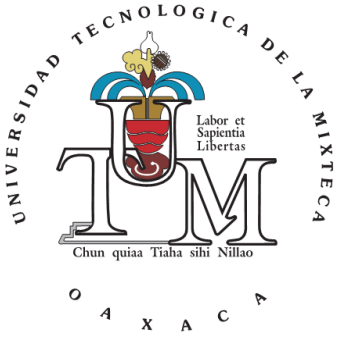


UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"SISTEMA DE COMUNICACIONES BASADO EN ETHERNET PARA EL CONTROL DE SISTEMAS EMPOTRADOS"

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA

PRESENTA

DAVID MÉNDEZ BAUTISTA

DIRECTOR

M. C. JOSÉ ANTONIO MORENO ESPINOSA

HUAJUAPAN DE LEÓN, OAX. DICIEMBRE DE 2009

Tesis presentada el 16 de diciembre de 2009
ante los siguientes sinodales:

M.C. Jacob Javier Vásquez Sanjuán
Ing. Heriberto Ildefonso Hernández Martínez
Dr. Alejandro Ernesto Ramírez González

Director de tesis:

M. C. José Antonio Moreno Espinosa

Dedicatoria

El presente trabajo de tesis está dedicado a mi familia, quienes de manera indirecta realizaron un sacrificio para que pudiera culminar esta etapa de mi vida. Su apoyo incondicional es lo que me ayuda a continuar día a día, y que finalmente me llevó a terminar este trabajo; el cual dedico a las dos personas que me trajeron hasta aquí el día de hoy:

A mi madre, María del Socorro Bautista López (Q. E. P. D.), quien proporcionó fortaleza a mi persona, me enseñó que la constancia es el camino al éxito y que caerse está permitido, pero levantarse es una obligación. Tu amor siempre está presente aunque ya no estés conmigo.

A mi padre, Fernando Manuel Méndez Manzano, quien ha puesto su mano en mi hombro en cada uno de mis fracasos, pronunciando palabras de aliento que me llevaron a salir del hoyo una y otra vez. Me has proporcionado la inteligencia y me has mostrado el camino para ser una persona de bien.

David

Agradecimientos

Las victorias y derrotas que preceden a este trabajo solo tienen sentido reconociendo el trabajo de cada una de las personas que estuvieron conmigo durante este proceso.

A cada uno de los integrantes de mi familia: María del Socorro (Q. E. P. D.), Fernando Manuel, Azucena, Ester, Fernando Ezequiel, Marcela, Lehi, Claudia, María Fernanda; y a mis sobrinos: Lehi Omar, Fernando, Mae Yaretzi y Naomi, por haberme regresado a la realidad y demostrarme que la vida tiene sentido.

Quiero realizar un agradecimiento especial al M.C. José Antonio Moreno Espinosa, por haber confiado en mí desde un principio, por haber tenido la visión clara de los alcances de este proyecto, por su apoyo y tiempo incondicional para alcanzar los objetivos.

Un agradecimiento a mis sinodales: M.C. Jacob J. Vásquez Sanjuán, Ing. Heriberto I. Hernández Martínez, Dr. Alejandro E. Ramírez González, por su tiempo y aportaciones para mejorar el contenido de este trabajo.

A Heriberto, por su amistad sincera, por enseñarme que es posible ser día a día una mejor persona y por orientarme en los momentos de desesperación.

A mis amigos: Alejandro (Ponzo), Luis Daniel (Alemán), Eric, Aldo (Pelucas), Eduardo (Garras), Keneth (Borrego), Eduardo (Borreguito), Andrea, Julio César (Maldad), Julio Alfredo (Lean Coach), Cristóbal (Nagas) y a todos aquellos que iniciaron esta aventura y se quedaron en el camino.

Un reconocimiento a Fidel López Domínguez, por su aportación a este trabajo y su ayuda desinteresada para realizar la PCB.

A los encargados del Laboratorio Avanzados de Electrónica: José Manuel (Vince), Josué (Boti), Canseco, Jehú, Miguel y Juan Carlos, por la ayuda prestada durante los largos días de estudiante, por su tiempo y paciencia para explicarme temas complejos de una forma sencilla.

David

Índice

Lista de tablas

Lista de figuras

Resumen

El presente trabajo tiene como objetivo diseñar e implementar un sistema de comunicaciones que permita la interacción entre una interfaz de usuario basada en web y un sistema basado en microcontrolador, dicho sistema de comunicaciones utilizará el protocolo Ethernet y el conjunto de protocolos TCP/IP. Cuenta con una base de datos para almacenar las acciones realizadas.

Un sistema empotrado tiene elementos tanto de *hardware* (HW) como de *software* (SW), los cuales son desarrollados de forma paralela. El proceso de desarrollo del Sistema de Comunicaciones Basado en Ethernet para el Control de Sistemas Empotrados se describe mediante la metodología de sistemas empotrados.

Como resultado del presente trabajo se obtuvo un sistema de gran flexibilidad en cuanto a los módulos desarrollados, permitiendo que el sistema pueda evolucionar hacia nuevas tecnologías.

Abstract

The major objectives of this work are; i) to design a system that allows communication between a GUI web based and a microcontroller-based system and ii) to implement it. The implemented system directly uses the Ethernet and the *suite* TCP/IP protocols and it makes use of a database to self-storage actions that were previously done.

The implemented system comprises of an embedded system containing *hardware* (HW) and *software* (SW) parallely arranged elements. The embedded system, based on Ethernet, is described in detail herein using the embedded system methodology.

As a conclusion, the implemented system is flexible and efficient due to its adaptable modules and due to its potential to expand with new technologies.

Introducción

El presente trabajo aborda el área de redes de computadoras como medio de comunicación entre sistemas empotrados (*embedded systems*) y los usuarios del sistema. Se presenta un dispositivo (microcontrolador) que realiza operaciones de control y monitoreo periódico de algún sistema, los datos que se generan se almacenan en algún medio para su procesamiento posterior. La información que se presenta al usuario se realiza gráficamente en algún medio para tal propósito.

Un microcontrolador (MCU) es un circuito electrónico programable que ejecuta instrucciones codificadas en una memoria. Un sistema basado en MCU generalmente se encarga de obtener, procesar y presentar la información al usuario mediante el uso de dispositivos periféricos de entrada/salida; en la mayoría de los casos los procesos de obtención, procesamiento y presentación se realizan a nivel local, es decir, dentro del mismo MCU. Sin embargo, actualmente se requieren sistemas complejos cuyos procesos se ejecutan de forma remota mediante sistemas de comunicaciones para realizar sus tareas. El sistema de comunicaciones de datos más usado es Internet[1].

Los procesos de interacción entre los sensores/actuadores y el MCU son de forma local y los procesos de recolección y presentación de datos son de forma remota. El desarrollo de sistemas digitales puede llegar a presentar problemas en la entrega de resultados finales, los cuales pueden ser de carácter parcial o total, y ser establecidos en un tiempo periódico; así mismo, estos resultados deben ser mostrados en los dispositivos de salida del sistema; se tiene la posibilidad de que el sistema cuente con capacidad de almacenamiento temporal para una cierta cantidad de datos, en este caso existe la posibilidad de tener contratiempos de implementación e incluso pérdida de los datos; otro posible problema es la incapacidad de configurar remotamente los parámetros de funcionamiento del sistema.

La presentación de la información en el MCU generalmente es mediante indicadores binarios (encendido/apagado). Es preferible tener la información de salida en un *software* especial basado en una terminal de órdenes simple (C*ommand Console*) o una interfaz gráfica de usuario (GUI, *Graphic User Interface*), dependiendo de los requisitos y flexibilidad del

1 Se hace la diferencia en el uso de Internet e internet, siendo la primera sinónimo de la red que se usa en las comunicaciones globales y la segunda como un conjunto de redes locales interconectadas mediante un *router*.

sistema. Además, existe la posibilidad de utilizar un navegador web[2] en lugar de un *software* especial, con la ventaja de que dicho navegador existe prácticamente en cualquier computadora conectada a Internet.

Los conceptos anteriores se engloban en un área que se conoce como monitoreo remoto (*Remote Monitoring*) y se define como un sistema que permite realizar controles rutinarios a través de redes de comunicaciones extensas como puede ser Internet [15]. A continuación se mencionan algunas de sus principales áreas de aplicación:

- Terminales ATM (A*utomated Teller Machine*).
- Sistemas de verificación para pago con tarjetas de crédito (*Credit card and Check-verification systems*).
- Recolección de datos (*Data collection*).
- Máquinas despachadoras de combustible (Gas pump).
- Sistemas de monitoreo remoto industriales y médicos (*Industrial and medical remote monitoring system*s).
- Terminales de puntos de venta (*Point-of-sale terminals*).
- Diagnósticos remotos (*Remote diagnostics*).
- Medición remota (*Remote metering*).
- Sistemas de seguridad (*Security systems*).
- Sistemas de acceso a servicios de TV digital (*Television set-top boxes*).
- Máquinas de venta electrónicas (*Ticketing machines*).

Las aplicaciones mencionadas son de consumo general, se pueden observar en lugares públicos o dentro de instalaciones industriales privadas. El propósito de este trabajo es realizar un sistema genérico de monitoreo remoto que contenga los elementos mínimos de una aplicación de este tipo.

Antecedentes

Algunos de los trabajos realizados en el área de monitoreo remoto, con características similares (uso de Internet) son:

2 El término Web es utilizado cuando se refiere a la WWW (*World Wide Web*) y el término web cuando se habla de elementos relacionados con Internet.

- *Sistema de gestión para redes de pequeñas y medianas empresas (Instituto Politécnico Nacional, México) [6].*

- *Monitoreo remoto de procesos a través del canal de voz de teléfonos celulares GSM (Universidad Peruana de Ciencias Aplicadas, Perú) [4].*

- *DaMA-Web*: *Un programa para el monitoreo y control local y remoto vía web, de la adquisición de los datos* (*Universidad Nacional de Salta, Argentina*) [5].

- *Desarrollo de una red de monitoreo por sensores remotos de la calidad de agu*a (*Instituto Tecnológico de Costa Rica*) [9].

- *Desarrollo de un sistema flexible de control local y remoto para una red domótica en edificios inteligentes* (*Universidad Pública de Navarra, España*) [25].

- *Vigilando el desierto, Descripción del Sistema de Telemetría de la Estación Experimental de Zonas Áridas* (*Almería, España*) [34].

- *Desarrollo de un Servidor Web para adquisición de Datos en Tiempo Real* (*Universidad de A. Coruña, España*) [39].

- *Laboratorios Remotos para las Prácticas de Ingeniería de Sistemas y Automática* (*Universidad Miguel Hernández, España*) [22].

- *Control remoto del tanque de experimentación hidroacústica del Instituto de Acústica del Consejo Superior de Investigación Científica* (*CSIC, España*) [40].

Los trabajos mencionados son de propósito específico, de los cuales no se cuenta con acceso a su desarrollo técnico a nivel ingeniería por lo que se consideran como referencia de desarrollo en el área de monitoreo remoto.

Planteamiento del problema

La idea inicial que generó este trabajo es la implementación de un sistema de vigilancia dentro de la Universidad Tecnológica de la Mixteca (UTM), haciendo uso de la infraestructura actual de comunicaciones, la cual se basa en Ethernet.

El presente trabajo de tesis propone diseñar e implementar un sistema de comunicaciones para que los usuarios interactúen con elementos de monitoreo de forma remota, haciendo uso de tecnologías estandarizadas (TCP/IP) y la infraestructura de la red existente en la UTM. El sistema será capaz de enviar y recibir datos desde cualquier lugar de la universidad donde se cuente con una conexión Ethernet para configurar y/o presentar los datos de monitorización obtenidos por el sistema, además de tener la capacidad de almacenarlos dentro de una base de datos (BD). El sistema se basará en un MCU de propósito general que haga uso de Internet para comunicarse con una computadora mediante una

pasarela RS232–Ethernet.

El trabajo a desarrollar propone un sistema dividido en dos etapas: la etapa de *software* (SW) y la etapa de *hardware* (HW) con la finalidad de que la solución sea completamente entendible y realizable.

Los elementos funcionales del *software* tienen como entrada del sistema la interacción de un administrador o usuario final y como salida órdenes dirigidas a los dispositivos gestionables (Figura 1). La tarea de la interfaz de usuario es proporcionar información de dichos dispositivos y obtener órdenes del usuario para su procesamiento. La información proveniente, tanto de la interfaz de usuario como de los dispositivos, tiene que ser revisada por el gestor de nodos, el cual se encarga de traducir las órdenes generadas de ambos elementos y establecer una comunicación entre ellos. Por otra parte, dicho módulo tiene comunicación con un gestor de base de datos, al cual se le envía información acerca de los eventos ocurridos en el sistema. La comunicación entre los elementos mencionados se realiza a través de TCP/IP. Con respecto al *software* de presentación, se propone utilizar un navegador web, lo cual requiere el uso de *sockets* implementados en algún lenguaje estándar de programación.

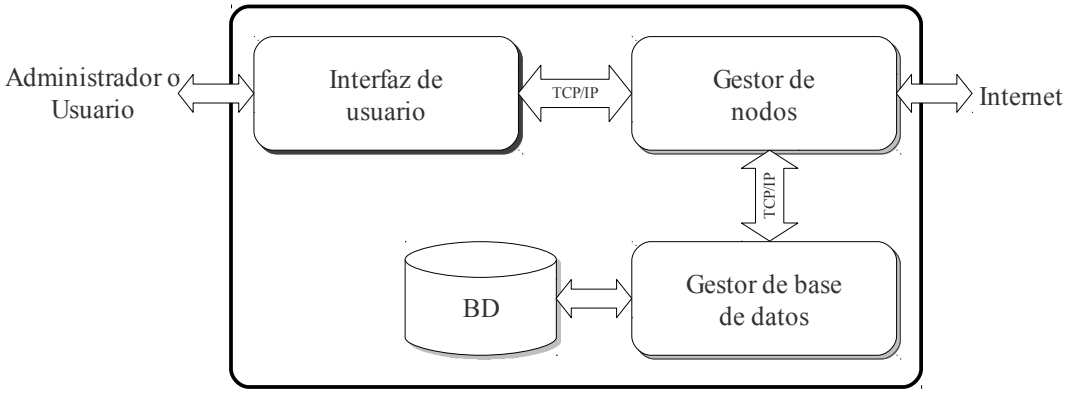


Figura 1. Elementos funcionales del software.

Los elementos funcionales de *hardware* tienen como entrada información proveniente del gestor de nodos y como salida la interacción con sensores y actuadores (Figura 2). La pasarela RS232-Ethernet se encarga de realizar la conexión TCP/IP con los elementos de *software* y establecer la comunicación con el MCU mediante el protocolo RS232; como pasarela RS232-Ethernet se usará el dispositivo MT100SEM de la firma Multitech. El sistema basado en MCU cuenta con la capacidad de interactuar con sensores y actuadores para realizar diversas tareas propias de este tipo de sistemas; como MCU se propone el dispositivo Atmega8 de la firma Atmel.

Los elementos funcionales de *hardware* y de *software* son una abstracción del sistema, durante el desarrollo de este documento se explicará el diseño y desarrollo detallado de cada elemento.

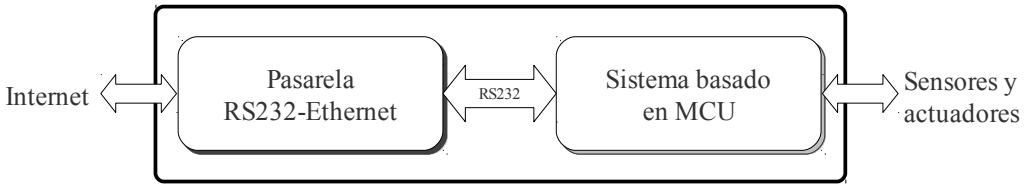


Figura 2. Elementos funcionales del hardware.

Objetivos

El objetivo principal del presente trabajo es diseñar e implementar un sistema de comunicaciones mediante Ethernet que permita controlar dispositivos basados en MCU, además de permitir al usuario interactuar con una interfaz web y ofrecer almacenamiento temporal de eventos ocurridos en el sistema dentro de una base de datos.

Para cumplir con el objetivo principal se proponen los siguientes objetivos secundarios:

- Utilizar un MCU que permita disponer de cuatro entradas analógicas, cuatro entradas digitales y cuatro salidas digitales.
- Realizar la comunicación entre el MCU y el MT100SEM mediante el protocolo RS232 con un *baudrate* de 9600 bps.
- Establecer comunicación entre el MCU y el módulo de funciones de gestión mediante el dispositivo MT100SEM.
- Establecer comunicación entre el MT100SEM y el módulo de funciones de gestión mediante TCP/IP.
- Diseñar e implementar un gestor de nodos que permita atender 64 usuarios simultáneamente y 8 dispositivos MT100SEM.
- Diseñar e implementar un protocolo para la comunicación entre el MCU y el gestor de nodos.
- Diseñar e implementar un protocolo para la comunicación entre el gestor de nodos y la interfaz web.
- Diseñar e implementar una interfaz web para el control del sistema.
- Realizar el proceso de almacenamiento de datos mediante un gestor de base de datos.

Justificación

En la UTM no se tiene implementado un sistema de monitoreo remoto abierto, esto es que permita agregar diversos elementos tales como sensores de presencia, temperatura, intensidad luminosa, etc.

El sistema propuesto se enfoca al monitoreo de los edificios y la infraestructura en general del campus de la UTM. Debido a la similitud entre los campus de la UTM y las demás universidades pertenecientes al Sistema de Universidades Estatales de Oaxaca (SUNEO) en el futuro será posible implementar este sistema sin requerir cambios substanciales.

Las mayoría de las herramientas *software* a utilizar son de uso libre, por lo que el sistema no representa costos por licencia.

La arquitectura que se propone para el sistema permite realizar cambios de los elementos de forma independiente (Figura 1 y Figura 2), lo cual facilita la evolución del sistema.

Metodología de desarrollo

El modelo de sistemas empotrados propuesto por la autora Tammy Noergaard introduce conceptos y fundamentos de sistemas empotrados; el concepto de arquitectura de sistemas empotrados es usado para describir cómo se integra el *hardware* y el *software* dentro de un sistema empotrado [33]. El nivel más alto de descripción en este modelo contempla tres elementos: La capa de *hardware*, la capa de *software* del sistema y la capa de *software* de aplicación (Figura 3).

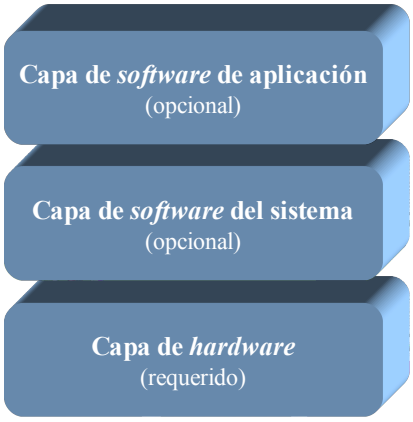


Figura 3. Modelo de sistemas empotrados [33].

Todos los sistemas empotrados comparten esta similitud en el nivel más alto de abstracción [33], es decir, los sistemas empotrados tienen por lo menos una capa (*hardware*) o todas las capas (*hardware*, *software* del sistema y *software* de aplicación), dentro de las cuales se encuentran a detalle cada uno de los componentes del sistema. La capa de *hardware* contiene la mayor parte de los componentes físicos, normalmente en una tarjeta de desarrollo,

la capa de *software* del sistema y de *software* de aplicación contienen todas las herramientas necesarias para su funcionamiento, es ahí donde empieza el proceso de sistema empotrado.

El modelo presentado es esencialmente en capas (modular), representando una arquitectura de sistemas empotrados desde la cual se puede derivar una estructura arquitectónica modular. El concepto de capas no se aplica únicamente al diseño de sistemas empotrados, modelos como OSI (*Open Systems Interconnection*) y TCP/IP se describen también en capas; el uso de capas es una herramienta muy útil para visualizar las posibles combinaciones entre los componentes *hardware* y *software* que pueden ser usados en el diseño de sistemas empotrados.

Un sistema empotrado es una aplicación de sistemas de computadoras y es diferenciado con otro tipo de sistemas de computadoras tales como las computadoras personales o supercomputadoras. Aunque la definición es difícil de describir ya que se está en constate evolución debido a los avances tecnológicos y a un decremento en los costos de implementación en el *hardware* y en el *software*, se pueden mencionar descripciones comunes de sistemas empotrados [33]:

- *Un sistema empotrado es más limitado en funcionalidad de hardware y/o* software *que una computadora personal*: En términos de *hardware*, esto puede significar limitaciones en el desempeño de procesamiento, consumo de energía, memoria, funcionalidad del *hardware*, entre otras. En términos de *software*, típicamente significa limitaciones relativas a una PC, pocas aplicaciones, sin sistema operativo (SO) o un sistema operativo limitado, nivel bajo de abstracción. Aunque la definición es parcialmente verdadera, la tarjeta madre y el *software* contenidas en las primeras PC y en las actuales tienen la misma definición, la diferencia es que estas últimas han sido reempaquetadas en diseños más complejos de sistema empotrado.

- *Un sistema empotrado es diseñado para desempeñar una función dedicada*: Muchos dispositivos empotrados son diseñados principalmente para desarrollar una función específica. Por ejemplo, en la actualidad se tienen asistentes de datos personales (PDA, *Personal Data Assistant*) y/o teléfonos celulares híbridos, que son sistemas empotrados diseñados con la capacidad de hacer una o varias funciones básicas. También las televisiones más recientes incluyen aplicaciones interactivas que desarrollan una gran variedad de funciones generales como correo electrónico, navegación web y juegos.

- *Un sistema empotrado es un sistema de computadora con requerimientos de alta calidad y fiabilidad en comparación con otros sistemas*: Algunos dispositivos empotrados necesitan un cierto umbral en los requerimientos de calidad y fiabilidad. Por ejemplo, si un dispositivo de control en un automóvil falla mientras se está en un camino transitado o si un dispositivo médico falla durante una operación quirúrgica, se pueden tener resultados catastróficos. En cambio, existen también dispositivos empotrados como televisiones, juegos o teléfonos celulares en los cuales un mal funcionamiento es un inconveniente, pero no es una situación que pone en riesgo vidas humanas.

Principales aplicaciones de los sistemas empotrados

Los dispositivos electrónicos pueden ser clasificados por su área de consumo (mercado) y tienen un amplio rango de aplicación, tales como:

- *Automotriz*: Sistemas de encendido, control del motor, sistema de frenado.
- *Aparatos domésticos*: Televisores analógicos y digitales, DVD, VCR, PDAs, aparatos de cocina (refrigeradores, tostadores, hornos de microondas), juguetes/juegos, teléfonos fijos, teléfonos celulares, cámaras, sistemas de posicionamiento global (GPS).
- *Control industrial*: Robots y sistemas de control (manufactura).
- *Estaciones de red*: Ruteadores, concentradores (HUB), pasarelas (*gateways*).
- *Automatización de oficinas*: Máquina de fax, fotocopiadora, impresoras, escáner.

Ciclo de vida de un sistemas empotrados

El ciclo de vida de un sistema empotrado puede ser descrito en base a varios modelos; desde el punto de vista de ingeniería de sistemas, dichos modelos se basan en uno o en una combinación de los modelos de desarrollo descritos a continuación [33] [43]:

- El modelo de gran explosión (*Bing-bang model*): Es esencial no planificar y procesar los requisitos antes o durante el desarrollo del sistema.
- El modelo codificar-arreglar (*Code-and-fix model*): Se definen los requerimientos del producto pero no los procesos formales antes de empezar el desarrollo.
- El modelo de cascada (*Waterfall model*): Existen procesos definidos para desarrollar el sistema basado en fases, donde el resultado de una fase sirven como entrada para la siguiente fase.
- El modelo de desarrollo evolutivo: Se entrelazan las actividades como la especificación, desarrollo y la validación. Se desarrolla rápidamente un sistema inicial a partir de especificaciones abstractas y se refina basándose en las peticiones del cliente para producir un sistema que satisfaga los objetivos. Los modelos evolutivos son iterativos y existen principalmente dos enfoques: El modelo incremental y el modelo en espiral.
- El modelo de ingeniera basada en componentes: Se basa en la existencia de un número significativo de componentes reutilizables. El proceso principal de desarrollo se centra en integrar los componentes existentes dentro del sistema más que en desarrollarlos desde cero.

Como se mencionó anteriormente, los modelos de procesos genéricos se utilizan ampliamente en la práctica de ingeniería de sistemas. Los modelos no tienen por que ser excluyentes entre sí, a menudo se utilizan combinaciones de ellos, especialmente para desarrollo de sistemas grandes [43].

El diseño general del sistema se basa en la metodología de sistemas empotrados, la cual define siete fases de desarrollo: Especificación del producto, División HW y SW, Iteración e implementación, Diseño detallado HW y SW, Integración de componentes HW y SW, Prueba y liberación del producto, Mantenimiento y actualización [3] [8]. La última fase queda fuera del ámbito de este trabajo.

Contenido del documento de tesis

En el capítulo 1 se describe la parte teórica que sustenta el desarrollo de este trabajo; los conceptos y herramientas mencionadas en este capítulo se abordan brevemente ya que la información referente a estos elementos son descritos a detalle por los autores y/o desarrolladores. Existe bibliografía especializada en cada uno de los temas abordados.

El capítulo 2 presenta el desarrollo del sistema, en el cual se detallan las fases de la metodología de sistemas empotrados. Seguir dicha metodología permite evaluar los avances en el sistema.

En el capítulo 3 se presenta el sistema final (también conocido como producto final). Para presentar los resultados, el sistema se divide conforme al modelo de sistemas empotrados, es decir, en la capa de *hardware* (nodo), capa de *software* del sistema (gestor de nodos) y la capa de *software* de aplicación (GUI), permitiendo ver con claridad cada elemento del sistema.

El capítulo 4 presenta las conclusiones obtenidas y las futuras líneas de investigación.

Por último, se presentan las referencias bibliográficas y los anexos de la información relevante para este trabajo de tesis.

Capítulo 1
Marco teórico

El término sistema es ampliamente utilizado. En el ámbito tecnológico se tiene una definición general: «*Un sistema es una colección de componentes interrelacionados que trabajan conjuntamente para cumplir algún objetivo*» [43]. La definición anterior se refiere a sistemas en general y es aplicable a un gran número de sistemas. Los sistemas que incluyen componentes de *software* son clasificados en dos categorías:

- Sistemas técnicos informáticos: Son aquellos sistemas que incluyen *hardware* y *software* pero no procedimientos y procesos. Algunos ejemplos de este tipo de sistemas son televisores, teléfonos celulares y la mayoría del *software* de computadora. Los usuarios utilizan sistemas para algún fin, pero el conocimiento de este fin no es parte del sistema.

- Sistemas socio-técnicos: Este tipo de sistemas contemplan uno o más sistemas técnicos e incluyen el conocimiento de cómo debe usarse para alcanzar un objetivo más amplio. Estos sistemas han definido procesos operativos que incluyen a personas (los operadores) como partes inherentes del sistema.

1.1 Sistemas empotrados

En los sistemas empotrados todos los componentes electrónicos están contenidos en una placa de desarrollo, también conocida como PCB (*Printed Circuit Board*), esta placa normalmente se realiza en fibra de vidrio o baquelita. Las conexiones eléctricas entre componentes se basan en líneas de cobre que permiten el paso de señales eléctricas entre las terminales de los componentes. Todos los componentes (incluyendo la placa) están contenidos en la capa de *hardware* del modelo de sistemas empotrados (Figura 1.1).

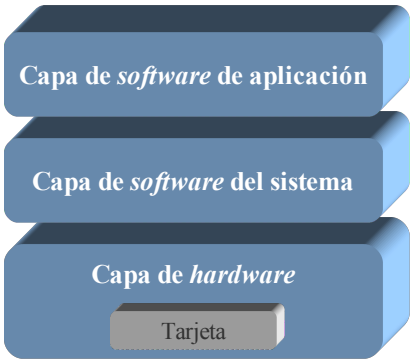


Figura 1.1. Tarjeta y componentes en el modelo de SE.

En el nivel más alto de abstracción, los componentes de la capa de *hardware* son clasificados en cinco categorías, según la funcionalidad del componente [33]:

- Unidad central de proceso (CPU): Es el componente primario de un sistema.
- Memoria.
- Dispositivos de entrada.
- Dispositivos de salida.
- Rutas de conexión/bus de datos.

Las cinco categorías mencionadas se definen en la arquitectura von Neumann (Figura 1.2), la cual es una herramienta para explicar la arquitectura de *hardware* de algunos dispositivos[3]. El hecho de que un sistema empotrado sea un tipo de sistema de computadora permite explicar la relación existente entre los componentes usando la arquitectura mencionada [33].

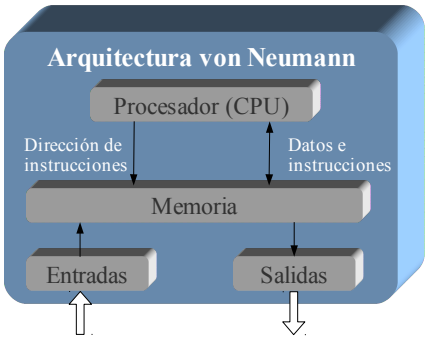


Figura 1.2. Arquitectura de von Neumann.

En la realidad las arquitecturas de microprocesadores (μP) son mucho más complejas que la arquitectura von Neumann presentada en la Figura 1.2, sin embargo muchos procesadores están basados en dicha arquitectura o en una modificación de la misma. Una variante de esta arquitectura se conoce como arquitectura Harvard (Figura 1.3).

3 La arquitectura von Neumann es el resultado del trabajo publicado por John von Neumann en 1945, en el cual se definen los requerimientos de una computadora electrónica de propósito general. La arquitectura von Neumann también se conoce como arquitectura Princeton [18].

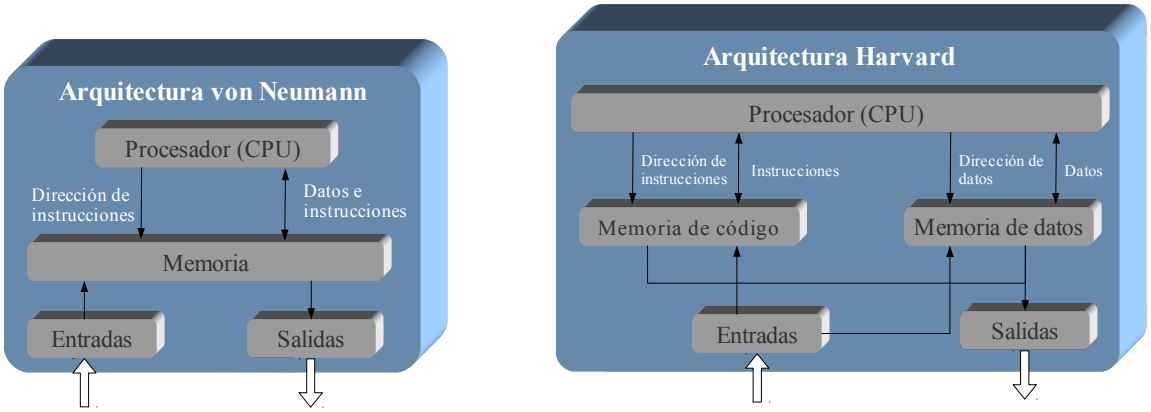


Figura 1.3. Arquitectura von Neumann vs arquitectura Harvard.

Estas arquitecturas difieren principalmente en el área de memoria; mientras la arquitectura Princeton define un espacio simple de memoria para almacenar instrucciones y datos, la arquitectura Harvard define dos espacios de memorias independientes, uno para instrucciones y otra para datos, logrando realizar simultáneamente el ciclo *fetch* y soportar concurrencia. La principal razón de elegir entre una arquitectura Princeton o una arquitectura Harvard es el rendimiento; manejar la memoria de instrucciones y de datos por separado permite incrementar la cantidad de procesamiento de información por unidad de tiempo.

1.1.1 Microprocesador vs microcontrolador

El procesador es la unidad principal de un sistema empotrado, su función principal es procesar instrucciones y datos. Un dispositivo electrónico como éste, cuenta por lo menos con un procesador denominado *maestro* que actúa como unidad central de control pudiendo tener otros procesadores llamados *esclavos*. Los dispositivos *esclavos* pueden tener funciones complementarias como controladores de USB, controlador Ethernet, etc. La complejidad del procesador *maestro* determina su clasificación, µP o MCU. Tradicionalmente, un µP está compuesto de elementos mínimos de memoria, puertos de entrada/salida y tiene que hacer uso de elementos externos para realizar sus funciones (Figura 1.4a); en cambio un MCU contiene todos los elementos (memoria de instrucciones/datos, entradas, salidas, reloj oscilador, etc.) dentro del mismo encapsulado (Figura 1.4b) [3].

Existen ventajas al usar un diseño basado en MCU con respecto a un µP, algunas de ellas se mencionan a continuación [11]:

- El circuito impreso es mucho más pequeño debido a que sus componentes se encuentran en el mismo encapsulado.
- El costo de desarrollo de un sistemas es menor al reducir el número de componentes.
- Se eliminan los problemas de ruido que pueden afectar a un µP, ya que los componentes se encuentran en el mismo encapsulado.
- El tiempo de desarrollo se reduce considerablemente.

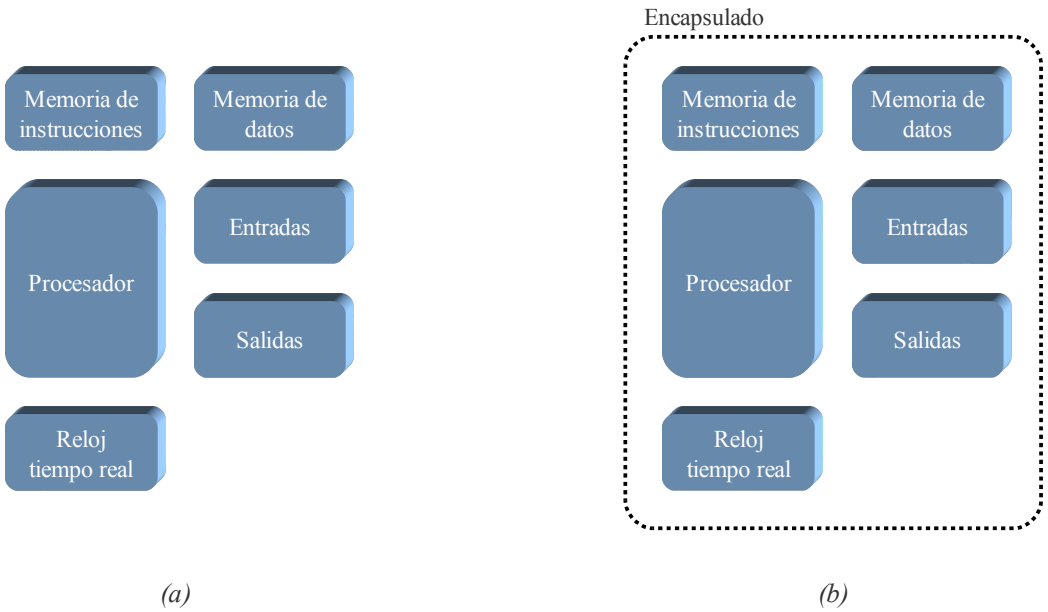


Figura 1.4. Elementos de un: a. Microprocesador; b. Microcontrolador.

1.1.2 Microcontrolador

El MCU es un circuito electrónico programable capaz de ejecutar instrucciones almacenadas en su memoria de código. Está formado por los elementos mostrados en la Figura 1.4b y algunos otros elementos que permiten tener características especiales. Los MCU están basados en el modelo von Neumann o alguna de sus variantes y cuentan con una CPU, memoria de datos/código, entradas y salidas.

Existen varias empresas que se dedican al desarrollo de MCU: Microchip, Atmel, FreeScale (antes Motorola), Hitachi, Holtek, Intel, National Semiconductor, Nec, Parallax, Texas Instruments, Zilog, Silab, entre otros. Básicamente los MCU de cada empresa tienen un funcionamiento común. La selección de un MCU depende de la tarea que se quiera realizar y las características varían de un dispositivo a otro.

Por lo general, los componentes que integran un MCU son [18]:

- CPU: Es la unidad central de proceso encargada de procesar las instrucciones almacenadas en la memoria de código. La CPU está compuesta por la unidad aritmética-lógica (ALU), el decodificador de instrucciones y el circuito de control.

- Memoria de código: En la memoria de código están almacenadas las instrucciones que forman el programa. Cuando el programa es muy extenso es posible, para algunos dispositivos, dividir el programa en secciones de memoria interna y/o externa. Normalmente este tipo de memoria es no volátil, ya sea EPROM, EEPROM o flash.

- Memoria de datos: La memoria RAM (*Random Memory Access*) es la memoria de datos del MCU y se utiliza para almacenar datos de forma temporal. La CPU utiliza la memoria de datos para almacenar los valores de los registros, tales como registros de propósito general, registros de propósito específico, espacio de I/O o pila. La pila se utiliza para almacenar el PC (*Program Counter*) con el fin de regresar de una llamada de subrutina o una interrupción.

- Reloj oscilador: El MCU ejecuta las instrucciones contenidas en la memoria de código a una frecuencia específica, a la cual se conoce como frecuencia del oscilador. Existen distintos tipos de osciladores: osciladores RC, compuestos por una resistencia y un capacitor, osciladores de tipo *cristal;* algunos dispositivos cuentan con un oscilador interno, con lo que no es necesario elementos externos para su funcionamiento.

- *Reset* y detector de nivel de voltaje: El circuito de *reset* asegura que todos los componentes inicien a un determinado tiempo y que los registros estén inicializados de manera conocida. El circuito detector de voltaje realiza un monitoreo al voltaje de alimentación, si en algún momento el voltaje es menor del requerido realiza un *reset* para evitar la corrupción de los registros o el contenido de la memoria, lo que puede ocasionar un mal funcionamiento del MCU.

- Puerto serie: El puerto serie es utilizado para comunicar el MCU con otro dispositivo. El puerto serie funciona a diferentes velocidades y diferentes configuraciones de envío de bits. Es posible hacer la comunicación de forma síncrona o asíncrona.

- Puertos de E/S digitales: Se cuenta con entradas y salidas digitales para interactuar con el medio exterior.

- Puertos de E/S analógicos: Para realizar la lectura de entradas analógicas es necesario contar con un convertidor análogo-digital (ADC). Los valores entregados por el ADC pueden tener diferentes resoluciones (8, 9, 10 bits, etc.). Para realizar el proceso inverso (tener una salida analógica) se debe contar con un convertidor digital-analógico (DAC), la mayoría de los MCUs cuentan con PWMs (*Pulse Width Modutalors*) que junto con un filtro pasabajas realiza la misma función.

- Temporizador: El temporizador se utiliza para generar un evento a un determinado tiempo. La frecuencia de ocurrencia del evento depende directamente de la frecuencia del oscilador que maneje el MCU. El temporizador también se utiliza para contar eventos externos, en este caso el temporizador es llamado contador.

1.2 Estándares y redes de comunicaciones

Una red de comunicación es la conexión entre dos o más dispositivos que pueden enviar y/o recibir datos. Si un sistema empotrado necesita comunicarse con otro sistema, ya sea una terminal, un servidor u otro dispositivo empotrado, éstos deberán tener el mismo esquema de conexión. Para que la comunicación sea satisfactoria deben contar con el mismo

sistema de interconexión y el mismo protocolo de red que permitan tener una interoperabilidad.

Los sistemas que son desarrollados en base a metodologías específicas llamadas estándares, pueden interactuar de forma correcta con otros sistemas basados en tales estándares. Para construir aplicaciones que deban ejecutarse en red existen varios estándares, algunos de los cuales se listan en la Tabla 1.1.

Tabla 1.1. Estándares en el ámbito de redes.

Estándares de comunicaciones	
TCP (*Transmission Control Protocol*)/IP (*Internet Protocol*)	Protocolo basado en los RFC (*Request For Comments*) 791 (IP) y 793 (TCP), en los cuales se definen los componentes de *software* del sistema.
PPP *(Point-to-Point Protocol)*	Componentes de *software* del sistema basado en los RFC 1661, 1332 y 1334.
IEEE (*Institute of Electronic and Electrical Engineers*) 802.3 Ethernet	Protocolo de red que define el *hardware* y los componentes *software* del sistema para redes de área local (LAN, *Local Area Network*).
Estándares para construir aplicaciones en red	
HTTP (*Hypertex Transfer Protocol*)	La WWW (*World Wide Web*) define este protocolo en los documentos RFC 2616, 2016, 2069, 2109. Este protocolo pertenece a la capa de aplicación (modelo OSI) y es implementado usando un navegador web.
Lenguajes de programación	
HTML (*HyperText Markup Lenguage*)	Lenguaje de tipo *script* que interpreta etiquetas y muestra el resultado en un navegador web.

Para entender cuáles son los componentes de redes necesarios para un dispositivo empotrado se requieren dos pasos:

- Determinar en qué tipo de arquitectura de red se va a conectar el sistema empotrado.
- Utilizar algún modelo de red que describa su funcionamiento, por ejemplo el modelo OSI o el modelo TCP/IP.

Es necesario contemplar tres características: distancia entre los dispositivos, medio físico de transmisión del dispositivo empotrado con respecto a la red y estructura de red. Es importante entender claramente en qué tipo de red se va a trabajar, ya que sus características determinan los estándares a implementar dentro del sistema empotrado (Figura 1.5).

Una arquitectura de red no es lo mismo que una topología, esta última es el arreglo físico de cómo están conectados los dispositivos y se determina por la arquitectura, el medio de transmisión (alámbrico o inalámbrico) y la distancia entre los dispositivos conectados en alguna red en particular [33].

1.2.1 Modelo de referencia OSI

Existen estándares que permiten que las computadoras de diferentes capacidades, fabricantes y sistemas operativos puedan tener interoperabilidad ente ellas [46]. Debido a la complejidad de los sistemas de comunicaciones, es recomendable dividirlos en elementos manejables, estructurándose en una arquitectura de comunicaciones. Por esa razón la Organización Internacional de Estandarización (ISO, *International Organization for Standarization*) estableció el modelo de referencia OSI [49].

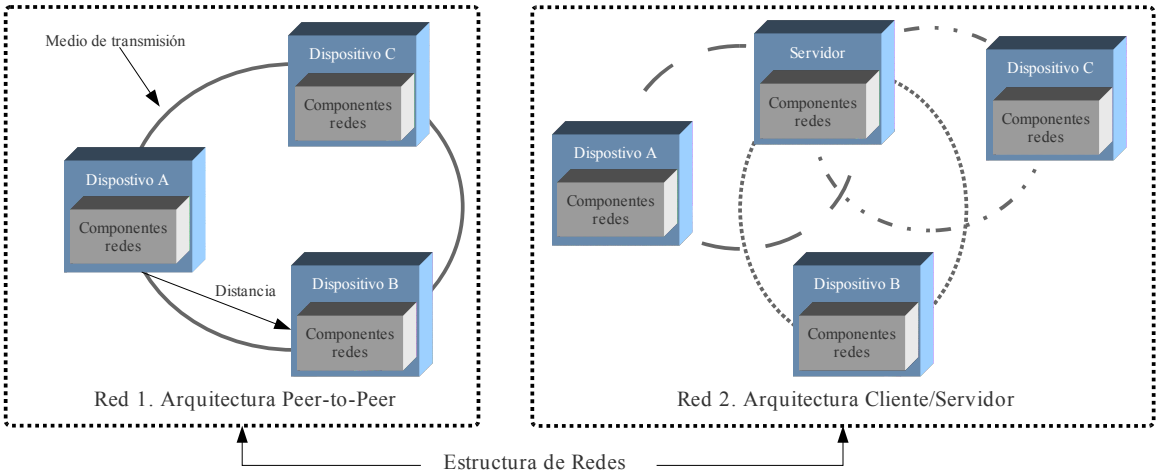


Figura 1.5. Diagrama a bloques de una arquitectura de red [33].

El modelo OSI (Figura 1.6) se utiliza como referencia para introducir los conceptos relacionados a la interconexión de redes. Este modelo está dividido por capas jerárquicas y los elementos que componen a cada capa realizan un conjunto de tareas similares. La comunicación se realiza solo entre capas adyacentes. Cada capa ofrece servicios a la capa superior y recibe servicios de la capa inferior [45].

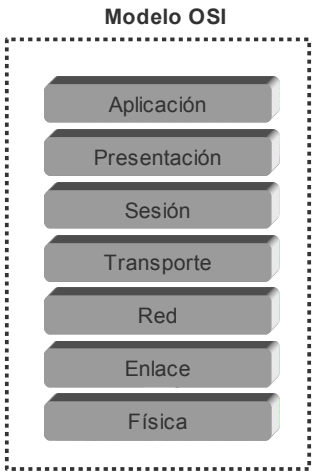


Figura 1.6. Capas del modelo OSI [49].

Cada uno de los elementos conectados a una red puede contener todas las capas o solo algunas de ellas. El modelo de referencia OSI establece que la comunicación entre elementos que componen la red solo puede llevarse a cabo entre capas del mismo nivel. A continuación se menciona la descripción de las funciones de cada una de las capas:

- Capa física: Se encarga de la transmisión de cadenas de bits no estructuradas sobre el medio físico; está relacionada con las características mecánicas, eléctricas, funcionales y de procedimiento para acceder al medio físico. Los protocolos de la capa física definen el *hardware* de red del dispositivo empotrado.

Los componentes *hardware* de la capa física se conectan al sistema por algún medio de transmisión; la distancia entre los dispositivos conectados son definidos por la arquitectura de red utilizada, pudiendo ser clasificada como una red LAN (*Local Area Network*) o una WAN (*Wide Area Network*), que a su vez se subdividen con respecto al medio de transmisión con el que establecen la conexión con la red (alámbrico o inalámbrico).
La capa física define, maneja y procesa (vía *hardware*) la señales de datos dependiendo del tipo de codificación que se utiliza.

- Capa de enlace de datos: Proporciona un servicio de transferencia de datos fiable a través del enlace físico; envía bloques de datos (llamados tramas) llevando a cabo la sincronización, el control de errores y el flujo de datos.

- Capa de red: Proporciona independencia a los niveles superiores respecto a las técnicas de conmutación y de transmisión utilizadas para conectar los sistemas; especifica la asignación de las direcciones y el reenvío de paquetes de un extremo a otro.

- Capa de transporte: Proporciona una transferencia transparente y fiable de datos entre los puntos finales, además proporciona procedimientos de recuperación de errores y control de flujo origen-destino.

- Capa de sesión: Proporciona el control de la comunicación, establece, gestiona y cierra las conexiones (sesiones) entre las aplicaciones.

- Capa de presentación: Proporciona a los procesos de aplicación independencia respecto a las diferencias en la representación de los datos (sintaxis).

- Capa de aplicación: Proporciona el acceso al entorno OSI para los usuarios y también proporciona servicios de información distribuida.

Se puede situar el modelo de sistemas empotrados (Figura 3) en el modelo de referencia OSI (Figura 1.6). La capa física del modelo OSI se asocia a la capa de *hardware* del modelo de sistemas empotrados; las capas de transporte, red y enlace de datos se asocian a la capa de *software* del sistema; y las capas restantes (aplicación, presentación y sesión) se relacionan con la capa de *software* de aplicación (Figura 1.7).

1.2.2 Protocolos usados en Internet

El término internet se refiere a un conjunto de redes interconectadas mediante dispositivos llamados *routers*. Cada *router* es una computadora de propósito especial que conecta a dos o más redes; aunque cuenta con una memoria y un procesador, está dedicado a mover datos entre las redes a las que interconecta.

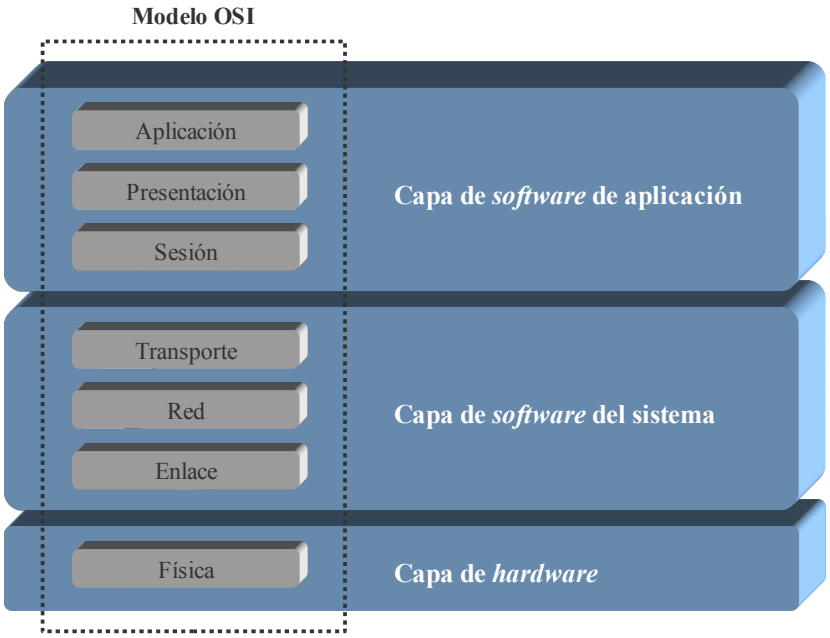


Figura 1.7. Diagrama a bloques del modelo OSI y el modelo de sistemas empotrados [33].

Los dispositivos direccionables que se conectan a una internet se llaman *host*. El *host* puede ser de gran capacidad (por ejemplo una supercomputadora) o pequeña capacidad (una computadora personal o una tarjeta de red); puede ser conectada usando distintos elementos de *hardware* y/o distintos sistemas operativos.

La Internet actual, que es un conjunto de redes que cubre todo el planeta y la cual forma parte de las actividades rutinarias de millones de seres humanos, evolucionó de una red de conmutación de paquetes llamada ARPANET[4]. Esta red está compuesta por un gran número de tecnologías de red, sistemas operativos y diferentes arquitecturas de redes. Todas ellas proveen el mismo tipo de servicio básico independientemente del *hardware* y del *software* que se esté utilizando.

La interoperabilidad en la Internet es posible debido al uso de un conjunto o *suite* de protocolos conocidos como TCP/IP. Esta *suite* de protocolos establece la manera en que deben funcionar las capas de red, transporte y aplicación dejando fuera de la arquitectura las capas físicas y de enlace de datos, que básicamente están relacionadas a la tecnología de red que se utiliza. Lo anterior forma una red virtual homogénea. En la Figura 1.8 se muestra la relación que existe entre los protocolos de la capa de red, la capa de transporte y algunos de la capa de aplicación.

4 Proyecto financiado por el Departamento de Defensa (DOD) de los Estados Unidos.

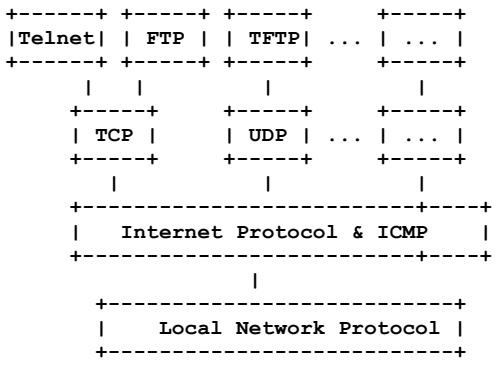


Figura 1.8. Relación entre protocolos de la familia TCP/IP [37].

Las tareas realizadas por las diferentes capas de la arquitectura de protocolos TCP/IP son las siguientes:

- Capa de enlace: Conocida también como capa de interfaz de red (*network interface layer*), normalmente incluye la tarjeta de red, los controladores (*driver*) de dicha tarjeta, también comprende la interfaz física del cableado (o el medio de comunicación disponible).

- Capa de red: También conocida como la capa de Internet (*Internet layer*), es la encargada de mover o enrutar (*routing*) los paquetes dentro de la red.

- Capa de transporte: Provee el flujo de datos entre dos *hosts*. En la *suite* de protocolos TCP/IP básicamente existen dos protocolos de transporte, TCP y UDP.

- Capa de aplicación: Está relacionada con aplicaciones particulares, entre ellas: Telnet (Acceso remoto), FTP (*File Transfer Protocol*). SMTP (*Simple Mail Transfer Protocol*), SNMP (*Simple Network Management Protocol*).

Es posible relacionar la *suite* de protocolos TCP/IP, el modelo de sistemas empotrados y el modelo de referencia OSI (Figura 1.9). En este caso, la capa de enlace del modelo TCP/IP es compartida entre la capa física y la capa de enlace del modelo OSI y entre la capas de *hardware* y *software* del sistema correspondientes al modelo de sistemas empotrados. La capa de red y transporte están situadas en la capa de *software* del sistema. Finalmente la capa de aplicación de TCP/IP se integra en la capa de *software* de aplicación del modelo de sistemas empotrados.

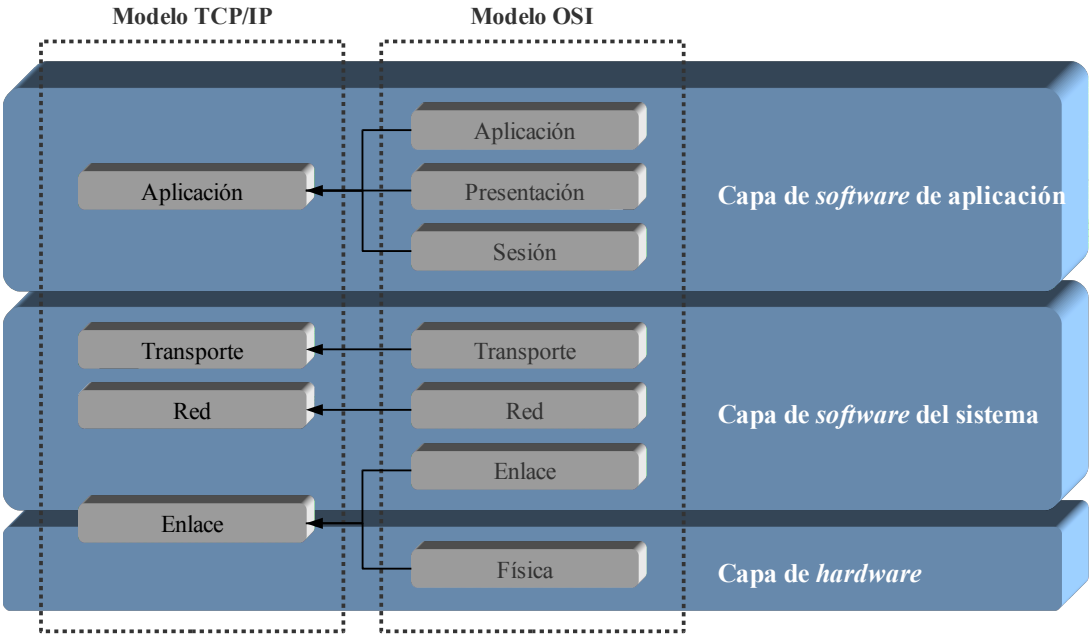


Figura 1.9. El modelo TCP/IP y OSI relacionados con el modelos de SE [33].

1.2.2.1 Los protocolos IP e ICMP

El protocolo principal de la capa de red es el IP, documentado en el RFC 791; este protocolo tiene información de direccionamiento e información de control que permite el enrutamiento de paquetes. El IP tiene dos funciones principales: ofrecer la entrega de datagramas basada en el mejor esfuerzo y sin conexión a través de una internet, y ofrecer la fragmentación y reensamble de datagramas para soportar los enlaces de datos con tamaños diferentes [16].

Debido a la sencillez de IP, éste funciona como un protocolo no orientado a conexión. Existe un protocolo conocido como Protocolo de Mensajes de Control de Internet (ICMP, *Internet Control Message Protocol*), documentado en el RFC 792, que permite comunicar al *software* de IP en un *host* con el *software* correspondiente en otro *host*. Esta comunicación permite mejorar el desempeño de IP, ya que se envían mensajes de control y error.

Los mensajes de error que se reportan mediante ICMP solo reportan la condición de error sin establecer el procedimiento a seguir, y son reportados a la fuente original del paquete que causó el error. Entre los errores que se reportan usando ICMP están: *host*, protocolo o puerto inalcanzable. Además puede realizar funciones administrativas como solicitudes de respuestas de eco. El mensaje de respuesta al eco del ICMP indica que es posible llegar al destino [16].

1.2.2.1.1 Esquema de direccionamiento

El direccionamiento es un concepto importante dentro de Internet. Para dar apariencia de un sistema único y transparente, cada uno de los *host* conectados a una internet deben usar un esquema de direccionamiento[5]. Aunque los esquemas de direccionamiento son abstracciones de *software*, las direcciones de protocolo se usan como destinos en una red virtual, y de manera análoga las direcciones de *hardware* son destinos usados en la red física.

Las direcciones *hardware* también son conocidas como direcciones MAC (*Media Access Control*), las cuales son un subconjunto de direcciones de la capa de enlace. Una dirección MAC está construida con 48 bits de longitud y se expresa como una agrupación de doce dígitos hexadecimales. Los primeros 6 dígitos de la dirección son asignadas por el IEEE e identifican al fabricante (OUI, *Organizationally Unique Identifier*) y los restantes seis dígitos son asignados por el fabricante para identificación interna del dispositivo [44].

El protocolo de Internet oculta los detalles relacionados con la parte física y ofrece características de una red virtual. La red virtual opera casi de la misma manera que un red física, permitiendo que las computadoras transmitan y reciban paquetes de información. La diferencia entre una red física y una red virtual es que esta última es tan solo una abstracción propuesta por los diseñadores y es creada totalmente por *software*. En una red virtual se tiene la libertad de seleccionar direcciones, formatos de paquetes y técnicas de entrega independientemente de los detalles de *hardware*.

En la *suite* de protocolos TCP/IP, el protocolo IP especifica el direccionamiento, este protocolo se encarga de asignar a cada *host* un número único de 32 bits, conocido comúnmente como dirección IP[6]. Dentro de una red basada en TCP/IP, cada paquete enviado contiene la dirección IP del transmisor (origen) y la dirección IP del receptor (destino).

Cada dirección IP de 32 bits se puede dividir en dos partes: prefijo y sufijo; el prefijo de la dirección identifica la red física a la que se está conectada y el sufijo identifica las computadoras de esa red. Dichas direcciones existen en un espacio jerárquico de direcciones, también se les conoce como direcciones lógicas o direcciones virtuales.

El esquema de direccionamiento divide el espacio de direcciones IP en cinco clases, en donde el prefijo y sufijo tienen tamaños diferentes; la clasificación está determinada por los primeros cuatro bits de la dirección. La división del espacio de direcciones se muestra en la Tabla 1.2, las clases A, B y C se llaman clases primarias porque son usadas para direcciones de *host*; la clase D se usa para multitransmisión, lo que permite entregar información a un grupo de computadoras; la clase E es reservada para uso futuro.

---

5 «*Para ofrecer un direccionamiento uniforme en una interred, el protocolo define un esquema de direccionamiento abstracto que asigna a cada host una dirección única. Los usuarios, los programas de aplicación y las capas superiores del protocolo usan las direcciones abstractas para comunicarse*» [10].

6 Direccionamiento IPv4.

Tabla 1.2. Rango de direcciones, cantidades de redes y hosts en las diferentes clases.

Clase de dirección	Rango de cifras	Bits del prefijo	Cant. Máx. de redes	Bits del sufijo	Cant. Máx. de host por red
A	0 a 127	7	128	24	16777216
B	128 a 191	14	16384	16	65535
C	192 a 223	21	2097152	8	256
D	224 a 239	-	-	-	-
E	240 a 255	-	-	-	-

En una internet, cada prefijo de red debe ser único. En las redes conectadas a Internet, las organizaciones reciben sus números de red de compañías de comunicaciones denominadas proveedores de servicios de Internet (ISP). Los proveedores se coordinan con una organización central llamada la Autoridad de Números Asignados en Internet (IANA, *Internet Assigned Numer Authority*), quien se encarga de asegurar que cada prefijo de red sea único en toda la Internet. Dado que un *host* puede o no estar conectado a Internet, se hace la diferencia de su dirección IP según su alcance en una red:

- Dirección pública: Los *host* con direcciones públicas son visibles desde toda la Internet, es decir es posible que un *host* conectado a Internet pueda establecer comunicación con cualquier otro *host* que tenga una dirección pública.

- Dirección privada: Los *host* con direcciones privadas son solo accesibles entre *hosts* de la misma red. Los *host* con direcciones privadas pueden tener acceso a Internet mediante un *host* con una dirección pública.

Las características y datos mencionados anteriormente se aplican a la versión conocida como Ipv4, cuya principal desventaja es que tiene un espacio de direccionamiento limitado. En sus inicios solo existían unas cuantas redes y era posible manejarlas con direcciones de 32 bits, en la actualidad la demanda de prefijos está llegando a sus límites y pronto no será posible mayor crecimiento [10].

El espacio de direccionamiento de IPv4 tiene un límite teórico de 4.3 billones de direcciones y los métodos de distribución para asignar direcciones son ineficientes. Si fuera posible reorganizar el espacio de direccionamiento de IPv4 podría organizarse de manera más efectiva, pero este proceso no es posible, una reorganización y renumeración global no es práctica [20].

IPv6 es la evolución de IPv4, la cual conserva muchas de las características de su predecesor. Este nuevo protocolo opera sin conexiones (cada datagrama tiene una dirección destino y se enruta independientemente). Como en IPv4, la cabecera de cada datagrama tiene una cantidad máxima de saltos que pueden hacerse antes de ser descartado. Además IPv6 conserva la mayor parte de las facilidades generales ofrecidas por las opciones IPv4.

A pesar de que IPv6 mantiene gran similitud con IPv4, los detalles cambian por completo. Por ejemplo, usa direcciones más grandes (direcciones de 128 bits) y un formato de cabecera de datagrama completamente nuevo; también usa una serie de cabeceras de longitud

fija para manejar la información crucial, en lugar de una sola cabecera con un campo de opciones con longitud variable.

1.2.2.2 Los protocolos TCP y UDP

El protocolo IP se encarga de transmitir de un punto a otro de la Internet unidades de datos conocidos como datagramas; generalmente, cada datagrama se trata de forma independiente. En la mayoría de las ocasiones es necesario enviar de un *host* a otro un conjunto de datos de mayor tamaño que un solo datagrama, por lo que es necesario dividir esta información de tal manera que sea manejable para el protocolo IP. La capa de transporte es la encargada de realizar este proceso. En la *suite* de protocolos TCP/IP existen dos opciones para la capa de transporte que son los protocolos TCP y UDP. Un *host* puede ejecutar ambos protocolos simultáneamente.

1.2.2.2.1 Protocolo TCP

El protocolo TCP (*Transmission Control Protocol*), descrito en el RFC 793, proporciona un flujo de datos confiable entre dos *host, s*e encarga de dividir la información recibida de la capa de aplicación en elementos de tamaño apropiada para enviarlos a la capa inferior; también se encarga de unir en el extremo destino dicha información fragmentada, recibida en la capa de red, para entregarlo a la capa de aplicación.

Cuando se habla de comunicación punto a punto se refiere a que las conexiones TCP tienen dos puntos terminales. Entre los servicios que ofrece TCP están [10]:

- El servicio orientado a conexión indica que la aplicación debe solicitar primero una conexión al destino y luego emplearla para transferir datos.

- La confiabilidad garantiza que los datos enviados por una conexión se entregarán exactamente igual, sin datos faltantes ni desordenados.

- La operación *fulll-duplex* permite que los datos fluyan en ambos sentidos y que cualquiera de los programas de aplicación transmitan datos en cualquier momento.

- La interfaz de flujo permite transmitir una secuencia continua de octetos por una conexión.

- El arranque confiable de una conexión requiere que, cuando se establece una conexión entre dos aplicaciones, ambas acepten.

Las conexiones que ofrece TCP se llaman conexiones virtuales porque se establecen en el *software*. El sistema de internet no ofrece apoyo de *hardware* o de *software* a las conexiones, sino que son módulos TCP de dos máquinas las que intercambian mensajes y crean la ilusión de una conexión.

TCP establece una comunicación en la que el origen envía un paquete y espera la confirmación del receptor destino; en caso de que durante un periodo de tiempo no reciba confirmación, el origen reenvía el paquete. Es posible que el paquete enviado o la confirmación se pierda, en cualquiera de los casos, el origen envía nuevamente el paquete y el receptor sabe perfectamente qué fue lo que ocurrió al poner el paquete en el lugar correspondiente. TCP es considerado un protocolo seguro porque cuenta con el mecanismo de *checksum* que permite saber si el paquete ha llegado con algún error o no, si esto ocurre el destino vuelve a pedir la retransmisión de dicho paquete.

1.2.2.2.2 Protocolo UDP

El protocolo UDP (*User Datagram Protocol*), descrito en el RFC 768, no es orientado a conexión y es un servicio mucho más sencillo, solo envía paquetes de datos llamados datagramas desde un *host* a otro, por lo que no se garantiza que lleguen bien a su destino.

UDP es un protocolo que envía datagramas sin esperar respuesta, por lo que usa menos recursos que TCP, lo que lo hace más rápido ya que tiene menos información adicional sobre el datagrama. A diferencia de TCP, UDP no agrega a IP funciones de confiabilidad, control de flujo y recuperación de errores [16].

UDP es útil en situaciones donde no se requieren los mecanismos de confiabilidad que ofrece TCP, por ejemplo, cuando un protocolo de las capas superiores ofrezca las funciones de recuperación de errores y control de flujo. En algunos casos resulta más práctico utilizar este protocolo si no existe mucha información a enviar y no sobrepasa el tamaño máximo de un datagrama (implica que el datagrama no será fragmentado). La conveniencia de usar UDP en lugar de TCP depende de las condiciones de la aplicación que se quiera realizar.

En cada *host* solo existe un módulo TCP y un módulo UDP. En general existen varias aplicaciones que requieren de los servicios de la capa de transporte. TCP y UDP usan el concepto de *puerto* para distinguir entre las diferentes aplicaciones.

Un puerto, en el ámbito de TCP/IP, es un número de 16 bits que hace referencia a un servicio. Existen servicios que están asociados a números de puertos especiales, a esos puertos se les denomina *puertos bien conocidos* y están comprendidos en el rango de 1 a 1023[7]. Como ejemplo se tiene a FTP (20), Telnet (23), HTPP (80), etc. Las aplicaciones de usuarios pueden usar cualquier otro número de puerto no contemplados en esa lista [46].

1.2.2.3 Protocolos de aplicaciones usadas en Internet

1.2.2.3.1 Protocolo de transferencia de hipertexto

El protocolo de transferencia de hipertexto HTTP (*Hypertext Transfer Protocol*),

7 El rango de números son normalmente usados por servicios del SO Unix, la IANA define el rango entre 1 y 1023 como puertos bien conocidos.

descrito en el RFC 2616, es el protocolo base de la WWW (*World Wide Web*) y puede utilizarse en cualquier aplicación que haga uso de hipertextos. El nombre del protocolo puede ser un tanto desafortunado, ya que HTTP no es un protocolo para transferir únicamente hipertexto sino que es un protocolo para transmitir información de distintos tipos pero que cuenta con la eficiencia necesaria para efectuar saltos de hipertexto. Los datos transferidos por el protocolo pueden ser texto nativo, hipertexto, audio, imágenes u otros tipos información de los que están disponible en Internet [45].

HTTP es un protocolo cliente-servidor (sección 1.2.3) orientado a transacciones. El uso más habitual se produce entre un navegador y un servidor web. Para proporcionar fiabilidad, HTTP hace uso de TCP, sin embargo, HTTP es un protocolo "sin estados", cada transacción se trata independientemente, por consiguiente, una implementación típica creará una conexión nueva entre el cliente y el servidor para cada transacción, y la cerrará tan pronto como se complete la transacción.

El caso más sencillo de conexión es cuando el cliente establece una conexión directa con el servidor. El cliente inicia la solicitud, como es el caso de un navegador web actuando de parte del usuario final; el servidor es el que contiene los recursos de interés. Otro caso en que no exista una conexión directa de extremo a extremo, en su lugar, existen uno a más sistemas intermedios con conexiones TCP entre sistemas lógicamente adyacentes. Cada sistema intermedio actúa como un retransmisor, de forma que una solicitud iniciada por el cliente se retransmite a través de los sistemas intermedios hasta el servidor y la respuesta del servidor se retransmite de vuelta al cliente. Se definen tres tipos de sistemas intermedios en la especificación de HTTP:

- *Proxy*: Un *proxy* actúa en nombre de otros clientes y presenta las solicitudes de éstos a un servidor. El *proxy* actúa como servidor cuando interactúa con un cliente y como cliente cuando interactúa con un servidor. Existen dos escenarios que requieren el uso de un *proxy*: Intermediario de seguridad y diferentes versiones de HTTP. Si el cliente y el servidor ejecuta diferentes versiones, el *proxy* puede implementar ambas versiones y realizar las traducciones necesarias. Un *proxy* es un agente de reenvío que recibe solicitudes de objetos URL[8], modifica las solicitudes y las reenvía hacia el servidor identificado en la URL.

- *Gateway*: Es un servidor que se presenta al cliente como si se tratase de un servidor origen. Actúa en nombre de otros servidores que no pueden comunicarse directamente con un cliente. Existen dos escenarios en los que se pueden utilizar *gateways*: intermediarios de seguridad y servidores que no usan HTTP.

- *Tunnel*: A diferencia del *proxy* y del *gateway*, el túnel no realiza operaciones sobre las solicitudes y respuestas HTTP. En su lugar, un túnel es simplemente un punto de retransmisión entre dos conexiones TCP y los mensajes HTTP se transfieren sin modificaciones, como si hubiera una única conexión HTTP entre el agente usuario y el servidor.

8 URL (*Uniform Resource Locator*), localizador uniforme de recursos.

1.2.2.3.2 Sistema de nombres de dominio

El sistema de nombres de dominio (DNS, *Domain Name System*) es una base de datos distribuida usada en Internet para mantener la relación entre el nombre de un *host* y su dirección IP, también proporciona información de ruteo al servicio de correo electrónico. El término base de datos distribuida se emplea porque no existe una sola base de datos que contenga todos y cada uno de los nombres de domino (*hostnames*); en cambio, cada organización, universidad o departamento de una empresa, mantiene su propia información de *hostnames* y de direcciones IP [46].

La base de datos distribuida DNS es indexada por nombres de dominio. En esencia, cada nombre de dominio es una ruta en un diagrama parecido a un árbol invertido (Figura 1.10). El árbol tiene su inicio en un elemento llamado raíz, las intersecciones de las rutas se llaman nodos. La profundidad del árbol está limitada a una profundidad de 127 niveles [24].

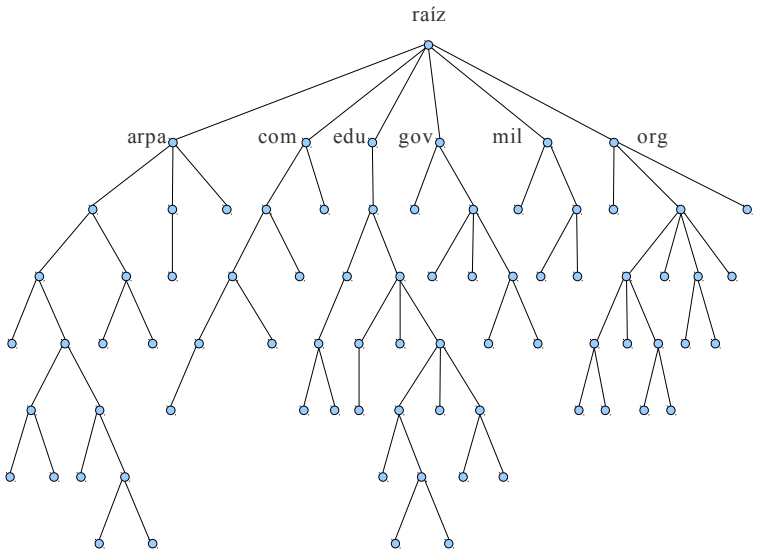


Figura 1.10. Estructura de nombres de dominio DNS.

Cada nodo en el árbol está etiquetado con un máximo 63 caracteres; se reserva la etiqueta nula para la raíz del árbol. El nombre de dominio completo para cualquier nodo en el árbol es la secuencia de etiquetas siguiendo la ruta desde el nodo raíz. Los nombres de dominio siempre son leídos desde el nodo más profundo del árbol hacia arriba hasta llegar a la raíz y son separados por puntos. Un dominio es simplemente un subárbol de los nombres de espacio de dominio.

Los conceptos y facilidades proporcionadas por el uso de un DNS están documentadas en el RFC 1034, los detalles de la implementación y las especificaciones de desarrollo se encuentran en el RFC 1035.

1.2.3 Modelo cliente-servidor

El sistema de redes interconectadas ofrece un canal de comunicación entre aplicaciones, el protocolo no puede iniciar ni aceptar el contacto con una computadora remota. Para establecer cualquier comunicación deben participar dos programas de aplicación: uno que inicia la comunicación y otro que la acepte.

El paradigma cliente-servidor contempla una aplicación que espera pasivamente a que otra inicie la comunicación. Los términos cliente y servidor se refieren a las dos aplicaciones que participan en la comunicación. La aplicación que inicia la comunicación se denomina cliente y la que espera pasivamente se denomina servidor. Las aplicaciones desarrolladas con este paradigma utilizan protocolos de la capa de transporte para comunicarse [10].

El propósito del servidor es ofrecer un servicio al cliente, como puede ser: almacenar archivos de propósito general (FTP), almacenar código HTML con información para páginas web, gestionar el envío de mensajes de correo electrónico y otros más. Un cliente realiza funciones como traer archivos almacenados desde un servidor, desplegar páginas web, escribir/leer mensaje de correo electrónico y otras funciones. Se pueden diferenciar a los servidores en dos clases [46]:

- Servidores iterativos: Están vinculados a una serie de pasos repetitivos; para explicar el funcionamiento de estos servidores se tiene el siguiente algoritmo:

  1. Esperar que la petición del cliente ocurra.
  2. Procesar la petición del cliente.
  3. Enviar la respuesta generada hacia el cliente.
  4. Regresar al paso 1.

  El problema con un servidor de este tipo es que en el paso 2 se detiene el proceso de recepción de peticiones mientras se atiende a la petición actual.

- Servidor concurrente: Soluciona el problema anterior usando los siguientes pasos:

  1. Esperar que la petición del cliente ocurra.
  2. Iniciar un nuevo servicio en el servidor para atender a la petición entrante. Esto involucra crear un nuevo proceso, tarea o hilo dependiendo del sistema operativo.
  3. Regresar al paso 1.

La ventaja de un servidor concurrente es que el servidor genera otros procesos servidores para manejar las solicitudes del cliente. En esencia cada cliente tiene su propio servidor. Suponiendo que el sistema operativo permita multitarea, se pueden atender múltiples clientes al mismo tiempo. Como regla general, los servidores concurrentes se basan en TCP y los servidores iterativos en UDP [46].

La información intercambiada entre clientes y servidores puede fluir en ambas direcciones; aunque muchos servicios acuerdan que el cliente mande peticiones y el servidor responda, se pueden dar otras interacciones [10].

1.2.4 Interfaz de socket

La interfaz que usan las aplicaciones para interactuar con el protocolo de la capa de transporte se conoce como interfaz de programación de aplicaciones (API, *Application Programming Interface*). La API es un conjunto de procedimientos que utilizan las aplicaciones para cada operación básica (Tabla 1.3).

Tabla 1.3. Procedimientos básicos de sockets usados en TCP/IP [47].

Primitiva	Significado
Socket	Crea una nueva comunicación de punto final.
Bind	Agrega la dirección IP local al *socket*.
Listen	Anuncia que está disponible para aceptar conexiones.
Accept	Atiende una conexión entrante.
Connect	El *socket* actual estableció una conexión.
Send	Envía datos sobre la conexión establecida.
Receive	Recibe datos desde una conexión remota.
Close	Libera la conexión.

La API de *socket*[9] se originó en un proyecto de la Universidad de California en Berkeley, es por ello que a la API de *socket* se le conoce también como interfaz de *sockets* de Berkeley; dicha interfaz se elaboró y distribuyó en una versión del sistema operativo Unix BSD (*Barkeley Software Distribution*) con protocolos de interconectividad TCP/IP. Un *socket* se puede considerar como un punto final en una comunicación [45].

Los *sockets* pueden crearse desde un programa (en lenguajes como C o java, entre otros), permitiendo al programador proporcionar funciones y aplicaciones de red. El mecanismo de programación de *socket* incluye una semántica suficiente para permitir que se comuniquen dos procesos independientes en *host* diferentes o en el mismo (Figura 1.11). En la programación de *sockets* se debe especificar un protocolo de la capa de transporte (TCP o UDP), proveer la dirección de protocolo de la máquina remota, el puerto remoto y definir si se comportará como un cliente o como un servidor [10] [46].

La interfaz de *socket* de Berkeley es un estándar *de facto* para la API *sockets* que sirve para el desarrollo de aplicaciones de red. La API proporciona acceso genérico a servicios de comunicación entre procesos [10].

9 La API de *socket* se abrevia también como *sockets*.

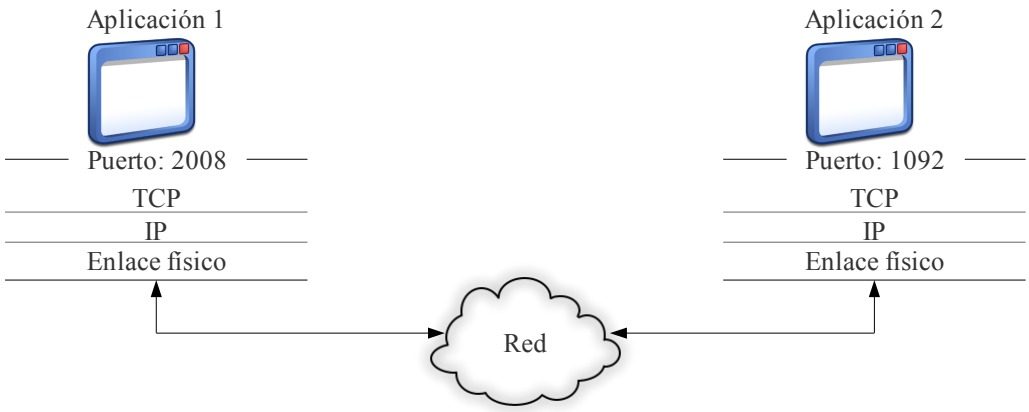


Figura 1.11. Interfaz de programación de aplicaciones de socket.

1.3 Herramientas de desarrollo para aplicaciones de Internet

El usuario final de un sistema requiere una interfaz que sea sencilla, comprensible, configurable, segura, fácil de instalar y algunas otras características que son ofrecidas por las interfaces basadas en web. El código de estas interfaces se encuentra almacenado en un servidor y se escriben usando alguno(s) de los lenguajes mostrada en la Tabla 1.4. El más usado de ellos es HTML, aunque últimamente tiende a usarse en combinación con otros lenguajes. Una interfaz construida utilizando únicamente HTML, ofrece una sola presentación, es estática y no permite almacenar datos.

Tabla 1.4. Algunos lenguajes de programación disponibles para desarrollos web.

Lenguaje	Descripción
ActionScript (Flash)	Diseños basados en el reproductor Adobe Flash.
AJAX	(*Asynchronous JavaScript And XML*) diseño de aplicaciones interactivas.
ASP	(*Active Server Page*) Creado por Microsoft para desarrollos web.
ASP.NET	La versión ASP que trabaja con Frameworks.
CSS	Hojas de estilos para formato de documentos.
HTML	Lenguaje de programación basado en etiquetas. Muy común en paginas web.
JavaScript	Versión popular para ejecutar *scripts*.
JSP	(*Java Server Pages*) Otra versión de *scripts* basado en java.
VBScript	*Scripts* basados en lenguaje de programación Visual Basic.
Macromedia ColdFusion	Servidor de páginas web de la empresa Macromedia.
Perl	Parecido a lenguaje C, lenguaje interpretado *shell* y LISP entre otros.
PHP	Lenguaje que se puede empotrar a código HTML.
Python	Parecido a Perl. Uso de lenguaje interpretado.
Ruby	Lenguaje de programación interpretado, reflexivo y orientado a objetos.
Web 2.0	Orientado a la interacción con el usuario.
XML	Genera un formato a documentos HTML.

La interfaz web puede ser desplegada en una PC, en un PDA, en un teléfono celular u otros dispositivos. Cada uno de ellos presenta características distintas en cuanto a tamaño, velocidad y resolución, por lo que un solo tipo de presentación no es la mejor opción para todos ellos. Las hojas de estilo permiten que los documentos en HTML puedan presentarse de forma diferente dependiendo del dispositivos que lo haga.

Las interfaces para controlar sistemas, además de presentar información al usuario, requieren que éste envíe datos u órdenes. La interactividad implícita en estas acciones se puede lograr con código anexo al HTML y que puede ejecutarse del lado del servidor o del lado del cliente. Ejecutarlo del lado del cliente implica tener riesgos en la seguridad ya que es susceptible a modificaciones malintencionadas por parte de los usuarios. Para ejecutar código del lado del servidor que permita interactividad, una de las opciones es usar el lenguaje de programación PHP.

Muchas aplicaciones como los sistemas de seguridad necesitan almacenar datos que permitan al usuario tener información del funcionamiento del mismo. Una opción viable es utilizar una base de datos, un manejador de bases de datos permite ser integrado en interfaces web, tal es el caso de MySQL.

1.3.1 Lenguaje de marcas de hipertexto

El lenguaje de marcas de hipertexto (HTML, *HyperText Markup Language*) es un lenguaje de programación que permite al programador dar las guías generales de presentación e indicar el contenido de un documento. Un documento de hipertexto disponible en Internet se llama página web.

HTML es un lenguaje basado en etiquetas que no incluye instrucciones detalladas de formato, sin embargo, permite que el documento tenga guías generales de presentación y que el navegador[10] se encargue de los detalles. En consecuencia, dos navegadores pueden presentar de manera diferente un mismo documento HTML.

Los navegadores web tienen una estructura general muy similar entre ellos (Figura 1.12). El software, que forma la presentación dirigida a la pantalla, se conoce como motor de renderizado. La Tabla 1.5 muestra los motores más usados y los navegadores que se basan en ellos [23].

10 Un navegador es un programa de aplicación que permite al usuario ver información de la WWW.

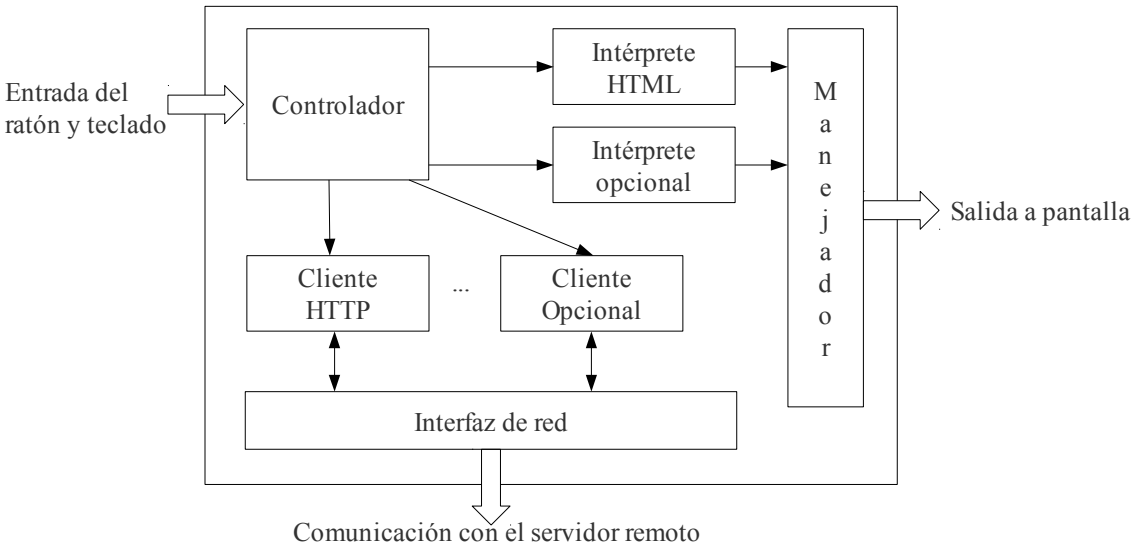


Figura 1.12. Arquitectura de un navegador web [10]

Tabla 1.5. Navegadores web disponibles en Internet.

Basado en	Nombre	Sistema Operativo
Gecko	Mozilla Firefox	Multiple
	Nestcape Navigator	Linux, Mac OS, Windows
	Galeon	Unix
	Epiphany	Unix
	Camino	Mac OS
Trident	Internet Explorer	Mac OS, Solaris, Windows
	MSN Explorer	Windows
	Windows Explorer	Windows
GRML	Pioneer Report MDI	Windows
	Tree MDI	Windows
	Bar Graph MDI	Windows
KHTML	Konqueror	Linux
	Safari	Mac OS, Windows

Un documento HTML se divide en dos partes, la cabecera y el cuerpo. La cabecera tiene información referente al documento y el cuerpo tiene la información que se presenta en la página web. Existe una gran cantidad de literatura disponible donde se describen las posibilidades de HTML: Introducción a XHTML [13], HTML, XHTML, and CSS [7], Head First HTML with CSS & XHTML [17], Build Your Own Web Site The Right Way Using HTML & CSS [26], HTML, XHTML, and CSS All-in-One Desk Reference For Dummies [21], HTML & XHTML: The Definitive Guide [30].

1.3.2 Hojas de estilo

La utilización de CSS (*Cascading Style Sheets*) en el desarrollo de páginas web permite separar las tareas de los diseñadores y las de los escritores de contenido. CSS define el diseño de la presentación o cómo se va a imprimir el documento en un archivo separado de la información [19].

CSS se utiliza para dar presentación a documentos HTML y XML (*eXtensible Markup Language*), separando el contenido de la presentación. CSS funciona en base a reglas, las cuales definen la apariencia de cada etiqueta HTML. Las reglas tiene dos partes, selector y declaración, y la declaración está compuesta por una propiedad y un valor.

La W3C (*World Wide Web Consortium*) propuso la creación de hojas de estilo para representar la misma información de páginas web en distintos dispositivos. Originalmente existieron dos propuestas para resolver el problema de los navegadores: CHSS (*Cascading HTML Style Sheets*) propuesto por Hakon Wium Lie y SSP (*Stream-based Style Sheet Proposal*) propuesto por Bert Bos. A finales de 1994 y principios de 1995, Lie y Bos se reunieron para definir un nuevo lenguaje que tomara lo mejor de cada propuesta, consiguiendo lo que hoy se conoce como CSS [12].

En 1995, la W3C estandarizó CSS y fue añadido al grupo de trabajo de HTML. A finales de 1996, la W3C publicó su primera recomendación de hojas de estilo llamada “CSS nivel 1”. El 12 de Mayo de 1998 se publicó la segunda versión de CSS llamada “CSS nivel 2”. El primer navegador que dio soporte completo a “CSS nivel 1” fue la versión Internet Explorer para el sistema operativo de Macintosh. Hasta el momento ningún navegador web soporta por completo “CSS nivel 2”.

La literatura que se puede consultar para el tema: Introduccion a CSS [12], CSS avanzado [14], CSS Cookbook [41], CSS: The Definitive Guide [27], The Zen of CSS Design: Visual Enlightenment for the Web [42].

1.3.3 PHP Hypertext Pre-processor

«*PHP, acrónimo de “PHP: Hypertext Preprocessor”, es un lenguaje “Open Source” interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender*» [48].

PHP fue desarrollado por el *PHP Group* en el año de 1995 y es influenciado por los lenguajes de programación C, C++, Perl, Java, Python y Ruby. PHP es un lenguaje que se ejecuta en el lado del servidor. La última versión estable liberada al público es la 5.2.8, fechada el 8 de diciembre de 2008 bajo la licencia *PHP License ver 3.01* [48].

La función que hace PHP dentro del desarrollo de páginas web es generar código HTML. El código generado se basa en la toma de decisiones, bucles u otras operaciones

internas propias del programador. El lenguaje PHP puede establecer comunicación con bases de datos, siendo éstas una de las fortalezas del lenguaje. También tiene la capacidad de usar *sockets* y realizar comunicación con otras aplicaciones que cuenten con esta característica. La Tabla 1.6 lista las bases de datos soportadas por la versión 5.2.8 de PHP [36] [48].

Tabla 1.6. Base de datos soportados en PHP.

Base de datos			
Adabas D	FilePro (read-only)	InterBase	PostgreSQL
dBase	Hyperwave	MySQL	Solid
Direct MS-SQL	IBM DB2	ODBC	Sybase
Empress	Informix	Oracle (OCI7 and OCI8)	Velocis
FrontBase	Ingres	Ovrimos	Unix dbm

1.3.4 MySQL

MySQL es un sistema de gestión de base de datos SQL; lo desarrolla, distribuye y soporta la empresa MySQL AB (derivada de la empresa *Sun Microsystems*), y se distribuye baja licencia GNU.

«*El software MySQL® proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido*» [31].

Una base de datos es una colección estructurada de datos; por ejemplo una lista de productos para comprar, un listado de clientes de una fábrica o el directorio de los contactos de una empresa.

MySQL es un gestor de base de datos relacionales, las cuales tiene como característica el tener tablas de registros separadas, con lo que se logra una mejor velocidad en las consultas y una mayor flexibilidad de los datos. SQL es un lenguaje estandarizado en el ámbito de las bases de datos y está definido en el documento ANSI/ISO SQL. Existen diversas versiones de este documento distinguidas por el año de aparición: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008.

MySQL cuenta con características que se requieren en el desarrollo de aplicaciones, algunas de estas características son:

- Escrito en C y C++.
- Probado en un amplio rango de compiladores diferentes.
- Funciona en diversas plataformas (Linux, Sun Solaris, Windows).
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y Tcl.
- Uso completo de *multi-threaded* mediante *threads* del kernel.
- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.

- Usa un sistema de reserva de memoria muy rápido basado en hilos.
- Tablas *hash* en memoria, usadas como tablas temporales.

La seguridad que maneja MySQL se basa en la verificación de contraseña, el proceso de validación se encuentra en el servidor y todo el tráfico generado en las operaciones está totalmente cifrado. Las operaciones realizadas entre el gestor MySQL y los clientes se manejan usando TCP/IP.

Capítulo 2
Diseño e implementación

La metodología de desarrollo de sistemas empotrados [3] [8] y el modelo de sistemas empotrados [33], describen las fases de diseño e implementación utilizadas durante el desarrollo de este trabajo. Estos modelos permiten describir de forma detallada los componentes contenidos en un sistema.

La Figura 2.1 muestra el nivel más alto de abstracción del sistema y su correspondencia con las capas del modelo de sistemas empotrados.

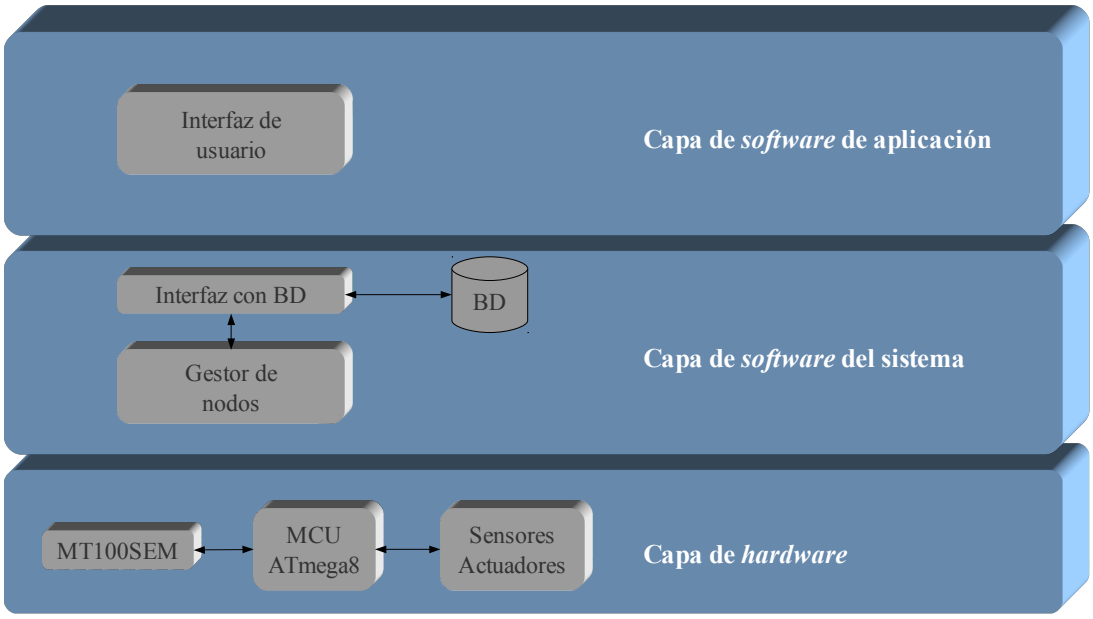


Figura 2.1. Desarrollo del sistema usando el modelo de sistemas empotrados.

La capa de *software* de aplicación contiene la GUI, la cual corresponde al elemento que interactúa con el usuario. La capa de *software* del sistema contempla todas las funciones del sistema (a nivel de lenguaje de programación) y su comunicación con la base de datos. La capa de *hardware* contempla los elementos *hardware* del sistema, siendo estos los dos dispositivos principales: MT100SEM y MCU Atmega8.

2.1 Especificación del producto

El objetivo de este proyecto es establecer comunicación entre un sistema empotrado (nodo) y un centro de información (gestor de nodos). El intercambio de información entre un nodo y el gestor de nodos se realiza usando protocolos de la *suite* TCP/IP.

2.1.1 Comunicación Ethernet

El encargado de implementar el protocolo IP es el dispositivo MT100SEM de la firma *Multitech System*s y tiene las siguientes características [28]:

- Soporte para IEEE 802.3 y TCP/IP (IP v4).
- Señalización de estados (velocidad, enlace, actividad, colisión y modo *dúplex*).
- Soporte para el estándar RS232 a velocidades de transferencia entre 300 y 115.2 kbps.

El MT100SEM tiene dos modos de operación:

- Modo órdenes (predeterminado): En este modo el MT100SEM acepta órdenes del tipo AT para configurar su funcionamiento.
- Modo datos: Este modo permite transmitir y recibir datos usando un *socket* TCP/IP; en este modo no es posible utilizar las órdenes AT.

2.1.2 Administrador de nodo

El MCU Atmega8 [1] de la firma Atmel es el dispositivo encargado de enviar las órdenes AT y controlar el intercambio de información con el MT100SEM. Al MCU se le puede considerar como administrador del nodo pues es quien se encarga de configurar y controlar los dispositivos periféricos del nodo. Las tareas del administrador de nodo son:

- Configurar y gestionar el MT100SEM.
- Configurar y enviar datos al LCD (*Liquid Crystal Display*)[11].
- Controlar E/S digitales y entradas analógicas.
- Detectar la presencia de enlace en el dispositivo MT100SEM.
- Detectar *reset* externo del dispositivo MT100SEM.
- Generar *reset* por *software* del dispositivo MT100SEM.
- Comunicación serial mediante el estándar RS232 a un *bauderate* de 9.6 kbps.

2.1.3 Gestor de nodos

El gestor de nodos realiza intercambio bidireccional de información con los nodos y con la GUI. Las tareas que componen este módulo son las siguientes:

- Obtener datos de los nodos registrados desde la base de datos (BD).
- Buscar usuarios registrados en la BD.

11 Este elemento es opcional y tiene la finalidad de depurar el funcionamiento del nodo. Para hacer uso de este elemento es necesario eliminar dos salidas digitales.

- Buscar el *socket* de comunicación con el nodo.
- Establecer si el nodo está activo y/o registrado.
- Establecer si la orden del usuario es válida.
- Establecer la activación y desactivación del nodo.
- Generar la lista de nodos para información de la GUI.
- Gestionar las peticiones/respuestas de la GUI.
- Establecer el protocolo de comunicación entre los nodos y la GUI.

2.1.4 Gestor de base de datos

El gestor de base de datos realiza la comunicación entre el gestor de nodos y la base de datos. *MySQL Connector* 5.1 de la firma Sun Microsystems es el encargado de realizar la tarea de intercambio de información. Las tareas realizadas por este modulo son [32]:

- Asignar la dirección del servidor y el puerto remoto.
- Asignar el usuario y la contraseña.
- Seleccionar la BD a utilizar.
- Leer y escribir registros en alguna tabla.
- Filtrar la información almacenada de alguna tabla.

2.1.5 Interfaz de usuario

La GUI es el elemento de interacción con el usuario final y es realizado en base a una interfaz web. Las tareas realizadas en la interfaz son las siguientes:

- Identificar al usuario.
- Mostrar nodos activos y no activos.
- Interactuar con las salidas del nodo seleccionado.
- Mostrar las entradas digitales/analógicas del nodo seleccionado.
- Mostrar información de algún nodo.

2.2 División hardware y software

La metodología de desarrollo para sistemas empotrados indica que se deben identificar y dividir las tareas *hardware* (HW) y las tareas *software* (SW) (Tabla 2.1).

Tabla 2.1. División de las tareas HW y SW.

Hardware	Software
Comunicación TCP/IP	Configuración del MT100SEM
Comunicación RS232	Configuración del LCD
Señalización de estados	Generar *reset* del MT100SEM
Gestión de MT100SEM	Gestión de E/S digitales y entradas analógicas
Control de LCD	Obtener información de nodos registrados
Detección de MT100SEM	Buscar usuarios registrados

Hardware	Software
Detección de *reset* externo de MT100SEM	Buscar *socket* de comunicación con el nodo
	Establecer sí el nodo está activo y/o registrado
	Establecer sí la orden es válida
	Activar y desactivar el nodo
	Generar lista de nodos
	Gestión de petición/respuesta de la interfaz
	Establecer el protocolo de comunicación
	Asignar la dirección y puerto del servidor remoto
	Asignar usuario y contraseña
	Seleccionar la base de datos
	Leer y escribir registro en alguna tabla
	Filtrar información de alguna tabla
	Identificar al usuario
	Mostrar nodos activos y no activos
	Interactuar con las salidas del nodo seleccionado
	Mostrar las entradas digitales y analógicas del nodo
	Mostrar información de algún nodo

2.2.1 Selección de HW y SW

2.2.1.1 Comunicación Ethernet

El presente proyecto parte del objetivo de establecer comunicación vía Ethernet entre dos elementos. Se tienen varias opciones para realizar dicha tarea (Tabla 2.2).

Tabla 2.2. Comparativa de algunos dispositivos que soportan Ethernet.

Dispositivo	Fabricante	Comunicación	MAC	Protocolos	10/100 Base-T	Actualización Firmware	Precio[12]
LXT972A [35]	Intel	MII	No	-	Sí	-	-
ENC28J60	Microchip	SPI	Sí	-	10Base-T	-	$5.37 [13]
PIC18F97J60	Microchip	SPI, I^2C, RS232	Sí	-	10Base-T	Sí	$15.26 [13]
MT100SEM	Multitech	RS232	Sí	ICMP, TCP/IP	Sí	Sí	$43.22 [14]
RCM4200	Rabbit	SPI, RS232	Sí	TCP/IP	Sí	Sí	$81.00 [15]

Los dispositivos LXT972A y ENC28J60 son pasarelas, para su uso es necesario que el dispositivo maestro (MCU) tenga los módulos adecuados para establecer comunicación. En el caso del LTX972A es necesario que el MCU cuente con una Interfaz de Medios Independientes (MII, *Media Independent Interface*) para que funcione correctamente; el MCU

12 Los precios manejados están en dolares. Fecha de cotización: Abril 2009.
13 Newark, México. http://mexico.newark.com.
14 AG Electrónica, Newark, México. http://mexico.newark.com.
15 Rabbit Core Modules. http://www.rabbit.com.

DS80C400 es un elemento que proporciona funcionalidad al dispositivo mencionado como se propone en [35]. El dispositivo ENC28J60 requiere una comunicación SPI (*Serial Peripheral Interface*) que es común en dispositivos MCU. El tiempo de desarrollo que implica utilizar una pasarela no es el apropiado para este trabajo, con lo cual se eliminan las dos opciones. Los tres últimos dispositivos de la Tabla 2.2 cumplen con las características deseadas para el desarrollo del proyecto, la selección depende del estudio de factibilidad y el presupuesto con que se cuenta. Cabe señalar que en la mayoría de los casos, la elección del dispositivo queda a criterio de los desarrolladores [8].

La Tabla 2.3 muestra las características del dispositivo MT100SEM de la firma Multitech, del cual se cuenta con dos elementos en el Cuerpo Académico de Redes de Instrumentación del Instituto de Electrónica y Computación (IEC) de la Universidad Tecnológica de la Mixteca.

Tabla 2.3. Principales características del dispositivo MT100SEM.

Características	
Protocolos soportados: ARP, DHCP, FTP, ICMP, IP, TCP, TELNET	Control de flujo RST/CTS
Soporte para 10BaseT Ethernet	Configuración RS232
Soporte para 10/100BaseT Ethernet	Función *MAC Bridge*
Memoria flash de 2 Mb	Velocidad máxima de transferencia serial de 230 Kbps
Almacenamiento para 256 *frames*	Voltaje de operación 3.3 – 5 VCD
Half y Full Duplex	Dimensiones 1” x 2.5”

El empaquetado del dispositivo MT100SEM es compatible con otros dispositivos de la misma firma, como son MT800SWM (comunicación inalámbrica vía Wi-Fi®) y MTS2BTSMI (comunicación inalámbrica vía Bluetooth®), sin requerir realizar cambios mayores en la configuración del sistema.

2.2.1.2 Administrador del nodo

La selección de las herramientas de desarrollo determinan las características del sistema. Para el desarrollo del administrador de nodo se seleccionaron herramientas de la firma Atmel, tanto para el *hardware* como para el *software*.

Para definir las características del MCU se realizó un sistema prototipo inicial, con el cual se llegó a una aproximación de las características generales para cumplir con los objetivos:

- Memoria de código de 2 Kb.
- Memoria de datos de 1 Kb.
- Memoria de almacenamiento de 512 bytes.
- Cuatro entradas analógicas.
- Ocho E/S digitales.

La herramienta HW seleccionada para programar al MCU es la tarjeta de desarrollo

AVR Dragon de la firma Atmel (Figura 2.2), la cual es un programador de MCU de las familias: *ATtiny*, *ATmega*, *ATcan*, *ATpwm* y *ATusb* [2].

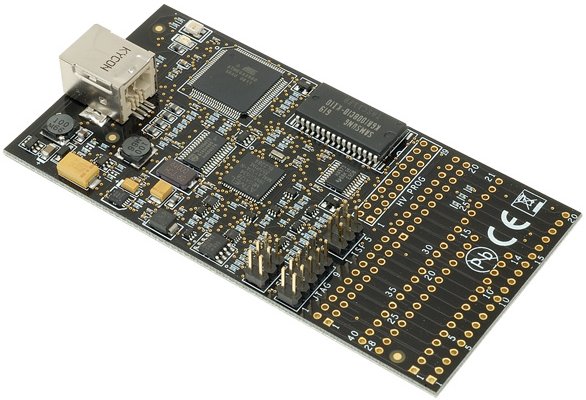


Figura 2.2. Tarjeta de desarrollo AVR Dragon de la firma Atmel.

La selección de un MCU depende de las características y propósitos para el que fue desarrollado, desechando desde un principio las familias *ATcan*, *ATpwm* y *ATusb*, debido a que tienen una área de aplicación diferente.

De acuerdo con los objetivos de este trabajo y las características deseables del MCU, se realiza una comparativa entre algunos de los MCU de las familias *ATtiny* y *ATmega* (Tabla 2.4); los dispositivos que cumplen con dichos requerimientos son: ATtiny84, ATmega8, ATmega16, ATmega32. ATmega64, ATmega128.

Tabla 2.4. Comparativa de algunos MCU soportados por la herramienta AVR Dragon.

Dispositivo	Memoria Flash [Kb]	Memoria SRAM	Memoria EEPROM	Velocidad Máx. [MHz]	Interrupción Externa	Canales ADC	Terminales E/S
ATtiny12	1	-	64b	8	1	-	6
ATtiny13	1	64b	64b	20	1	4	6
ATtiny2313	2	128b	128b	20	2	-	18
ATtiny24	2	128b	128b	20	1	8	12
ATtiny26	2	128b	128b	16	1	11	16
ATtiny44	4	256b	256b	20	1	8	12
ATtiny84	8	512b	512b	20	1	8	12
ATtiny85	8	512b	512b	16	1	4	6
ATmega8	8	1Kb	512b	16	2	6	23
ATmega16	16	1Kb	1Kb	16	2	8	32
ATmeg32	32	2Kb	2Kb	16	3	8	32
ATmeg64	64	4Kb	4Kb	16	8	8	53
ATmega128	128	4Kb	4Kb	16	8	8	53

Los dispositivos ATmega16, ATmega32, ATmega64 y ATmega128 cumplen con las características deseables pero exceden las características mínimas, por lo que la selección del

MCU se reduce al uso del ATtiny84 o ATmega8. El ATtiny84 se descarta debido a que utilizaría todo sus recursos para ser utilizado en este proyecto sin ofrecer la posibilidad de expandir el sistema; en cambio el ATmega8 permite satisfacer los requerimiento y dando la posibilidad de aumentar los requerimiento del sistema dado que cuenta con más recursos.

Como resultado de la comparativa de los MCU se elige como administrador de nodo el ATmega8 por las siguientes características:

- Arquitectura Harvard.
- Tarjeta de desarrollo *AVR Dragon* compatible con el MCU ATmega8 y el *software* AVR Studio.
- Velocidad de operación máxima de 16 MHz.
- Dos *timer/counter* de 8 bits, un modo de comparación.
- Un *timer/counter* de 16 bits, un modo de comparación y un modo de captura.
- Tres canales PWM (*Pulse Witdh Modulation*).
- Seis canales multiplexados de ADC (*Analog to Digital Converter*).
- Soporte para protocolo TWI (*Two-wire Serial Interface*).
- Soporte para USART (*Universal Synchronous/Asynchronous Receiver/Transmitter*).
- Soporte para interfaz serial SPI (*Serial Peripheral Interface*).

Teniendo en cuenta que las decisiones de selección de SW son de menor riesgo que las hechas por el HW, se muestran las herramientas para el diseño HW y SW, simulación e implementación:

- MT100SEM como pasarela RS232-Ethernet.
- ATmega8 como administrador del nodo.
- SW de aplicación AVR Studio® 4.13.
- Tarjeta de desarrollo *AVR Dragon*.
- Lenguaje de programación PHP.
- Hojas de estilo CSS.
- Gestor de bases de datos MySQL.

2.3 Iteración e implementación

El desarrollo de este trabajo requirió iteraciones para cumplir con los objetivos. Cada iteración resolvió algún objetivo, permitiendo depurar con claridad cada uno de los elementos que componen al sistema. El trabajo fue dividido en cinco iteraciones generales, cada una con un propósito específico.

- Iteración 1: Familiarización con el dispositivo MT100SEM.
- Iteración 2: Control del MT100SEM por el MCU.
- Iteración 3: Comunicación del nodo con el gestor de nodos.
- Iteración 4: Comunicación del gestor de nodos con la GUI.
- Iteración 5: Comunicación básica entre los elementos del sistema.

La primera iteración está integrada por una PC, el dispositivo MT100SEM y una conexión a una red local (Figura 2.3). Como primer tarea se establece la comunicación entre la PC y el MT100SEM por medio del protocolo RS232, posteriormente se configura la velocidad de transferencia (*baudrate*) a 9600 bps[16] para obtener las distintas respuestas a las órdenes utilizadas. Una vez configurado el dispositivo MT100SEM, se continua con la tarea de hacer un *ping* para determinar si el MT100SEM puede comunicarse a nivel IP con la PC. Para finalizar se establece la comunicación TCP/IP entre el MT100SEM y la PC. Con los pasos anteriores se comprueba el correcto funcionamiento del MT100SEM.

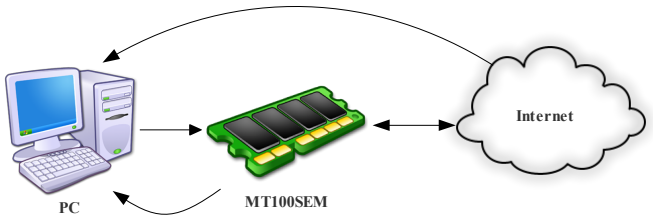


Figura 2.3. Familiarización con el dispositivo MT100SEM.

Una vez que se conoce el comportamiento del dispositivo MT100SEM, se procede a realizar la segunda iteración donde se integra el MCU como nuevo elemento (Figura 2.4). La tarea principal del MCU es configurar el dispositivo MT100SEM e interactuar con los dispositivos de E/S (digitales y analógicos). En base a la lista de órdenes y respuestas obtenidas en la primera iteración, el MCU se programa para realizar las operaciones de configuración y gestión de la comunicación TCP/IP.

16 El dispositivo MT100SEM tiene como valor predeterminado una velocidad de transferencia de 115.2 Kbps.

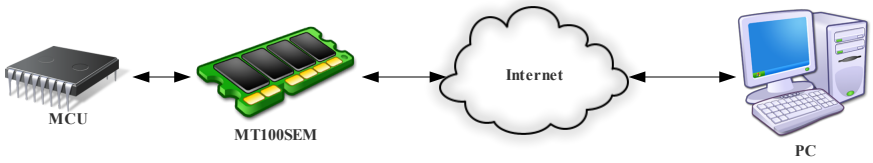


Figura 2.4. Reconocimiento y comunicación con una PC.

Las tareas de configuración consisten en asignar: una dirección IP (*IP address*), la máscara de sub-red (*net mask*), la puerta de enlace predeterminada (*default gateway*), el servidor DNS, la dirección remota para realizar un *ping*, y la dirección y el puerto remoto para establecer la comunicación TCP/IP. Las tareas de gestión consisten en identificar el dispositivo y manejar la lectura o escritura en una terminal del MCU. Esta iteración comprueba la configuración y la comunicación TCP/IP entre el dispositivo MT100SEM y la PC.

Una vez establecida la comunicación TCP/IP entre los elementos inicial (MCU) y final (PC), se continua con la tercera iteración, que tiene como objetivo principal desarrollar el gestor de nodos (Figura 2.5). Después de que el MCU establece la comunicación TCP/IP, es necesario que envíe el nombre y la dirección MAC (*Media Access Control*) del dispositivo MT100SEM con el que se estableció la comunicación[17], esto se debe a que el gestor de nodos necesita identificar al dispositivo para procesar y almacenar su información.

La tarea del gestor de nodos es identificar al dispositivo entrante, registrar su hora y fecha de ingreso, registrar los cambios realizados en las terminales de E/S dentro del MCU e información generada por el nodo. Esta iteración comprueba el manejo de los nodos, los cambios ocurridos en cada uno y la parte del protocolo diseñado entre el nodo y el gestor de nodos.

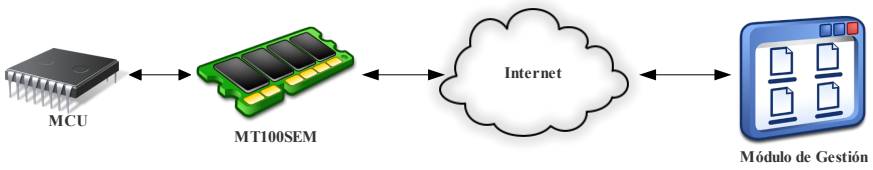


Figura 2.5. Identificación del nodo con el Módulo de Gestión.

La cuarta iteración consiste en realizar la GUI y el protocolo de comunicación entre dicha interfaz y el gestor de nodos (Figura 2.6). Esta iteración tiene como tarea principal establecer comunicación TCP/IP con el gestor de nodos, es necesario utilizar *sockets* para cumplir con el objetivo; cabe señalar que PHP permite realizar la comunicación requerida. La iteración no contempla la comunicación con otros elementos del sistema.

17 La asignación del nombre y la dirección MAC se realiza en modo de diseño, antes de ser compilado el código del MCU y ser programado el dispositivo.

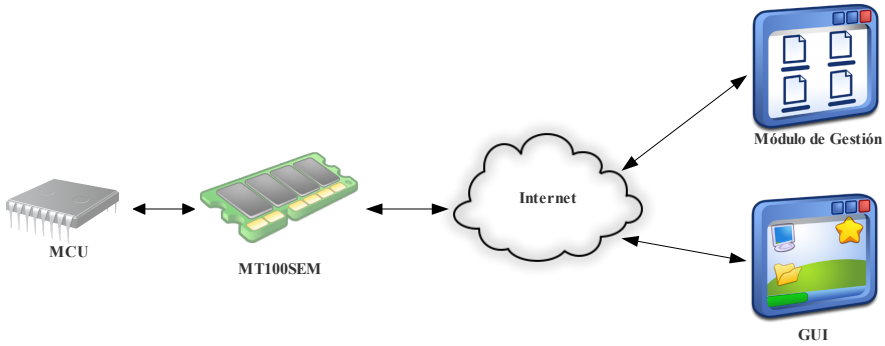


Figura 2.6. Comunicación entre el Módulo de Gestión y la GUI.

En la última iteración se integran todos los elementos restantes, empezando con la comunicación del MCU con el módulo de gestión e ingresando los datos de actualización en la base de datos, finalizando con el acceso desde la interfaz de usuario para interactuar con el MCU (Figura 2.7).

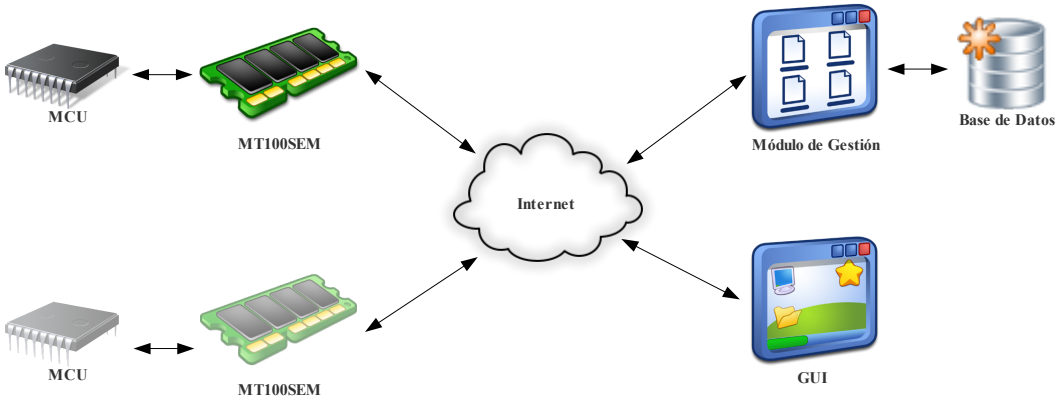


Figura 2.7. Comunicación completa entre componentes.

2.4 Diseño detallado de hardware y software

Para el mejor desarrollo, el diseño del sistema se dividió en dos secciones diferentes, *hardware* y *software* (Tabla 2.5), con lo que se puede separar al sistema en cada uno de sus elementos.

Tabla 2.5. División en el diseño hardware y software.

Hardware	Software
Diseño *hardware* del nodo	Diseño *software* del gestor de nodos
Diseño *software* del nodo	Diseño *software* de GUI

2.4.1 Diseño hardware del nodo

Los dos elementos más importantes en el diseño HW son el dispositivo de comunicación RS232-Ethernet (MT100SEM) y el MCU ATmega8. En el caso del MT100SEM no es posible realizar cambios a la arquitectura, sin embargo tiene la posibilidad de actualizar el *firmware*, proceso que se describe en el Anexo C. Las terminales del MT100SEM se dividen en cinco áreas (Figura 2.8) [29]: comunicación Ethernet, comunicación RS232, voltaje de alimentación, indicadores de estado y *reset*. Las terminales referentes a la comunicación son utilizadas para establecer el control basado en *hand-shaking*[18] y quedan descartadas para este trabajo.

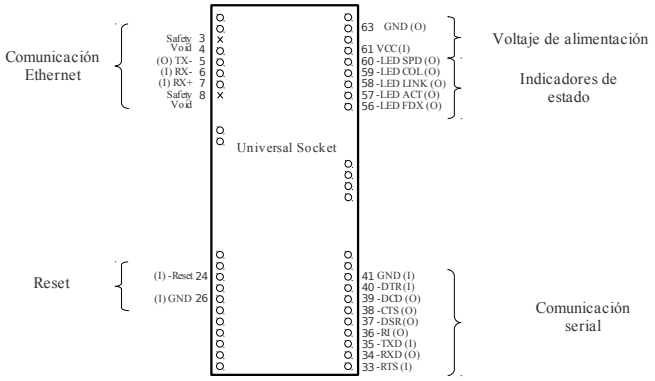


Figura 2.8. Terminales de E/S del dispositivo MT100SEM.

El MCU ATmega8 es el encargado de administrar el nodo, este dispositivo cuenta con 28 terminales (Figura 2.9). Se definen las E/S (cuatro entradas analógicas, cuatro entradas digitales y cuatro salidas digitales) basadas en la especificación del producto (sección 2.1).

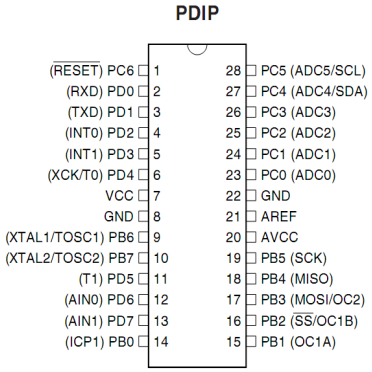


Figura 2.9. MCU ATmega8 de la firma Atmel.

La descripción detallada de la configuración de las terminales del MCU se encuentra en el Anexo A.I.

18 Disponible a partir de la versión *firmware* v1.01c.

2.4.2 Diseño software del nodo

El programa que se desarrolla dentro del MCU se divide en dos partes funcionales:

- Modo inicial (o *reset*): El modo inicial se encarga de la configuración del MCU y del MT100SEM (Figura 2.10a).

- Modo normal de comunicación: En el modo normal se realizan tres acciones continuamente (Figura 2.10b), envío de *ping*, apertura de la conexión TCP/IP y comunicación mediante el protocolo diseñado (sección 2.4.3); en caso de que la respuesta generada en cualquiera de las acciones no sea satisfactoria, se regresa al punto inicial del modo normal para reiniciar las operaciones correspondientes.

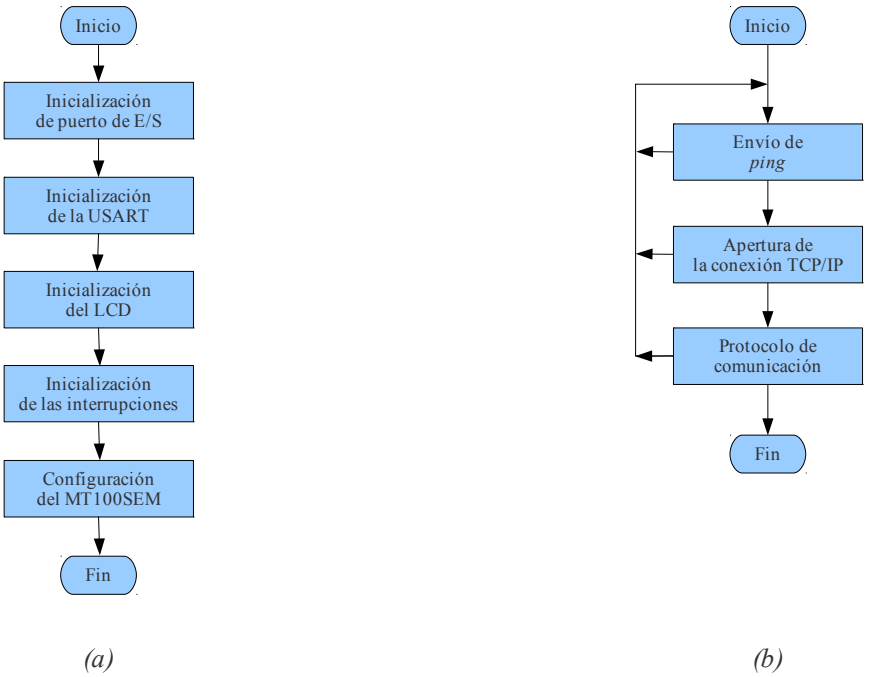


Figura 2.10. Funcionamiento global del MCU.
a. Modo de inicial (o reset); b. Modo normal de operación.

El modo normal de comunicación hace uso de las dos interrupciones externas del MCU para generar nuevas tareas de reconfiguración en el flujo de programa:

- Detectar la presencia de enlace en el dispositivo MT100SEM (*INT0*): Ocurre cuando se conecta y desconecta el cable Ethernet en el nodo. En caso de que el cable se desconecte, el programa entra en modo de espera (no hace nada) hasta que se vuelva a detectar el cambio en el estado del enlace (*link*), en donde el MCU reconfigura al MT100SEM para reiniciar las operaciones de comunicación con el gestor de nodos.

- Detectar *reset* externo del dispositivo MT100SEM (*INT1*): Ocurre cuando se presiona el botón *reset* del MT100SEM, lo cual requiere reconfigurar el dispositivo.

2.4.3 Diseño software del gestor de nodos

El gestor de nodos es una parte importante del sistema, ya que es el encargado de realizar la comunicación entre la GUI y los nodos disponibles. Sus dos tareas principales son:

- Aceptar conexiones TCP/IP (peticiones) y entregar una respuesta en el caso de que la conexión se establezca desde la GUI. Si la conexión se realiza desde algún nodo, éste tendrá que identificarse y registrarse en la lista de nodos activos.

- Establecer el protocolo para la comunicación entre los elementos del sistema.

El diagrama de flujo de la Figura 2.11a muestra la configuración inicial del gestor de nodo y su preparación para aceptar múltiples conexiones simultáneamente (servidor concurrente). Para ello es necesario crear nuevas conexiones y si es posiblemente reutilizar conexiones abandonadas o cerradas, tal y como se muestra en la Figura 2.11b[19].

Una vez que se establece la conexión con el *socket* (usando un *socket* secundario) se introduce el protocolo de comunicación. El diagrama de flujo de la Figura 2.11c muestra una forma sencilla de lograr que la información recibida sea procesada por el protocolo y éste genere una respuesta apropiada a la petición.

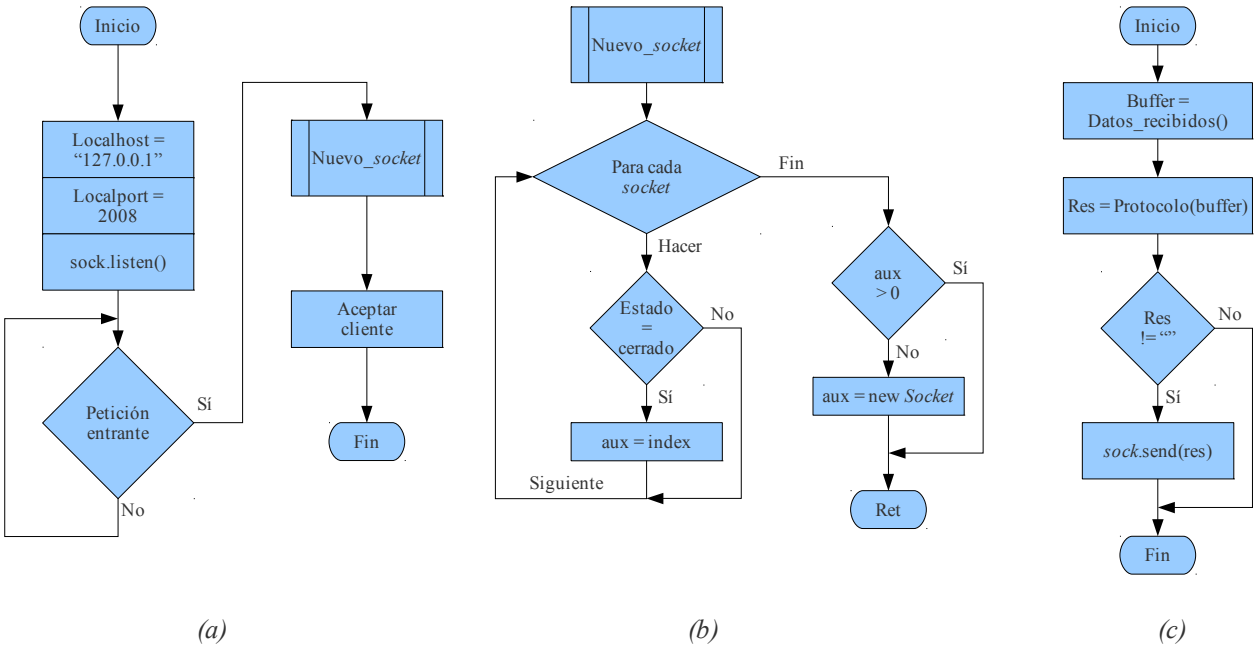


Figura 2.11. Diagrama de flujo del funcionamiento del gestor de nodos.
a. Socket de espera (servidor); b. Creación de un nuevo socket; c. Socket secundario para responder al cliente.

El desarrollo del protocolo de comunicaciones que se implementa se realiza por secciones. Las secciones funcionales del gestor de nodos son:

19 La actividad concurrente es necesaria para que los nodos y la GUI se conecten/comuniquen simultáneamente.

- Identificación del nodo.
- Reporte de mediciones en el nodo.
- Obtención de la lista de nodos.
- Estado de algún nodo.
- Estado del servicio (gestor de nodos).
- Escritura de valor en el nodo desde la GUI.
- Lectura de valor en el nodo desde la GUI.

Las órdenes que se utilizan en el protocolo son triadas de números; los número pares corresponden a la comunicación entre algún nodo y el gestor de nodos, de la misma forma los número impares corresponden a la interacción entre la GUI y el gestor de nodos (Tabla 2.6).

Tabla 2.6. Órdenes y argumentos en el protocolo implementado.

Orden	Argumentos	Descripción
100	MT100 00:00:00	Activa algún nodo registrado en la BD.
101		Verifica la disponibilidad del servicio.
102	$*cmd* $*val*	Reporta lecturas realizas.
103	$*cmd*	Escribe un valor en una terminal del nodo.
104	$*val*	Retorna en valor escrito en la terminal del nodo.
105		Solicita la lista de nodos registrados en la BD.
106	$*val*	Retorna el valor leído en unta terminal del nodo.
107	$*cmd* $*val*	Lectura del valor en una terminal del nodo.
109	MT100 00:00:00	Verificar que el nodo esté activo.

Las secciones que involucran solamente a los nodos y al gestor de nodos son la identificación del nodo y el reporte de mediciones en el nodo, se hace la agrupación de estas dos secciones ya que ambas presentan un funcionamiento similar.

La identificación por parte del nodo ocurre una vez establecida la comunicación TCP/IP con el gestor de nodo, ya que es necesario reconocer la conexión (hilo) donde está el dispositivo. El proceso de identificación se forma de la siguiente manera: `100 MT100 00:00:00`. El número `100` corresponde a la petición identificación; el primer parámetro es el tipo de dispositivo con el que se establece la comunicación[20] y el segundo parámetro indica los tres últimos octetos de la dirección MAC (en formato hexadecimal). Las respuestas posibles para esta orden son las siguientes: `200 OK`, `400 Bad request` y `401 Unauthorized`.

El gestor de nodos es el encargado de determinar si la identificación tuvo éxito, si el número de parámetros es correcto o si el dispositivo no se encuentra registrado. Cabe mencionar que la respuesta `200` es sinónimo de éxito en la operación, en otro caso ocurrió algún tipo de error en la ejecución. En la Figura 2.12a se muestra el desarrollo de esta orden.

La forma de reportar valores, tanto digitales como analógicos (usando el MCU del

20 La identificación del tipo de dispositivo se debe a la posibilidad de que en una versión posterior del sistema la conexión sea realizada con dispositivos de la misma familia, por ejemplo: MT800SWM (Wi-Fi®) o MTS2BTSMI (Bluetooth®).

nodo), es mediante la petición 102 A:B, siendo A el puerto y la terminal donde se lee el valor y B el valor obtenido en la lectura. La Figura 2.12b muestra cómo se realiza esta operación y sus posibles respuestas. Es necesario mencionar que las dos peticiones (100 y 102) son generadas desde el nodo y sirven para entregar información al gestor de nodos.

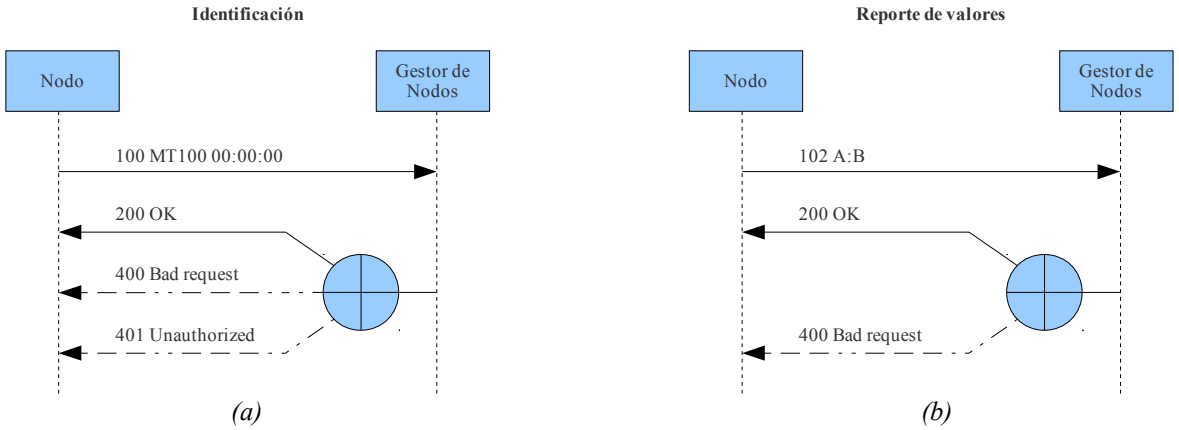


Figura 2.12. Órdenes que involucran al nodo y al gestor de nodos.
a. Identificación del nodo; b. Reporte de mediciones en el nodo

La segunda agrupación de secciones que involucran procesos de comunicación entre el gestor de nodos y la GUI, dichos procesos son: obtención de la lista de nodos, estado del nodo (activo/no activo) y estado del gestor de nodos.

Para la obtención de la lista de nodos (Figura 2.13a) se necesita enviar la petición 105 sin ningún parámetro, teniendo como respuesta satisfactoria el número 200 OK <NodeList>; la forma en que está integrada la lista de nodos es: nombre, dirección MAC, dirección IP, puerto remoto y estado (activo/no activo). La información está separada por el símbolo ';' (punto y coma) con lo que se tiene una sola línea de información, un ejemplo donde se puede apreciar una línea es el siguiente: 200 OK MT100|00:00:01|192.168.2.1|2008|1;MT100|00:00:02|192.168.2.2|2008|1.

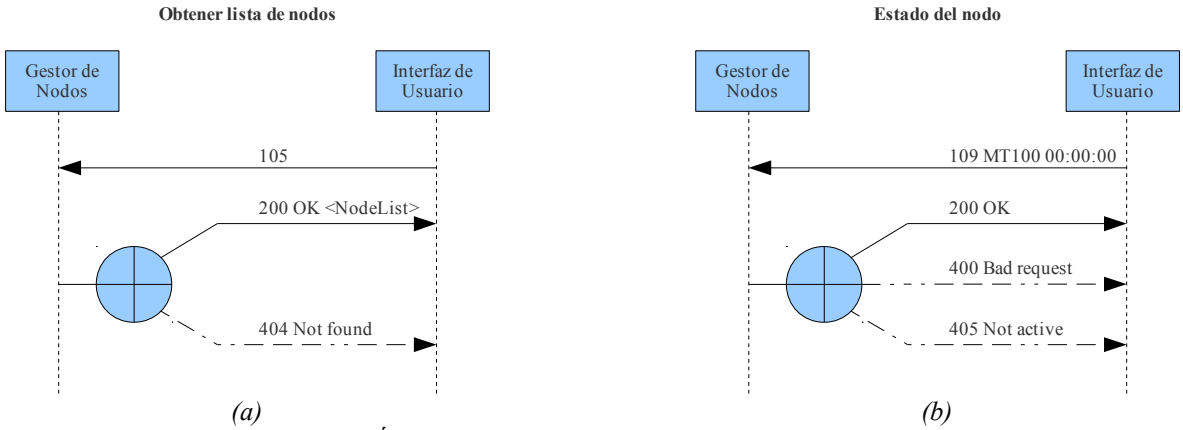


Figura 2.13. Órdenes que involucran al gestor de nodos y a la GUI.
a. Obtención de la lista de nodos registrados; b. Estado (activo/no activo) de algún nodo.

El proceso estado del nodo tiene la finalidad de ayudar a la GUI antes de realizar algún cambio sobre el nodo y determinar sí es posible realizar la operación deseada. La Figura 2.13b muestra cómo se realiza dicha petición, teniendo la sintaxis: `109 MT100 00:00:00`. El número `109` representa la acción de preguntar sobre algún nodo, seguido de dos parámetros, nombre y dirección MAC.

El proceso estado del servicio (Figura 2.14) tiene como principal función determinar sí el servicio (gestor de nodos) está disponible. Únicamente es necesario enviar la petición `101` y esperar la respuesta `200 OK`. En el supuesto de que el gestor de nodos no esté activo, el *socket* tiene un mecanismo de detección de errores en la conexión, en este caso notificará que no se ha logrado establecer comunicación con la dirección IP destino.

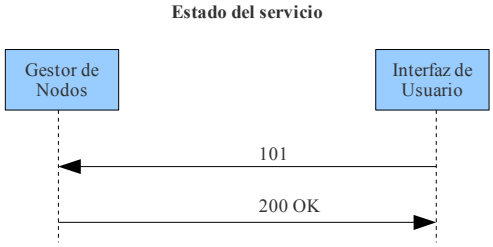


Figura 2.14. Estado del gestor de nodos (activo/no activo).

La última agrupación de secciones se compone por procesos del protocolo en los que intervienen los tres elementos (nodo, gestor de nodos y GUI), en estos procesos se debe tener cuidado con las peticiones, sobre todo las de escritura ya que en el caso de que más de dos usuarios intenten escribir un valor en el mismo nodo (al mismo tiempo o en un periodo de tiempo muy corto) se puede tener un conflicto en la escritura, algunas de las posibles soluciones son:

- Limitar el número de usuarios que pueden hacer uso del servicio (objetivo de este trabajo).
- Realizar retardos de *n*-segundos después de realizar la operación de escritura logrando asegurar que el estado de valor se mantenga por un tiempo.
- Restringir la GUI a usuarios registrados, esto implica tener un proceso de inicio de sesión y que se pueda jerarquizar a los usuarios. Así el usuario con jerarquía superior podrá efectuar cambios mucho antes que los usuarios con jerarquías menores.

El proceso escritura de un valor en el nodo desde la GUI requiere que la petición sea realizada desde la GUI. La petición que hace dicha acción es `103 MT100 00:00:00 <acción>`, el número `103` representa la operación de escritura y necesita tres parámetros para su funcionamiento, nombre, dirección MAC y acción a realizar.

La acción a realizar se representa en forma decimal (0 – 255) y los bits que la representan en su formato binario tienen un significado propio (Tabla 2.7). El parámetro

denominado <acción> en la operación de escritura/lectura es un número codificado, el formato utilizado está compuesto de la siguiente forma: E PPPP XXX.

donde:

- E es el estado de la terminal (1= Encendido, 0 = Apagado).
- P es el puerto donde se quiere realizar la operación.
- X es el número de la terminal donde se quiere realizar la operación.

Tabla 2.7. Lista de acciones para las operaciones de lectura y escritura.

Salidas digitales				
	PB1	PB2	PC4	PC5
Encender	193	194	172	173
Apagar	65	66	44	45
Entradas digitales				
	PD4	PD5	PD6	PD7
Lectura	20	21	22	23
Entradas analógicas				
	PC0	PC1	PC2	PC3
Lectura	168	169	170	171

De esta forma, el nodo puede procesar las peticiones y respuestas de forma rápida; la Tabla 2.8 muestra las acciones base para cada puerto. En la operación de lectura el bit más significativo 'E' se ignora.

Tabla 2.8. Direcciones de lectura/escritura en los dispositivos E/S del nodo.

Puerto	Dirección	Descripción
Puerto B	192 - 199	Operación con el PB0, .., PB7. Encendido.
	64 - 71	Operación con el PB0, .., PB7. Apagado.
Puerto C	168 - 175	Operación con el PC0, .., PC7. Encendido.
	40 - 47	Operación con el PC0, .., PC7. Apagado.
Puerto D	144 - 151	Operación con el PD0, .., PD7. Encendido.
	16 - 23	Operación con el PD0, .., PD7. Apagado.
ADC	208 - 214	Operación con el ADC0, .., ADC5.

La función del gestor de nodos es determinar si la acción es válida para el nodo actual, pudiendo eliminar errores en la ejecución del bucle principal del nodo. La Tabla 2.9 muestra cómo está compuesta la acción, en este caso el encendido/apagado de cada terminal del puerto B correspondiente al MCU.

La petición de escritura se genera desde la GUI, la petición pasa por el gestor de nodos, quien se encarga de determinar el estado del nodo (activo/no activo); en caso de que el nodo esté activo se reenvía la información necesaria al nodo, este último responde con el número 104 y el valor presente en el puerto después de la escritura, con esto el gestor de nodos puede generar una respuesta satisfactoria dirigida a la GUI (Figura 2.15).

Tabla 2.9. Encapsulamiento de algunas acciones con el puerto B.

Decimal	Hex.	Binario	Descripción
192	C0	1 1000 000	Encendido del puerto B, terminal 0.
193	C1	1 1000 001	Encendido del puerto B, terminal 1.
64	40	0 1000 000	Apagando del puerto B, terminal 0.
65	41	0 1000 001	Apagando del puerto B, terminal 1.
71	47	0 1000 111	Apagando del puerto B, terminal 7.

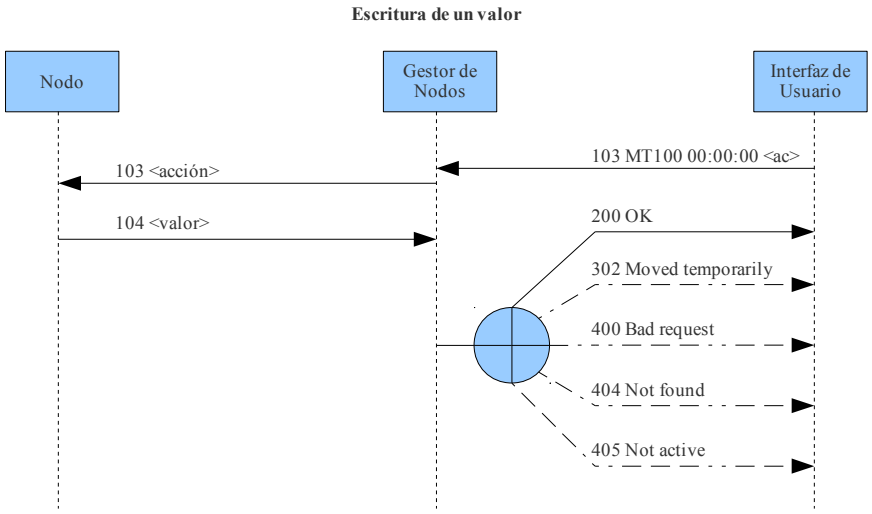


Figura 2.15. Orden de escritura proveniente de la GUI hacia el nodo.

El proceso de lectura de un valor en el nodo (Figura 2.16) es similar a la escritura, con la diferencia de que el valor devuelto por el nodo es únicamente el valor de la terminal (64 si no está activo ó 128 si está activo); en caso de lectura de los ADCs se utiliza todo el byte.

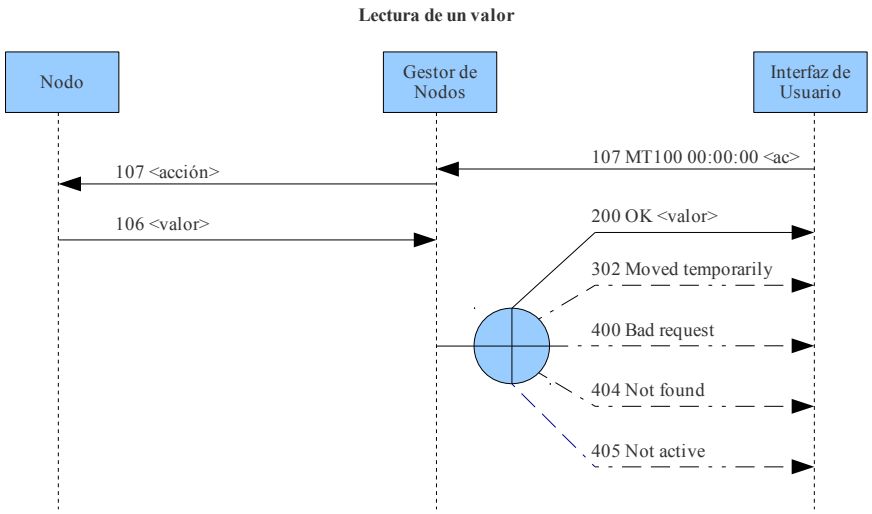


Figura 2.16. Orden de lectura desde la GUI y la respuesta generada.

2.4.4 Diseño software de la GUI

La GUI es el último elemento del sistema y es el único con el que el usuario final puede interactuar. La Tabla 2.10 muestra los elementos que contiene la GUI.

El lenguaje de programación seleccionado es PHP (sección 1.3.3), ya que permite ser combinado con HTML, de forma tal que el contenido de la GUI se genera por código de PHP y se muestra al usuario como HTML.

Tabla 2.10. Elementos de la GUI.

Sección	Descripción
Inicio	Esta sección será la primera página que verá el usuario final y contendrá información referente al sistema desarrollado.
Dispositivo	La sección de dispositivos mostrará una lista de los dispositivos registrados con una breve descripción y el estado del dispositivo (activo/no activo).
Reporte	En esta sección se podrá visualizar la información de los eventos ocurridos en un periodo de tiempo.

2.4.4.1 Inicio

La pantalla de Inicio proporciona información del desarrollo del proyecto. La codificación de la información se realiza mediante HTML y CSS para mantener una formato uniforme en toda la GUI, además de permitir que una presentación se adecue a distintos dispositivos.

2.4.4.2 Dispositivos

La sección Dispositivos contiene una lista de los dispositivos registrados en la BD. Lo prioritario en esta sección es determinar sí el nodo está disponible (conexión TCP/IP establecida con el gestor de nodos) y que el gestor de nodos se encuentre funcionando.

Para detectar errores en la conexión, PHP proporciona una función llamada *header* que se utiliza para verificar errores de lectura de código. En este caso proporciona la capacidad de verificar el estado del gestor de nodos y evitar errores posteriores. La Figura 2.17 muestra el diagrama de flujo que describe el comportamiento de esta sección y las tareas que se realizan son:

- Verificar el estado del gestor de nodos.
- Solicitar la lista de usuarios al gestor de nodos.
- Crear una tabla que contenga cada uno de los elementos registrados.
- Esperar la selección de algún nodo.
- Después de la selección, verificar que el nodo esté activo.
- Mostrar información del nodo.
- Esperar algún cambio en el estado de los dispositivos de E/S.
- Notificar al gestor de nodos.

La lista mostrada en esta sección contiene los nombres de los nodos registrados y su estado (activo/no activo), permitiendo interactuar solo con los nodos activos.

Como condición adicional se agrega un *refresh* de la página cada 20 segundos para determinar el estado del gestor de nodos[21]. En el caso que dicho servicio no esté disponible, se redireccionará a una página de error indicando el problema ocurrido.

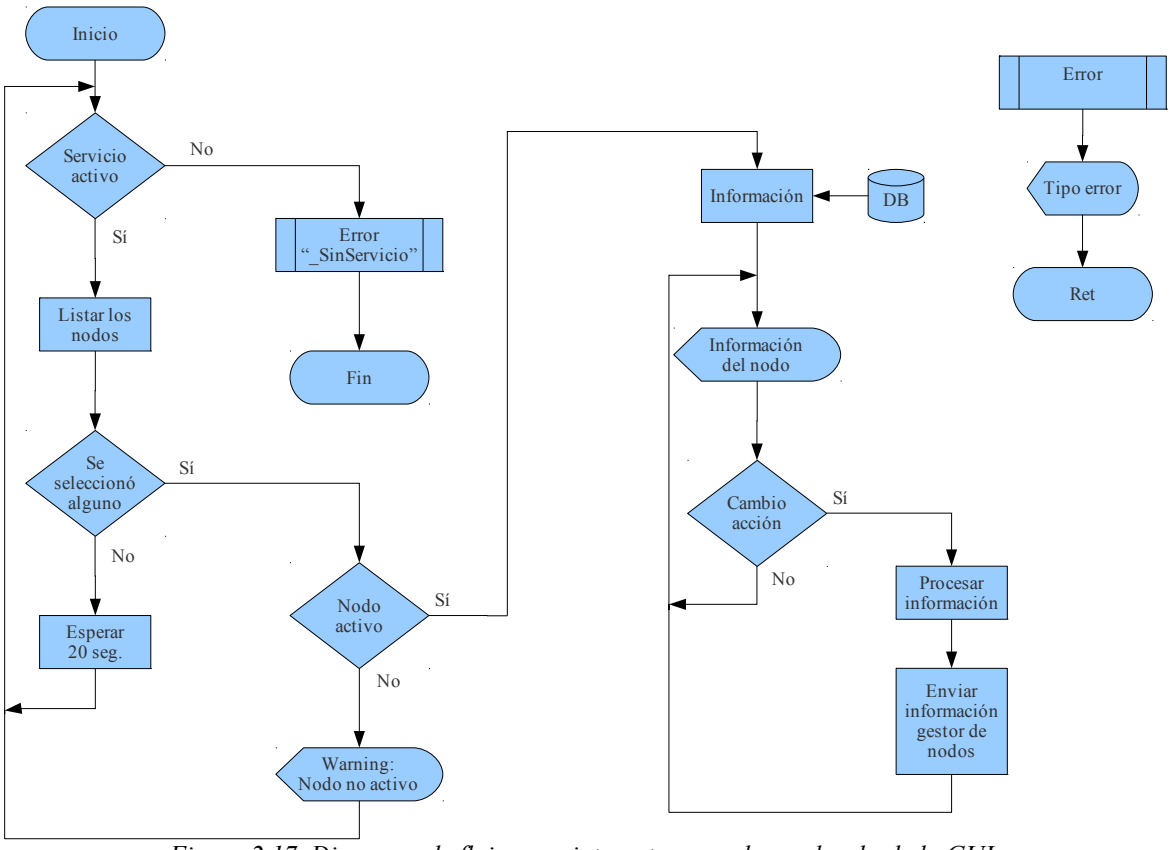


Figura 2.17. Diagrama de flujo para interactuar con los nodos desde la GUI.

2.4.4.3 Reporte

La sección de reporte muestra información correspondiente a los eventos ocurridos durante un periodo de tiempo, pudiendo filtrar la información de la siguiente manera:

- Selección de un nodo registrado.
- Por la fecha actual (predeterminado).
- En una fecha específica.

Una característica importante del gestor de BD es el uso de funciones de filtrado, esto permite tomar registros de alguna tabla y seleccionar solo los registros que cumplan con un criterio de búsqueda. La Figura 2.18 muestra el diagrama de cómo se realiza el reporte de las actividades de los nodos.

21 Esto se debe a las limitaciones propias de interfaces basadas en HTML y PHP, ya que no pueden recibir eventos de notificación provenientes del servidor.

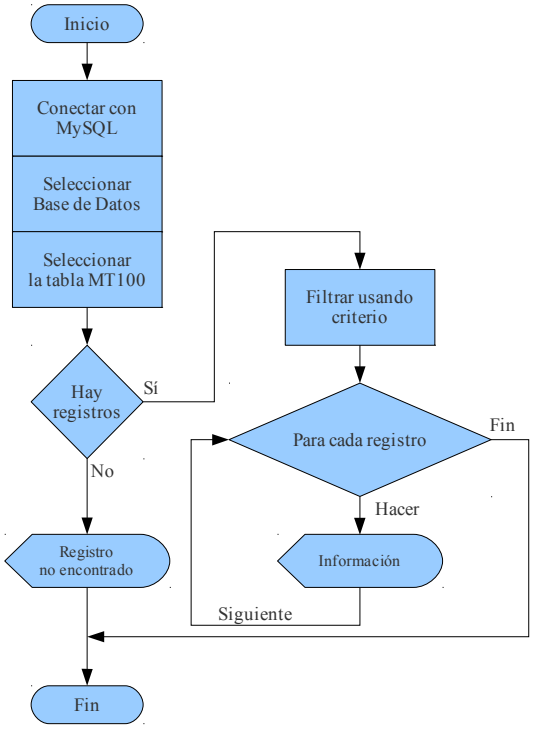


Figura 2.18. Diagrama de flujo para realizar un reporte sencillo.

2.5 Integración de componentes Hardware y Software

La fase de integración de componentes HW y SW consiste en anexar cada uno de los elementos del sistema: nodo, gestor de nodos e GUI. La Figura 2.19 muestra cómo está integrado el sistema de forma global, teniendo a Internet como punto central de la comunicación.

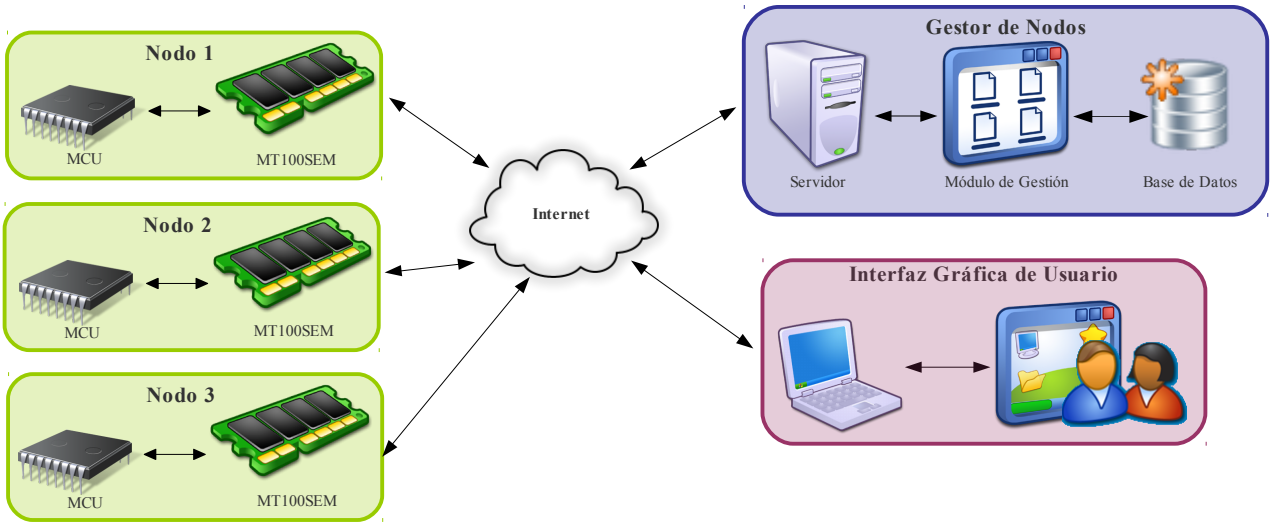


Figura 2.19. Integración de componentes del sistema.

2.6 Verificación del producto

Para que la verificación del producto se pueda realizar con claridad, se divide en base a las capas del modelo de sistemas empotrados (sección Metodología de desarrollo, Figura 3).

2.6.1 Verificación del hardware

Las pruebas de verificación de *hardware* se enlistan a continuación (Figura 2.20a):

- Verificación detallada de las conexiones: La verificación de las conexiones se llevó a cabo mediante un multímetro para comprobar la continuidad en las pistas del PCB.

- Medición de voltaje y corriente: El voltaje de alimentación es de 5 V y tiene un consumo promedio de 215 mA en el nodo.

- Control de transmisión/recepción RS232 (sincronización): En la comprobación de la comunicación no fue necesario el uso de un osciloscopio, el nodo cuenta con un LCD que muestra el estado de la comunicación, esto permite determinar el estado del proceso actual en la comunicación. Además, el nodo cuenta con un CI max232 que permite establecer comunicación RS232 con una PC, pudiendo establecer un modo de depuración en la comunicación. Cabe mencionar que el uso del max232 es para la configuración directa del MT100SEM vía RS232.

- Verificación de inicialización: El LCD muestra información referente al modo de inicialización, desplegando la siguiente información: nombre y dirección IP del nodo (Figura 2.21a), estado del cable Ethernet y la configuración de red para el MT100SEM (Figura 2.21b).

- Verificación de la operación normal: En el LCD se muestra la petición y respuesta de lectura/escritura para el nodo. Se cuenta con *leds* indicadores que indican el estado actual de la terminal[22].

Se realizaron dos pruebas más para verificar el comportamiento del nodo en dos casos:

- Establecimiento/pérdida de la conexión física Ethernet (Figura 2.20b): Cuando el nodo pierde la comunicación Ethernet (cable desconectado), el flujo del programa principal es afectado, se detiene el proceso de comunicación y el LCD muestra un mensaje de error (Figura 2.21c). Cuando se restablece la comunicación (cable nuevamente conectado), el LCD muestra un mensaje de recuperación de la comunicación y reinicia la operación de configuración y establecimiento de la comunicación (Figura 2.21d).

22 Las terminales de E/S digitales son activas en bajo.

- Detección de *reset* externo al MT100SEM (Figura 2.20c): Este evento ocurre cuando se presiona el *reset* externo y es necesario reiniciar la operación de configuración del dispositivo[23] (Figura 2.21a y Figura 2.21b).

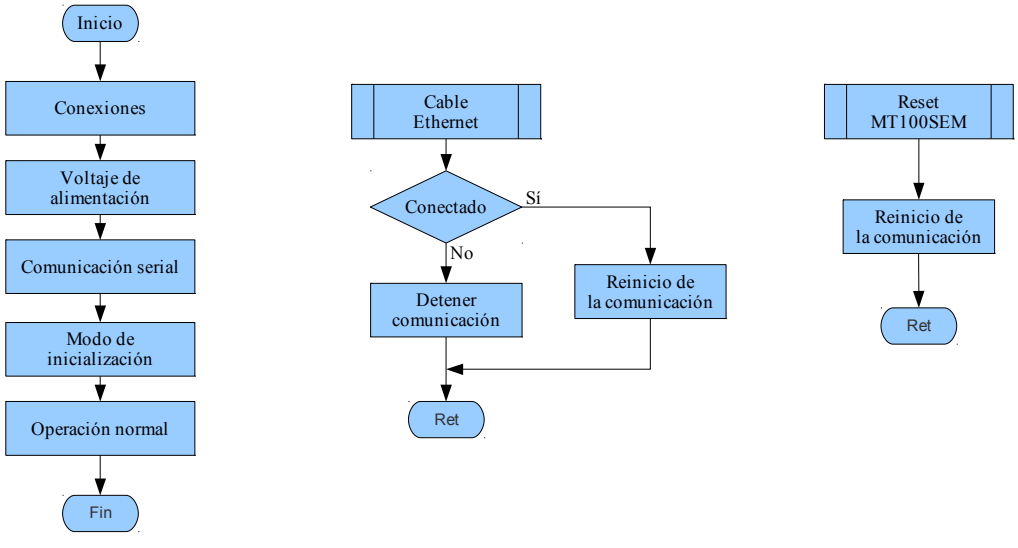


Figura 2.20. Pruebas de verificación del hardware.
a. Verificación del funcionamiento global del nodo; b. Verificación de la conexión; c. Verificación del reset.

Figura 2.21. Mensajes de notificación en el nodo.
a. Presentación de nombre del nodo; b. Envío de orden IPADDR; c. Mensaje mostrado cuanto se pierde la conexión Ethernet; d. Mensaje mostrado al restablecer la conexión.

23 El dispositivo MT100SEM permite guardar en su memoria de trabajo (orden AT&W) los valores de configuración predeterminados, se omite esta característica en este trabajo.

2.6.2 Verificación del software del sistema

Las pruebas de verificación del software del sistema se enlistan a continuación (Figura 2.22):

- Verificación del estado del gestor de BD.
- Recepción de peticiones: Las peticiones contempladas en el protocolo de comunicación son los números 100, 101, 102, 103, 104, 105, 106, 107 y 109.
- Verificación de almacenamiento de las acciones en la BD.

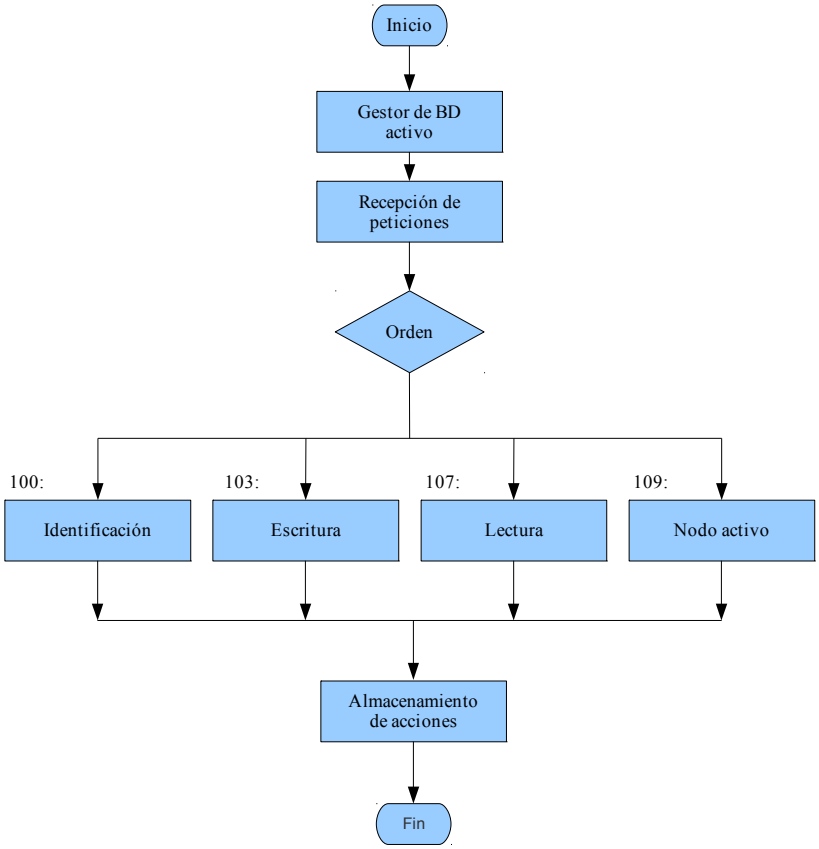


Figura 2.22. Pruebas de verificación del software del sistema.

Realizar pruebas para cada una de las órdenes contempladas en el protocolo de comunicación resulta repetitivo, por esa razón solo se muestra el comportamiento de la orden 103, la cual requiere comunicación de los tres elementos más importantes del sistema (nodo, gestor de nodos y GUI).

La prueba de verificación se llevó a cabo usando el *software* Wireshark[24], el cual es un software para analizar protocolos de redes y permite dar seguimiento a un paquete Ethernet

24 Wireshark es *software* libre bajo licencia GNU GLP v2.

(trama) dentro de una red local. La prueba consistió en realizar una escritura desde la GUI y observar los paquetes generados en la red local, pasando por el gestor de nodos, el reenvío del paquete hacia el nodo y la respuesta obtenida (Figura 2.23).

Al realizar esta prueba surge una desincronización en la comunicación ya que el protocolo está basado en petición/respuesta. La desincronización ocurre cuando las peticiones dirigidas a un nodo son enviadas rápidamente (a nivel programación), la pila TCP/IP une las peticiones y las envía en la misma trama lo cual provoca pérdidas de algunas peticiones. Las posibles soluciones a la falta de sincronización son las siguientes:

- Realizar un *parser* en el nodo para que verifique más de una petición contenida en los datos recibidos.
- Que el gestor de nodos genere un *flush* en la pila TCP/IP para asegurar la segmentación de la trama.
- Insertar retardos en la GUI (aproximadamente de 200 ms) para asegurar la separación de las tramas.

La solución más viable es insertar retardos en la GUI.

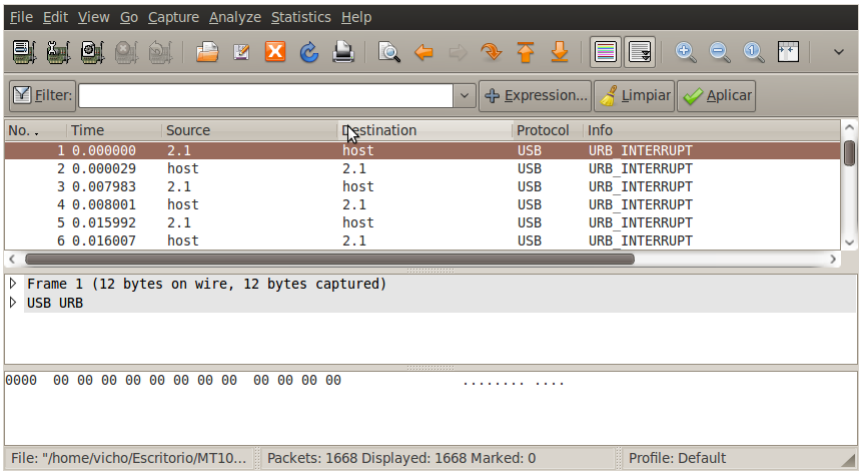


Figura 2.23. Captura de tramas en la comunicación del sistema.

2.6.3 Verificación del software de aplicación

A continuación se enlistan las pruebas realizadas para la verificación de la GUI (Figura 2.24):

- Verificar que el servicio HTTP esté activo: Se logra verificando que el servidor que proporciona el servicio tenga abierto el puerto 80.

- Verificar que el gestor de nodos esté presente: El servicio que proporciona el gestor de nodos es primordial en el funcionamiento del sistema. El sistema no puede proporcionar información a la GUI si este servicio no está activo. La GUI muestra un mensaje de error cuando el gestor de nodos no está activo (Figura 2.25).

- Verificar que el nodo esté activo: El gestor de nodos proporciona la información necesaria sobre algún nodo, esto permite notificar si se ha perdido la comunicación con el nodo actual y muestra un mensaje de advertencia notificando que la última acción realizada puede que no se haya realizado.

- Pruebas de lectura/escritura en las terminales de E/S del nodo.

- Verificación de reporte de las actividades de los nodos[25].

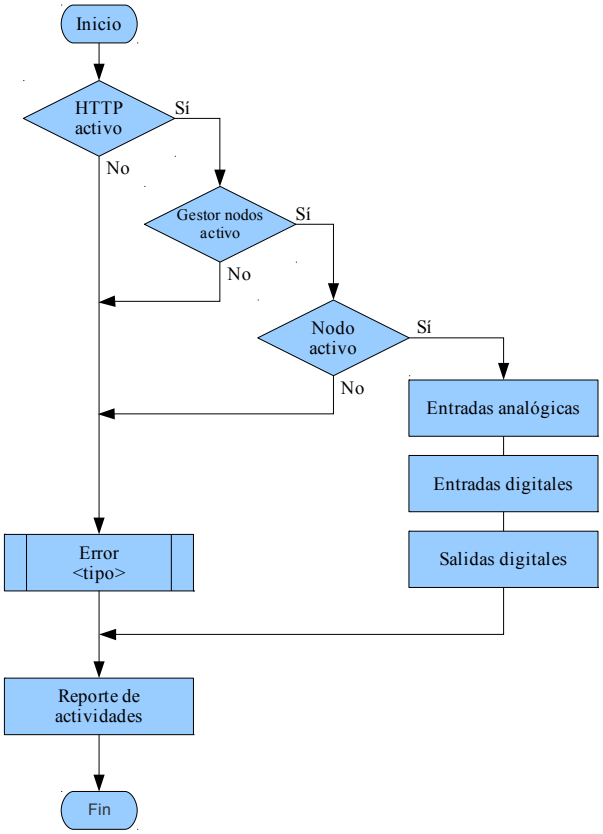


Figura 2.24. Pruebas de verificación del software de aplicación.

Si el servicio HTTP no está disponible, el navegador web realiza la notificación al usuario, indicando que no se encontró el servicio.

25 Esta verificación se realiza en conjunto con el *software* de aplicación.

Figura 2.25. Mensaje mostrado cuando el gestor de nodos no está activo.

2.6.4 Verificación del sistema final

En la verificación final del sistema se realizó la lectura de una medición analógica en el nodo (Figura 2.26). La condición de la prueba analógica es conocida, un potenciómetro usado como nivel de referencia (1.5 V) mostrado en el lector del multímetro. El proceso de comunicación inicia con la petición de lectura desde la GUI (orden `107 MT100SEM D2:81:25 168`), es codificada y reenviada al nodo por el *software* del sistema; como respuesta del nodo se recibe la orden `106 L` (la letra 'L' corresponde al valor 76 en el código *ascii*) y se vuelve a reenviar a la GUI como respuesta de lectura. En la GUI se muestra el valor decimal de lectura obtenida en el nodo, la conversión de número decimal a voltaje se realiza con la ecuación siguiente: (*valor*) * *5 / 255*, dando como valor 1.5098 V.

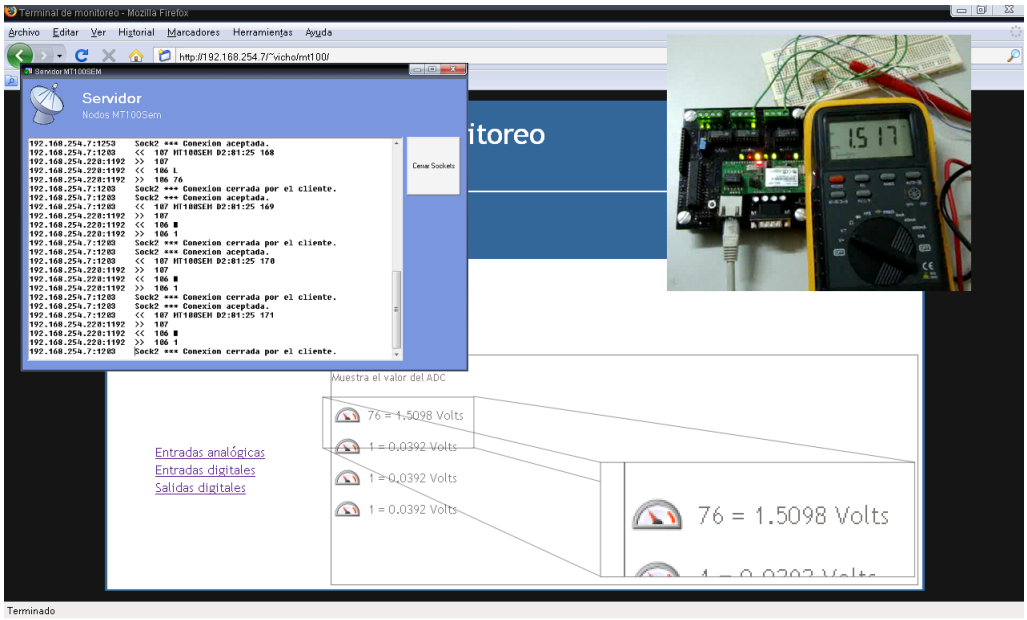


Figura 2.26. Lectura de un valor analógico en el nodo y resultado en la GUI.

La verificación de la lectura de las entradas digitales se deja a un lado, ya que el proceso de lectura es simular a las entradas analógicas, con la diferencia que solo se leen los valores 128 y 64.

En la verificación de escritura (Figura 2.27), se envía la orden 103 MT100SEM D2:81:25 193 desde la GUI para activar una salida en el nodo. La petición es atendida por el *software* del sistema y reenviada al nodo; la respuesta contiene el estado actual de la terminal del nodo (activo o no activo) y es entregada a la GUI para su representación. Debido a que los valores de escritura son binarios, se tiene menos problemas en la representación del estado de la terminal.

Figura 2.27. Activación de las salidas digitales.

La verificación final da como resultado el correcto funcionamiento de cada uno de los elementos que componen al sistema. La verificación individual de cada elemento permite depurar los posibles errores en el proceso de comunicación que se realizan entre los elementos finales del sistema (nodo y la GUI).

Capítulo 3
Resultados

3.1 Producto final

La Figura 3.1 muestra el nivel más alto de abstracción del sistema desarrollado en este trabajo, dicha abstracción sirve para hacer la división de los elementos basados en el modelo de sistemas empotrados.

- La capa de *hardware* contiene los siguientes elementos: Dispositivo MT100SEM, cuya función principal es realizar la comunicación TCP/IP; MCU Atmega8, encargado de administrar y configurar al nodo (vía RS232), permite detectar los cambios en el estado del enlace de la comunicación (*link*), generar y detectar un *reset* en el MT100SEM; elementos de protección en las terminales del MCU basado en un OPAMP (*Operational Amplifier*) y los elementos de entrada y salida del sistema.

- La capa de software del sistema contiene los siguientes elementos: Gestor de nodos, permite concentrar la información generada en el sistema, pudiendo determinar el estado de cada uno de los nodos conectados al sistema, reinterpretar las órdenes provenientes de la GUI y entregar una respuesta; y el gestor de base de datos MySQL, encargado de establecer comunicación con la BD para leer/escribir información en las distintas tablas.

- La GUI contiene los siguientes elementos: Un identificador de usuario (sesión), que evita que usuarios ajenos al sistema ingresen a la GUI, la información de los usuarios aceptados se encuentra contenida la base de datos. La GUI cuenta con una cabecera de título, información del usuario activo, un menú con 3 opciones (Inicio, Dispositivos y Resumen) y una área de trabajo. La función primordial de la GUI es la comunicación con los nodos, en la opción Dispositivos se despliega una lista de los nodos registrados y su estado, se pueden seleccionar e interactuar solo con los nodos activos.

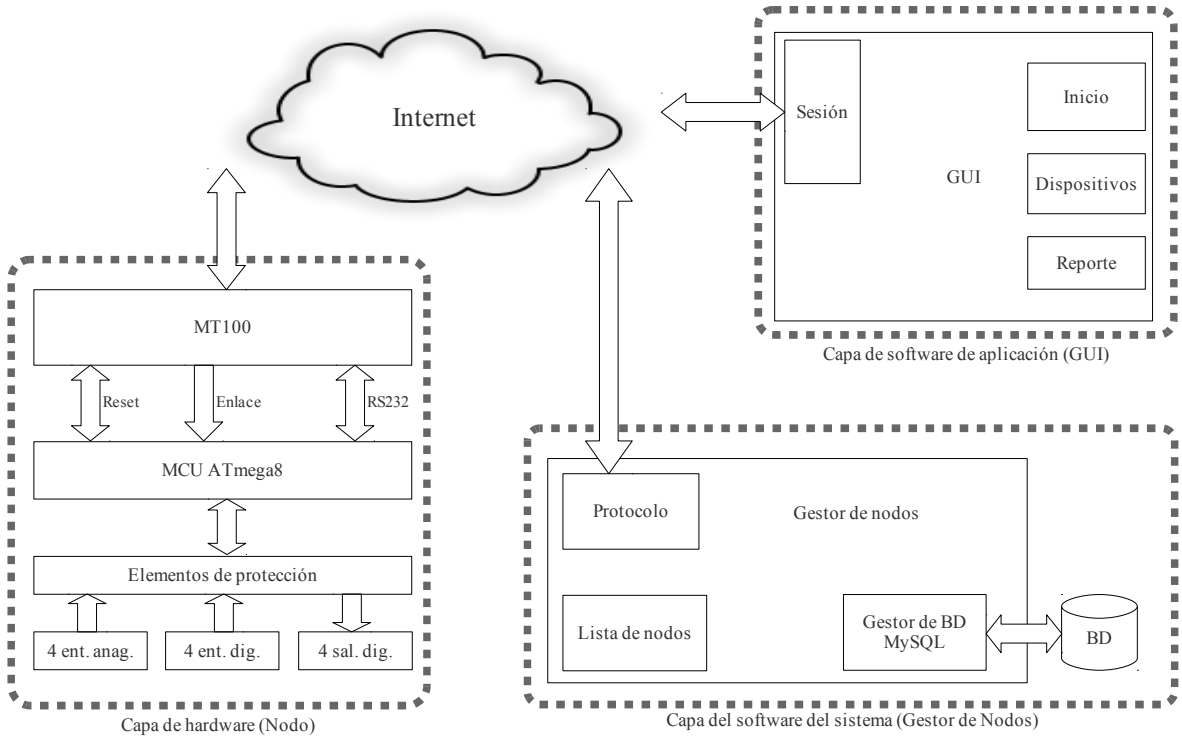


Figura 3.1. Diagrama a bloques del sistema realizado.

3.1.1 Nodo

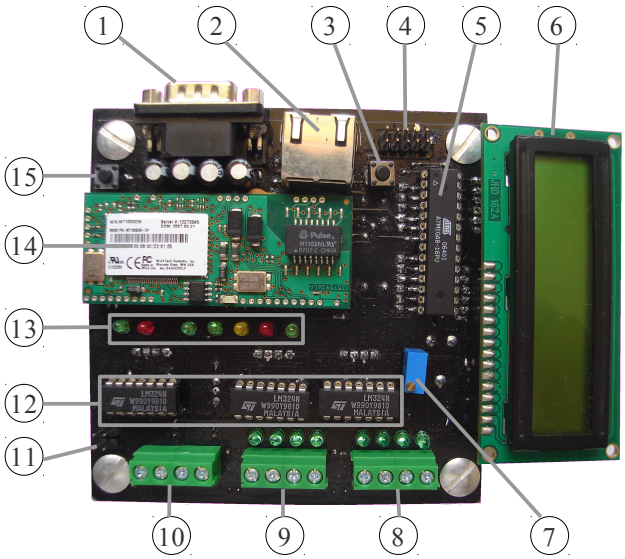
El *hardware* del nodo final contiene los siguientes elementos (Figura 3.2): dispositivo MT100SEM, MCU ATmega8, conector RJ45 hembra, conector RS232 macho, terminales de programación SPI, visualizador LCD, indicador de encendido, indicador de *reset*, indicadores de estado del MT100SEM, *reset* MT100SEM, *reset* MCU, elementos de protección (OPAMP LM324) e indicadores de estado E/S digitales.

Los detalles del diagrama esquemático del nodo se muestra en el Anexo B.I, en donde se tiene la configuración de las conexiones entre el MT100SEM, el MCU ATmega8 y los demás elementos. Los elementos de protección para las terminales del MCU se consideraron como elementos extras y se describen en el Anexo B.II.

El diagrama de la posición de los elementos en el nodo (Anexo B.III) proporciona un mapa de cómo están montados los elementos dentro de la PBC. La superficie de cobre superior e inferior del PCB se muestran en el Anexo B.IV y Anexo B.V respectivamente.

Uno de los objetivos secundarios de este trabajo propone utilizar ocho nodos simultáneamente, pero se cuentan con solo dos dispositivos MT100SEM, por este motivo se emularon los 6 dispositivos restantes. La Figura 3.3 muestra la interfaz de aplicación desarrollada para este trabajo que emula a los nodos; teniendo en cuenta que los nodos hacen

uso de TCP/IP se pueden sustituir la conexión por una aplicación *software*. Los nodos emulados siguen el protocolo de comunicación diseñado y dentro del sistema no se diferencian con los reales.



1. Conector DB9 macho.
2. Conector RJ45.
3. *Reset* MCU.
4. Programador SPI MCU.
5. MCU ATmega8.
6. LCD AND491.
7. Ajuste de comparadores.
8. Salidas digitales.
9. Entradas digitales.
10. Entradas analógicas.
11. Voltaje de alimentación.
12. Comparadores LM324.
13. Indicadores MT100SEM.
14. MT100SEM.
15. *Reset* MT100SEM.

Figura 3.2. Aspecto final del nodo.

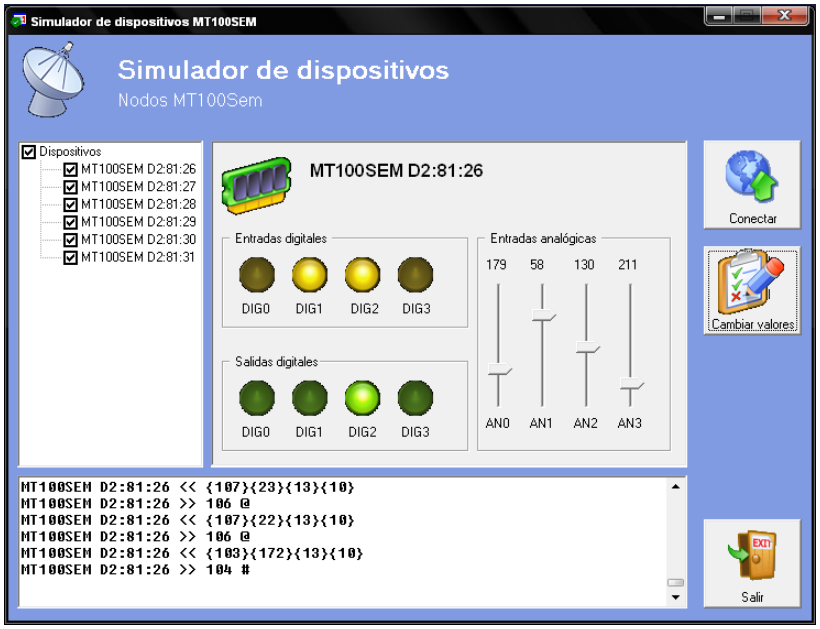


Figura 3.3. Emulador de nodos.

3.1.2 Gestor de nodos

El gestor de nodos tiene una interfaz gráfica simple, cuenta con una caja de texto (*textbox*) en donde se muestran las direcciones IP, el puerto de comunicación, la dirección del mensaje y la orden usada en las acciones del sistema; en algunos casos se muestra el cambio en el estado de la conexión de cada *socket* (Figura 3.4).

El gestor de nodos también se encarga de establecer la comunicación con la BD y almacena la información requerida en una tabla. El almacenamiento de la información permite que la GUI de usuario final muestre información generada en el sistema en un tiempo anterior.

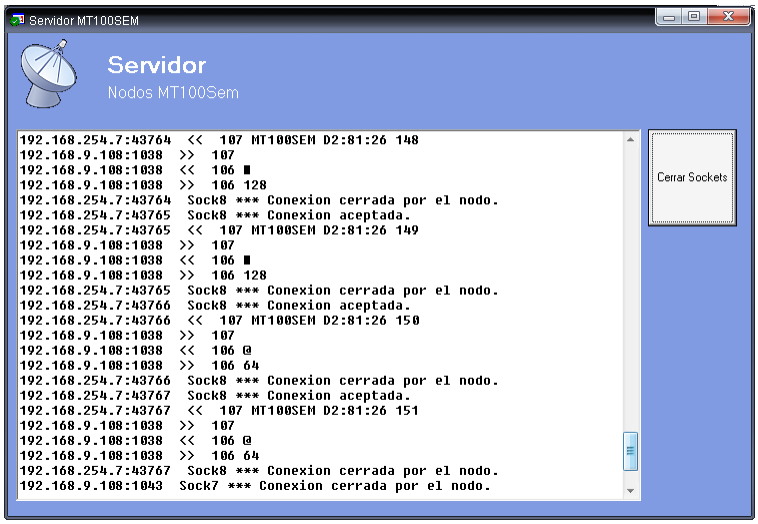


Figura 3.4. Apariencia del gestor de nodos (mensajes de notificación).

3.1.3 Interfaz de usuario

Como elemento de seguridad, en el acceso a la GUI se incorpora un inicio de sesión, siendo necesario ingresar el nombre de usuario y la contraseña para tener acceso al sistema. Algunos navegadores web tienen la capacidad de generar la ventana de identificación (Figura 3.5).

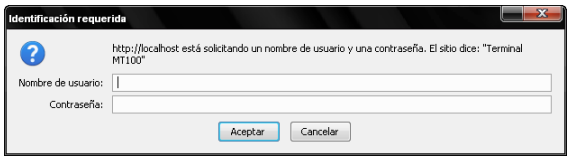


Figura 3.5. Inicio de sesión en de la GUI.

Como resultado de los tres elementos contenidos en la GUI, se tiene la pantalla de Inicio (Figura 3.10a) que contiene la información referente a este trabajo.

En el menú de Dispositivos (Figura 3.6), se puede seleccionar algún elemento activo (ingresar al nodo) si el icono se muestra afirmativamente (palomita); una vez que se tiene acceso al nodo, se puede elegir entre los diferentes elementos de E/S del nodo (entradas analógicas, entradas digitales y salidas digitales). Las entradas analógicas y digitales son solamente informativas, los únicos elementos con los que se pueden interactuar son con las salidas digitales (Figura 3.7).

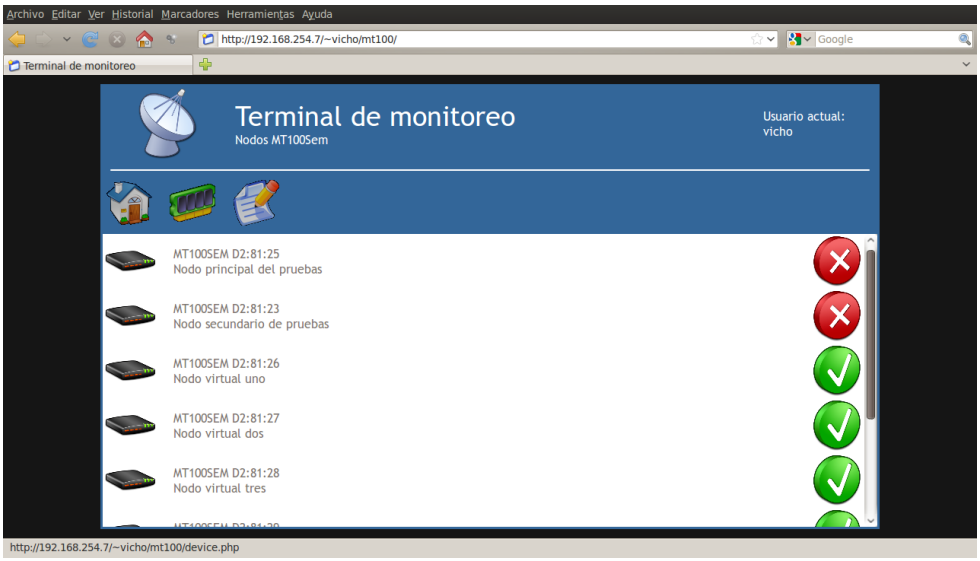


Figura 3.6. Pantalla para seleccionar un nodo.

Figura 3.7. Interacción con los elemento de E/S del nodo.

Accediendo al menú Reporte (Figura 3.8) se muestra la información contenida en la BD, el filtro predeterminado que utiliza es: mostrar todos los eventos ocurrido en la última hora. Para realizar otro criterio de filtrado, se pueden seleccionar los eventos ocurrido en el día o la semana actual; también es posible elegir un filtro que dependa de algún nodo o algún usuario en particular. Los filtros se pueden combinar para tener distinta información de filtrado.

La información obtenida de la BD y presentada en el área de información es: la fecha y hora en que ocurrió el evento, el usuario o dirección MAC donde surgió la petición, la dirección IP remota y la descripción del evento.

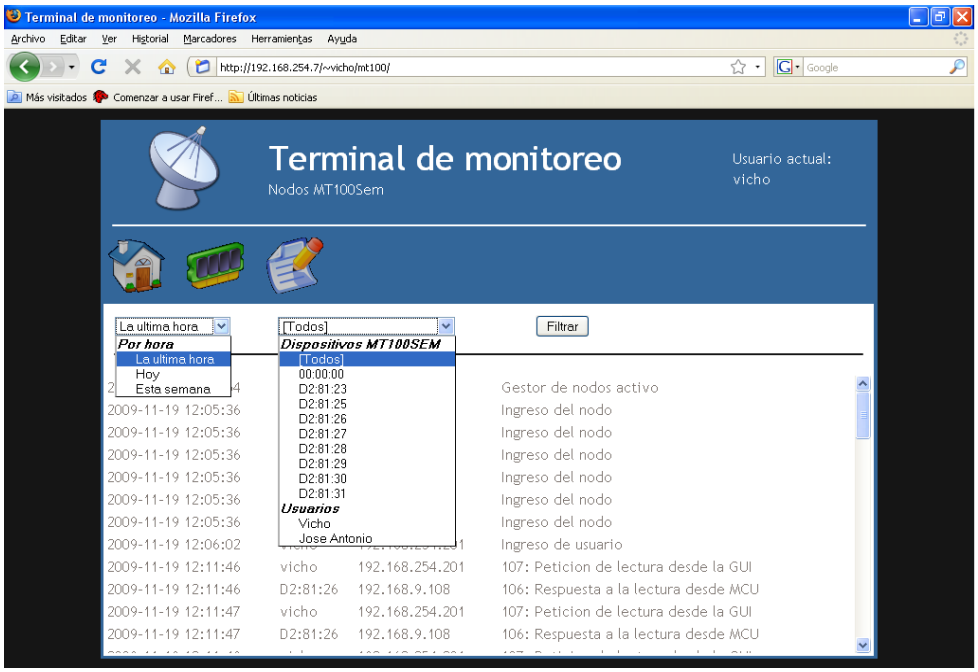
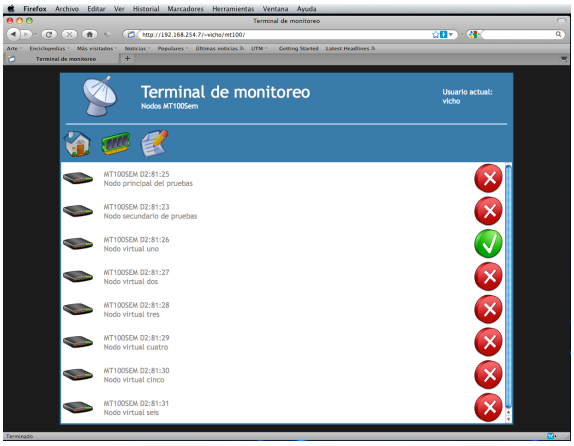
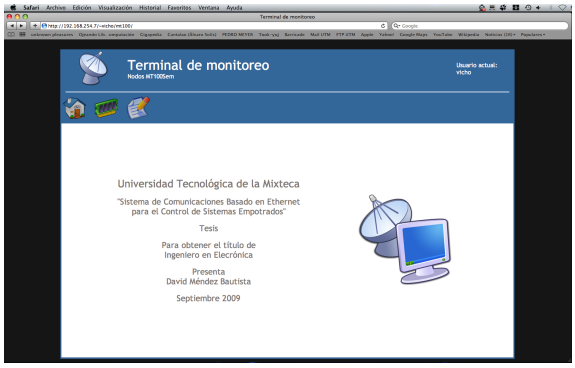


Figura 3.8. Criterios de filtrado en la BD.

La GUI desarrollada presenta portabilidad para distintos sistemas operativos: Macintosh (Figura 3.9), Linux (Figura 3.10) y Windows (Figura 3.11). La portabilidad es proporcionada por navegadores web como: Chrome, Mozilla Firefox y Safari. La W3C establece las reglas de funcionamiento de CSS, los navegadores mencionados anteriormente tienen compatibilidad con CSS por lo que la información es presentada de manera similar en los navegadores. En el caso del navegador Konqueror (Linux) no soporta completamente los elementos de CSS usados en la GUI, ya que se observa una variante en el formato de la presentación de la información sin que esto llegue a interferir con el proceso de comunicación con el gestor de nodos. El navegador Internet Explorer no es compatible con CSS y aunque la información es presentada en la pantalla, esta información no se posiciona en el lugar correcto, de igual manera que el navegador Koqueror, la presentación no interfiere en el proceso de comunicación con el gestor de nodos.



(a)



(b)

Figura 3.9. Aspecto final de la GUI funcionando en el SO Macintosh.
a. Navegador Mozilla Firefox; b. Navegador Safari.

(a)

(b)

Figura 3.10. Aspecto final de la GUI en el SO Linux.
a. Navegador Chrome; b. Navegador Firefox.

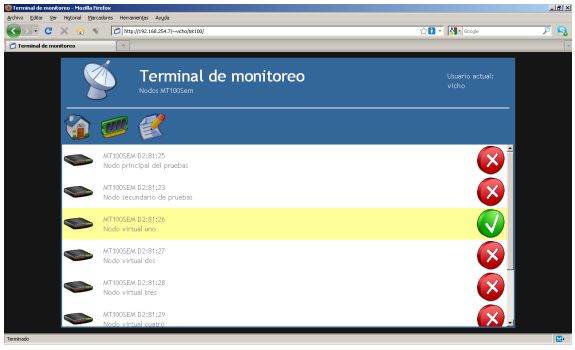


Figura 3.11. Aspecto final de la GUI en Windows (Firefox).

Capítulo 4
Conclusiones y líneas futuras de investigación

El presente trabajo de tesis se desarrolla en el ámbito del monitoreo remoto y se establece un sistema de comunicaciones basado en Ethernet para controlar dispositivos basados en MCU.

Para lograr los objetivos de este trabajo se aplicó la metodología de Sistemas Empotrados [8], siendo una de las formas que describen la interrelación entre *hardware* y *software*. Su implementación se basa en el dispositivo MT100SEM de la firma Multitech y en el MCU ATmega8 de la firma Atmel para la parte *hardware* y principalmente el lenguaje de programación PHP para la parte *software*.

Al finalizar el sistema de comunicaciones, se ha cumplido con los objetivos establecidos en el inicio de este trabajo, obteniendo las siguientes características:

- El sistema cuenta con nodos que tienen cuatro entradas analógicas, cuatro entradas digitales y cuatro salidas digitales con indicadores de estado para E/S digitales (sección 3.1.1, Figura 3.2).
- La comunicación RS232 entre el MCU y el dispositivo MT100SEM se estableció a una velocidad de 9.6 kbps, logrando obtener un 0.2% de Tasa de Error de Bit (BER), según especificaciones del fabricante (sección 2.6.1).
- Se estableció la comunicación simultánea con dos nodo basados en el dispositivo MT100SEM. Se emularon 6 dispositivos para cumplir con uno de los objetivos iniciales (sección 3.1.1, Figura 3.2 y Figura 3.10).
- Se estableció la comunicación entre los Nodos y la GUI (sección 3.1.3, Figura 3.7).
- Se desarrolló una GUI basada en web (sección 3.1.3).
- Los eventos generados en el sistema se almacenaron en una BD usando el gestor MySQL (sección 3.1.3, Figura 3.8).

Existen diferentes dispositivos que realizan la comunicación TCP/IP como por ejemplo los que presenta el fabricante Rabbit Co. [38], que son dispositivos que ya cuentan con un

MCU como parte de su arquitectura, teniendo implementado un servidor HTTP dentro de sí mismos. Esa característica es una referencia de hacia donde va dirigido este trabajo.

El trabajo de tesis desarrollado brinda la posibilidad de evolucionar el sistema de forma modular, dejando un desarrollo de trabajos futuros:

- Los productos de la firma Multitech comparten la característica de usar un *socket* universal (usado en la tarjeta de desarrollo MTSMI-UDK del mismo fabricante), gracias a este diseño es posible realizar cambios en el sistema para que tenga diferentes medios de comunicación, usando dispositivos como SocketModem® CDMA – MTSMC-C, SocketWireless® Wi-Fi® – MT800SWM o SocketWireless® Bluetooth® – MTS2BTSMI [29]. Aunque el diseño actual del sistema no soporta a ninguno de los dispositivos mencionados, se deja la posibilidad de que el sistema evolucione y soporte dos o más dispositivos del fabricante.

- Los desarrollos Web actuales tienden a utilizar tecnologías con mayor capacidad, usando el paradigma de programación orientada a objetos (POO). Una de las herramientas que mejoraría el desempeño del sistema es la denominada Web 2.0.

- El sistema final no cuenta con elementos de seguridad en la comunicación TCP/IP, siendo deseable que se cuente con algún tipo de seguridad en el intercambio de información. Los certificados de seguridad, conexiones de tipo SSL (*Socket Secure Layer*) usadas en HTTPS (*HTTP Secure*) permiten elevar el nivel de seguridad en una conexión TCP/IP.

Bibliografía

[1] ATMEL, Co. Atmel: ATmega8, Data Sheet. Rev. 2486Q–AVR–10/06. 2006. 309 p.

[2] ATMEL, Co. AVR Studio 4.3: AVR tools guide. 2007.

[3] BERGER, Arnold, Embedded Systems Design: An Introduction to Processes, Tools, and Techniques. CMP Books, 2002. 237 p.

[4] BODERO, Luis. SARMIENTO, Diego. Monitoreo Remoto de Procesos a través del Canal de Voz de Teléfonos Celulares GSM. Universidad Peruana de Ciencias Aplicadas. Perú, 2005.

[5] CARRANZA, C. MORALES, S. CARDÓN, L. DaMA-Web. Un programa para el monitoreo y control local y remoto vía web, de la adquisición de los datos. Avances en Energías Renovables y Medio Ambiente. Vol 11. 2007.

[6] CASTILLO, H. CASTAÑEDA, B. Sistema de gestión para redes de pequeñas y medianas empresas. Instituto Politécnico Nacional. México, 2004.

[7] CASTRO, Elizabeth. HTML, XHTML, and CSS. 6a. Ed. Peachpit Press, 2006. 456 p.

[8] CHAMÚ, Carlos. Desarrollo de un Sistema Educativo para la Enseñanza del protocolo de Comunicaciones CAN. Trabajo de titulación (Ingeniero en Electrónica). México : Universidad Tecnológica de la Mixteca, Abril 2005. 136 p.

[9] CHAVES, Adolfo, ARAYA, R. Freddy. Desarrollo de una red de monitoreo por sensores remotos de la cantidad de agua. Tecnología en Marcha. Vol. 18 No. 2. 2002.

[10] COMER, Douglas. Redes de Computadoras Internet e Interredes. Prentice Hall, 1997. 506 p.

[11] DUQUE, Edison. Curso básico de microcontroladores PIC. Cekit S A, 1997. 115 p.

[12] EGUÍLUZ, Pérez Javier. Introducción a CSS [en línea]. 2008. Disponible en: http://www.librosweb.es/css/.

[13] EGUÍLUZ, Pérez Javier. Introducción a XHTML [en línea]. 2008. Disponible en: http://www.librosweb.es/xhtml/.

[14] EGUÍLUZ, Pérez Javier. CSS avanzado [en línea]. 2009. Disponible en: http://www.librosweb.es/css_avanzado/.

[15] ESARDA Working Group on C/S. Guidelines for Developing Unattended and Remote Monitoring and Measurement Systems. JRC Ispra, 2006. Disponible en: http://esarda2.jrc.it/bulletin/bulletin_33/. Fecha de consulta: 23 de Octubre del 2009.

[16] FORD, Merille. Tecnologías de interconectividad de redes. Prentice Hall, 1998. 736 p.

[17] FREEMAN, Elisabeth. FREEMAN, Eric. Head First HTML with CSS & XHTML. O'Reilly Media Inc., 2005. 658 p.

[18] GADRE, Dhananjay. Programming and customizing the AVR microcontroller [en línea]. McGraw Hill, 2001. Disponible en: http://books.google.com.mx.

[19] Guía Breve de CSS. W3C. Disponible en: http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo. Fecha de consulta: 21 de Febrero del 2009.

[20] HAGEN, Silvia. IPv6 Essentials. O'Reilly Media Inc, 2006. 436 p.

[21] HARRIS, Andy. MCCULLOH, Chris. HTML, XHTML, and CSS. All-in-One Desk Reference For Dummies. Pap/Cdr ed, 2008. 960 p.

[22] JIMÉNEZ, L. REINOSO, O. Laboratorios remotos para las prácticas de ingeniería de sistemas y automática en la universidad Miguel Hernandez. España. 2003.

[23] Lista de navegadores. Mas adelante. Disponible en: http://www.masadelante.com/faq-listado-buscadores.htm . Fecha de consulta: 21 de Febrero del 2009.

[24] LIU, Cricket. ALBITZ, Paul. DNS and BIND. O'Reilly Media Inc., 1998. 502 p.

[25] LLOBET, J. MATÍAS, I. ARREGUI F. Desarrollo de un sistema flexible de control local y remoto para una red domótica en edificios inteligentes. Universidad Pública de Navarra : Dpto. Ingeniería Eléctrica y Electrónica. 1998.

[26] LLOYD, Ian. Build Your Own Web Site The Right Way Using HTML & CSS. SitePoint, 2008. 488 p.

[27] MEYER, Eric. CSS: The Definitive Guide. O'Reilly Media Inc., 2006. 536 p.

[28] MULTITECH Systems. SocketEthernet IP: AT Commands Reference Guide. Rev. S000426F-F. 2008.

[29] MULTITECH Systems. Universal Socket Connectivity Embedded, Device Networking Solutions: Hardware Guide for Developers. Rev. S000342I-I. 2008.

[30] MUSCIANO, Chuck. KENNEDY, Bill. HTML & XHTML: The Definitive Guide. 4a. ed. O'Reilly Media Inc., 2006. 672 p.

[31] MYSQL 5.1 Reference Manual. MySQL AB. Disponible en: http://dev.mysql.com/doc/refman/5.1/en/. Fecha de consulta: 21 de Febrero del 2009.

[32] MYSQL Downloads: MySQL Connector/ODBC 5.1. Disponible en: http://dev.mysql.com/downloads/. Fecha de consulta: 3 de Marzo del 2009.

[33] NOERGAARD, Tammy. Embedded Systems Architecture: A comprehensive guide for engineers and programmers. Elsevier Inc., 2005. 640 p.

[34] ORDIALES, R. MAGÁN, H. VIDAL, S. Vigilando el desierto. Descripción del Sistema de Telemetría de la Estación Experimental de Zonas Áridas (Almería). Ecosistemas, Revista de Ecología y Medio Ambiente, Septiembre 2001.

[35] OVIEDO, Rubén. Sistema mínimo de propósito general basado en el microcontrolador DS80C400 con operación en un sistema de red. Trabajo de titulación (Ingeniero en Electrónica). México : Universidad Tecnológica de la Mixteca, Julio 2007. 111 p.

[36] PHP: Hypertext Preprocessor. The PHP Group. Disponbile en: http://www.php.net/. Fecha de consulta: 21 de Febrero del 2009.

[37] POSTEL, Jon. RFC 791: Internet Protocol [en línea]. 1981. Disponible en: http://www.ietf.org/rfc/rfc791.txt. Fecha de consulta: 23 de Septiembre del 2009.

[38] RABBITCORE Modules. RABBITCORE. Disponible en: http://www.rabbit.com/products/CoreModules/index.shtml. Fecha de consulta: 11 de Octubre 2009.

[39] RABUÑAL, J. OLIVEIRA, S. Desarrollo de un Servidor Web para Adquisición de Datos en Tiempo Real. Universidad de A Coruña, Departamento de Tecnologías de la

Información y las Comunicaciones. España. 2002.

[40] RANZ, Carlos. GARCÍA, Mario. Control remoto del tanque de experimentación hidroacustica del Instituto de Acústica del CSIC. España. 1999.

[41] SCHMITT, Christopher. CSS Cookbook. O'Reilly Media Inc., 2006. 528 p.

[42] SHEA, Dave. HOLZSHLAG, Molly. The Zen of CSS Design: Visual Enlightenment for the Web. Peachpit Press, 2005. 304 p.

[43] SOMMERVILLE, Ian. Ingeniería del software. 7a. ed. Pearson Education, 2005. 712 p.

[44] SPURGEON, Charles. Ethernet: The definitive guide. O'Reilly Media Inc., 2000. 527 p.

[45] STALLINGS, William. Comunicaciones y redes de computadoras. 7a. ed. Prentice Hall Hispanoamericana, 2004. 831 p.

[46] STEVENS, Richard. TCP/IP Illustrated: The Protocols. Addison-Wesley, 2002. 576p.

[47] TANENBAUM, Andrew. Computer Networks. 4a. ed. Prentice Hall, 2002. 891 p.

[48] The PHP License version 3.01. The PHP Group. Disponible en: http://www.php.net/license/ 3_01.txt. Fecha de consulta 21 de Febrero del 2009.

[49] ZIMMERMANN, Hubbert. OSI Reference Model: The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications. Vol. 28. No. 4. Abril 1980. pp. 425 – 432.

Anexo A Asignación de terminales en el nodo

A.I MCU Atmega8

En la Tabla A.1 se muestra el número y nombre de cada terminal con su función alternativa (usada en algunos casos), su configuración (entrada/salida) y la función que realiza dentro del nodo.

Tabla A.1. Configuración de terminales en el ATmega8.

No terminal	Nombre	Función alternativa	E/S	Función	Descripción
1	PC6	RST	E	*Reset*	Regresa al estado inicial del MCU.
2	PD0	RXD	E	MT100_RXD	Receptor de comunicación.
3	PD1	TXD	S	MT100_TXD	Transmisor de comunicación.
4	PD2	INT0	E	MT100_LNK	Interrupción externa. Estado del enlace.
5	PD3	INT1	E/S	MT100_RST	Detección de *reset* externo y generación de *reset* por *software*.
6	PD4	XCK/T0	E	IN_DIG0	Entrada digital.
7	VCC	-	-	-	Voltaje de alimentación.
8	GND	-	-	-	Tierra eléctrica.
9	PB6	XTAL1	E	-	Entrada del cristal oscilador.
10	PB7	XTAL2	E	-	Entrada del cristal oscilador.
11	PD5	T1	E	IN_DIG1	Entrada digital.
12	PD6	AIN0	E	IN_DIG2	Entrada digital.
13	PD7	AIN1	E	IN_DIG3	Entrada digital.
14	PB0	ICP1	S	LCD_DB4	Data-Bit 4 para la interfaz con LCD.
15	PB1	OC1A	S	OUT_DIG0/LCD_DB4	Salida digital. *Data-Bit* 5 para la interfaz con LCD.
16	PB2	OC1B	S	OUT_DIG1/LCD_DB4	Salida digital. *Data-Bit* 6 para la interfaz con LCD.
17	PB3	MOSI	S	LCD_DB7	*Data-Bit* 7 para la interfaz con LCD.
18	PB4	MISO	S	LCD_RS	Instrucción/Dato en la comunicación con LCD.
19	PB5	SCK	S	LCD_E	Activación de comunicación con LCD.
20	AVCC	-	-	-	Voltaje de alimentación para los ADCs.
21	AREF	-	-	-	Voltaje de referencia para los ACDs.
22	GND	-	-	-	Tierra eléctrica.
23	PC0	ADC0	E	IN_AN0	Entrada analógica.
24	PC1	ADC1	E	IN_AN1	Entrada analógica.
25	PC2	ADC2	E	IN_AN2	Entrada analógica.
26	PC3	ADC3	E	IN_AN3	Entrada analógica.
27	PC4	ADC4	S	OUT_DIG2	Salida digital.
28	PC5	ADC5	S	OUT_DIG3	Salida digital.

Anexo B Diagrama esquemático y PCB del nodo

B.I Esquemático

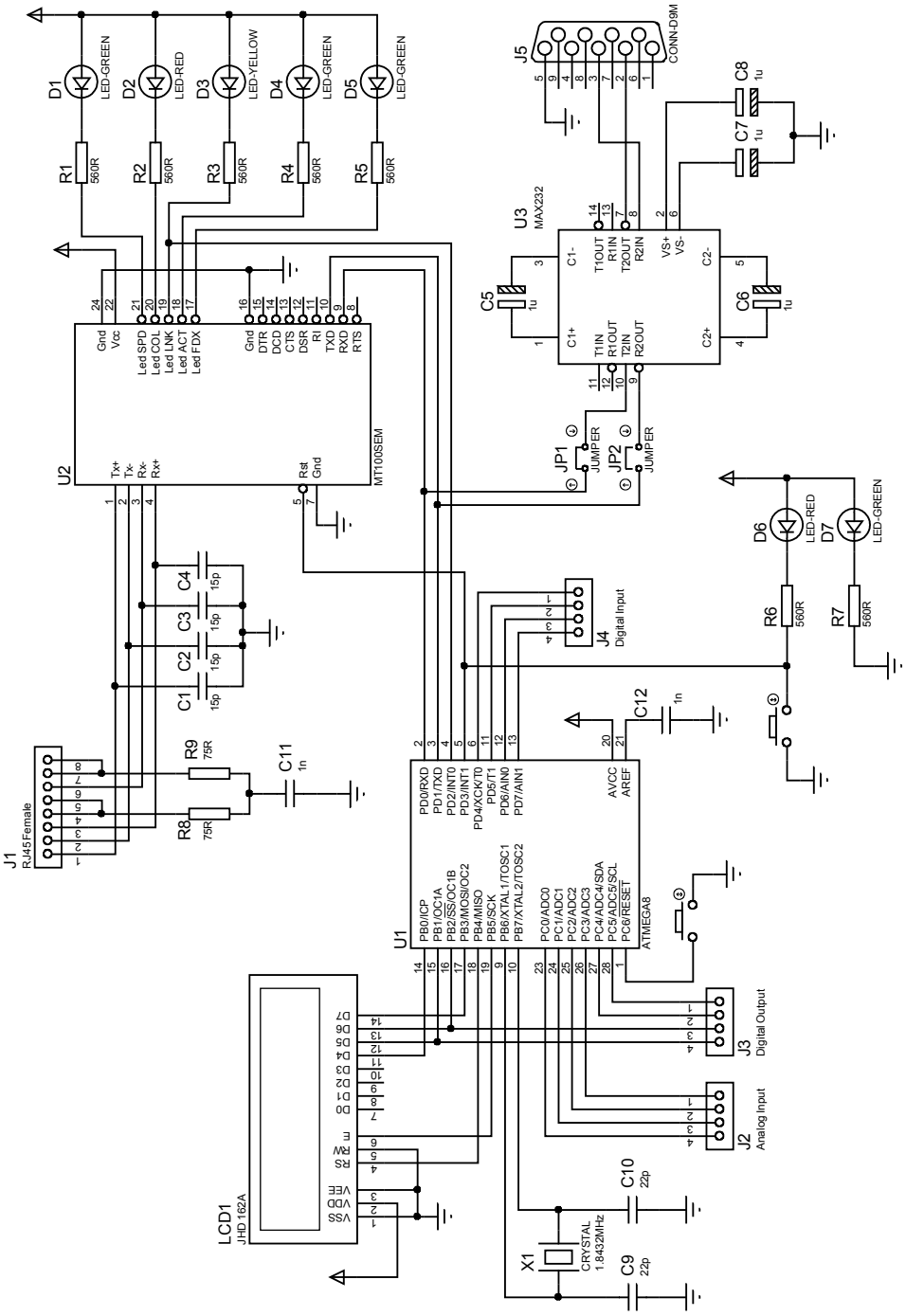


Figura B.1. Diagrama esquemático del nodo.

B.II Elementos de protección para el MCU

Para evitar que las terminales del MCU Atmega8 tengan sobrecarga de corriente (5 mA por terminal), se colocaron elementos de protección. El elemento propuesto es un amplificador operacional (LM324N de la firma ST) que proporciona protección en las terminales. La configuración para las E/S digitales es la de un comparador de voltaje (Figura B.2a y Figura B.2b) y para las entradas digitales se utiliza un seguidor de voltaje (Figura B.2c).

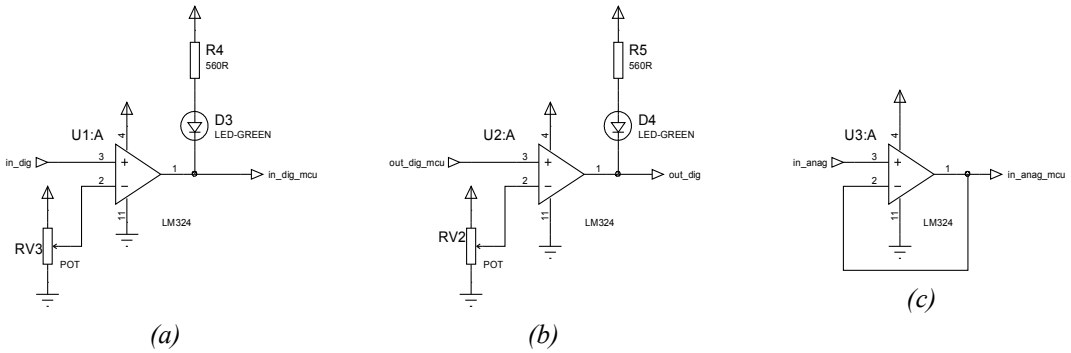


Figura B.2. Elementos de protección para el MCU.
a. Entradas digitales al nodo; b. Salidas digitales del nodo. c. Entradas analógicas al nodo.

B.III Diagrama de posición de elementos

El diagrama de posición de los elementos (*top and botttom skill*) proporciona al desarrollador de *hardware* un esquema de posicionamiento de los elementos en el nodo (Figura B.3).

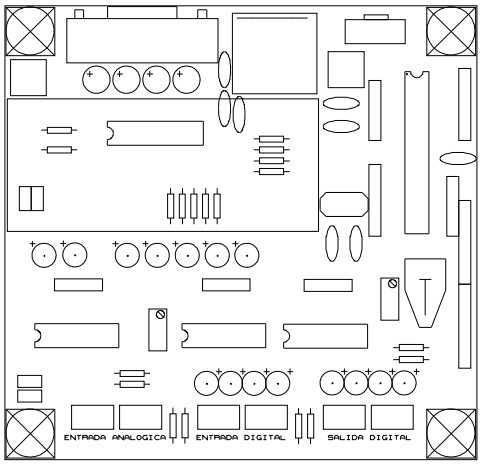


Figura B.3. Posición de los elementos en el nodo.

B.IV Capa superior

La Figura B.4 muestra la capa superior (*top copper*) del PCB diseñado.

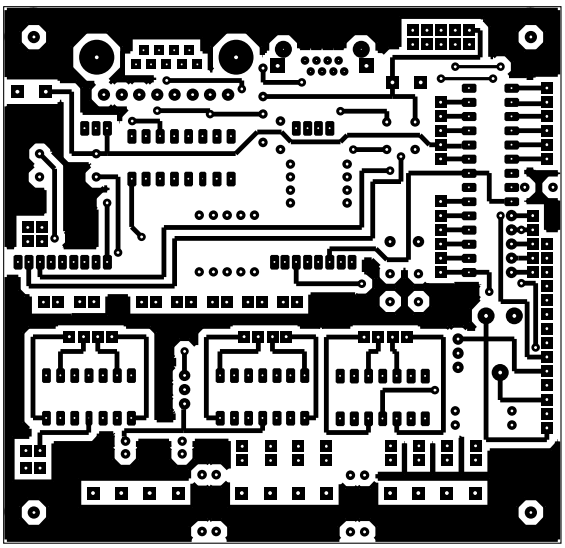


Figura B.4. Capa superior del nodo.

B.V Capa inferior

La Figura B.5 muestra la capa inferior (*bottom copper*) del nodo.

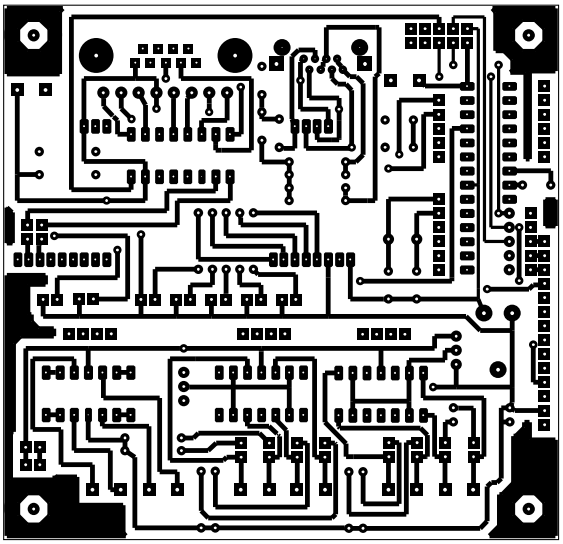


Figura B.5. Capa inferior del nodo.

Anexo C Actualización de Firmware MT100SEM

El dispositivo MT100SEM utilizado en este trabajo contenía la versión *firmware* v1.00c, la cual no cumplía las expectativas del sistema. El soporte técnico de Multitech proporcionó una actualización con la cual se tiene una mayor seguridad en el dispositivo, la versión actualizada es la v1.01c.

Para realizar la actualización del *firmware* es necesario tener el *software* llamado flashwiz2002.exe y el archivo HnQx101c.hex (*firmware* actualizado). Una vez instalado el programa es necesario colocar el archivo de actualización dentro del directorio de instalación del *software* y se continua con el proceso de actualización*.

Cuando se ejecuta el *Firmware Update Wizard* tarda aproximadamente 20 segundos en reconocer el dispositivo conectado en un puerto RS232 (Figura C.1). Una vez que el asistente termina el escaneado de puertos, muestra el dispositivo reconocido y el puerto al que se encuentra conectado (Figura C.2).

Figura C.1. Ventana inicial del Firmware Update Wizard.

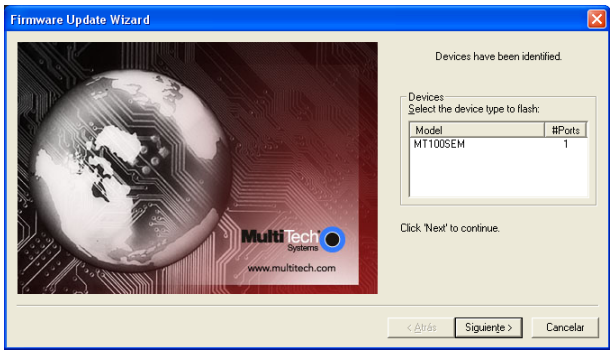


Figura C.2. Detección del dispositivo MT100SEM.

* El fabricante recomienda tener extrema precaución durante la actualización del dispositivo debido a que existe la posibilidad de dañarlo a causa de una interrupción inesperada en el proceso.

Al hacer clic en el botón siguiente, el *Firmware Update Wizard* reconoce el archivo *firmware* actualizado (HnQx101c.hex) que está en el directorio de la instalación (Figura C.3). Se muestra información referente al país de distribución y la versión del archivo entre otros.

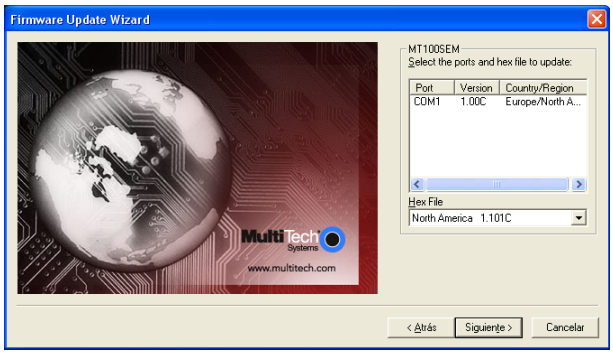


Figura C.3. Reconocimiento del firmware actualizado.

El proceso siguiente es el envío de datos de la actualización (Figura C.4), en el cual se muestra un barra de estado donde se puede observar el porcentaje de la actualización. En ese momento no debe haber algún tipo de interrupción en el proceso. Una vez terminado el envío de datos se muestra una confirmación de finalización (Figura C.5).

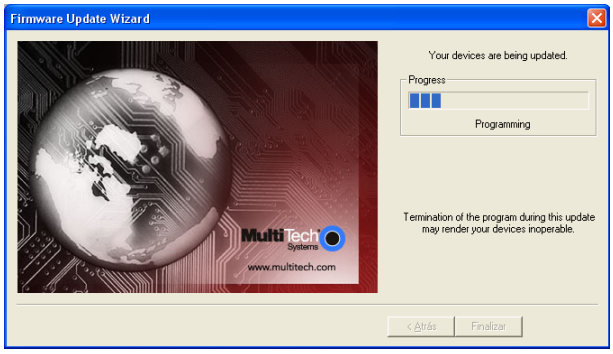


Figura C.4. Proceso de actualización de firmware.

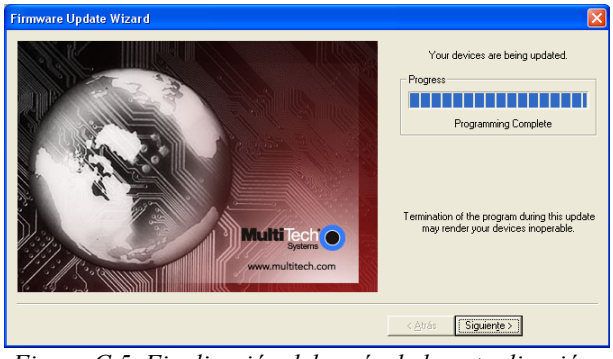


Figura C.5. Finalización del envío de la actualización.

Para terminar la fase de actualización, al hacer clic en el botón siguiente se muestra la pantalla final de *Firmware Update Wizard* e indica el total de dispositivos actualizados y el estado de la actualización (Figura C.6).

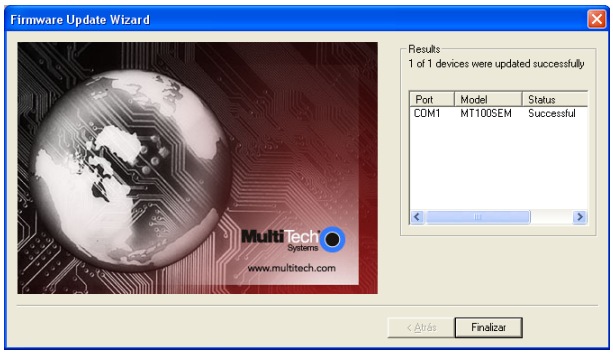


Figura C.6. Estado final del la actualización.

C.I Versiones firmware del MT100SEM

En este trabajo se realizó la actualización *firmware* del dispositivo MT100SEM por razones de seguridad. La actualización mejora el funcionamiento de la conexión TCP/IP e incluye seguridad de algunos elementos. Las características más sobresalientes de la actualización del dispositivo son (Tabla C.1):

- Protección por contraseña en el acceso en la conexión por Telnet.
- Soporte de *hand-shaking* en la comunicación RS232.
- Protección por contraseña de la función auto-discovery.

Tabla C.1. Comparativa entre versiones firmware del MT100SEM.

Versión 1.00c	Versión 1.01c
AT#VALL #FTPGETFILENAME: "" #FTPGETPATH: "" #FTPMODE: 0 #FTPPORT: 21 #FTPPUTFILENAME: "" #FTPPUTPATH: "" #FTPPW: "" #FTPSERV: "" #FTPTYPE: i #FTPUN: "" #POP3HEADERMODE: 1 #POP3PORT: 110 #POP3PW: "" #POP3SERV: "" #POP3UN: "" #DOMAIN: "" #SENDERADDR: "" #SENDERNAME: "" #SMTPPORT: 25 #SMTPPW: "" #SMTPSERV: "" #SMTPUN: ""	AT#VALL #FTPGETFILENAME: "" #FTPGETPATH: "" #FTPMODE: 0 #FTPPORT: 21 #FTPPUTFILENAME: "" #FTPPUTPATH: "" #FTPPW: "" #FTPSERV: "" #FTPTYPE: i #FTPUN: "" #POP3HEADERMODE: 1 #POP3PORT: 110 #POP3PW: "" #POP3SERV: "" #POP3UN: "" #DOMAIN: "" #SENDERADDR: "" #SENDERNAME: "" #SMTPPORT: 25 #SMTPPW: "" #SMTPSERV: "" #SMTPUN: ""

Versión 1.00c	Versión 1.01c
#SMTPAUTH: 1 #BODY1: "" #CCREC1: "" #REC1: #SUBJ1: "" #BODY2: "" #CCREC2: "" #REC2: #SUBJ2: "" #BODY3: "" #CCREC3: "" #REC3: #SUBJ3: "" #DLEMODE: 1 , 1 #TCPSERV: 1 , "" #TCPPORT: 1 , 0 #TCPTXDELAY: 1 , 100 #DLEMODE: 2 , 1 #TCPSERV: 2 , "" #TCPPORT: 2 , 0 #TCPTXDELAY: 2 , 100 #UDPSERV: 1 , "" #UDPPORT: 1 , 0 #UDPTXDELAY: 1 , 100 #UDPSERV: 2 , "" #UDPPORT: 2 , 0 #UDPTXDELAY: 2 , 100 #PINGDELAY: 1 #PINGNUM: 4 #PINGREMOTE: "" #DHCP: 0 #IPADDR: "192.168.2.3" #IPGATEWAY: "192.168.2.1" #IPNETMASK: "255.255.255.0" #DNSSERV1: "" #DNSSERV2: "" #TELNET: 0 +IPR: 115200 +ICF: 2,4 #AUTODISC: 1 #AUTODISCPORT: 1020 #AUTODISCTIMER: 10 #AUTODISCHOST: "MT100SEM" OK	#SMTPAUTH: 1 #BODY1: "" #CCREC1: "" #REC1: #SUBJ1: "" #BODY2: "" #CCREC2: "" #REC2: #SUBJ2: "" #BODY3: "" #CCREC3: "" #REC3: #SUBJ3: "" #DLEMODE: 1 , 1 #TCPSERV: 1 , "" #TCPPORT: 1 , 0 #TCPTXDELAY: 1 , 100 #DLEMODE: 2 , 1 #TCPSERV: 2 , "" #TCPPORT: 2 , 0 #TCPTXDELAY: 2 , 100 #UDPSERV: 1 , "" #UDPPORT: 1 , 0 #UDPTXDELAY: 1 , 100 #UDPSERV: 2 , "" #UDPPORT: 2 , 0 #UDPTXDELAY: 2 , 100 #PINGDELAY: 1 #PINGNUM: 4 #PINGREMOTE: "" #DHCP: 0 #IPADDR: "192.168.2.3" #IPGATEWAY: "192.168.2.1" #IPNETMASK: "255.255.255.0" #EMACSPD: 0 #DNSSERV1: "" #DNSSERV2: "" #TELNET: 1 #TELNETPORT: 23 #TELNETUSER: "admin" #TELNETPASSWORD: "" +IPR: 115200 +ICF: 2,4 +IFC: 0,0 #AUTODISC: 1 #AUTODISCPORT: 1020 #AUTODISCTIMER: 10 #AUTODISCHOST: "MT100SEM" #AUTODISCUSER: "admin" #AUTODISCPASSWORD: "" V: 1 &S: 0 &C: 0 OK

Anexo D Auto-Discovery Manager

El *Auto-Discovery Manager* es un *software* propietario que permite encontrar los distintos dispositivos de la firma Multitech conectados a una determinada red. Esta herramienta permite a los administradores conocer información del dispositivo, tal como dirección MAC, dirección IP, máscara de sub-red, si está habilitado el DHCP, nombre *host*, puerto de escaneo y versión *firmware*. La Figura D.1 muestra la ventana principal de la herramienta donde se presenta la información de los dos dispositivos usados en este trabajo de tesis.

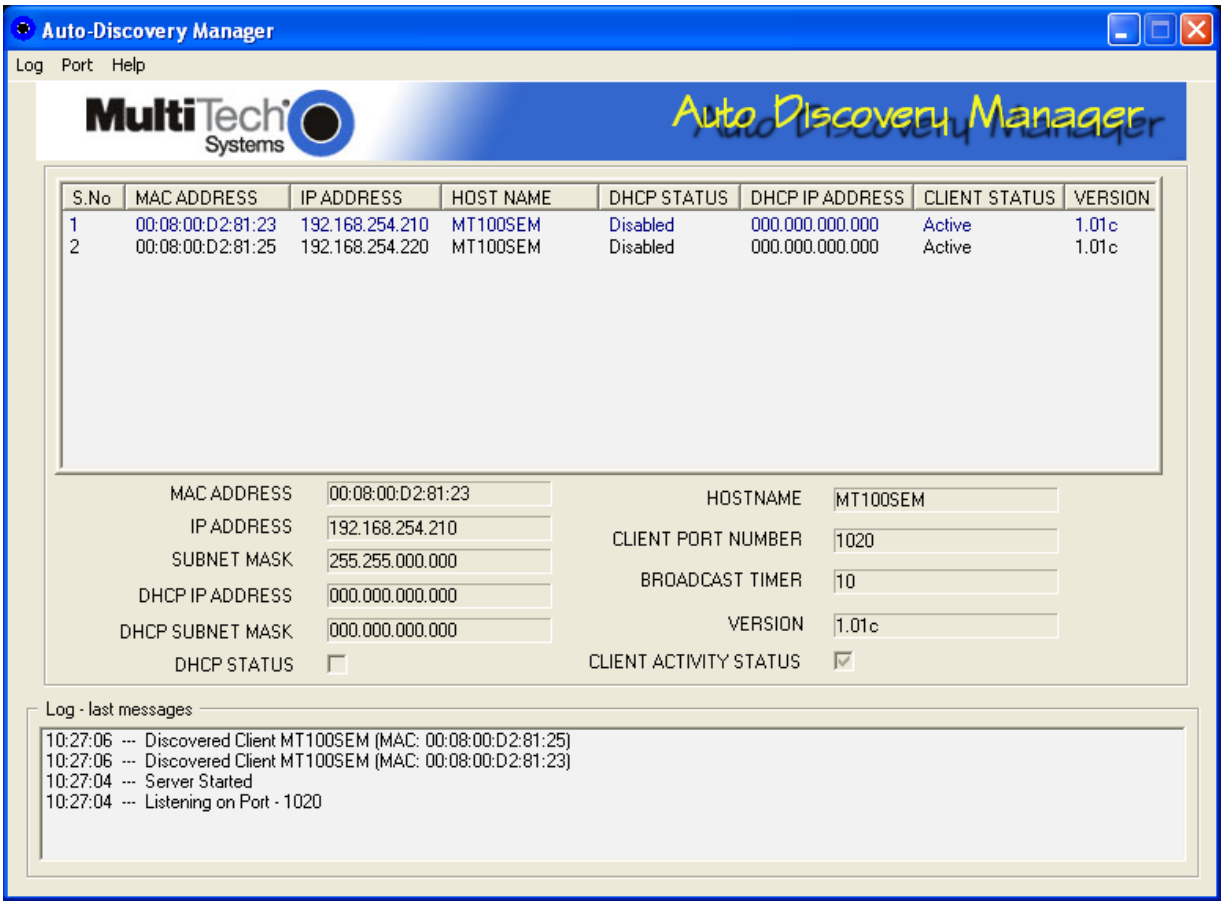


Figura D.1. Herramienta para encontrar dispositivos MT100SEM en la red local.