de voltaje en las señales que transmiten y reciben son distinta, sin embargo, el controlador *ENC22J60* ha sido diseñado para soportar señales de hasta 5 voltios, y en el caso del microcontrolador *Atmega328-P*, éste interpreta los niveles de 3.3 voltios como un *"uno lógico"*, por lo que se omitió la circuitería considerada en un principio, la cuál constaba en un *diodo Zener* y una resistencia en configuración para ajustar los voltajes.



31 Figura: Configuración de Zener para regular voltajes.

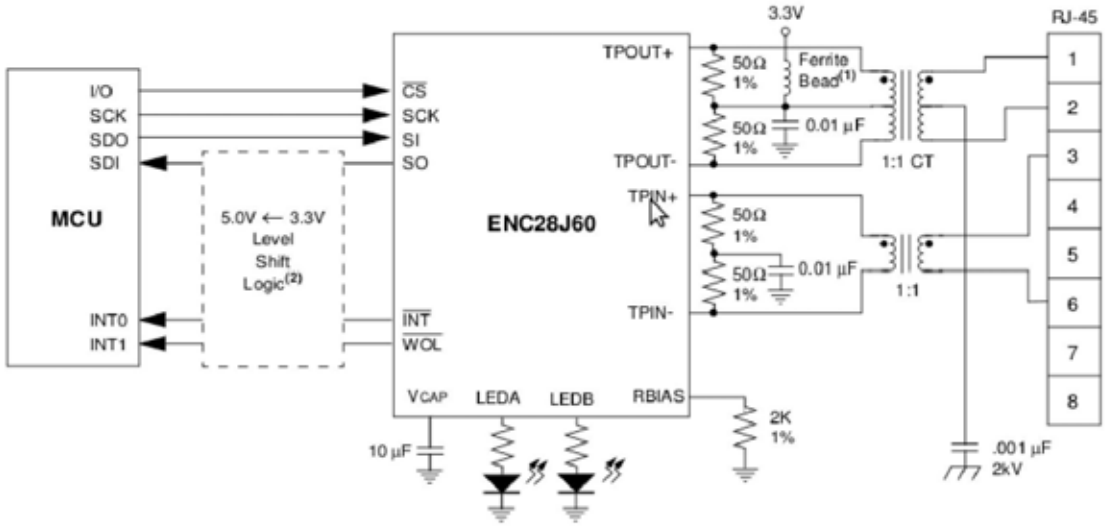
La comunicación entre el controlador de red *ENC22J60* y el microcontrolador *Atmega328-P*, se realiza apoyándose en el protocolo SPI. Esto evita el tener que diseñar un protocolo de comunicación entre los dos microcontroladores, haciendo más eficiente aún la comunicación entre ellos.

La comunicación SPI para controlador *ENC22J60* fué diseñada en un principio para trabajar con los microcontroladores de Microchip: los llamados PICs. Sin enmbargo, se ha demostrado con la experiencia que trabaja muy bien con microcontroladores de otras familias y de otras empresas incluso.



32 *Figura: Esquema del conector de red HR911105 con filtros, resistencias, inductores y LEDs internos.*

Para tener acceso a la red de manera física, se usó un conector RJ45 con filtros y leds incluidos, esto para facilitar la conexión y ahorrar espacio y componentes. Para su ensamble con el *ENC22J60*, sólo se ocuparon 4 resistencias, dos condensadores y un inductor.

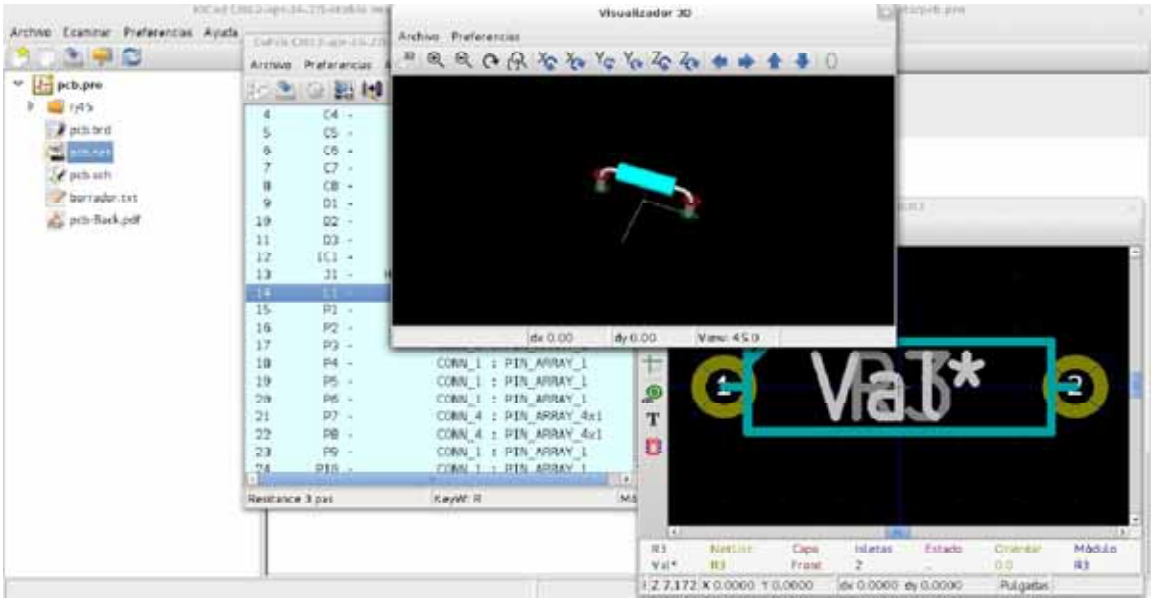


33 *Figura: Modelo de interfaz del Atmega328P con el ENC28J60 a través del protocolo SPI, y el conector RJ45 convencional.*

Teniendo bien definido el diseño del dispositivo, se utilizó la herramienta de software libre *Kicad*, la cual facilita el diseño e impresión de circuitos electrónicos en placas de PCB para ser soldadas.

Con el diseño en la herramienta *Kicad*, cada elemento del circuito fue asociado con un componente físico para así, poder diseñar el circuito impreso.

Los componentes se eligieron de acuerdo a su geometría y características eléctricas.



34 Figura: Asociación de componentes con Kicad.

Durante este proceso surgió un problema. La herramienta de *Kicad* no tiene incluida las especificaciones del conector RJ45 *HR911105*, por lo que se procedió a diseñarla para poder tener el objeto para asociarlo al componente y tener en el diseño de la PCB de manera adecuada.

Para construir el componente se escribió una biblioteca:

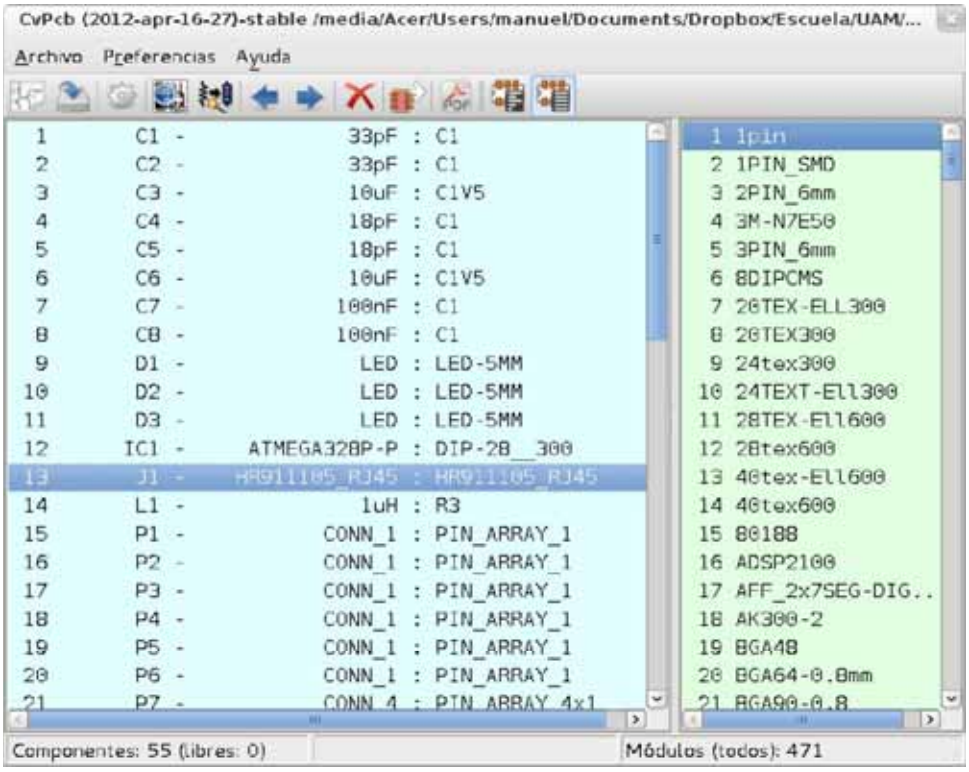
```
EESchema-LIBRARY Version 2.3  Date: 15/07/2012 19:22:14
#encoding utf-8
#
# HR911105_RJ45
#
DEF HR911105_RJ45 J 0 40 Y Y 1 F N
F0 "J" 150 650 60 H V C CNN
F1 "HR911105_RJ45" 550 150 60 V V C CNN
DRAW
...
...
...
ENDDEF
#
#End Library
```

Y también se escribió su archivo de módulo con las características geométricas y eléctricas:

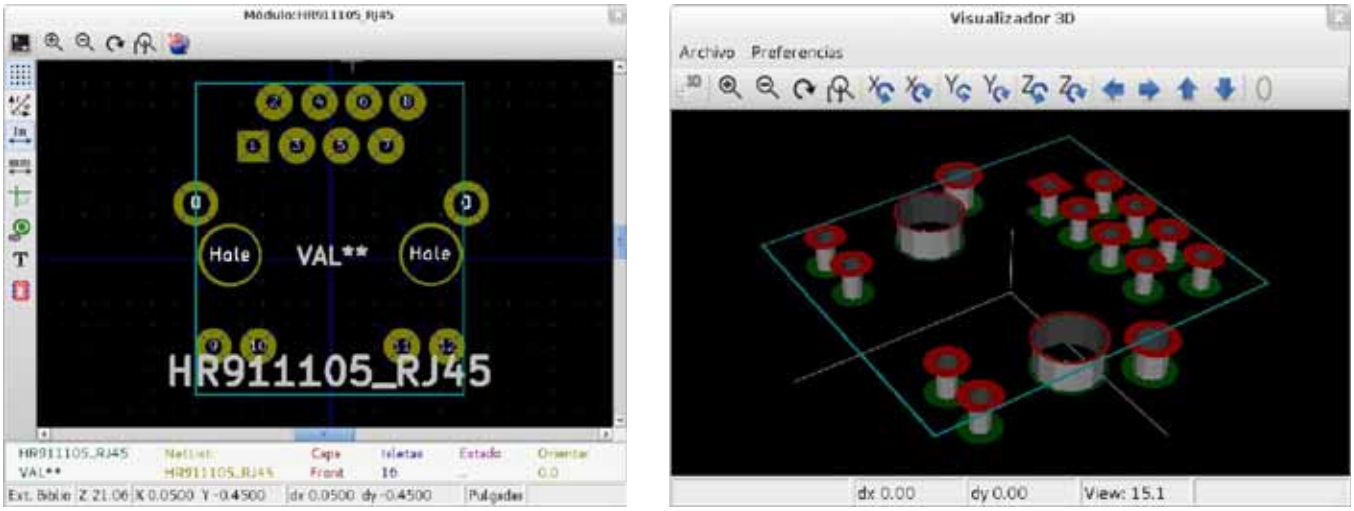
```
PCBNEW-LibModule-V1  dom 08 jun 2014 17:25:52 CDT
# encoding utf-8
$INDEX
HR911105_RJ45
```

```
$EndINDEX
$MODULE HR911105_RJ45
...
...
...
DS 3000 -4000 -3000 -4000 50 21
DS -3000 -4000 -3000 3100 50 21
$PAD
Sh "Hole" C 1417 1417 0 0 0
Dr 1280 0 0
...
...
...
At STD N 00F0FFFF
Ne 0 ""
Po 3051 -1260
$EndPAD
$EndMODULE  HR911105_RJ45
$EndLIBRARY
```

De modo que en el diseño, ya se tuvo el componente listo para incluirse en el proyecto:



35 *Figura: Asociación de componentes con Kicad, se incluye el componente HR911105_RJ45.*



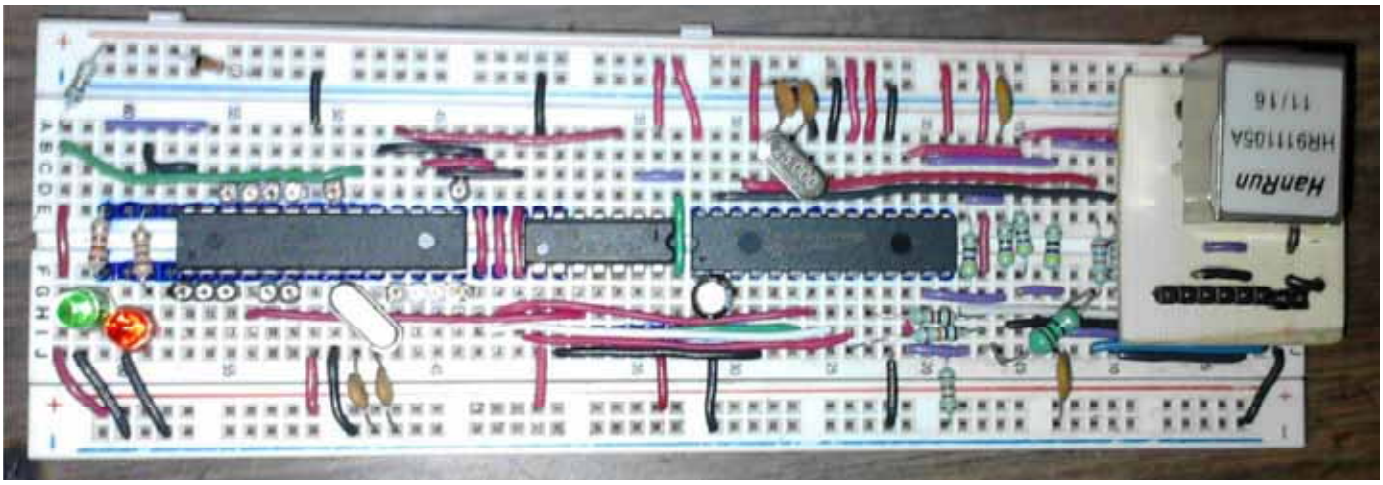
36 Figura: Vista previa del componente y en tercera dimensión (HR911105_RJ45).

Enseguida se listan los componentes que se utilizaron para el diseño del proyecto:

- 2 Condensadores de 33 pf.
- 2 Condensadores de 18 pf.
- 2 Condensadores de 10 uf.
- 2 Condensadores de 100 nf.
- 3 LED de 5 mm.
- 3 Resistencias de 270 Ohms.
- 4 Resistencias de 50 Ohms.
- 2 Resistencias de 10 KOhms.
- Resistencia de 2.8 KOhms.
- Cristal de 25 Mhz.
- Cristal de 16 Mhz.
- Inductor de 10 uH.
- Microcontrolador Atmega328P.
- Controlador de red ENC28J60.
- Conector RJ45 HR91115.
- Regulador LM7833.
- Componentes adicionales como conectores para las entradas y salidas, y la alimentación.

Una vez bien dedefinidos los componentes y sus conexiones, se realizó el montaje en una tarjeta de prototipos *(protoboard)*. Con ello se lograron encontrar errores en el cableado físico de los componentes y tomar lecturas reales de los valores eléctricos, temperaturas y correcto funcionamiento del prototipo. Por ejemplo, se ajusto la resistencia del Bias a un valor de 2.8 Kohms lo cual redujo la temperatura del controlador de red, el cual presentaba sobrecalentamiento por la disipación de potencia. Otro error que se

encontró fue que el nivel de voltaje obtenido con un arreglo de diodos Zener y resistencias no era el adecuado, por lo que se decidió controlar el voltaje con un regulador *LM7833*. También se comprobó que el valor de inductancia a 10 uH que entrega el inductor de cerámica propuesto en el diseño, trabajaba de manera correcta para la transmisión de tramas con el controlador de red, esto debido a que armar un inductor con núcleo de hierro es una tarea no trivial y que aumenta la probabilidad de error, por lo que se eligió un inductor de valor comercial.



37 *Figura: Prototipo construido a partir del diseño final. El conector de red RJ45 HR11105 fuen montado en una pequeña placa con un cableado fijo para facilitar el montaje.*

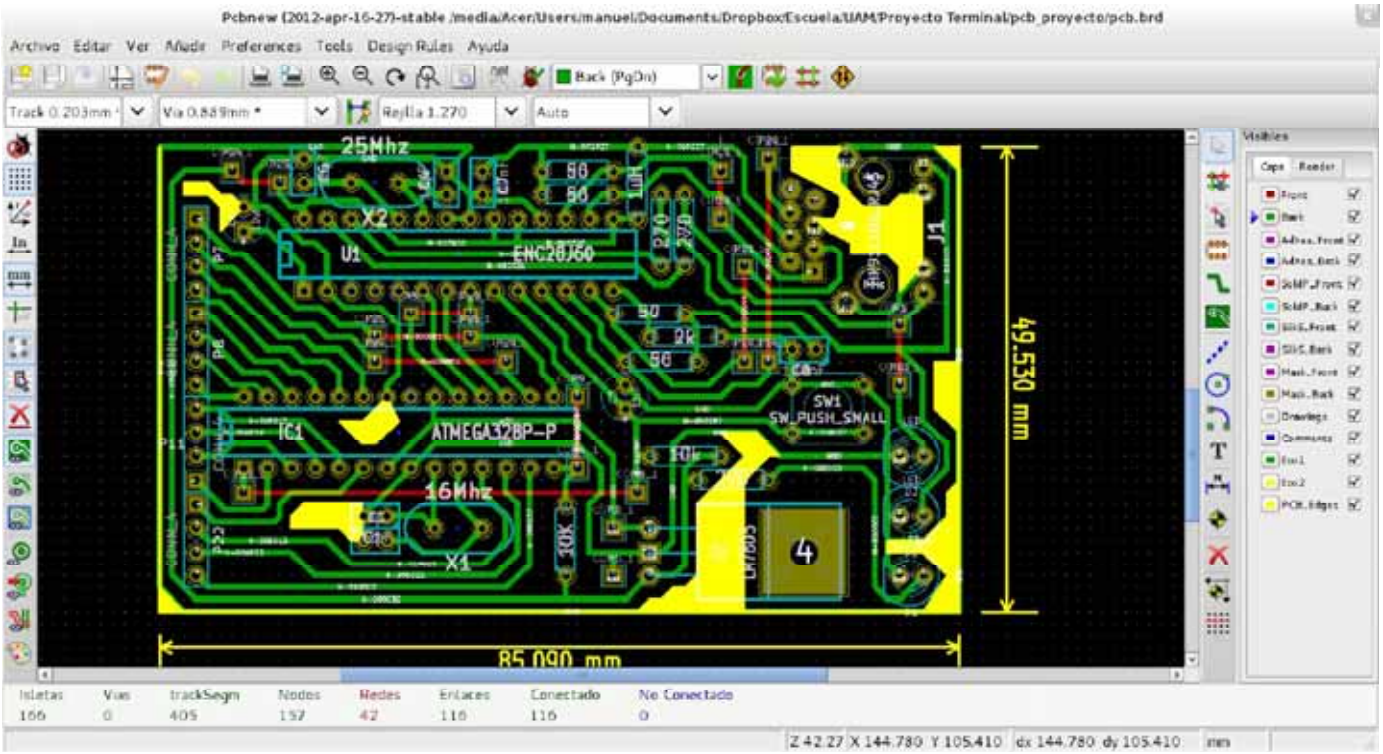
Con los componentes probados y montados en el prototipo, se continuó con el diseño del modelo de la placa impresa, que al igual que el esquema eléctrico y la asociación de componentes, también se realizó con *Kicad.*

Con los detalles corregidos en el prototipo se actualizaron valores y conexiones en el diseño, con el fin de tener una buen diseño para ser montado y soldado, ya que una vez soldados los componente, es difícil corregir errores.

La distribución de los componentes en la tarjeta, se realizó de manera que, el tamaño de la misma no excediera los 5 cm de ancho y los 10 cm de largo, ya que las medidas comerciales de las placas PCB, están dados en múltiplos de 5 cm. Algunas conexiones se diseñaron a través de puentes, con el fin de evitar el diseño de una placa de doble cara, la cuál aumenta la dificultad y el costo de fabricación.

Para los puertos de entrada y salida, se colocaron *cabeceras de pines*, las cuales permiten una fácil conexión con cables para la tarjeta de prototipos.

Una parte importante en el diseño, fue el agregar la *masa*, que es en realidad, aprovechar los espacios vacíos en el diseño para una pista que conduzca *tierra*, lo que ayuda a disminuir el ruido en el sistema.

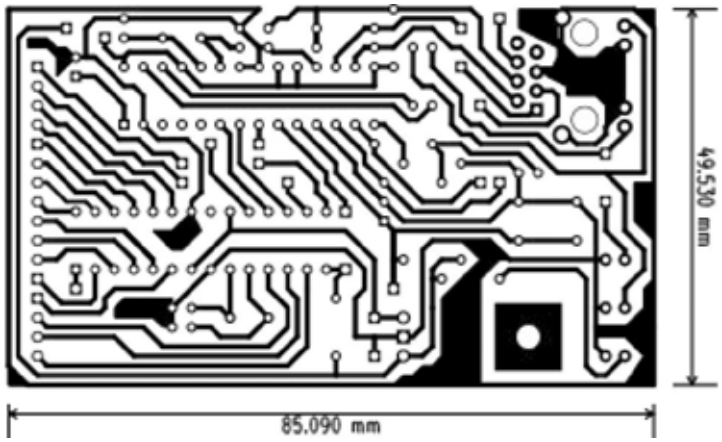


38 *Figura: Diseño del circuito impreso con Kicad.*

Una vista previa del circuito antes de su construcción física:

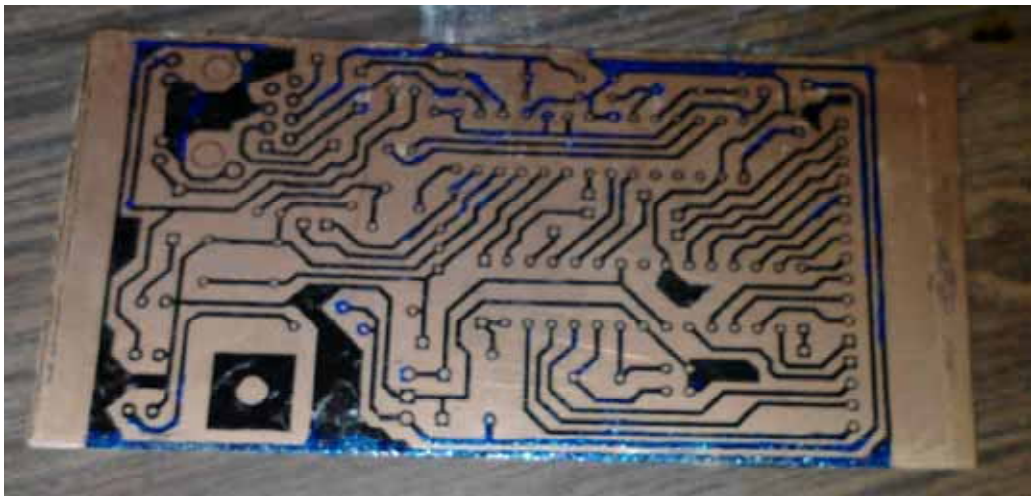
39 *Figura: Vista preliminar del diseño en tercera dimensión con Kicad.*

Para elaborar la placa PCB con el circuito impreso, se empleó la técnica de planchado de tarjeta. Para ello es necesario imprimir la capa de pistas de cobre en papel que permita transferencia térmica en una impresora láser. Con la capa de pistas impresa, se corta el pedazo de placa al tamaño definido para el circuito, y se coloca la hoja de transferencia de calor con el circuito impreso, para por último, ser planchado sobre la placa para que con el calor la tinta del *toner* se adhiera a la superficie de cobre de la placa.



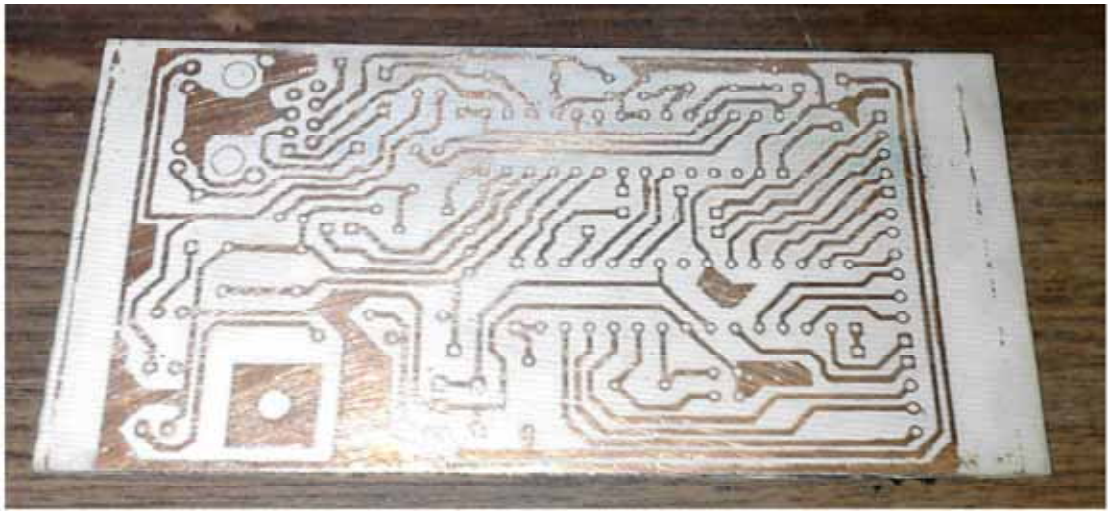
40 *Figura: Impresión del circuito en la hoja de transferencia de calor.*

Después de planchar y retirar el papel de la placa, debe verse la impresión del circiuto transferida en la capa de cobre con una buena calidad y con el mínimo número de errores. Si existen errores se deben corregir con algún plumón de tinta indeleble, con el fin de corregir las pistas dañadas.



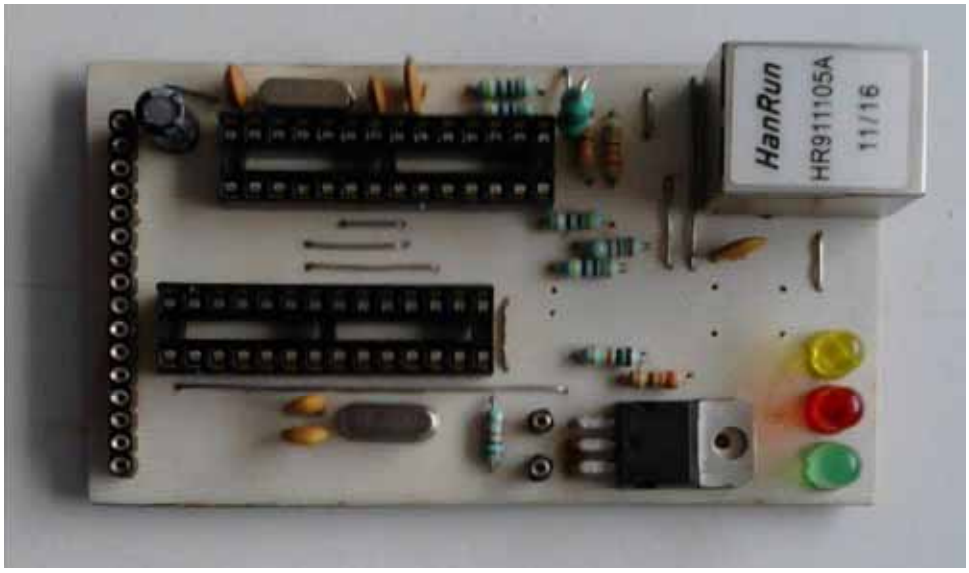
41 *Figura: Impresión del circuito en la placa. Los errores de transferencia de toner deben corregirse antes de someter la placa al ataque químico, esto se puede hacer utilizando algún plumón de tinta indeleble.*

Con la placa impresa y sus errores corregidos, se somete a un ataque químico con *cloruro férrico* el cuál es un corrosivo que ataca a la capa de cobre en sus zonas no protegidas por el toner y la tinta que protege las zonas dañadas o mal transferidas. Se debe mantener la placa sumergida en la solución hasta que el cobre de las zonas no protegidas esté completamente eliminado de la placa.



42 Figura: Placa con las pistas de cobre lista para ser perforada y ensamblada.

Para poder montar los componentes, es necesario hacer perforaciones en la placa en los puntos indicados para los pines. La perforación se realizó con una broca de 3/32 de pulgada. Ya con la placa lista para montar los componentes, se ensambló y se soldó.



43 Figura: Placa ensamblada y soldada.

Con esto, la parte electrónica del dispositivo hecho a la medida, es concluida.

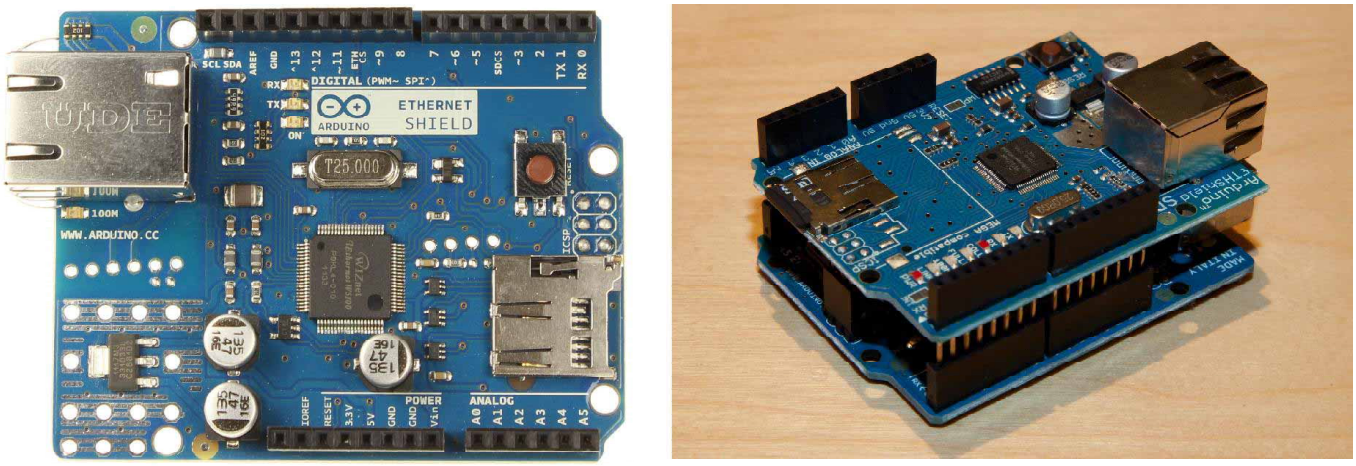
Plataforma Arduino

Para el caso de la plataforma *Arduino* el desarrollo de su electrónica es mínimo, ya que todos lo componentes que trabajan en ésta plataforma, están empaquetados en lo que se denomina *Shields*. Existen *shields* para conexión de red, para almacenamiento masivo, para control y sensores, etcétera.

En el caso del proyecto actual, se ocupa el *Ethernet Shield*. The Shield Arduino Ethernet se conecta la placa *Arduino* a Internet de manera sencilla. Sólo se tiene que conectar este módulo a la placa Arduino, conectarlo a la red con un cable UTP con conector RJ45. Como todos los elementos de la plataforma es de libre acceso

y de fuente abierta. Funciona con una alimentación de 5V, que se suministra a través de la placa *Arduino*. Está basado en el controlador de comunicación Ethernet *Wiznet W5100* con un bufer interno de 16 KB. Tiene una capacidad de conexión con velocidades de 10Mb y 100Mb . La conexión con Arduino se realiza a través del puerto SPI. El controlador *Wiznet W5100* proporciona la pila de protocolos IP, con la capacidad de trabajar con los protocolo TCP y UDP. Soporta hasta cuatro conexiones de *socket* simultáneas.

Para el almacenamiento, cuenta también con una ranura para tarjetas *micro SD*, que se puede utilizar para almacenar archivos para servir a través de la red. Es compatible con *Arduino Uno* y *Arduino Mega*. El lector de tarjetas *micro SD* se puede acceder a través de la Biblioteca *SD.h*. Cuando se trabaja con esta biblioteca, la señal *SS* es el pin 4 en la tarjeta.



44 *Figura: Placa Ethernet Shield para la comunicación por red basada en el controlador de red Wiznet W5100 .*

El uso de la biblioteca *Ethernet.h* facilita el desarrollo de aplicaciones que requieran comunicación a través de la red.

La parte de programación

La programación de la aplicación en los microcontroladores, consiste en desarrollar el programa que trabajará en el lado del servidor, y la interfaz que será presentada en el navegador Web del usuario.

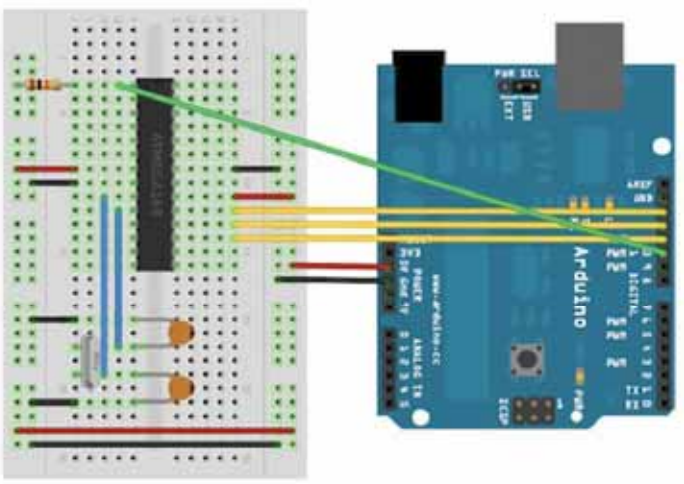
Dado que tanto el dispositivo hecho a la medida como el basado en Arduino trabajarán como servidores, se deben realizar ambos desarrollos.

Primero se explica la programación en el lado del servidor para cada dispositivo, y después se detalla la interfaz Web. La interfaz Web es exactamente la misma en ambos casos.

Sistema hecho a la medida

Aprovechando que el microcontrolador utilizado para el sistema hecho a la medida y el utilizado en la plataforma Arduino es el mismo, se utilizó la herramienta de Arduino para su programación, además de las bibliotecas disponibles construidas por la comunidad del *software libre*.

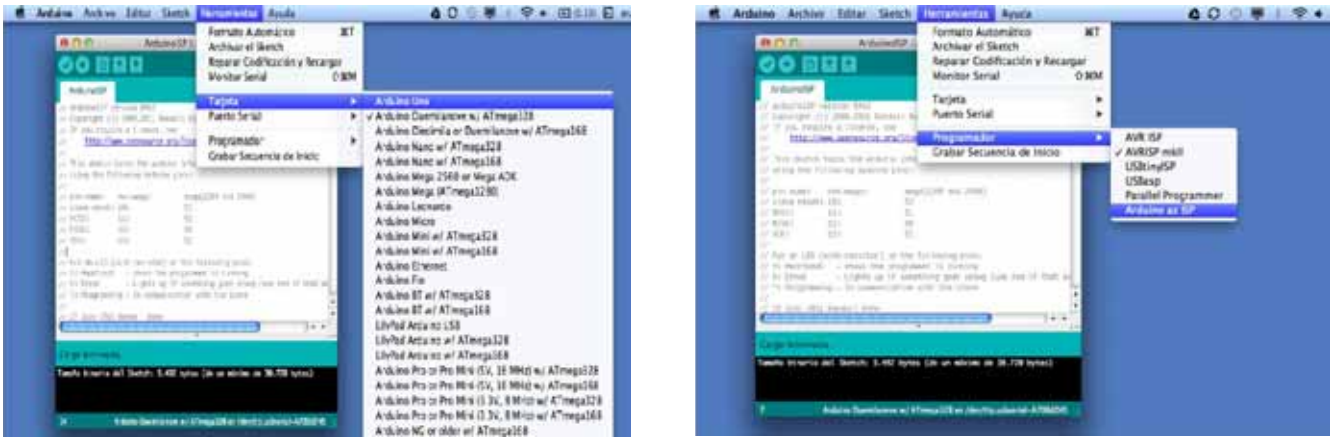
Para poder ejecutar el código compilado en la herramienta de Arduino, es necesario cargar un programa de arranque en el microcontrolador (*BIOS Arduino*). Para ello se debe armar un sistema mínimo con el microcontrolador objetivo, y a través de la interfaz SPI, comunicarlo con la tarjeta de desarrollo Arduino.



45 *Figura: Conexiones entre la placa Arduino y el sistema mínimo montado para la comunicación SPI.*

Teniendo comunicación entre el microcontrolador y la tarjeta Arduino, se conecta el cable de comunicación USB a la computadora con la aplicación de Arduino.

En la máquina se abre la aplicación y en el menú *Herramientas*, se elige la opción *Tarjeta* para seleccionar el tipo de la tarjeta *Arduino UNO*. Después de selecciona la opción *Programador* para elegir el modo de la programación *Arduino as SPI*. Por último se carga el programa de arranque en el microcontrolador Atmega328P dando clic en el menú *Herramienta* y eligiendo la opción *Cargar secuencia de inicio*.



46 *Figura: Configuración de la herramienta de Arduino para cargar el programa de arranque en el microcontrolador Atmega328P.*

El cargador de arranque ocupa 512 Bytes de la memoria de programa del microcontrolador, por lo que, el espacio total de 32 KB se ve reducido en medio KB. Las aplicaciones deben ser cuidadosamente diseñadas para no ocupar demasiada memoria de programa.

Para la programación de la aplicación se utilizó la biblioteca *EtherCard.h*, en la cuál se definen el espacio de nombres *Ethernet*, los tipos de datos *BufferFiller*, y las funciones asociadas a la estructura definida en el espacio de nombres *Ethernet* (métodos de *Ethernet*): *begin*, *staticSetup*, *tcpOffset*, *packetReceive*, *packetLoop*, *httpServerReply*; y las funciones asociadas a la estructura *bfill* (métodos de *bfill*): *emit_p* y *position*.

En el programa se agregaron instrucciones para reportar la actividad del programa a través del puerto *serial*,

tales instrucciones tienen como finalidad la monitorización de la ejecución del programa en el microcontrolador, y no son necesarias para el funcionamiento del sistema.

Debido a que el MTU no puede exceder el tamaño de la memoria disponible en el microcontrolador, se define con una longitud máxima de 1500 bytes. Por lo tanto, las cadenas enviadas no deben exceder esa cantidad. Cada elemento de la interfaz debe ser enviado en tramas con un espacio para datos de máximo 1500, por lo que se definió el búfer *Ethernet* como un arreglo de bytes de 1300.

El programa inicializa su contexto en la función *setup()*, en donde se define el modo de los pines a utilizar: los pines 3, 2, 1, 0, 4, 5, 6, 7 como salidas, y los pines 9, 10, A0, A1, A2, A3, A4, A5 como entradas.

Con el función (*método* en el paradigma orientado a objetos) *begin* de la estructura *Ethernet*, se inicializan los valores de la estructura tomando en cuenta la MAC definida en el arreglo de bytes *mymac*[]: 74:69:69:2D:30:FF (para efectos de pruebas en la red local). Y se indica con la función *staticSetup()* que el dispositivo tendrá una dirección lógica fija definida en el arreglo de bytes *myip*[]: 192.168.1.2.

En el ciclo infinito de la función *loop()*, se obtiene la palabra *word* que contiene el paquete recibido en la petición y su tamaño. Se lee el estado de todos los pines usados como entradas y salidas. Se obtiene la petición del cliente apuntando al desplazamiento del tamaño del paquete, llegando así, a la zona de datos de la trama recibida, logrando obtener en una cadena de caracteres toda la petición HTTP. Después de obtener la petición, se filtran los encabezados en busca del encabezado GET, que es el que contiene el nombre del archivo solicitado. De acuerdo al nombre del archivo solicitado, se envía como respuesta una trama con una cadena de caracteres que contiene el contenido solicitado dentro de su definición.

El código sintetizado:

```
#include <EtherCard.h>

static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0xFF };
static byte myip[] = { 192,168,1,2 };

int ldPin[] = { 3,2,1,0,4,5,6,7 };
int swPin[] = { 9,10,A0,A1,A2,A3,A4,A5 };

byte Ethernet::buffer[1300];
BufferFiller bfill;

void setup () {
  pinMode(ldPin[0], OUTPUT);
  ...
  pinMode(ldPin[7], OUTPUT);
  pinMode(swPin[0], INPUT);
  ...
  pinMode(swPin[7], INPUT);

  if (
    ether.begin(sizeof Ethernet::buffer, mymac) == 0){
  }
```

```
  ether.staticSetup(myip);
}

static word homePage(int valLdPin[8],int valSwPin[8]) {
  bfill = ether.tcpOffset();
  bfill.emit_p(
    PSTR(
      "<html>"
      ...
      "</html>"
    ),
    valLdPin[0],
    ...
    valLdPin[7],
    valSwPin[0],
    ...
    valSwPin[7]
    );
  return bfill.position();
}

void loop () {
  word len = ether.packetReceive();
  word pos = ether.packetLoop(len);

  char * solicitud;
  char * queryString;
  int ldPinSol;

  int valSwPin[8];
  int valLdPin[8];

  valLdPin[0] = bitRead(PORTD, 3);
  ...
  valLdPin[7] = bitRead(PORTD, 7);

  valSwPin[0] = digitalRead(swPin[0]);
  ...
  valSwPin[7] = digitalRead(swPin[7]);

  if (pos){
```

```
    solicitud = (char *)Ethernet::buffer + pos;

    if(strstr(solicitud, "GET /index.htlm") != 0) {
      ether.httpServerReply(homePage(valLdPin,valSwPin));
    }
  }
}
```

Sistema basado en Arduino

Para programar la aplicación en el lado del servidor sobre el dispositivo basado en la plataforma Arduino, se utilizan dos bibliotecas distintas a las usadas en el sistema hecho a la medida, esto debido a que el controlador usado en el modulo *Ethernet Shield* es distinto. Para el controlador *Wiznet W5100* se utiliza la biblioteca estándar de *Arduin*, *Ethernet.h*. En ésta biblioteca se define el tipo de dato *IPAddress* y las estructuras *EthernetClient* y *EthernetServer*, las cuales incorpora a las funciones *begin() print()*, *println()*, *connected()*, *available()*, *read()* y *stop()*. El uso de esta biblioteca, es mucho más sencillo. También se importa la biblioteca para la comunicación a través del protocolo SPI, *SPI.h*.

Al igual que en el caso anterior, el contenido Web que se envía como respuesta se define en una función.

En la función *setup()* se inicializa la estructura *EthernetServer* la cual desde su definición, tiene al puerto 80 como el puerto donde trabaja el sistema. Se inicializa con la MAC y la IP como parámetros a través de la función *begin()*.

En el ciclo infinito se inicializa la estructura *EthernetClient* a través de la función *available()* de la estructura *EthernetServer*. Si existe una petición, el programa revisa la conexión y disponibilidad del mismo a través de las funciones *connected()* y *available()*. Se lee la petición caracter por caracter con la función *read()* de la estructura *EthernetClient*, y se va concatenando a una cadena de caracteres que se utiliza como bufer. Si se encuentra un doble salto de linea en la petición, significa que el cliente ha terminado de transmitir, y en ese momento se devuelve la respuesta. La conexión termina con la función *stop()* de la estructura *EthernetClient*.

El código sintetizado:

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,100);
EthernetServer server(80);

int ldPin[] = { 3,2,1,0,4,5,6,7 };
int swPin[] = { 9,10,A0,A1,A2,A3,A4,A5 };

void setup()
{
  pinMode(ldPin[0], OUTPUT);
```

```
  ...
  pinMode(ldPin[7], OUTPUT);

  pinMode(swPin[0], INPUT);
  ...
  pinMode(swPin[7], INPUT);

  Ethernet.begin(mac, ip);
  server.begin();
}

String homePage(int valLdPin[8],int valSwPin[8]) {
  String cadena = "<html>...";

  cadena.concat(valLdPin[0]);
  ...
  cadena.concat(valLdPin[7]);

  cadena.concat(valSwPin[0]);
  ...
  cadena.concat(valSwPin[7]);
  cadena.concat("...</html>");

  return(cadena);
}

void loop()
{
  int ldPinSol;
  int valSwPin[8];
  int valLdPin[8];

  valLdPin[0] = bitRead(PORTD, 3);
  ...
  valLdPin[7] = bitRead(PORTD, 7);

  valSwPin[0] = digitalRead(swPin[0]);
  ...
  valSwPin[7] = digitalRead(swPin[7]);

  EthernetClient client = server.available();
```

```
  if (client) {
    boolean currentLineIsBlank = true;
    String cadena="";

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        cadena.concat(c);

        if (c == '\n' && currentLineIsBlank) {

            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println();

            client.println(homePage(valLdPin,valSwPin));
            break;
        }
        if (c == '\n') {
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }
    delay(1);
    client.stop();
  }
}
```

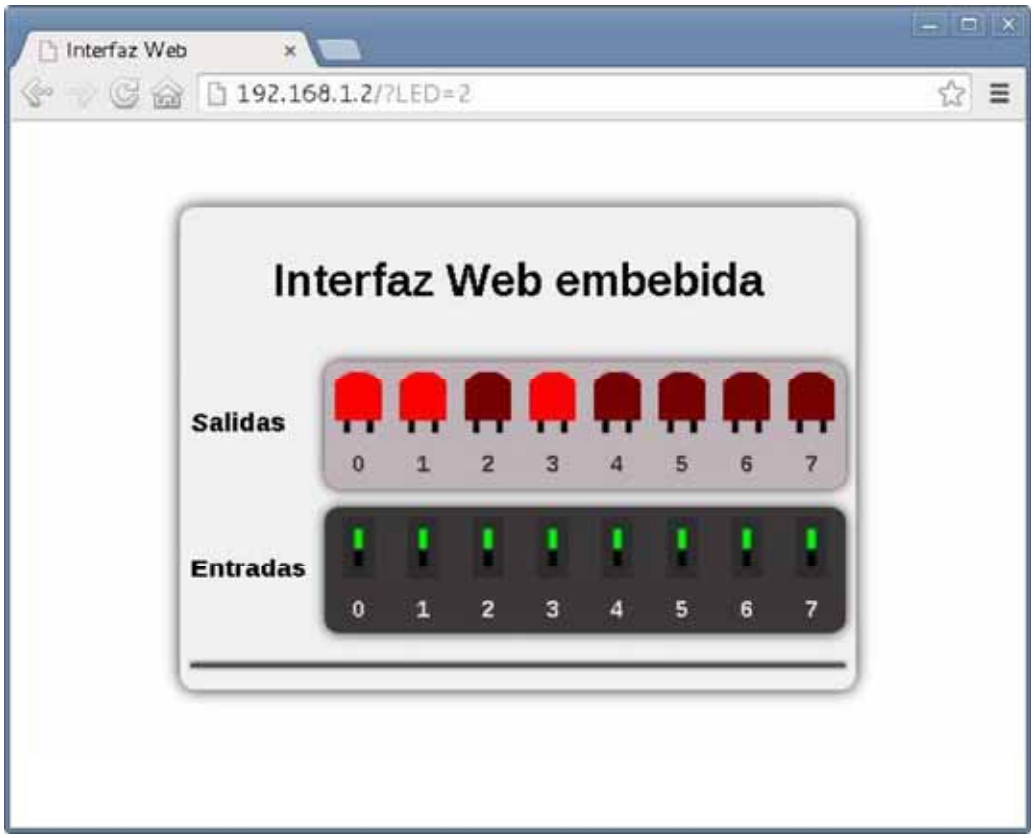
Programación de la interfaz Web

La pantalla que verá el usuario le permite controlar los puertos de entrada y salida de cada dispositivo. Es una página muy sencilla, sólo cuenta con ocho botones para encender y apagar LEDs y ocho imágenes que representan el estado de las entradas como interruptores.

La página cuenta con una hoja de estilos *CSS* y un archivo con código *JavaScript*, además del código *HTML* que compone el encabezado y el cuerpo de la página.

El número de caracteres de toda la página excede el número de bytes permitido por la memoria del microcontrolador, por lo que de a cuerdo al MTU establecido, no pueden transmitirse más de 1300 caracteres en cada transferencia. Cada archivo enviado tiene menos de 1300 caracteres. Esto se logró escribiendo el código de la siguiente manera:

- El HTML de la página principal cuenta con el encabezado, el título, llamadas a los archivos *scripts.js*, *imagenes.js* y *estilos.css*, y al cuerpo del documento.
- El archivo *scripts.js* construye de manera dinámica los botones que representan a los ocho LEDs y a los ocho interrupores.
- El archivo *imagenes.js* define a las imágenes en arreglos de cadenas con el código de las imágenes en *base64*. Esto debido a que en el microcontrolador no se pueden almacenar archivos bajo un sistema de archivos.
- El archivo *estilos.css* define las características visuales de los elementos de la página.



47 *Figura: Página Web de la aplicación.*

Cuando el cliente solicita el archivo *index.html*, en realidad realiza una serie de peticiones, primero llama a *index.html*, después llama a *scripts.js*, luego llama a *imagenes.js*, y al final llama a *estilos.css*. Este punto se observará en la etapa de pruebas, donde se capturan los paquetes para su análisis.

Además, existen otros archivos, el EDO y el LED=0, LED=1, LED=2, LED=3, LED=4, LED=5, LED=6 y LED=7, los cuáles trabajan como páginas dinámicas en el servidor. EDO obtiene el estado de las entradas y las salidas en una cadena de caracteres con valores binarios, que representan un estado de encendido o apagado. Y las páginas LED=0, LED=0, LED=1, LED=2, LED=3, LED=4, LED=5, LED=6 y LED=7 encienden o apagan una salida.

Esta interfaz, corre en una gran cantidad de navegadores, mismos que se evaluarán en la etapa de pruebas.

Etapa de evaluación

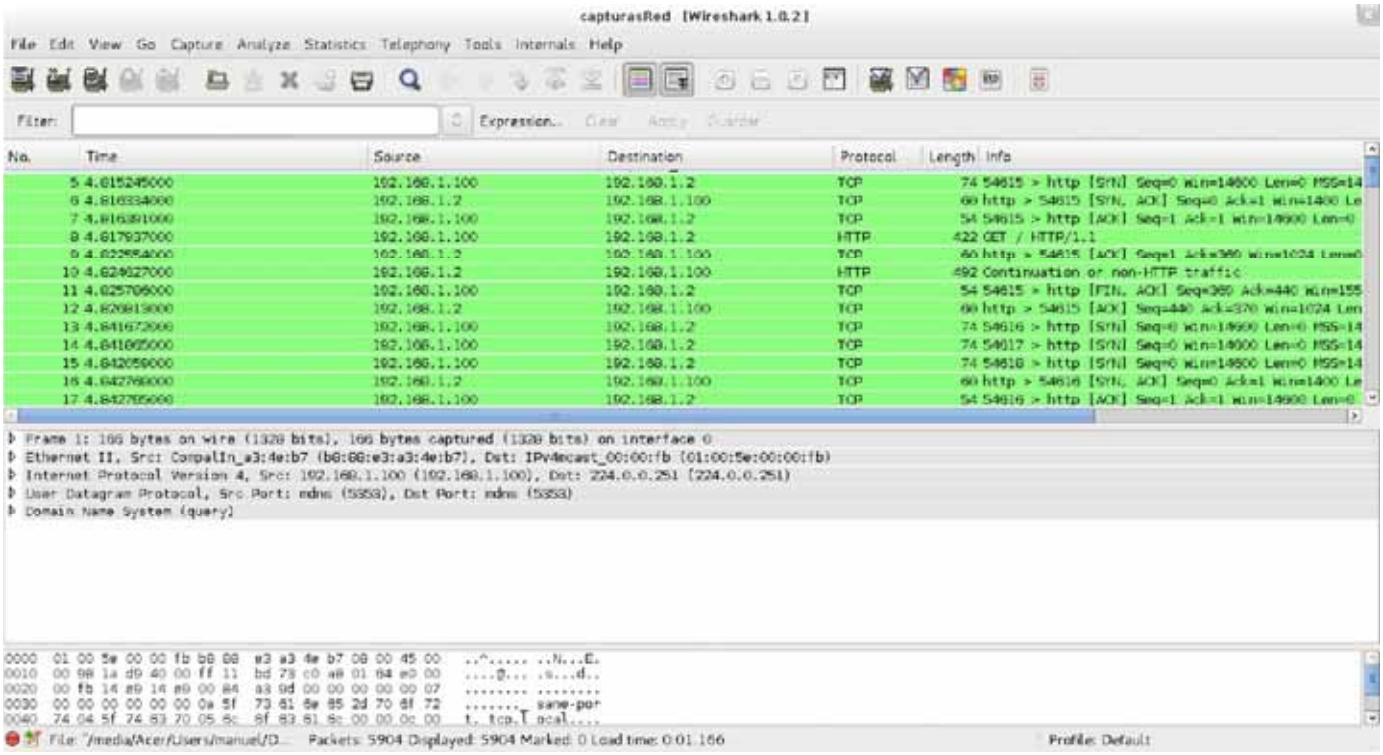
Para la evaluación técnica del los sistemas con una interfaz Web, se utilizó la herramienta para la captura de táfico de red *Wireshark*, el commando *Ping*, y además se construyó una aplicación con *Javascript*, para medir los tiempos de carga de los elementos de la interfaz. Las pruebas se ejecutaron en los navegadores Firefox, Chromium e Internet Explorer en máquinas con Debian Wheeze y Windows 8. Además se ejecutó la aplicación en un teléfono celular con Android.

Para evaluar los costos de tiempo, esfuerzo y dinero, se registraron precios de los componentes, tiempos de desarrollo y dificultades que se dieron durante el desarrollo de los sistemas.

Evaluación técnica

Comunicación

Corriendo los dispositivos en una red local, se capturó su tráfico con la herramienta *Wireshark*, comprobando que ambos ejecutan de manera correcta los protocolos de comunicación tanto en la capa de enlace como en la capa de red.

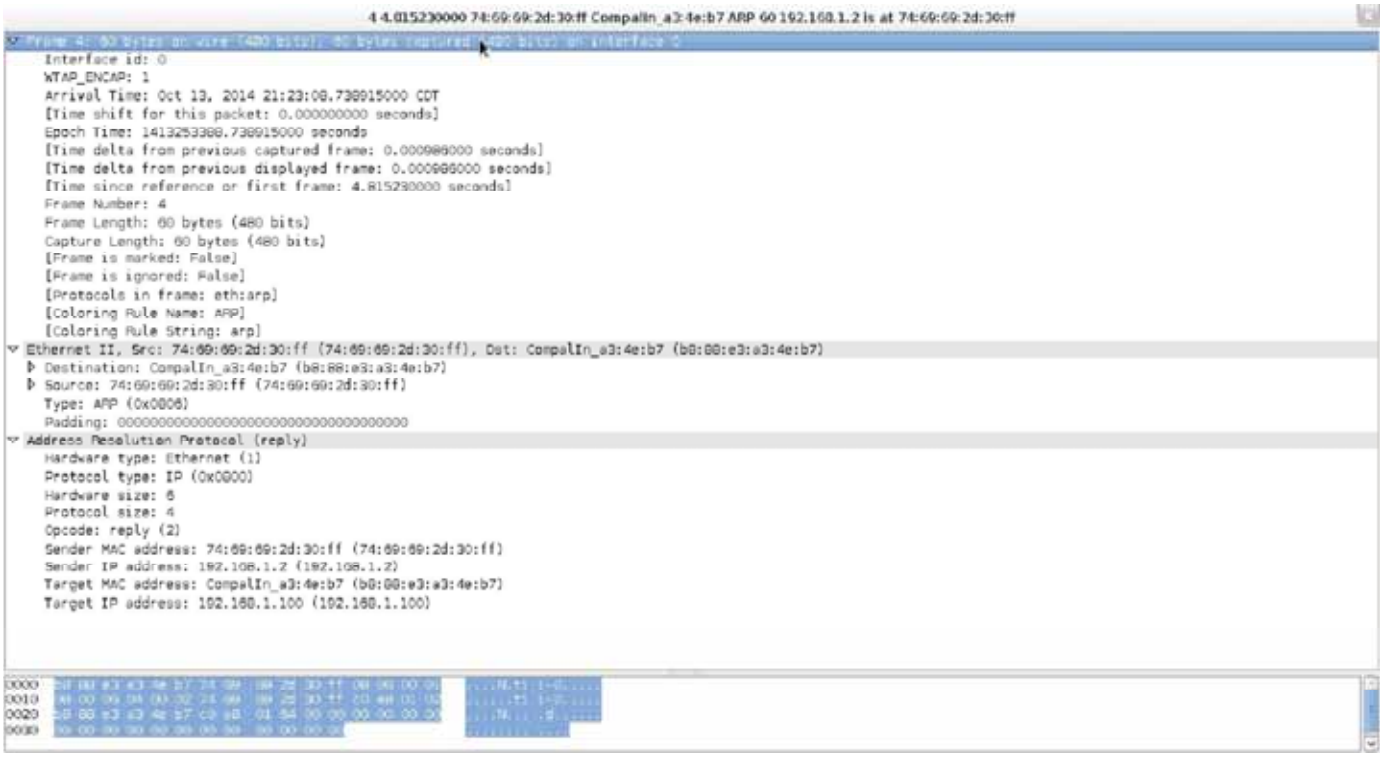


48 *Figura: Capturas de tráfico en la red de los dispositivos.*

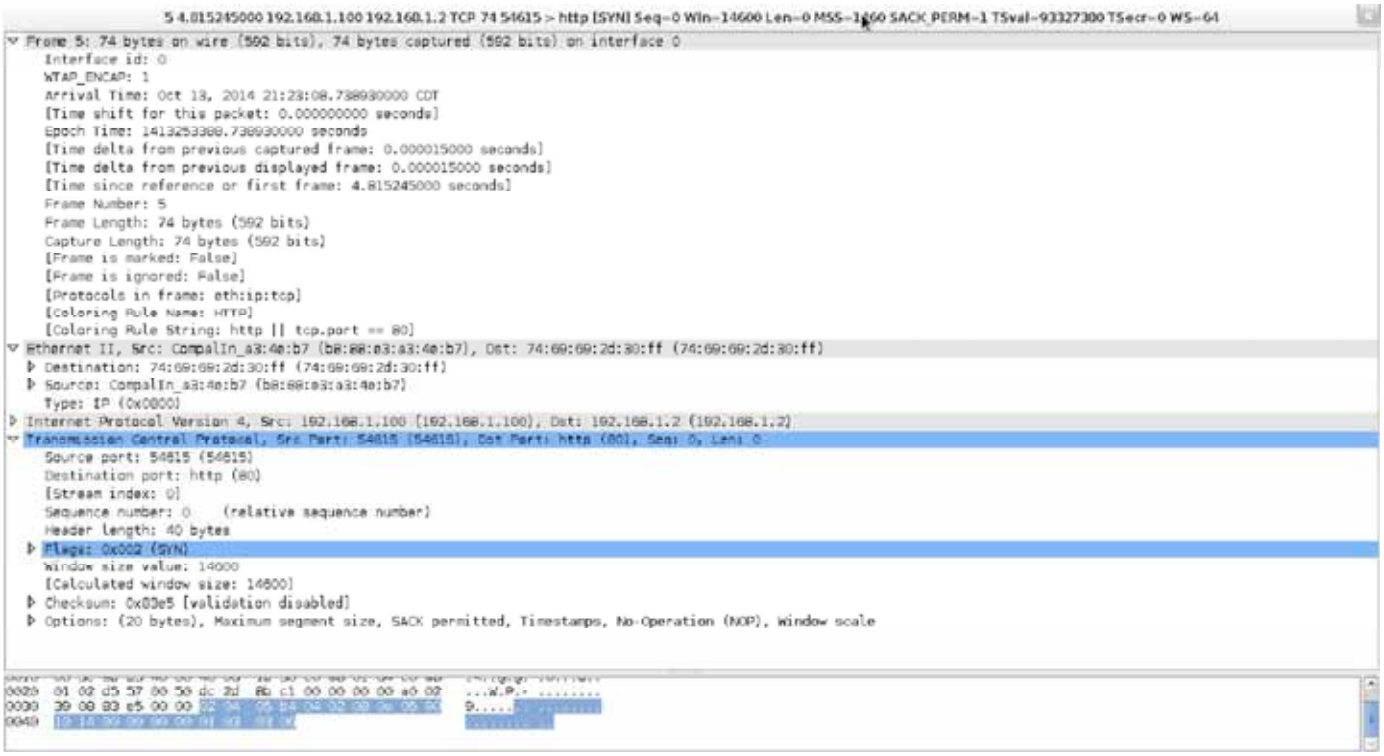
Ambos iniciaron enviando una trama ARP para obtener la dirección local de los dispositivos. Debido a que la comunicación en ese momento se realizaba por primera vez, es decir, la máquina que envió la petición no conocía la dirección física de los dispositivos.

Después los dispositivos contestaron con su dirección física, la cuál fue utilizada por la capa de enlace para realizar la comunicación. Teniendo las direcciones definidas en las tablas ARP, se solicitó a los dispositivos una petición HTTP, los dispositivos fueron respondiendo con respuestas que contenían los documentos solicitados.

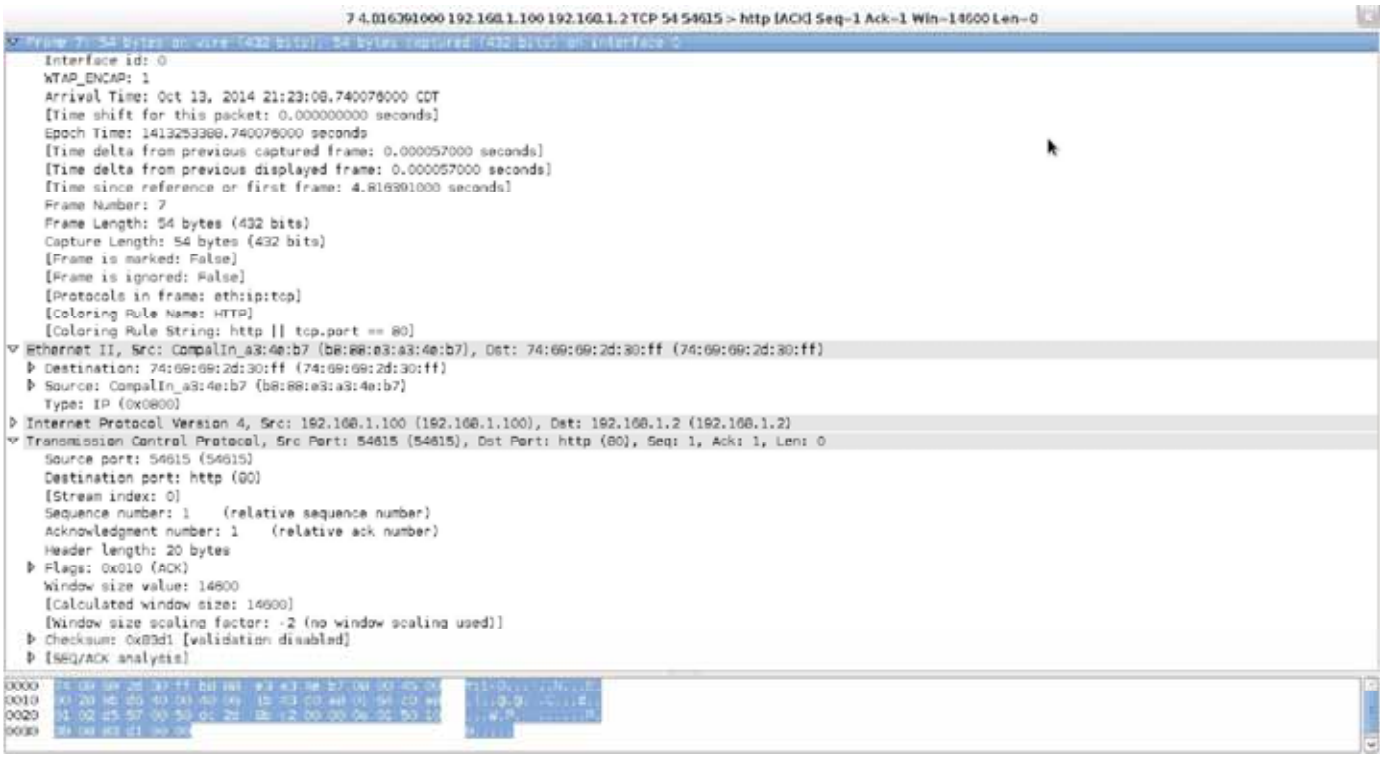
49 *Figura: Capturas de tráfico en la red de los dispositivos. Solicitud ARP realizada por el cliente.*



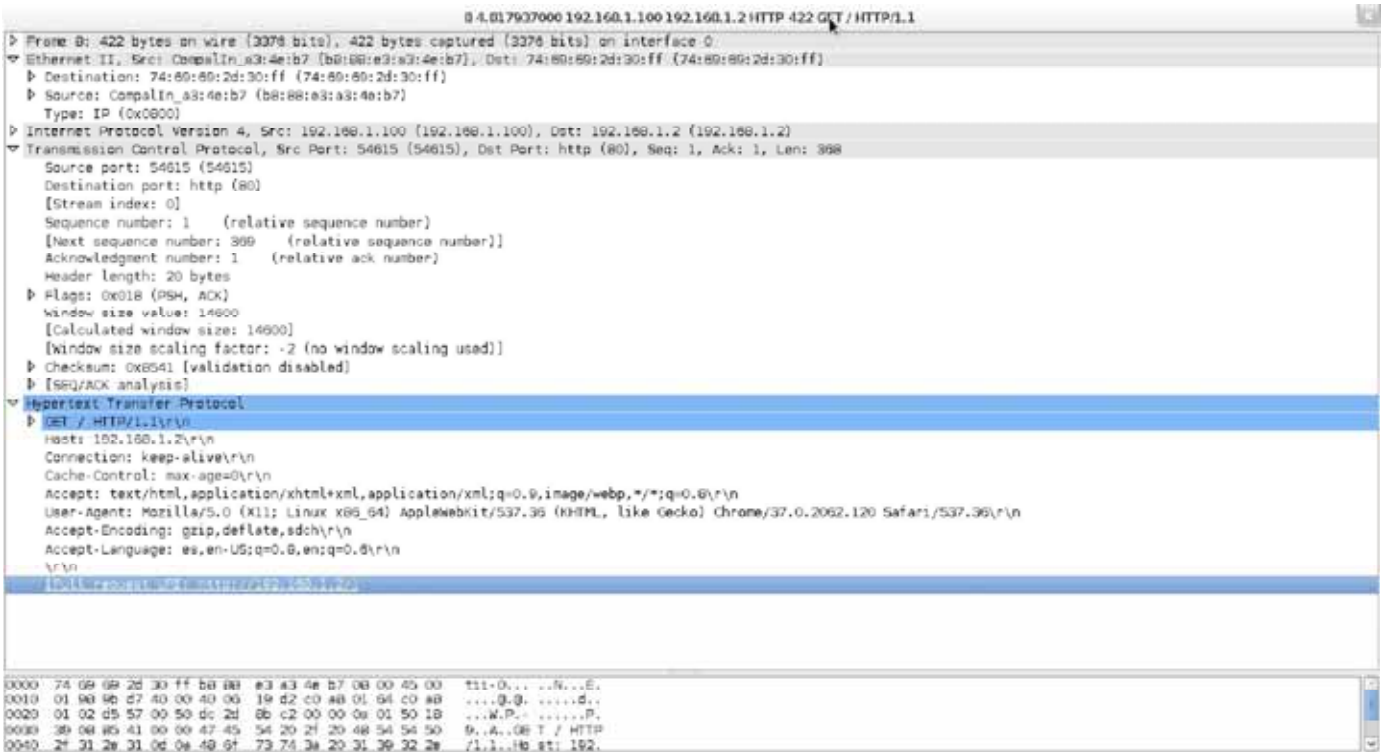
50 *Figura: Capturas de tráfico en la red de los dispositivos. Respuesta ARP realizada por cada dispositivo.*



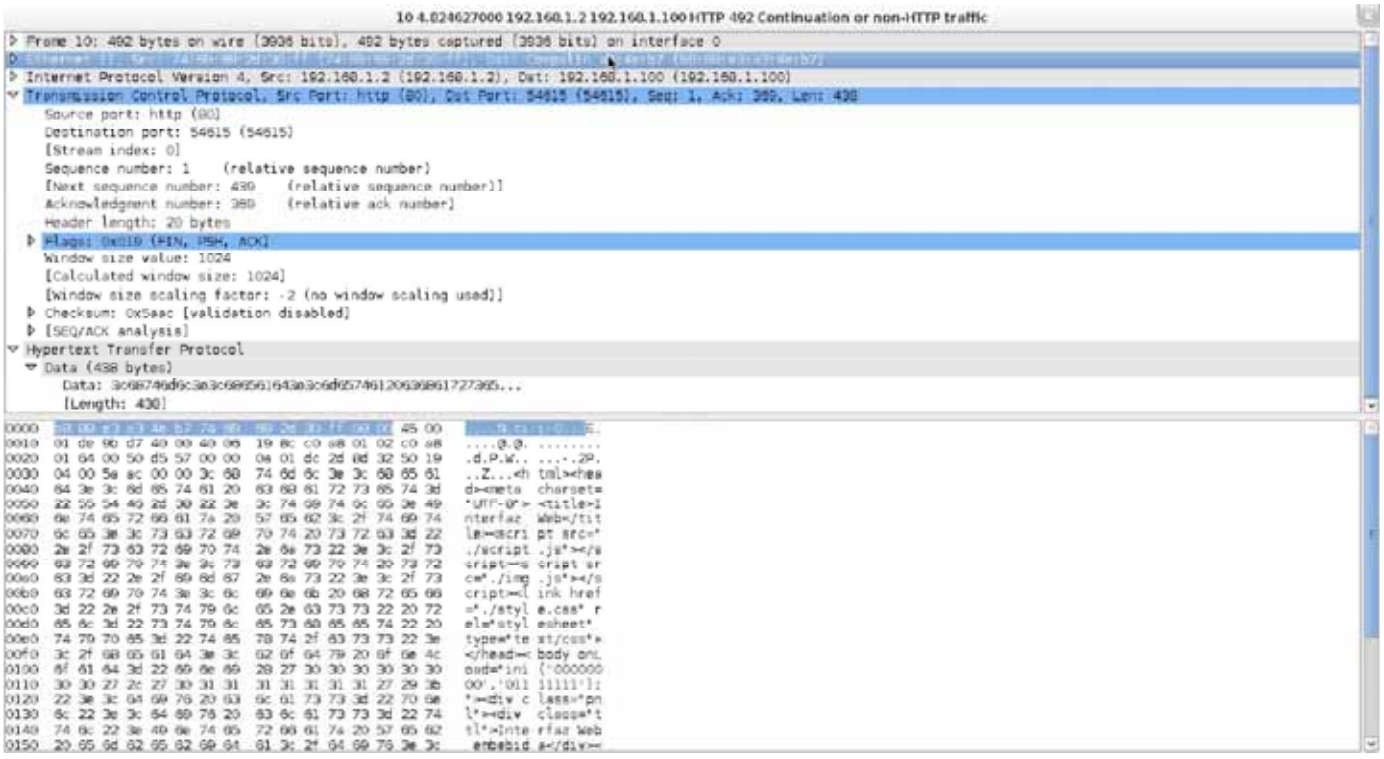
51 *Figura: Capturas de tráfico en la red de los dispositivos. Solicitud TCP realizada por el cliente (SYN).*



52 *Figura: Capturas de tráfico en la red de los dispositivos. Respuesta TCP realizada por el servidor (SYN, ACK).*



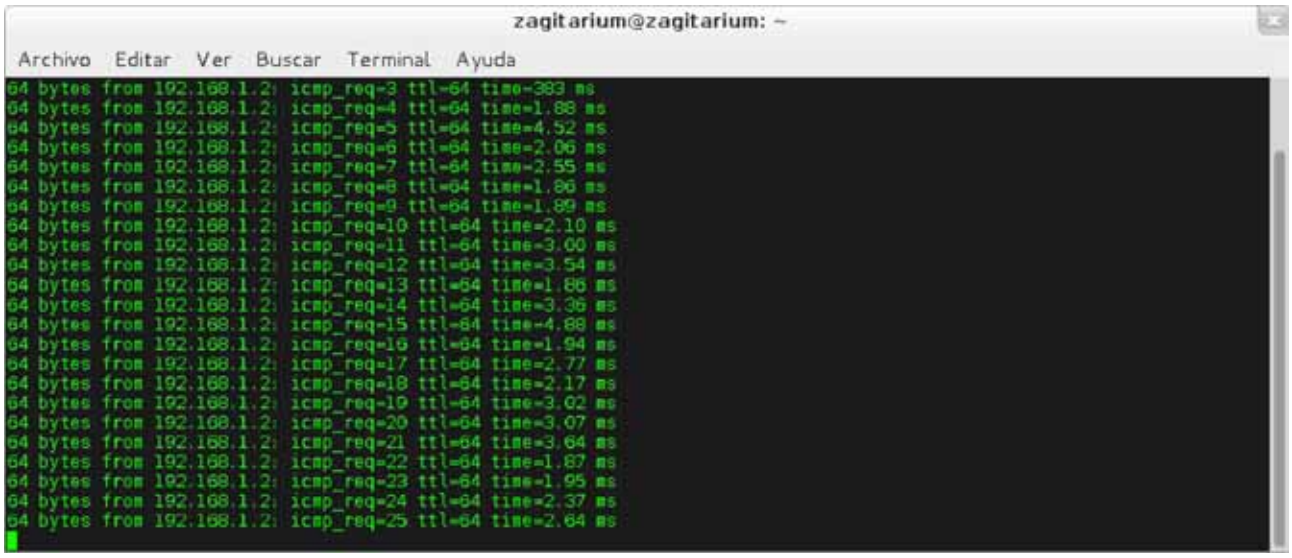
53 *Figura: Capturas de tráfico en la red de los dispositivos. Solicitud HTTP realizada por el cliente (GET).*



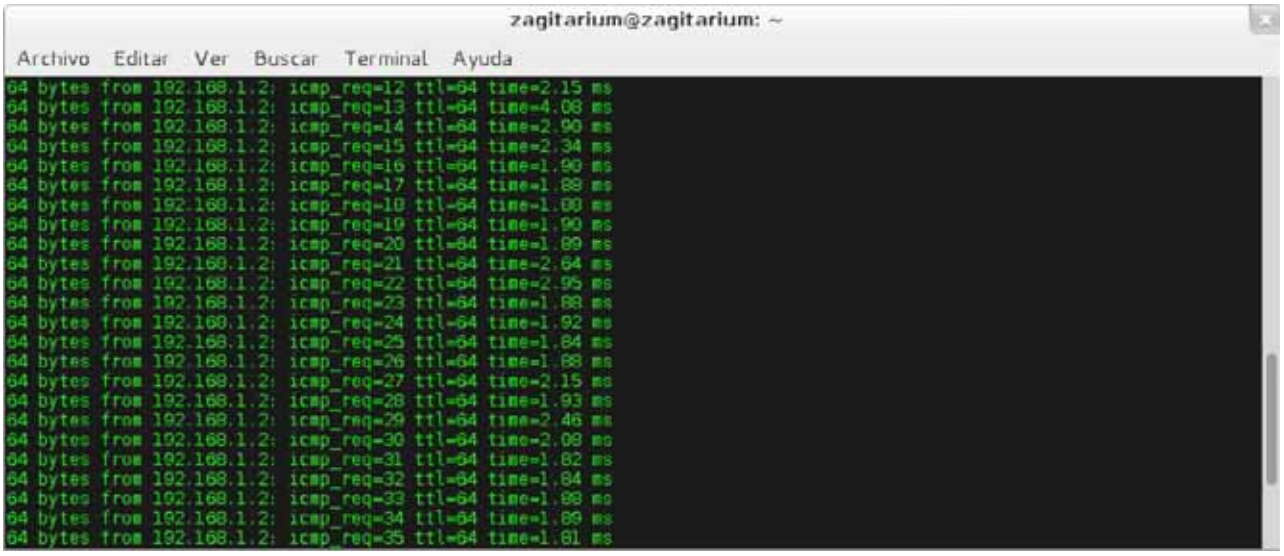
54 *Figura: Capturas de tráfico en la red de los dispositivos. Respuesta HTTP realizada por el servidor.*

De esta manera se observó que no importa el controlador ni la electrónica de los dispositivos, ambos se apegan a los protocolos del estándar de comunicaciones IEEE 802.3.

Con la ayuda del comando *Ping*, se tomaron datos acerca del tiempo de respuesta de los dispositivos.



55 *Figura: Ping recursivo enviado hacia el dispositivo hecho a la medida.*



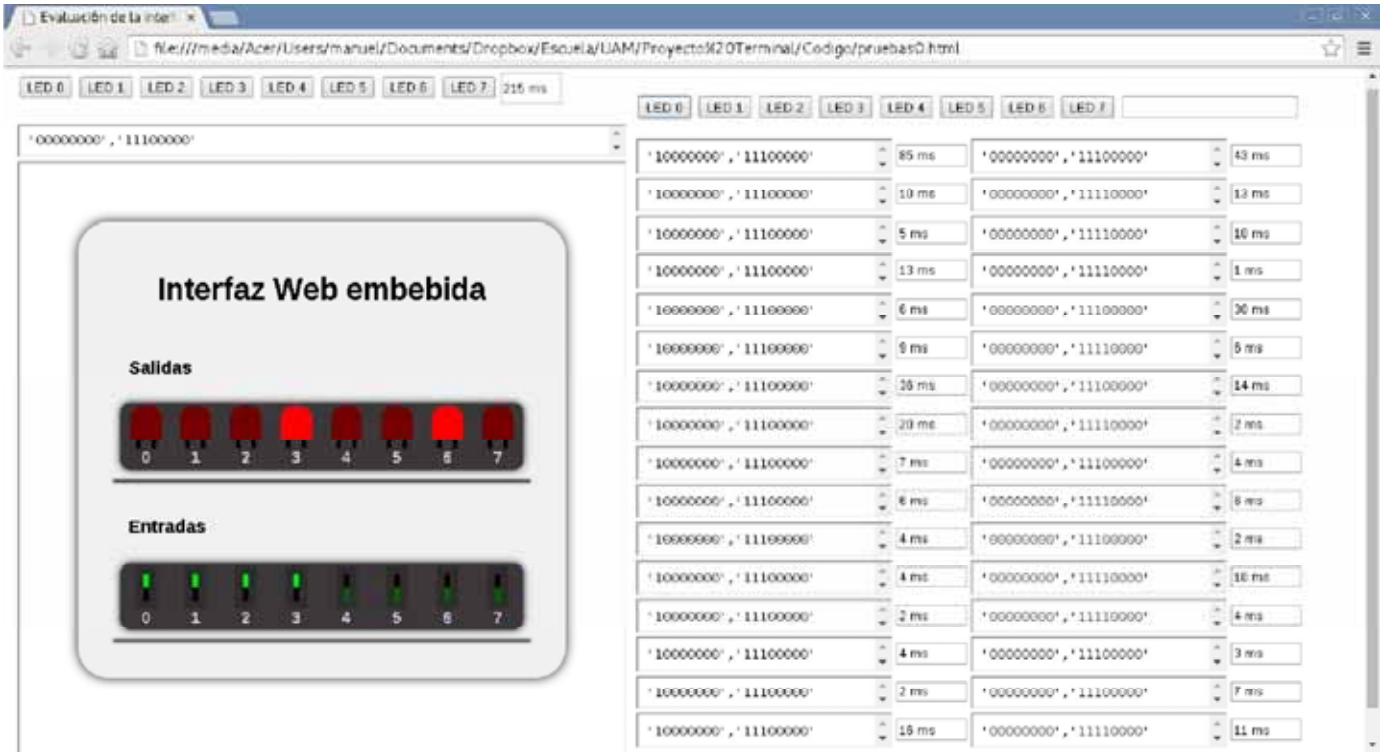
56 *Figura: Ping recursivo enviado hacia el dispositivo basado en Arduino.*

Desempeño

Para medir el desempeño de la interfaz servida por ambos dispositivos, se diseño una aplicación Web con código *JavaScript*, capaz de tomar el tiempo de respuesta para los comandos de la interfaz. Si algún comando fallara, esta aplicación también lo reportaría.

Se trata de una pantalla con la interfaz Web incrustada en un marco, acompañada de una serie de botones para ejecutar comandos y enviar las solicitudes al servidor. En los campos de texto se muestran los datos del tiempo de espera par cada petición al servidor. Un grupo de controles, permite realizar una sola solicitud por

comando, y el grupo más grande, realiza 32 solicitudes por cada ejecución de los comandos.



57 *Figura: Aplicación para la evaluación de la interfaz Web.*

La función *JavaScript* que toma el dato del tiempo está definida como:

```
function tiempo(){ var txtTiempo = document.getElementById("txtTiempo");
  var endTime = new Date();
  txtTiempo.value=endTime - startTime + ' ms';
  return (0);
}
```

Donde la variable *startTime* se define de manera global y se inicializa al inicio de la petición.

Con esta prueba, los dispositivos mostraron tiempos de respuesta muy similares, independientemente de la velocidad que pueden desarrollar para la transmisión. Esto se puede deber a que número de peticiones HTTP al servidor es mínimo, y en cada petición se aprovecha al máximo el MTU de la red local, por lo que no se presenta gran diferencia a la hora de transmitir. Quizá en una red más grande o con un contenido Web complejo, la diferencia comience a notarse. Recordemos que el dispositivo basado en *Arduino* incorpora el controlador de red *Wiznet W5100* el cual tiene capacidad para velocidades de 10 Mb/s y 100Mb/s, a diferencia del *ENC28J60* que soporta sólo 10Mb/s.

Compatibilidad

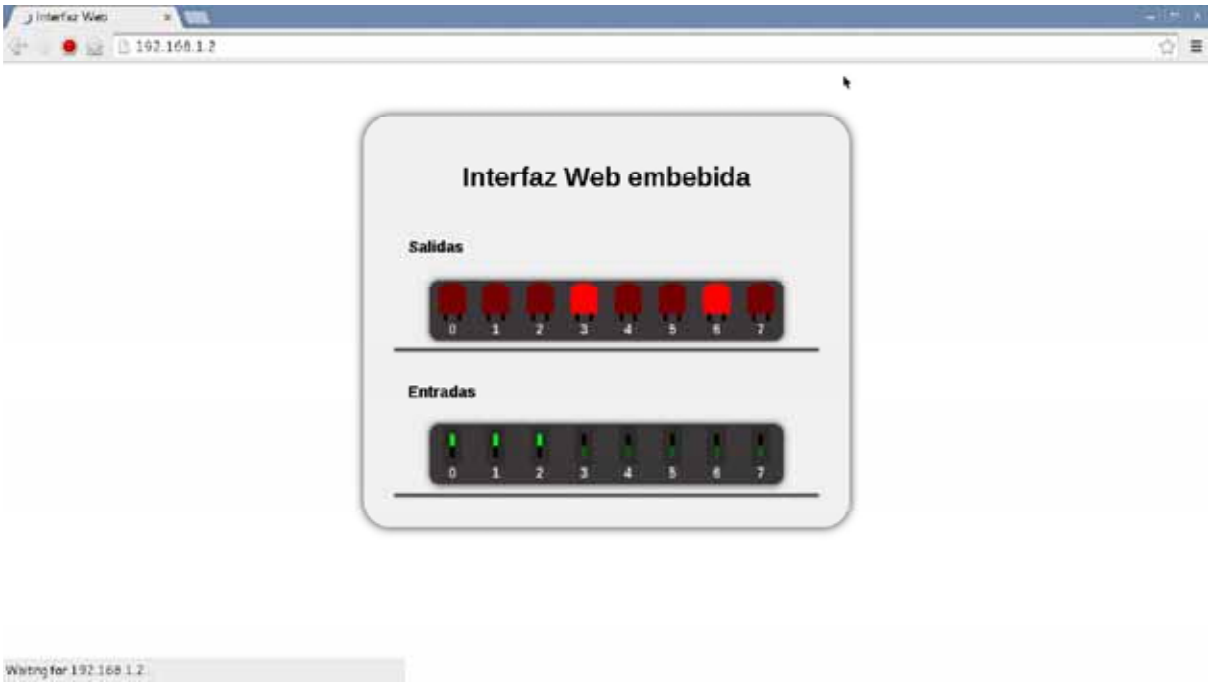
La interfaz se ejecutó en varios navegadores bajo distintos sistemas operativos, en una máquina de escritorio y en un dispositivo móvil.

La ejecución fue satisfactoria en todos los casos a excepción del navegador *Internet Explorer versión 7*, el cual no mostró de manera correcta los elementos de la página. Esto debido a que la implementación del DOM de HTML en el navegador de *Microsoft* no contempla al objeto *XMLHttpRequest()* con el cual se

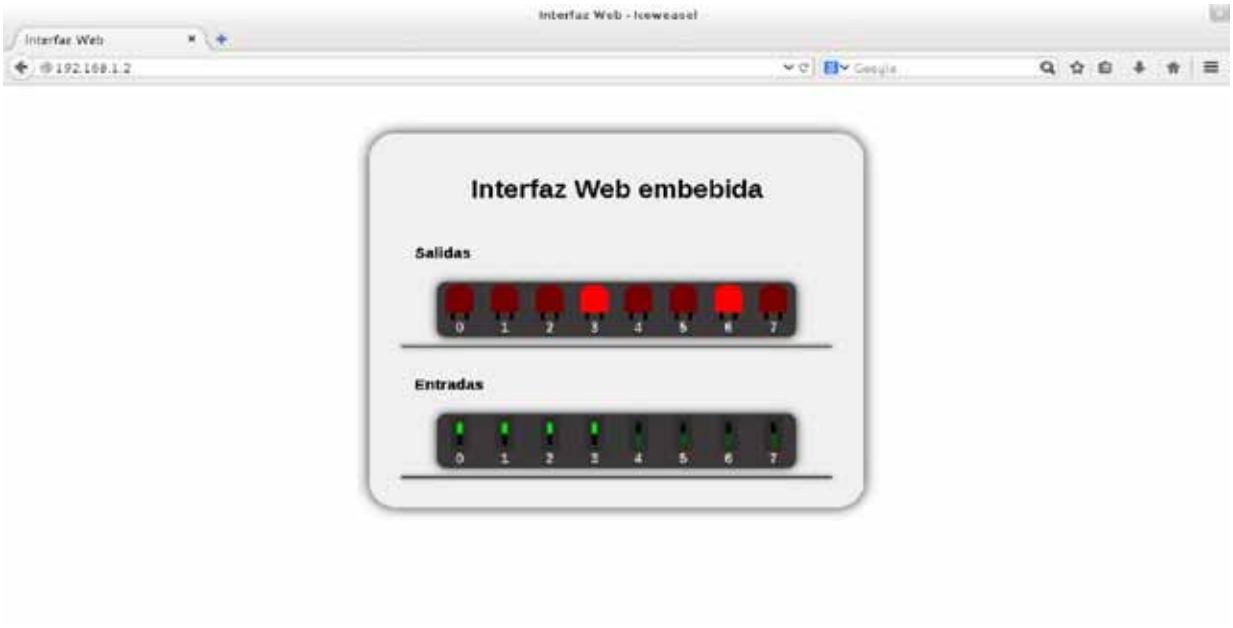
construyen los elementos de manera dinámica, y es usado como una estrategia *AJAX*, para evitar el *Postback* de la página y no generar tráfico innecesario para los dispositivos.

En seguida se muestran las capturas de pantalla de la ejecución en distintos dispositivos.

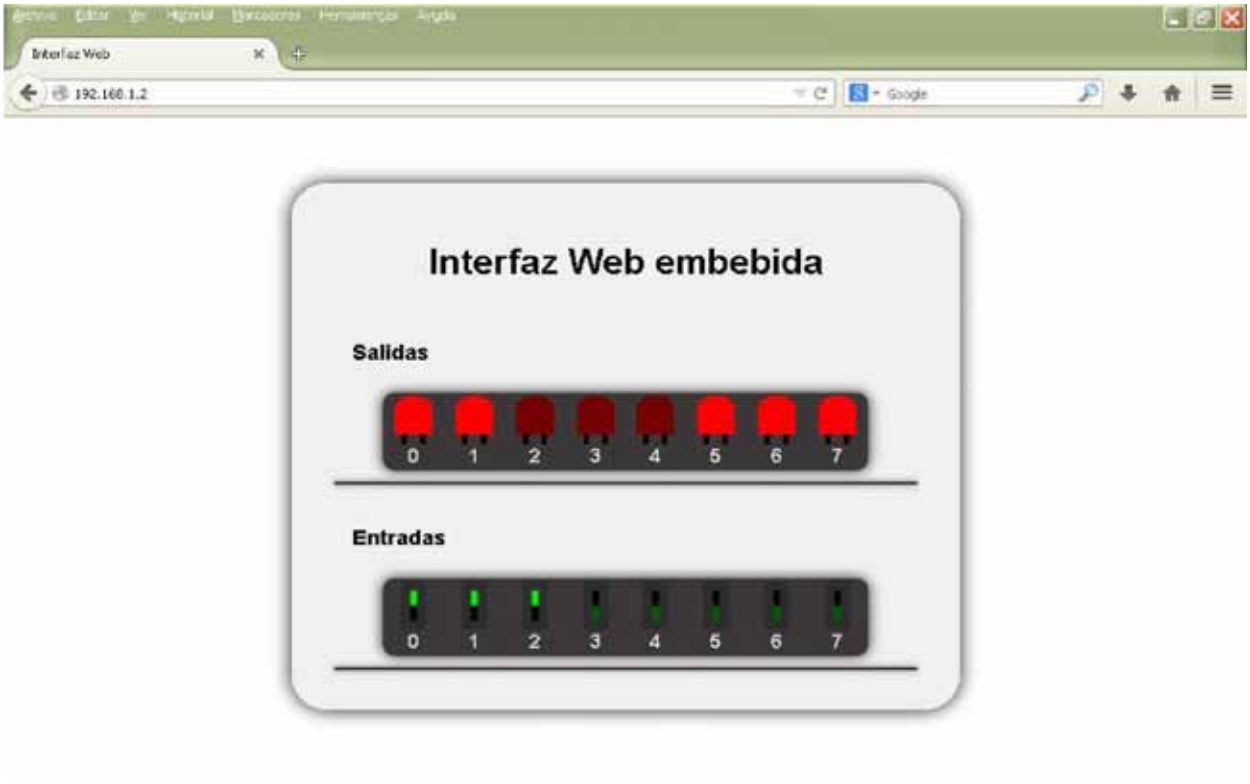
58 *Figura: Ejecución de la interfaz Web sobre un sistema Android con el navegador Google Chrome.*



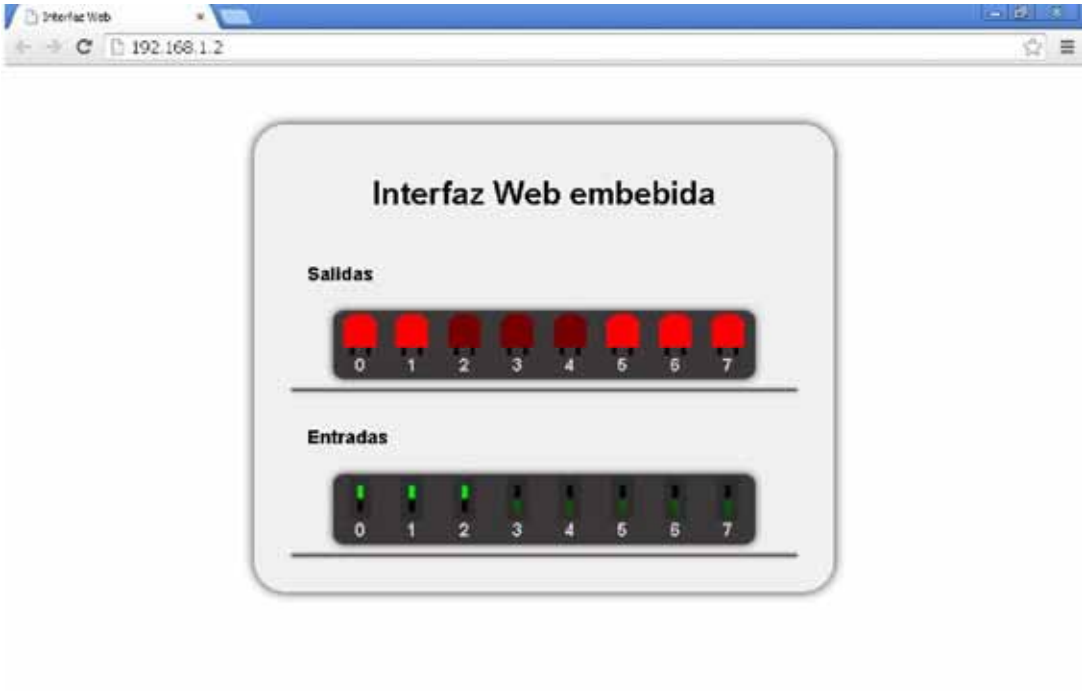
59 *Figura: Ejecución de la interfaz Web sobre un sistema Debian con Chromium.*



60 *Figura: Ejecución de la interfaz Web sobre un sistema Debian con Iceweasel.*

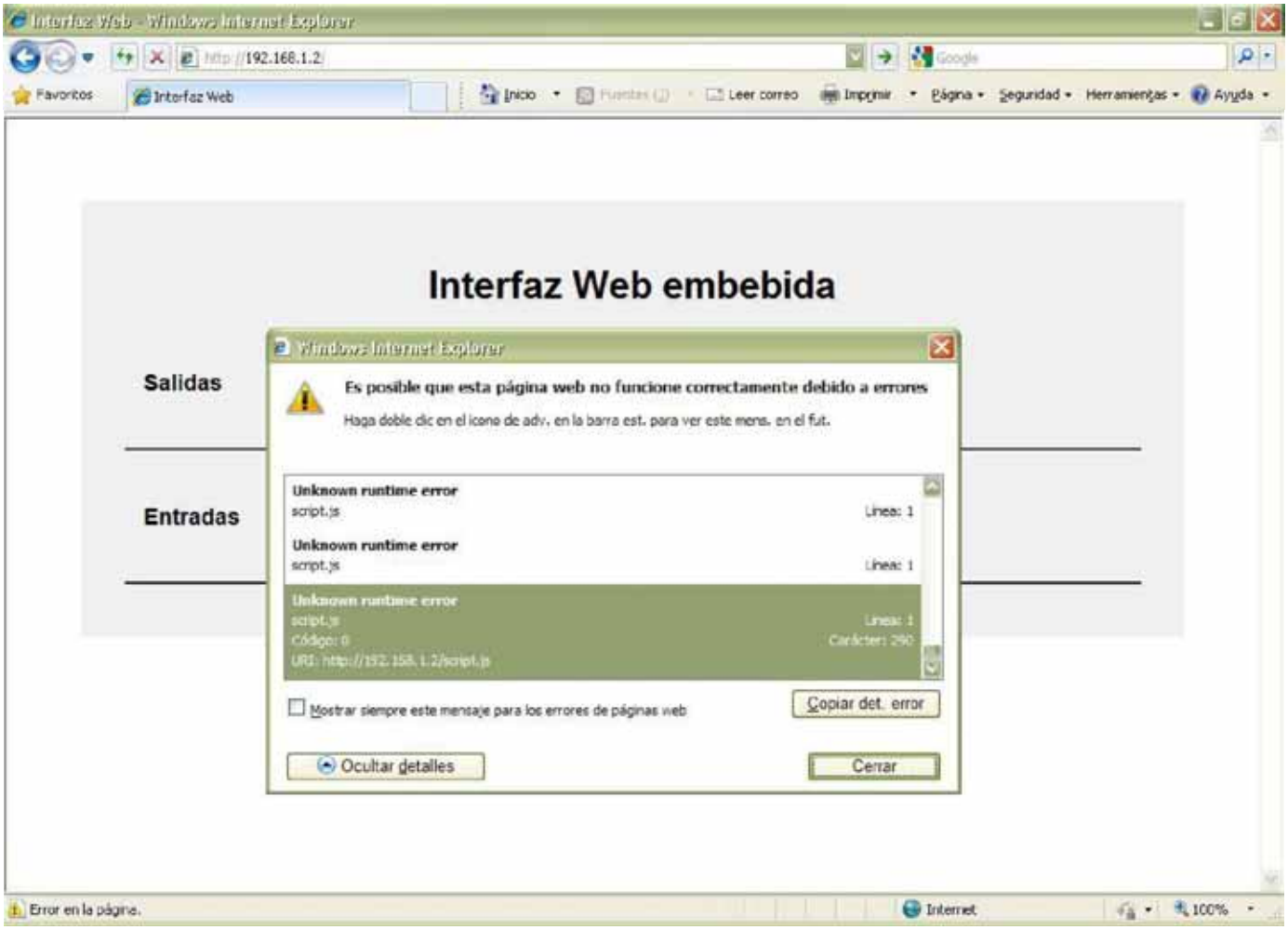


61 *Figura: Ejecución de la interfaz Web sobre un sistema Windows XP con Firefox.*



62 *Figura: Ejecución de la interfaz Web sobre un sistema Windows XP con Chromium.*

63 *Figura: Ejecución de la interfaz Web sobre un sistema Windows XP con Internet Explorer.*



64 *Figura: Ejecución de la interfaz Web sobre un sistema Windows XP con Internet Explorer, se muestra el error que ocasiona el uso del objeto XMLHttpRequest(), el cuál no se implementa en el árbol del DOM en el navegador de Microsoft.*

Como se ha mencionado antes, el único navegador problemático fue el *Internet Explorer 8* de *Microsoft.*

Costos

Para evaluar los costos se tomaron en cuenta los tiempos de desarrollo para cada sistema, el costo económico de cada uno y la dificultad para realizarlos.

Tiempos de desarrollo.

Los tiempos para el desarrollo del proyecto que se mencionaron en la propuesta, fueron modificados durante el transcurso del desarrollo.

Para el sistema hecho a la medida, el tiempo total de diseño y desarrollo e implementación del sistema se redujo de catorce semanas a sólo once.

El tiempo de desarrollo del sistema basado en *Arduino*, tomó dos semanas, en comparación con las cuatro semanas propuestas en un inicio.

Una parte importante del tiempo requerido para el desarrollo del sistema hecho a la medida, se llevó

consiguiendo los componentes adecuados para su construcción. Otra parte importante del tiempo total, se necesitó para el diseño del circuito. Con el diseño listo y probado en la placa de prototipos, la construcción del mismo y su programación, no excedieron las tres semanas.

Esfuerzo

Sin duda, el desarrollo del sistema hecho a la medida, requirió de un mayor esfuerzo comparado con la dificultad de implementar el sistema basado en Arduino.

Se necesitó conseguir los componentes, que en algunos casos fueron difíciles de conseguir, como es el caso de los controladores de red *ENC28J60*, los cuales tuvieron que comprarse a través de una página de Internet, y esperar a que fueran enviados por paquetería. Otro problema fue conseguir el inductor de valor comercial, debido a que se pretendió evitar el mayor número de errores en el diseño de la placa, se eligió un valor estándar, el cual resultó con el problema mencionado.

Aprender a utilizar la herramienta de diseño de circuitos electrónicos *Kicad*, implicó una tarea poco difícil, ya que no se contaba con una preparación previa en el diseño de circuitos, ni conocimientos de diseño de hardware. Sin embargo, el problema se superó y el diseño se realizó con éxito.

La falta de práctica en la elaboración de tarjetas de circuito impresas con el método de planchado, también ocasionó dificultades a la hora de fabricar la placa. Pero con el apoyo del asesor se logró su fabricación.

Para el caso del dispositivo basado en *Arduino*, la construcción y en general, el trabajo con su electrónica, no representó esfuerzo en absoluto, a excepción del empleado para adquirir la tarjeta y su módulo de red.

Dinero

La fabricación del dispositivo hecho a la medida, requirió de la compra de varios componentes, los cuáles tienen precios que van desde $1 peso, hasta los $80 pesos. De acuerdo a la lista de componentes obtenida a partir del diseño del circuito, es posible calcular el costo de su fabricación, el cual no excedió los $350 pesos.

El detalle en seguida:

Componente	Precio ($336 pesos en total)
2 Condensadores de 33 pf.	$4 pesos
2 Condensadores de 18 pf.	$4 pesos
2 Condensadores de 10 uf.	$3 pesos
2 Condensadores de 100 nf.	$3 pesos
3 LED de 5 mm.	$2 pesos
3 Resistencias de 270 Ohms.	$1.50 pesos
4 Resistencias de 50 Ohms.	$2 pesos
2 Resistencias de 10 KOhms.	$1 peso
Resistencia de 2.8 KOhms.	$0.50 pesos
Cristal de 25 Mhz.	$14 pesos
Cristal de 16 Mhz.	$14 pesos

Inductor de 10 uH.	$15 pesos
Microcontrolador Atmega328P.	$80 pesos
Controlador de red ENC28J60.	$80 pesos
Conector RJ45 HR91115.	$50 pesos
Regulador LM7833.	$20 pesos
PCB	$12 pesos
Componentes adicionales como conectores para las entradas y salidas, y la alimentación.	$30 pesos

La tarjeta de desarrollo y su módulo de red, fueron adquiridos como tal, no se necesitó comprar ningún elemento adicional. El precio de ambos fue de $550 pesos, un costo un poco elevado en comparación con el costo del sistema hecho a la medida.

Resultados

Los resultados del análisis a los dos sistemas se muestran en la siguiente tabla comparativa:

	Arduino	***Sistema hecho a la medida***
Alimentación	5 voltios	5 voltios
Capacidad de difusión (Broadcast)	Si	Si
Ejecución de protocolo ARP	Si	Si
Ejecución del protocolo ICMP	Si	Si
Ejecución del protocolo TCP	Si	Si
Ejecución del protocolo UDP	Si	Si
Ejecución del protocolo IP	Si	Si
Ejecución del protocolo HTTP	Si	Si
Calidad en CSMA/CD (colisiones)	Alta	Regular
TTL en paquetes (LAN)	64	64
Tiempo promedio de PING	2.17 ms	19.21 ms
Tiempo promedio de comandos (Web)	13.9 ms	20.78 ms
Compatibilidad con Windows	Si	Si
Compatibilidad con Linux	Si	Si

Compatibilidad con Android	Si	Si
Compatibilidad en Firefox	Si	Si
Compatibilidad en Iceweasel	Si	Si
Compatibilidad en Google Chrome	Si	Si
Compatibilidad en Chromium	Si	Si
Compatibilidad en Internet Explorer	No	No
Tiempo en adquirir componentes	1 semana	2 semanas
Tiempo de diseño de la electrónica	1 día	5 semanas
Tiempo de ensamblado de los componentes	1 día	3 semanas
Tiempo de pruebas de la electrónica	1 día	1 semana
Tiempo total de desarrollo de la electrónica	2 semanas	11 semanas
Tiempo de desarrollo del programa	2 semanas	2 semanas
Costo en dinero del desarrollo	$ 550.00	$ 336.00
Dificultad en el diseño de la electrónica	Muy baja	Alta
Dificultad en adquirir componentes	Baja	Alta
Dificultad en ensamblado	Muy baja	Alta
Dificultad en el desarrollo del programa	Media	Alta
Dificultad para realizar pruebas	Baja	Media
Dificultad para operar (conexiones, control)	Baja	Media
Conocimientos de electrónica	Muy básicos	Avanzados
Conocimientos de programación	Básicos	Avanzados
Conocimientos en redes	Muy básicos	Avanzados
Conocimientos en diseño de circuitos	Ninguno	Intermedios
Conocimientos de electricidad	Muy básicos	Intermedios
Conocimientos de diseño Web	Intermedios	Intermedios
Conocimientos en comunicaciones	Muy básicos	Intermedios

Conocimientos de instrumentación	Muy básicos	Intermedios
Espacio de tarjeta (tamaño)	Grande (40 cm^2, 2 placas)	Mediano (38 cm^2)
Capacidad de expansión (nuevas funciones)	Alta	Muy alta
Capacidad de rediseño	Baja	Muy alta
Licencia de uso y desarrollo	GNU/GPL	La que se elija
Operación de la interfaz Web	Fácil	Fácil
Almacenamiento	Si (programando)	No (es necesario implementar en electrónica)
Comunicación USB	Si	No (es necesario implementar en electrónica)
Comunicación Serial	Si	Si
Comunicación SPI	Si	Si
Velocidad de reloj	16 Mhrz	Hasta 20 Mhrz

Análisis y discusión de resultados

Después de observar los resultados obtenidos tras la evaluación de ambos dispositivos, se pudo conocer la diferencia en desempeño, dificultad, tiempo y costo de desarrollo de cada uno, tales datos facilitan la tarea de análisis de los resultados.

Para diseñar una aplicación de esta naturaleza y complejidad se deben tomar en cuenta aspectos técnicos bastante profundos, tales como la comunicación por red, el diseño y construcción de la electrónica, el diseño y programación de algoritmos y programas tanto para dispositivos pequeños como para máquinas potentes como lo son las computadoras personales y los servidores de diversos servicios en red. También se debe mencionar la parte del diseño de interfaces para la Web, en donde el paradigma es muy distinto a los utilizados para la programación en escritorio y en dispositivos embebidos.

Como se ha visto durante el desarrollo del proyecto, fue necesario implementar la pila de protocolos de red TCP/IP bajo el esquema del modelo OSI, lo cual implica un conocimiento suficientemente amplio. Aspectos propios del modelo, tales como las tramas Ethernet, los datagramas IP, y otros conceptos fueron revisados con bastante claridad en el marco teórico.

La construcción del dispositivo hecho a la medida, requirió de un conocimiento suficiente en el campo de la electrónica, tanto para conocer el funcionamiento de los componentes como para el diseño y fabricación del circuito.

La parte de la programación es una parte amplia e importante, ya que se utilizaron diversas tecnologías dentro de varios paradigmas de desarrollo, en primera instancia, se usó la programación en lenguaje C para el funcionamiento del lado del servidor. Para el lado del cliente se usaron técnicas de Javascript conocidas como Ajax, y lenguaje HTML para la construcción de la interfaz Web.

Las pruebas de rendimiento y funcionamiento, se deben hacer tanto para la parte de la electrónica como para la parte de la programación, además de evaluar la calidad de la interfaz con el usuario final. Esto implica revisar la compatibilidad con diversos dispositivos que trabajarán en el lado del cliente, así como las aplicaciones que se utilizan para ejecutar la interfaz. La parte de la comunicación es un aspecto fundamental para definir la fiabilidad de los dispositivos.

Para las pruebas de la electrónica se utilizaron herramientas para medir voltajes, corriente, etcétera. Lo cual implica conocimientos en la operación de instrumentos tales como voltímetros, amperímetros, entre otros. El diseño fue probado en un inicio en una tarjeta de prototipos, posteriormente fue montado y soldado en una placa PCB, la cuál también se sometió a tales pruebas. Una vez construido el dispositivo, se le aplicaron pruebas de comunicación.

Después de programar los dispositivos, también se le aplicaron pruebas. Para ello se desarrollaron herramientas que evaluaron el desempeño de la aplicación en cada dispositivo. Para la construcción de dichas herramientas se ocuparon tecnologías Web y de escritorio.

Tanto el desarrollo de la electrónica como las pruebas de la misma, se hicieron prácticamente sólo para el dispositivo hecho a la medida. Para el dispositivo basado en Arduino, realmente no se necesita ningún conocimiento avanzado ni en programación, ni en electrónica, ni en diseño de dispositivos electrónicos.

Las pruebas para la interfaz implementada en ambos dispositivos, fueron aplicadas en los dos casos, y las herramientas utilizadas fueron las mismas.

En el tema de los gastos, se evaluaron tanto los datos económicos como los datos de tiempo y esfuerzo. Como se pudo ver, el desarrollo del sistema hecho a la medida tomó mucho más tiempo y esfuerzo que el basado en Arduino, sin embargo, la cantidad de dinero invertido fue menor para el desarrollo del

dispositivo.

En los aspectos de desempeño, ambos mostraron una mínima diferencia, aunque, en la velocidad de transmisión, el sistema Arduino tuvo la ventaja debido al controlador de red implementado en el *Shield Ethernet*.

Los resultados de la comparación, señalan grandes ventajas y desventajas para ambos dispositivos. Los aspectos técnicos y los conocimientos necesarios son fundamentales para uno, mientras que el otro ofrece un tiempo y esfuerzo de desarrollo mínimos. Mientras que, el desempeño de la Interfaz es el mismo en ambos casos.

Las diferencias en los resultados son notorios para el desarrollador, no así para el usuario final, quien es el que hará uso del sistema a través de su interfaz.

Conclusiones

El dispositivo hecho a la medida demostró tener un desempeño similar al sistema basado en Arduino, tanto la comunicación como la operación y ejecución de la aplicación a través de la interfaz Web, fueron exitosos.

Para el desarrollo del sistema hecho a la medida, si hubo una gran diferencia frente a Arduino, el trabajo y el tiempo empleados fueron mucho mayores, sin embargo los costos resultaron mucho menores.

En un desarrollo que tenga limitaciones de tiempo y esfuerzo, el dispositivo basado en Arduino resulta ser la mejor opción.

Sin embargo, si lo que se busca es economía y la aplicación de conocimientos avanzados, sin duda, el sistema hecho a la medida ofrece mejores opciones.

Ambos dispositivos proveen de una base sólida y funcional para la operación a través de una interfaz Web, lo que da al usuario final una capa de abstracción lo suficientemente efectiva para que el usuario no tenga que preocuparse por aspectos técnicos, que implican los desarrollos que sostienen a dicha aplicación.

El sistema hecho a la medida podría ser de gran beneficio en el ambiente académico, donde su naturaleza económica lo hace asequible para los estudiantes, además de exigir la aplicación de los conocimientos y técnicas adquiridas en su formación.

Por otro lado, un sistema prefabricado y comercial, como lo es Arduino, es una buena opción para la industria, donde el tiempo de desarrollo y el esfuerzo son de suma importancia, aún más que la misma economía del desarrollo.

Sin embargo, sería una buena idea que la industria mexicana comenzara a tomar en cuenta ambas formas de desarrollo, logrando así, tener menos dependencia de la tecnología ajena o privativa. Cabe mencionar que en el país existe la capacidad tecnológica y científica para comenzar a producir tecnología.

Bibliografía

1: Alonso Figueroa, Israel, "Implementación de una VPN con el microcontrolador Rabbit para acceso a una red de cámaras web de la Universidad Autónoma Metropolitana", Universidad Autónoma Metropolitana, 2012.

2: López Jarquín, "Control distribuido de los dispositivos eléctricos de un inmueble mediante TCP/IP", Universidad Autónoma Metropolitana, 2011.

3: Avendaño López, Luis David, "Implementación de un router en un sistema empotrado", Universidad Autónoma Metropolitana, 2012.

4: V. Georgitzikis, O. Akribopoulos y I. Chatzigiannakis, "Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks", Institute of Electrical and Electronics Engineers (IEEE), 2012.

5: Hernández Méndez Mauricio, Montaño Sánchez Cesar E. y Vázquez Benítez Luís A, "Desarrollo de un sistema SCADA para casa habitación", ESCOM-IPN, 2009.

6: Alvarado Gaspar Hiram Ernesto, Saavedra Frausto Armando Joel, Valencia Gordillo Cecilia, "Móvil para telemetría controlado vía remota", ESCOM-IPN, 2008.

7: ANDREW S. TANENBAUM, "Redes de computadoras", Prentice-Hall, 2002

8: STEVENS, W. Richard, "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley Professional, 2003

9: COMER, Douglas, "Internetworking with TCP/IP, Vol. III: Client-Server Programming and Applications, Linux/Posix Sockets Version", Prentice-Hall, 2000

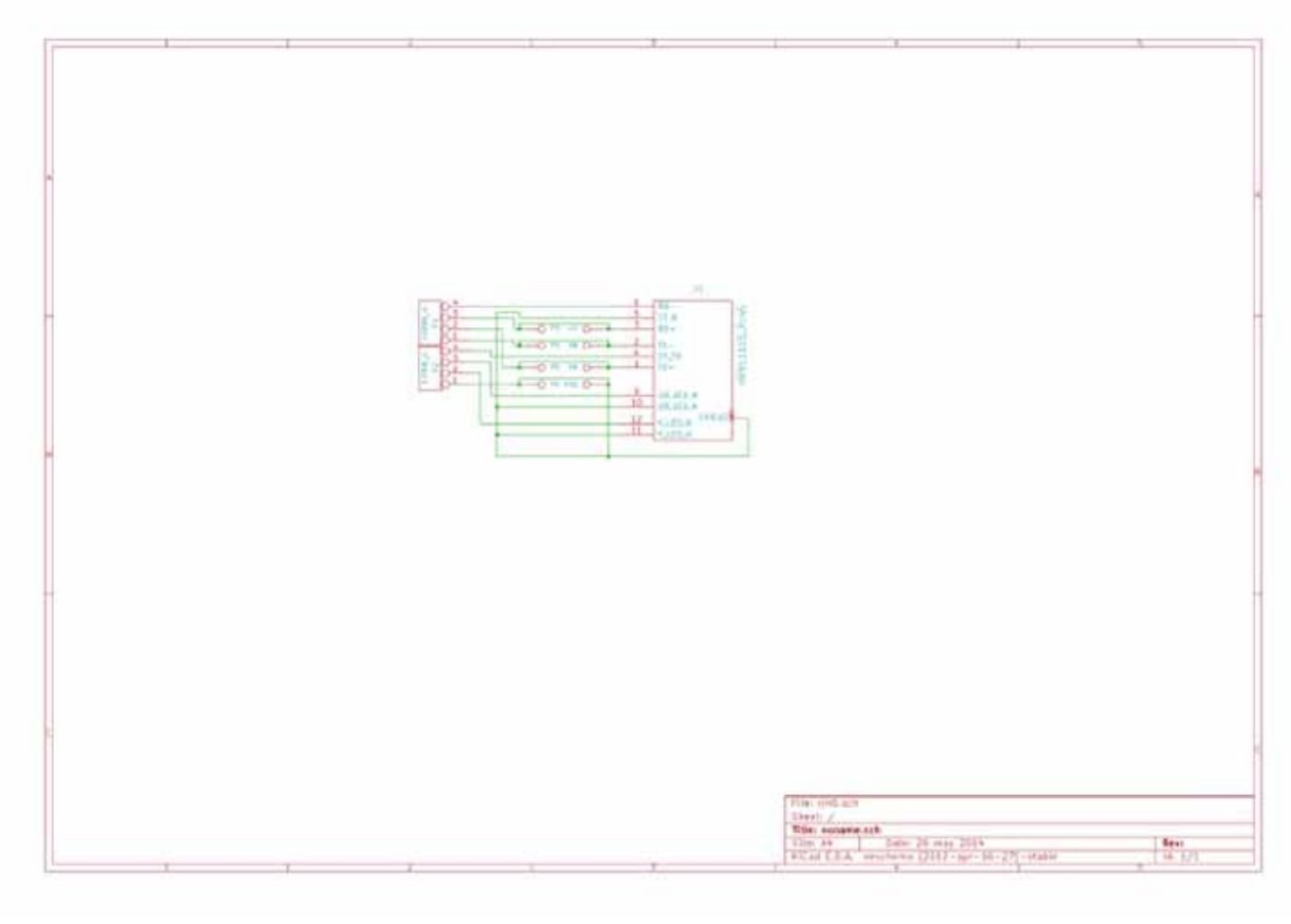
10: Jeremy Keith, "Dom Scripting: Web Design with JavaScript and the Document Object Model", Springer, 2005

11: Brian W. Kernighan, Dennis M. Ritchie, "El lenguaje de programación C", PRENTICE-HALL,

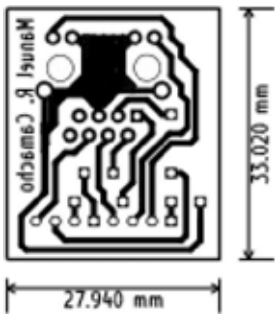
12: Michael Margolis, "Arduino Cookbook", O’Reilly Media,

13: Richard H. Barnett, Sarah Cox, Larry O'Cull, "Embedded C Programming and the Atmel AVR", Cengage Learning, 2006

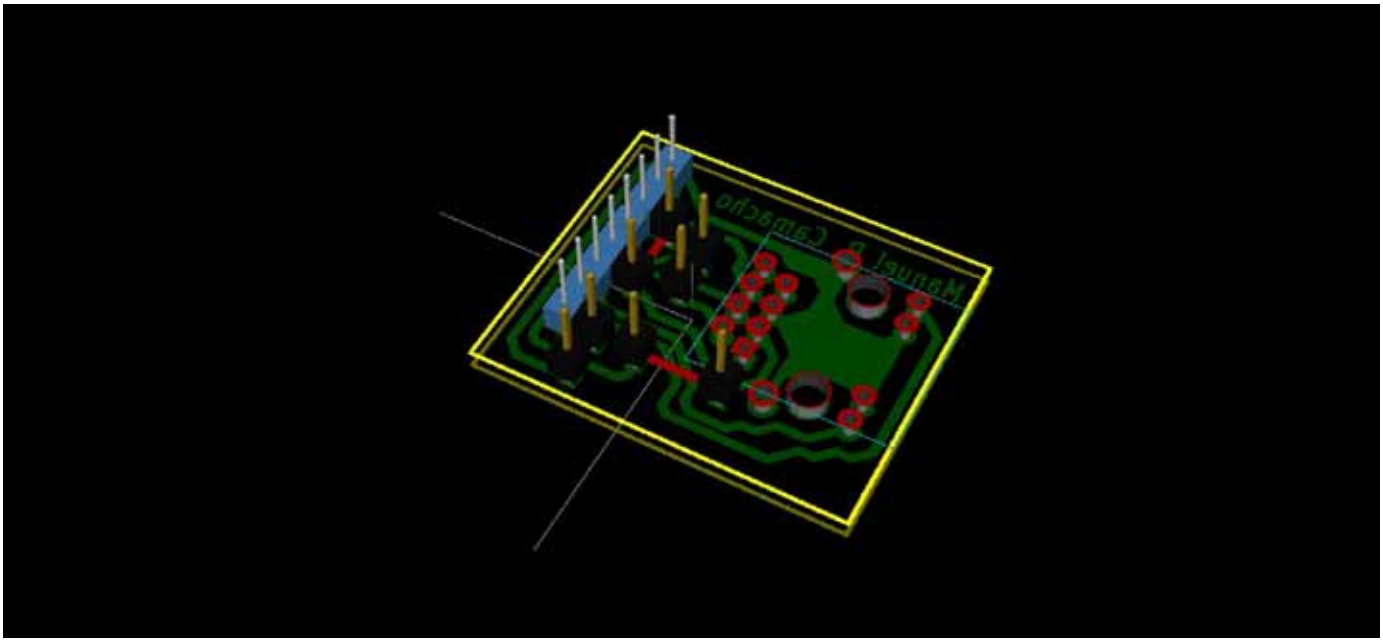
Apéndices



65 *Figura: Diagrama eléctrico de la placa auxiliar para el montaje del conector RJ45 HR911105.*



66 *Figura: Archivo de impresión de la placa auxiliar para el montaje del conector RJ45 HR911105 en la tarjeta de prototipos, para planchar en PCB.*



67 *Figura: Vista preliminar de la placa auxiliar para el montaje del conector RJ45 HR911105 en la tarjeta de prototipos.*

Archivo del módulo *hr911105_rj45.mod* para *Kicad*, del componente *HR911105*:

```
PCBNEW-LibModule-V1  dom 08 jun 2014 17:25:52 CDT
# encoding utf-8
$INDEX
HR911105_RJ45
$EndINDEX
$MODULE HR911105_RJ45
Po 0 0 0 15 5002FCB5 5394E1B9 ~~
Li HR911105_RJ45
Kw HR911105_RJ45
Sc 5394E1B9
AR HR911105_RJ45
Op 0 0 0
T0 -20 2598 600 600 0 120 N V 21 N "HR911105_RJ45"
T1 56 -40 394 394 0 80 N V 21 N "VAL**"
DS -3000 3100 3000 3100 50 21
DS 3000 3100 3000 -4000 50 21
DS 3000 -4000 -3000 -4000 50 21
DS -3000 -4000 -3000 3100 50 21
$PAD
Sh "Hole" C 1417 1417 0 0 0
Dr 1280 0 0
```

```
At STD N 00F0FFFF
Ne 0 ""
Po 2244 -98
$EndPAD
$PAD
Sh "Hole" C 1417 1417 0 0 0
Dr 1280 0 0
At STD N 00F0FFFF
Ne 0 ""
Po -2244 -79
$EndPAD
$PAD
Sh "1" R 700 700 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po -1732 -2579
$EndPAD
$PAD
Sh "2" C 800 800 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po -1260 -3583
$EndPAD
$PAD
Sh "3" C 800 800 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po -748 -2579
$EndPAD
$PAD
Sh "4" C 800 800 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po -236 -3583
$EndPAD
$PAD
Sh "5" C 800 800 0 0 0
```

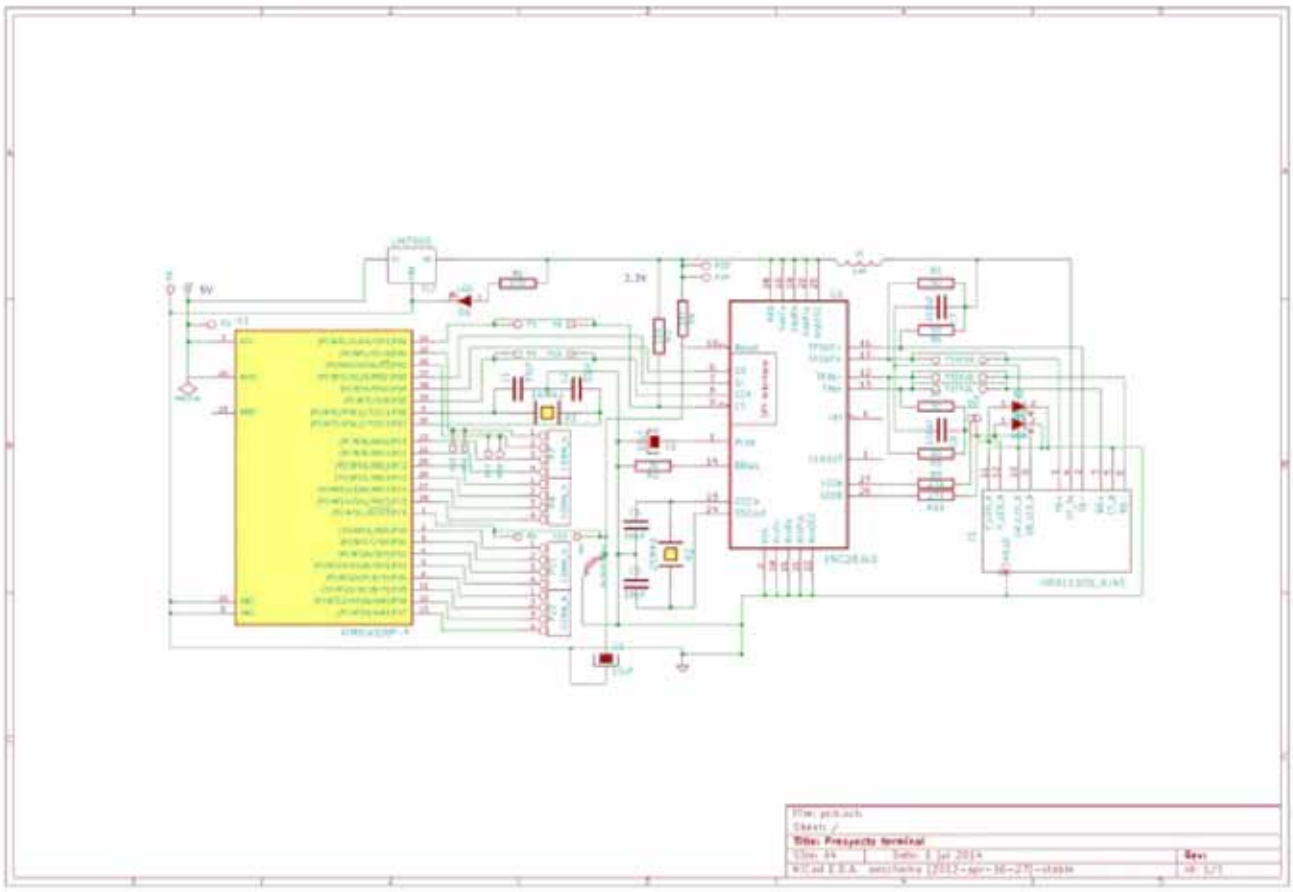
```
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po 236 -2579
$EndPAD
$PAD
Sh "6" C 800 800 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po 748 -3583
$EndPAD
$PAD
Sh "7" C 800 800 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po 1260 -2579
$EndPAD
$PAD
Sh "8" C 800 800 0 0 0
Dr 354 0 0
At STD N 00E0FFFF
Ne 0 ""
Po 1732 -3583
$EndPAD
$PAD
Sh "9" C 800 800 0 0 0
Dr 394 0 0
At STD N 00F0FFFF
Ne 0 ""
Po -2618 1996
$EndPAD
$PAD
Sh "10" C 800 800 0 0 0
Dr 394 0 0
At STD N 00F0FFFF
Ne 0 ""
Po -1614 1996
$EndPAD
$PAD
```

```
Sh "11" C 800 800 0 0 0
Dr 394 0 0
At STD N 00F0FFFF
Ne 0 ""
Po 1614 1996
$EndPAD
$PAD
Sh "12" C 800 800 0 0 0
Dr 394 0 0
At STD N 00F0FFFF
Ne 0 ""
Po 2618 1996
$EndPAD
$PAD
Sh "0" C 1000 1000 0 0 0
Dr 630 0 0
At STD N 00F0FFFF
Ne 0 ""
Po -3012 -1260
$EndPAD
$PAD
Sh "0" C 1000 1000 0 0 0
Dr 630 0 0
At STD N 00F0FFFF
Ne 0 ""
Po 3051 -1260
$EndPAD
$EndMODULE  HR911105_RJ45
$EndLIBRARY
```

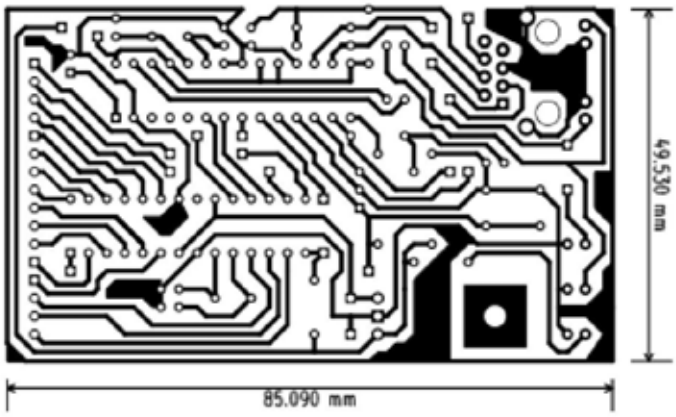
Biblioteca *hr911105_rj45.lib* para cargar módulo de construcción para *Kicad* del componente *HR911105*:

```
EESchema-LIBRARY Version 2.3  Date: 15/07/2012 19:22:14
#encoding utf-8
#
# HR911105_RJ45
#
DEF HR911105_RJ45 J 0 40 Y Y 1 F N
F0 "J" 150 650 60 H V C CNN
F1 "HR911105_RJ45" 550 150 60 V V C CNN
DRAW
```

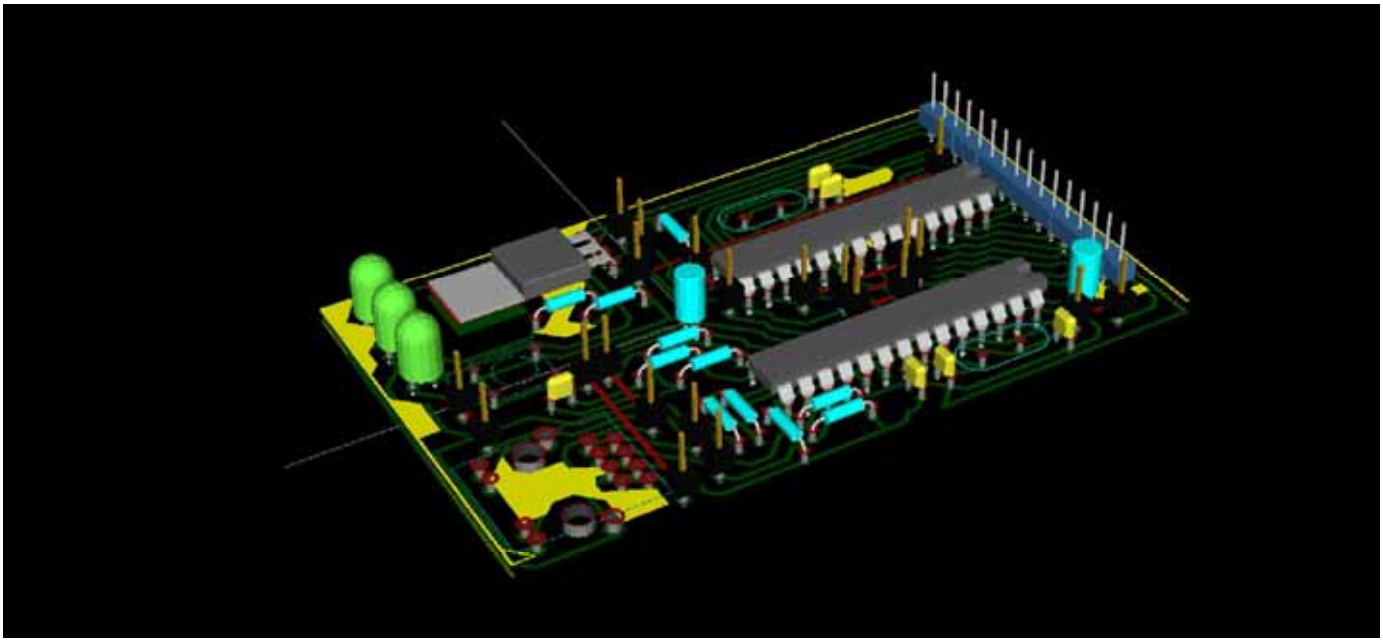
```
S -250 550 450 -700 0 1 0 N
X SHIELD 0 450 -500 0 L 50 50 1 1 W X
X TD+ 1 -550 -50 300 R 50 50 1 1 I
X TD- 2 -550 150 300 R 50 50 1 1 I
X RD+ 3 -550 300 300 R 50 50 1 1 I
X CT_TX 4 -550 50 300 R 50 50 1 1 I
X CT_R 5 -550 400 300 R 50 50 1 1 I
X RD- 6 -550 500 300 R 50 50 1 1 I
X GR_LED_A 9 -550 -300 300 R 50 50 1 1 I
X GR_LED_K 10 -550 -400 300 R 50 50 1 1 I
X Y_LED_K 11 -550 -650 300 R 50 50 1 1 I
X Y_LED_A 12 -550 -550 300 R 50 50 1 1 I
ENDDRAW
ENDDEF
#
#End Library
```



68 *Figura: Diagrama eléctrico del dispositivo hecho a la medida.*



69 *Figura: Impresión del circuito del dispositivo hecho a la medida para planchar PCB.*



70 *Figura: Vista preliminar del dispositivo hecho a la medida.*

Código del programa del sistema hecho a la medida:

```
#include <EtherCard.h>

static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0xFF };
static byte myip[] = { 192,168,1,2 };

int ldPin[] = { 3,2,1,0,4,5,6,7 };
```

```
int swPin[] = { 9,10,A0,A1,A2,A3,A4,A5 };

byte Ethernet::buffer[1300];
BufferFiller bfill;

void setup () {
  //Serial.begin(9600);
  //Serial.println( "Iniciando...");

  pinMode(ldPin[0], OUTPUT);
  pinMode(ldPin[1], OUTPUT);
  pinMode(ldPin[2], OUTPUT);
  pinMode(ldPin[3], OUTPUT);
  pinMode(ldPin[4], OUTPUT);
  pinMode(ldPin[5], OUTPUT);
  pinMode(ldPin[6], OUTPUT);
  pinMode(ldPin[7], OUTPUT);
  pinMode(swPin[0], INPUT);
  pinMode(swPin[1], INPUT);
  pinMode(swPin[2], INPUT);
  pinMode(swPin[3], INPUT);
  pinMode(swPin[4], INPUT);
  pinMode(swPin[5], INPUT);
  pinMode(swPin[6], INPUT);
  pinMode(swPin[7], INPUT);

  if (
    ether.begin(sizeof Ethernet::buffer, mymac) == 0){
//    Serial.println( "Fallo la conexion con el controlador Ethernet");
  }
  else{
//    Serial.println( "Se conecto al controlador Ethernet");
  }
  ether.staticSetup(myip);
  //Serial.println( "Iniciado");
}

static word estadoLS(int valLdPin[8],int valSwPin[8]) {
  bfill = ether.tcpOffset();
  bfill.emit_p(
    PSTR("'$D$D$D$D$D$D$D$D','$D$D$D$D$D$D$D$D'"),
```

```
    valLdPin[0],
    valLdPin[1],
    valLdPin[2],
    valLdPin[3],
    valLdPin[4],
    valLdPin[5],
    valLdPin[6],
    valLdPin[7],
    valSwPin[0],
    valSwPin[1],
    valSwPin[2],
    valSwPin[3],
    valSwPin[4],
    valSwPin[5],
    valSwPin[6],
    valSwPin[7]
  );
  return bfill.position();
}

static word homePage(int valLdPin[8],int valSwPin[8]) {
  bfill = ether.tcpOffset();
  bfill.emit_p(
    PSTR(
      "<html>"
      "<head>"
      "<meta charset=\"UTF-8\">"
      "<title>Interfaz Web</title>"
      "<script src=\"./script.js\"></script>"
      "<script src=\"./img.js\"></script>"
      "<link href=\"./style.css\" rel=\"stylesheet\" type=\"text/css\">"
      "</head>"
      "<body onLoad=\"ini('$D$D$D$D$D$D$D$D','$D$D$D$D$D$D$D$D');\">"
      "<div class=\"pnl\">"
      "<div class=\"ttl\">Interfaz Web embebida</div>"
      "<div class=\"lbl\">Salidas</div>"
      "<table id=\"lds\"></table>"
      "<hr />"
      "<div class=\"lbl\">Entradas</div>"
      "<table id=\"sws\"></table>"
      "<hr />"
```

```
      "</div>"
      "</body>"
      "</html>"
    ),
    valLdPin[0],
    valLdPin[1],
    valLdPin[2],
    valLdPin[3],
    valLdPin[4],
    valLdPin[5],
    valLdPin[6],
    valLdPin[7],
    valSwPin[0],
    valSwPin[1],
    valSwPin[2],
    valSwPin[3],
    valSwPin[4],
    valSwPin[5],
    valSwPin[6],
    valSwPin[7]
    );
  return bfill.position();
}

static word imgJS() {
  bfill = ether.tcpOffset();
  bfill.emit_p(
    PSTR(
      "var img=["

"\"
ZCB3aXRoIEdJTVAALAAAAAAWAB4AAAI5nB+pe40LmRuxKmdzeNrSnglgOJbmiabqyrbuC8fyTKMNgE8GD
ugD7wPqhBOiw3jLDZVF5tGZ7B0KADs=\","

"\"
ZCB3aXRoIEdJTVAALAAAAAAWAB4AAAI5nB+pe40LmRuxKmdzeNrSngngSJbmiabqyrbuC8fyHDfAPRk3k
A97/8sFJ0NH0YYTJolLYxPJOxQAADs=\","

"\"
ZCB3aXRoIEdJTVAALAAAAAAIABAAAAITjI+pKsKe2psJAGXxrRvlBYZGAQA7\","
```

```
"\"
SU1QACwAAAAACAAQAAACE4yPqQrAntqbSQhl8a0b5QWGRgEAOw==\"];"
    )
  );
  return bfill.position();
}

static word javaScript() {
  bfill = ether.tcpOffset();
  bfill.emit_p(
    PSTR(
      "var t=setInterval(function (){jx('./EDO')},500);"
      "function ini(lds,sws){"
      "var ldx='<tr>';"
      "var swx='<tr>';"
      "for(i=0;i<8;i++){"
      "ldx+='<td><img onclick=\\\"jx(\\'./LED=' + i + '\\');\\\" src=\\\"' +
img[1-lds[i]] + '\\\" /><br>' + i + '</td>';"
      "swx+='<td><img src=\\\"' + img[3-sws[i]] + '\\\" /><br>' + i + '</td>';"
      "}"
      "document.getElementById('lds').innerHTML=ldx + '</tr>';"
      "document.getElementById('sws').innerHTML=swx + '</tr>';"
      "}"
      "function jx(URL){"
      "xh=new XMLHttpRequest();"
      "xh.onreadystatechange=function(){"
      "if(xh.readyState==4 && xh.status==200){"
      "eval('ini(' + xh.responseText + ')');"
      "}\r\n"
      "}\r\n"
      "xh.open(\"GET\",URL,false);"
      "xh.send(null);"
      "}"
    )
  );
  return bfill.position();
}

static word cssStyle() {
  bfill = ether.tcpOffset();
  bfill.emit_p(
```

```
  PSTR(
    "body{"
    "padding: 50px;"
    "font-family: Arial;"
    "}"
    "img{"
    "cursor: pointer;"
    "height: 42px;"
    "}"
    "hr{"
    "margin: 10px;"
    "color: #1C1C1C;"
    "background-color: #3C383A;"
    "height: 2px;"
    "border: 0;"
    "box-shadow: 0 0 5px #1C1C1C;"
    "}"
    "div{"
    "padding: 25px;"
    "font-weight: 800;"
    "}"
    "table{"
    "margin: auto;"
    "width: 400px;"
    "font-weight: 800;"
    "border-radius: 10px;"
    "box-shadow: 0 0 10px #1C1C1C;"
    "background-color: #3C383A;"
    "color: #DFDFDF; "
    "}"
    "td{"
    "text-align: center;"
    "}"
    ".ttl{"
    "font-size: 30px;"
    "text-align: center;"
    "}"
    ".lbl{"
    "font-size: 18px;"
    "}"
    ".pnl{"
```

```
      "max-width: 500px;"
      "margin: auto;"
      "border-radius: 30px;"
      "box-shadow: 0 0 15px #1C1C1C;"
      "background-color: #F0F0F0;"
      "}"
    )
  );
  return bfill.position();
}

void loop () {
  word len = ether.packetReceive();
  word pos = ether.packetLoop(len);

  char * solicitud;
  char * queryString;
  int ldPinSol;

  int valSwPin[8];
  int valLdPin[8];

  valLdPin[0] = bitRead(PORTD, 3);
  valLdPin[1] = bitRead(PORTD, 2);
  valLdPin[2] = bitRead(PORTD, 1);
  valLdPin[3] = bitRead(PORTD, 0);
  valLdPin[4] = bitRead(PORTD, 4);
  valLdPin[5] = bitRead(PORTD, 5);
  valLdPin[6] = bitRead(PORTD, 6);
  valLdPin[7] = bitRead(PORTD, 7);

  valSwPin[0] = digitalRead(swPin[0]);
  valSwPin[1] = digitalRead(swPin[1]);
  valSwPin[2] = digitalRead(swPin[2]);
  valSwPin[3] = digitalRead(swPin[3]);
  valSwPin[4] = digitalRead(swPin[4]);
  valSwPin[5] = digitalRead(swPin[5]);
  valSwPin[6] = digitalRead(swPin[6]);
  valSwPin[7] = digitalRead(swPin[7]);

  if (pos){
```

```
    solicitud = (char *)Ethernet::buffer + pos;

//    Serial.println(solicitud);
    if(strstr(solicitud, "GET /EDO") != 0) {
    //  Serial.println("Se solicito estado");
      ether.httpServerReply(estadoLS(valLdPin,valSwPin));
    }
    else if(strstr(solicitud, "GET /img.js") != 0) {
    //  Serial.println("Se solicito imagenes javaScript");
      ether.httpServerReply(imgJS());
    }
    else if(strstr(solicitud, "GET /script.js") != 0) {
    //  Serial.println("Se solicito javaScript");
      ether.httpServerReply(javaScript());
    }
    else if(strstr(solicitud, "GET /style.css") != 0) {
    //  Serial.println("Se solicito css");
      ether.httpServerReply(cssStyle());
    }
    else if(strstr(solicitud, "GET /LED=") != 0){
      ldPinSol = strstr(solicitud, "GET /LED=")[9] - 48;
      //Serial.println("Se solicito Pin ");
      valLdPin[ldPinSol] = 1 - valLdPin[ldPinSol];
      digitalWrite(ldPin[ldPinSol], valLdPin[ldPinSol]);
      ether.httpServerReply(estadoLS(valLdPin,valSwPin));
    }
    else if(strstr(solicitud, "GET /LEDS=") != 0){
      solicitud = strtok(strstr(solicitud, "GET /LEDS="),"=");
      solicitud = strtok(NULL, "\n");
      //Serial.println("Se solicito Pin ");
      for(int i=0; i<8;i++){
        valLdPin[i] = solicitud[i] - 48;
        digitalWrite(ldPin[i], valLdPin[i]);
      }
      ether.httpServerReply(estadoLS(valLdPin,valSwPin));
    }
    //PRUEBAS
    else if(strstr(solicitud, "GET /prueba") != 0) {
    //  Serial.println("Se solicito imagenes javaScript");
      ether.httpServerReply(imgJS());
```

```
      ether.httpServerReply(cssStyle());
    }
    else{
     //  Serial.println("Se solicito index ");
      ether.httpServerReply(homePage(valLdPin,valSwPin));
    }
  }
}
```

Código del programa del sistema basado en *Arduino*:

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,100);
EthernetServer server(80);

int ldPin[] = { 3,2,1,0,4,5,6,7 };
int swPin[] = { 9,10,A0,A1,A2,A3,A4,A5 };

void setup()
{
  pinMode(ldPin[0], OUTPUT);
  pinMode(ldPin[1], OUTPUT);
  pinMode(ldPin[2], OUTPUT);
  pinMode(ldPin[3], OUTPUT);
  pinMode(ldPin[4], OUTPUT);
  pinMode(ldPin[5], OUTPUT);
  pinMode(ldPin[6], OUTPUT);
  pinMode(ldPin[7], OUTPUT);
  pinMode(swPin[0], INPUT);
  pinMode(swPin[1], INPUT);
  pinMode(swPin[2], INPUT);
  pinMode(swPin[3], INPUT);
  pinMode(swPin[4], INPUT);
  pinMode(swPin[5], INPUT);
  pinMode(swPin[6], INPUT);
  pinMode(swPin[7], INPUT);

  Ethernet.begin(mac, ip);
  server.begin();
```

```
}

String estadoLS(int valLdPin[8],int valSwPin[8]) {
  String cadena = "'";

  cadena.concat(valLdPin[0]);
  cadena.concat(valLdPin[1]);
  cadena.concat(valLdPin[2]);
  cadena.concat(valLdPin[3]);
  cadena.concat(valLdPin[4]);
  cadena.concat(valLdPin[5]);
  cadena.concat(valLdPin[6]);
  cadena.concat(valLdPin[7]);
  cadena.concat("','");
  cadena.concat(valSwPin[0]);
  cadena.concat(valSwPin[1]);
  cadena.concat(valSwPin[2]);
  cadena.concat(valSwPin[3]);
  cadena.concat(valSwPin[4]);
  cadena.concat(valSwPin[5]);
  cadena.concat(valSwPin[6]);
  cadena.concat(valSwPin[7]);
  cadena.concat("'");
  return(cadena);
}

String homePage(int valLdPin[8],int valSwPin[8]) {
  String cadena = "<html>..."
      "<head>"
      "<meta charset=\"UTF-8\">"
      "<title>Interfaz Web</title>"
      "<script src=\"./script.js\"></script>"
      "<script src=\"./img.js\"></script>"
      "<link href=\"./style.css\" rel=\"stylesheet\" type=\"text/css\">"
      "</head>"
      "<body onLoad=\"ini('";
  cadena.concat(valLdPin[0]);
  cadena.concat(valLdPin[1]);
  cadena.concat(valLdPin[2]);
  cadena.concat(valLdPin[3]);
  cadena.concat(valLdPin[4]);
```

```
  cadena.concat(valLdPin[5]);
  cadena.concat(valLdPin[6]);
  cadena.concat(valLdPin[7]);
  cadena.concat("','");
  cadena.concat(valSwPin[0]);
  cadena.concat(valSwPin[1]);
  cadena.concat(valSwPin[2]);
  cadena.concat(valSwPin[3]);
  cadena.concat(valSwPin[4]);
  cadena.concat(valSwPin[5]);
  cadena.concat(valSwPin[6]);
  cadena.concat(valSwPin[7]);
  cadena.concat("');\">"
      "<div class=\"pnl\">"
      "<div class=\"ttl\">Interfaz Web embebida</div>"
      "<div class=\"lbl\">Salidas</div>"
      "<table id=\"lds\"></table>"
      "<hr />"
      "<div class=\"lbl\">Entradas</div>"
      "<table id=\"sws\"></table>"
      "<hr />"
      "</div>"
      "</body>"
      "</html>");

  return(cadena);
}

String imgJS() {
  String cadena = "var img=["

"\"
ZCB3aXRoIEdJTVAALAAAAAAWAB4AAAI5nB+pe40LmRuxKmdzeNrSnglgOJbmiabqyrbuC8fyTKMNgE8GD
ugD7wPqhBOiw3jLDZVF5tGZ7B0KADs=\","

"\"
ZCB3aXRoIEdJTVAALAAAAAAWAB4AAAI5nB+pe40LmRuxKmdzeNrSngngSJbmiabqyrbuC8fyHDfAPRk3k
A97/8sFJ0NH0YYTJolLYxPJOxQAADs=\","

"\"
ZCB3aXRoIEdJTVAALAAAAAAIABAAAAITjI+pKsKe2psJAGXxrRvlBYZGAQA7\","
```

```
"\"
SU1QACwAAAAACAAQAAACE4yPqQrAntqbSQhl8a0b5QWGRgEAOw==\"];";
  return(cadena);
}

String javaScript() {
  String cadena = "var t=setInterval(function (){jx('./EDO')},500);"
    "function ini(lds,sws){"
    "var ldx='<tr>';"
    "var swx='<tr>';"
    "for(i=0;i<8;i++){"
    "ldx+='<td><img onclick=\\\"jx(\\'./LED=' + i + '\\');\\\" src=\\\"' + img[1-
lds[i]] + '\\\" /><br>' + i + '</td>';"
    "swx+='<td><img src=\\\"' + img[3-sws[i]] + '\\\" /><br>' + i + '</td>';"
    "}"
    "document.getElementById('lds').innerHTML=ldx + '</tr>';"
    "document.getElementById('sws').innerHTML=swx + '</tr>';"
    "}"
    "function jx(URL){"
    "xh=new XMLHttpRequest();"
    "xh.onreadystatechange=function(){"
    "if(xh.readyState==4 && xh.status==200){"
    "eval('ini(' + xh.responseText + ')');"
    "}\r\n"
    "}\r\n"
    "xh.open(\"GET\",URL,false);"
    "xh.send(null);"
    "}";
  return(cadena);
}

String cssStyle() {
  String cadena = "body{"
    "padding: 50px;"
    "font-family: Arial;"
    "}"
    "img{"
    "cursor: pointer;"
    "height: 42px;"
    "}"
```

```
    "hr{"
    "margin: 10px;"
    "color: #1C1C1C;"
    "background-color: #3C383A;"
    "height: 2px;"
    "border: 0;"
    "box-shadow: 0 0 5px #1C1C1C;"
    "}"
    "div{"
    "padding: 25px;"
    "font-weight: 800;"
    "}"
    "table{"
    "margin: auto;"
    "width: 400px;"
    "font-weight: 800;"
    "border-radius: 10px;"
    "box-shadow: 0 0 10px #1C1C1C;"
    "background-color: #3C383A;"
    "color: #DFDFDF; "
    "}"
    "td{"
    "text-align: center;"
    "}"
    ".ttl{"
    "font-size: 30px;"
    "text-align: center;"
    "}"
    ".lbl{"
    "font-size: 18px;"
    "}"
    ".pnl{"
    "max-width: 500px;"
    "margin: auto;"
    "border-radius: 30px;"
    "box-shadow: 0 0 15px #1C1C1C;"
    "background-color: #F0F0F0;"
    "}";
  return(cadena);
}
```

```
void loop()
{
  int ldPinSol;
  int valSwPin[8];
  int valLdPin[8];

  valLdPin[0] = bitRead(PORTD, 3);
  valLdPin[1] = bitRead(PORTD, 2);
  valLdPin[2] = bitRead(PORTD, 1);
  valLdPin[3] = bitRead(PORTD, 0);
  valLdPin[4] = bitRead(PORTD, 4);
  valLdPin[5] = bitRead(PORTD, 5);
  valLdPin[6] = bitRead(PORTD, 6);
  valLdPin[7] = bitRead(PORTD, 7);

  valSwPin[0] = digitalRead(swPin[0]);
  valSwPin[1] = digitalRead(swPin[1]);
  valSwPin[2] = digitalRead(swPin[2]);
  valSwPin[3] = digitalRead(swPin[3]);
  valSwPin[4] = digitalRead(swPin[4]);
  valSwPin[5] = digitalRead(swPin[5]);
  valSwPin[6] = digitalRead(swPin[6]);
  valSwPin[7] = digitalRead(swPin[7]);

  EthernetClient client = server.available();

  if (client) {
    boolean currentLineIsBlank = true;
    String cadena="";

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        cadena.concat(c);

        if (c == '\n' && currentLineIsBlank) {

            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println();
```

```
            if(cadena.indexOf("EDO")){
              client.println(estadoLS(valLdPin,valSwPin));
            }
            if(cadena.indexOf("img.js")){
              client.println(imgJS());
            }
            if(cadena.indexOf("scripts.js")){
              client.println(javaScript());
            }
            if(cadena.indexOf("styles.css")){
              client.println(cssStyle());
            }
            if(cadena.indexOf("LED=")){
              ldPinSol = cadena[cadena.indexOf("LED=") + 5] - 48;
              //Serial.println("Se solicito Pin ");
              valLdPin[ldPinSol] = 1 - valLdPin[ldPinSol];
              digitalWrite(ldPin[ldPinSol], valLdPin[ldPinSol]);
              client.println(homePage(valLdPin,valSwPin));
            }
            if(cadena.indexOf("index.html")){
              client.println(homePage(valLdPin,valSwPin));
            }
            break;
        }
        if (c == '\n') {
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }
    delay(1);
    client.stop();
  }
}
```

Código de la interfaz Web (*index.html*):

```
<html>
      <head>
      <meta charset="UTF-8">
```

```
            <title>Interfaz Web</title>
            <script src="./script.js"></script>
            <script src="./img.js"></script>
            <link href="./style.css" rel="stylesheet" type="text/css">
      </head>
      <body onLoad="ini('11000111','11100000');">
            <div class="pnl">
            <div class="ttl">Interfaz Web embebida</div>
            <div class="lbl">Salidas</div>
                  <table id="lds">
                  </table><hr />
                  <div class="lbl">Entradas</div>
                  <table id="sws">
                  </table>
                  <hr />
            </div>
      </body>
</html>
```

Código de la interfaz Web (*scripts.js*):

```
var t=setInterval(
      function (){
            jx('./EDO')
            },500);
      function ini(lds,sws){
            var ldx='<tr>';
            var swx='<tr>';
            for(i=0;i<8;i++){
                  ldx+='<td><img onclick=\"jx(\'./LED=' + i + '\');\" src=\"' +
img[1-lds[i]] + '\" /><br>' + i + '</td>';
                  swx+='<td><img src=\"' + img[3-sws[i]] + '\" /><br>' + i +
'</td>';
            }
      document.getElementById('lds').innerHTML=ldx + '</tr>';
      document.getElementById('sws').innerHTML=swx + '</tr>';
      }

function jx(URL){
      xh=new XMLHttpRequest();
      xh.onreadystatechange=function(){
            if(xh.readyState==4 && xh.status==200){
```

```
                eval('ini(' + xh.responseText + ')');
                }
           }
     xh.open("GET",URL,false);xh.send(null);
     }
```

Código de la interfaz Web (*img.js*):

```
var img=[
     "
XRlZCB3aXRoIEdJTVAALAAAAAAWAB4AAAI5nB+pe40LmRuxKmdzeNrSnglgOJbmiabqyrbuC8fyTKMNgE
8GDugD7wPqhBOiw3jLDZVF5tGZ7B0KADs=",
     "
XRlZCB3aXRoIEdJTVAALAAAAAAWAB4AAAI5nB+pe40LmRuxKmdzeNrSngngSJbmiabqyrbuC8fyHDfAPR
k3kA97/8sFJ0NH0YYTJolLYxPJOxQAADs=",
     "
XRlZCB3aXRoIEdJTVAALAAAAAAIABAAAAITjI+pKsKe2psJAGXxrRvlBYZGAQA7",
     "
CBHSU1QACwAAAAACAAQAAACE4yPqQrAntqbSQhl8a0b5QWGRgEAOw=="
     ];
```

Código de la interfaz Web (*styles.css*):

```
body{
     padding: 50px;
     font-family: Arial;
     }

img{
     cursor: pointer;
     height: 42px;
     }

hr{
     margin: 10px;
     color: #1C1C1C;
     background-color: #3C383A;
     height: 2px;
     border: 0;
     box-shadow: 0 0 5px #1C1C1C;
     }
```

```
div{
      padding: 25px;
      font-weight: 800;
      }

table{
      margin: auto;
      width: 400px;
      font-weight: 800;
      border-radius: 10px;
      box-shadow: 0 0 10px #1C1C1C;
      background-color: #3C383A;
      color: #DFDFDF;
      }

td{
      text-align: center;
      }

.ttl{
      font-size: 30px;
      text-align: center;
      }

.lbl{
      font-size: 18px;
      }

.pnl{
      max-width: 500px;
      margin: auto;
      border-radius: 30px;
      box-shadow: 0 0 15px #1C1C1C;
      background-color: #F0F0F0;
      }
```

Código de la aplicación para las pruebas de desempeño de la interfaz Web (*pruebas.html*):

```
<HTML>
<HEAD>
<meta charset="UTF-8">
<TITLE>
```

```
Evaluación de la interfaz
</TITLE>
      <SCRIPT>
      var startTime = new Date();

            function led(led){
                  var frmLeds = document.getElementById("frmLeds");
                  startTime = new Date();
                  frmLeds.src = 'http://192.168.1.2/LED=' + led;
                  return (0);
                  }

            function leds(led){
                  var frmLeds0 = document.getElementById("frmLeds0");
                  var frmLeds1 = document.getElementById("frmLeds1");
                  var frmLeds2 = document.getElementById("frmLeds2");
                  var frmLeds3 = document.getElementById("frmLeds3");
                  var frmLeds4 = document.getElementById("frmLeds4");
                  var frmLeds5 = document.getElementById("frmLeds5");
                  var frmLeds6 = document.getElementById("frmLeds6");
                  var frmLeds7 = document.getElementById("frmLeds7");
                  var frmLeds8 = document.getElementById("frmLeds8");
                  var frmLeds9 = document.getElementById("frmLeds9");
                  var frmLeds10 = document.getElementById("frmLeds10");
                  var frmLeds11 = document.getElementById("frmLeds11");
                  var frmLeds12 = document.getElementById("frmLeds12");
                  var frmLeds13 = document.getElementById("frmLeds13");
                  var frmLeds14 = document.getElementById("frmLeds14");
                  var frmLeds15 = document.getElementById("frmLeds15");
                  var frmLeds16 = document.getElementById("frmLeds16");
                  var frmLeds17 = document.getElementById("frmLeds17");
                  var frmLeds18 = document.getElementById("frmLeds18");
                  var frmLeds19 = document.getElementById("frmLeds19");
                  var frmLeds20 = document.getElementById("frmLeds20");
                  var frmLeds21 = document.getElementById("frmLeds21");
                  var frmLeds22 = document.getElementById("frmLeds22");
                  var frmLeds23 = document.getElementById("frmLeds23");
                  var frmLeds24 = document.getElementById("frmLeds24");
                  var frmLeds25 = document.getElementById("frmLeds25");
                  var frmLeds26 = document.getElementById("frmLeds26");
                  var frmLeds27 = document.getElementById("frmLeds27");
```

```
        var frmLeds28 = document.getElementById("frmLeds28");
        var frmLeds29 = document.getElementById("frmLeds29");
        var frmLeds30 = document.getElementById("frmLeds30");
        var frmLeds31 = document.getElementById("frmLeds31");
        startTime = new Date();
        frmLeds0.src = 'http://192.168.1.2/LED=' + led;
        frmLeds1.src = 'http://192.168.1.2/LED=' + led;
        frmLeds2.src = 'http://192.168.1.2/LED=' + led;
        frmLeds3.src = 'http://192.168.1.2/LED=' + led;
        frmLeds4.src = 'http://192.168.1.2/LED=' + led;
        frmLeds5.src = 'http://192.168.1.2/LED=' + led;
        frmLeds6.src = 'http://192.168.1.2/LED=' + led;
        frmLeds7.src = 'http://192.168.1.2/LED=' + led;
        frmLeds8.src = 'http://192.168.1.2/LED=' + led;
        frmLeds9.src = 'http://192.168.1.2/LED=' + led;
        frmLeds10.src = 'http://192.168.1.2/LED=' + led;
        frmLeds11.src = 'http://192.168.1.2/LED=' + led;
        frmLeds12.src = 'http://192.168.1.2/LED=' + led;
        frmLeds13.src = 'http://192.168.1.2/LED=' + led;
        frmLeds14.src = 'http://192.168.1.2/LED=' + led;
        frmLeds15.src = 'http://192.168.1.2/LED=' + led;
        frmLeds16.src = 'http://192.168.1.2/LED=' + led;
        frmLeds17.src = 'http://192.168.1.2/LED=' + led;
        frmLeds18.src = 'http://192.168.1.2/LED=' + led;
        frmLeds19.src = 'http://192.168.1.2/LED=' + led;
        frmLeds20.src = 'http://192.168.1.2/LED=' + led;
        frmLeds21.src = 'http://192.168.1.2/LED=' + led;
        frmLeds22.src = 'http://192.168.1.2/LED=' + led;
        frmLeds23.src = 'http://192.168.1.2/LED=' + led;
        frmLeds24.src = 'http://192.168.1.2/LED=' + led;
        frmLeds25.src = 'http://192.168.1.2/LED=' + led;
        frmLeds26.src = 'http://192.168.1.2/LED=' + led;
        frmLeds27.src = 'http://192.168.1.2/LED=' + led;
        frmLeds28.src = 'http://192.168.1.2/LED=' + led;
        frmLeds29.src = 'http://192.168.1.2/LED=' + led;
        frmLeds30.src = 'http://192.168.1.2/LED=' + led;
        frmLeds31.src = 'http://192.168.1.2/LED=' + led;
        return (0);
        }

function tiempo(){
```

```
                var txtTiempo = document.getElementById("txtTiempo");
                var endTime = new Date();
                txtTiempo.value=endTime - startTime + ' ms';
                return (0);
                }

           function tiempos(indice){
                var txtTiempo = document.getElementById("txtTiempos" + indice);
                var endTime = new Date();
                txtTiempo.value=endTime - startTime + ' ms';
                startTime = endTime;
                return (0);
                }
     </SCRIPT>
</HEAD>
<BODY>

<TABLE>
     <TR>
          <TD>
                <TABLE>
                     <TR>
                          <TD>
                               <FORM TARGET="leds" METHOD="POST">
                                    <INPUT TYPE="BUTTON" ID="btnLeds0"
VALUE="LED 0" onclick="led(0);">
                                    </INPUT>
                                    <INPUT TYPE="BUTTON" ID="btnLeds1"
VALUE="LED 1" onclick="led(1);">
                                    </INPUT>
                                    <INPUT TYPE="BUTTON" ID="btnLeds2"
VALUE="LED 2" onclick="led(2);">
                                    </INPUT>
                                    <INPUT TYPE="BUTTON" ID="btnLeds3"
VALUE="LED 3" onclick="led(3);">
                                    </INPUT>
                                    <INPUT TYPE="BUTTON" ID="btnLeds4"
VALUE="LED 4" onclick="led(4);">
                                    </INPUT>
                                    <INPUT TYPE="BUTTON" ID="btnLeds5"
VALUE="LED 5" onclick="led(5);">
```

```
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds6"
VALUE="LED 6" onclick="led(6);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds7"
VALUE="LED 7" onclick="led(7);">
                                        </INPUT>
                                        <INPUT TYPE="TEXT" ID="txtTiempo" text=""
STYLE="width:60px;height:30px" />
                                  </FORM>
                            </TD>
                      </TR>
                      <TR>
                            <TD>
                                  <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds" onload="tiempo();" STYLE="width:600px;height:30px"/>
                                  </IFRAME>
                            </TD>
                      </TR>
                      <TR>
                            <TD>
                                  <IFRAME SRC="http://192.168.1.2"
ID="frmPrincipal" STYLE="width:600px; height:600px">
                                  </IFRAME>
                            </TD>
                      </TR>
                </TABLE>
          </TD>
          <TD>
                <TABLE>
                      <TR>
                            <TD colspan="4">
                                  <FORM TARGET="leds" METHOD="POST">
                                        <INPUT TYPE="BUTTON" ID="btnLeds0"
VALUE="LED 0" onclick="leds(0);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds1"
VALUE="LED 1" onclick="leds(1);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds2"
VALUE="LED 2" onclick="leds(2);">
```

```
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds3"
VALUE="LED 3" onclick="leds(3);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds4"
VALUE="LED 4" onclick="leds(4);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds5"
VALUE="LED 5" onclick="leds(5);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds6"
VALUE="LED 6" onclick="leds(6);">
                                        </INPUT>
                                        <INPUT TYPE="BUTTON" ID="btnLeds7"
VALUE="LED 7" onclick="leds(7);">
                                        </INPUT>
                                        <INPUT TYPE="TEXTAREA" ID="txtTiempos"
text="" />
                                  </FORM>
                            </TD>
                      </TR>
                      <TR>
                            <TD>
                                  <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds0" onload="tiempos(0);" STYLE="width:250px;height:30px"/>
                                  </IFRAME>
                            </TD>
                            <TD>
                                  <INPUT TYPE="TEXTAREA" ID="txtTiempos0" text=""
STYLE="width:70px"/>
                            </TD>
                            <TD>
                                  <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds1" onload="tiempos(1);" STYLE="width:250px;height:30px"/>
                                  </IFRAME>
                            </TD>
                            <TD>
                                  <INPUT TYPE="TEXTAREA" ID="txtTiempos1" text=""
STYLE="width:70px" />
                            </TD>
                      </TR>
```

```
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds2" onload="tiempos(2);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos2" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds3" onload="tiempos(3);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos3" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds4" onload="tiempos(4);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos4" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds5" onload="tiempos(5);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos5" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
```

```
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds6" onload="tiempos(6);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos6" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds7" onload="tiempos(7);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos7" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds8" onload="tiempos(8);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos8" text=""
STYLE="width:70px"/>
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds9" onload="tiempos(9);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos9" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
```

```
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds10" onload="tiempos(10);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos10" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds11" onload="tiempos(11);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos11" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds12" onload="tiempos(12);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos12" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds13" onload="tiempos(13);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos13" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
```

```
ID="frmLeds14" onload="tiempos(14);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                               </TD>
                               <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos14" text=""
STYLE="width:70px" />
                               </TD>
                               <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds15" onload="tiempos(15);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                               </TD>
                               <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos15" text=""
STYLE="width:70px" />
                               </TD>
                          </TR>
                          <TR>
                               <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds16" onload="tiempos(16);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                               </TD>
                               <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos16" text=""
STYLE="width:70px" />
                               </TD>
                               <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds17" onload="tiempos(17);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                               </TD>
                               <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos17" text=""
STYLE="width:70px" />
                               </TD>
                          </TR>
                          <TR>
                               <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds18" onload="tiempos(18);" STYLE="width:250px;height:30px"/>
```

```
                              </IFRAME>
                        </TD>
                        <TD>
                              <INPUT TYPE="TEXTAREA" ID="txtTiempos18" text=""
STYLE="width:70px" />
                        </TD>
                        <TD>
                              <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds19" onload="tiempos(19);" STYLE="width:250px;height:30px"/>
                              </IFRAME>
                        </TD>
                        <TD>
                              <INPUT TYPE="TEXTAREA" ID="txtTiempos19" text=""
STYLE="width:70px" />
                        </TD>
                  </TR>
                  <TR>
                        <TD>
                              <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds20" onload="tiempos(20);" STYLE="width:250px;height:30px"/>
                              </IFRAME>
                        </TD>
                        <TD>
                              <INPUT TYPE="TEXTAREA" ID="txtTiempos20" text=""
STYLE="width:70px" />
                        </TD>
                        <TD>
                              <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds21" onload="tiempos(21);" STYLE="width:250px;height:30px"/>
                              </IFRAME>
                        </TD>
                        <TD>
                              <INPUT TYPE="TEXTAREA" ID="txtTiempos21" text=""
STYLE="width:70px" />
                        </TD>
                  </TR>
                  <TR>
                        <TD>
                              <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds22" onload="tiempos(22);" STYLE="width:250px;height:30px"/>
                              </IFRAME>
```

```
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos22" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds23" onload="tiempos(23);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos23" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds24" onload="tiempos(24);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos24" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds25" onload="tiempos(25);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos25" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds26" onload="tiempos(26);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
```

```
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos26" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds27" onload="tiempos(27);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos27" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds28" onload="tiempos(28);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos28" text=""
STYLE="width:70px" />
                              </TD>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds29" onload="tiempos(29);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos29" text=""
STYLE="width:70px" />
                              </TD>
                        </TR>
                        <TR>
                              <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds30" onload="tiempos(30);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                              </TD>
                              <TD>
```

```
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos30" text=""
STYLE="width:70px" />
                               </TD>
                               <TD>
                                    <IFRAME SRC="http://192.168.1.2/EDO"
ID="frmLeds31" onload="tiempos(31);" STYLE="width:250px;height:30px"/>
                                    </IFRAME>
                               </TD>
                               <TD>
                                    <INPUT TYPE="TEXTAREA" ID="txtTiempos31" text=""
STYLE="width:70px" />
                               </TD>
                          </TR>
                     </TABLE>
           </TD>
      </TR>
</TABLE>

</BODY>
</HTML>
```