UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO Y CONTROL REMOTO DE ENCENDIDO/APAGADO DE SISTEMAS DE AIRE ACONDICIONADO, ILUMINACION Y GRUPO ELECTRÓGENO EN AGENCIAS BANCARIAS Y/O GUBERNAMENTALES EMPLEANDO PROTOCOLO TCP/IP

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

RENZO RICARDO CANO SIALER

PROMOCIÓN

2005 - I

LIMA – PERÚ
2011

IMPLEMENTACION DE UN SISTEMA DE MONITOREO Y CONTROL REMOTO DE ENCENDIDO/APAGADO DE SISTEMAS DE AIRE ACONDICIONADO, ILUMINACIÓN Y GRUPO ELECTRÓGENO EN AGENCIAS BANCARIAS Y/O GUBERNAMENTALES EMPLEANDO PROTOCOLO TCP/IP

A Dios, mis padres, hermanos y a todas aquellas personas que colaboraron en muchos aspectos e hicieran posible que llegue a este punto.

SUMARIO

El presente informe trata de la implementación de un sistema de monitoreo y control en forma remota utilizando como hardware una tarjeta de control con microcontrolador (PIC WEB) que contiene un puerto de comunicación Ethernet esta tarjeta de control es enlazada a una tarjeta acondicionadora de señales de proceso, con respecto al software se ha desarrollado un conector en Java que permite por un lado conectarse a la tarjeta de control, y por otro lado conectarse a una base de datos, luego también se ha desarrollado una interfaz web que permite mostrar la información almacenada en la base de datos en cualquier dispositivo que esté conectado a internet, tanto el conector en Java, la base de datos y la interfaz web son alojadas en una estación servidora.

La red Ethernet por medio de los protocolos TCP, IP y HTTP fueron empleadas para poder comunicar este sistema.

ÍNDICE

PRÓLOGO 1

CAPÍTULO I
NECESIDAD 2

1.1. Descripción de la necesidad 2

1.2 Objetivos del diseño y la implementación 2

1.3 Alcances del proyecto 3

1.4 Limitaciones del proyecto 3

CAPÍTULO II
MARCO TEÓRICO GENERAL 4

2.1 Introducción a la automatización 4

2.2 Sensores y actuadores 5

2.3 Controlador lógico programable (PLC´s) 5

2.4 Microcontrolador 6

2.5 Tarjeta acondicionadora de señal 8

2.6 Protocolos y redes de comunicación de edificios 8

2.6.1 Protocolo Modbus 11

2.6.2 Protocolo Bacnet 13

2.6.3 Protocolo Lonwork 15

2.6.4 Protocolo KNX 16

2.7 La red Ethernet 16

2.7.1 Protocolo TCP 18

2.7.2 Protocolo UDP 24

2.7.3 Protocolo IP 26

2.7.4 Protocolo HTTP 27

2.7.5 Protocolo FTP 28

2.7.6 Protocolo SMTP 28

2.8 Sistema de monitoreo y control remoto (SCADA) 28

2.9 Software libre 29

2.9.1 Plataforma de desarrollo Java 29

2.9.2 Servidor web 29

2.9.3 Aplicaciones web 30

2.9.4 Servidor de base de datos 30

CAPÍTULO III
HARDWARE Y SOFTWARE IMPLEMENTADO PARA SATISFACER LA NECESIDAD 31

3.1 Arquitectura de monitoreo y control del sistema implementado 31

3.1.1 Elección de la red y protocolos utilizada 31

3.1.2 Elección de la tarjeta de monitoreo y control (hardware) 32

3.1.3 Implementación de la tarjeta acondicionadora de señal 33

3.1.4 Elección de la plataforma de software que se comunica con la tarjeta de control .. 36

3.2 Pantallas de supervisión 38

3.2.1 Elección de la base de datos 38

3.2.2 Elección del servidor web 39

3.3 Arquitectura de comunicación entre las diferentes plataformas 39

3.4 Declaración de variables en la tarjeta electrónica (PIC WEB) 40

3.4.1 Declaración de variables en la pila del microcontrolador 40

3.5 Conexión con la tarjeta electrónica (PIC WEB) 41

3.6 Conexión entre el conector en Java y la base de datos 43

3.7 Conexión entre el servidor apache y la base de datos 43

CAPÍTULO IV
ANÁLISIS Y PRESENTACIÓN DE RESULTADOS 44

4.1 Edificación a escala para las pruebas e implementación 44

4.2 Pruebas iniciales de comunicación con la tarjeta electrónica 45

4.3 Conexión con la plataforma Java 48

4.4 Conexión remota entre el usuario y la base de datos/ingreso al sistema 49

4.5 Pantallas principales 50

4.6 Participantes del proyecto 53

4.7 Cronograma de ejecución del proyecto 53

4.8 Costos del proyecto 53

CONCLUSIÓNES Y RECOMENDACIONES 57

ANEXO A
PROGRAMA EN C – DECLARACIÓN DE VARIABLES DIGITALES EN EL MICROCONTROLADOR 61

ANEXO B
PROGRAMA EN C – DECLARACIÓN DE VARIABLES ANALÓGICAS Y PROCESAMIENTO DE LAS SEÑALES EN EL MICROCONTROLADOR 63

ANEXO C
PROGRAMA CONECTOR ENTRE JAVA Y LA TARJETA PIC WEB 68

ANEXO D
PROGRAMA CONECTOR CON LA BASE DE DATOS 70

ANEXO E
CONECTOR ENTRE EL SERVIDOR WEB Y BASE DE DATOS 72

ANEXO F
PROGRAMA EN MICROCONTROLADOR PIC 18F67J60 SERVIDOR TCP 74

ANEXO G
PROGRAMA EN MICROCONTROLADOR PIC 18F67J60 CLIENTE TCP 78

ANEXO H
PROGRAMA EN MICROCONTROLADOR PIC 18F67J60 APLICACIÓN HTTP 83

ANEXO I
ESPECIFICACIONES TÉCNICAS DE LA TARJETA PIC WEB 89

ANEXO J
ESPECIFICACIONES TÉCNICAS DEL MICROCONTROLADOR PIC 18F67J60 91

ANEXO K
PANTALLA DE PRUEBA PIC WEB 94

ANEXO L
DATASHEET DEL TRANSMISOR DE PRESION DIFERENCIAL 96

ANEXO M
DIAGRAMA CIRCUITAL DE LA TARJETA PIC WEB 98

BIBLIOGRAFÍA 100

PRÓLOGO

El presente informe trata sobre la implementación y utilización de hardware de bajo costo y el empleo de software libre que permite optimizar procesos que actualmente resultaría demasiado costoso con equipos y dispositivos de fabricantes tradicionales, este informe contiene cinco capítulos:

El **"Capítulo I"**, presenta la descripción de la necesidad imperativa que se tiene respecto de la reducción de los costos en cuanto a sistemas de monitoreo y control, también abarca los alcances y limitaciones de este proyecto/implementación, en cuanto a que por lo pronto esta implementación no abarca técnicas de control avanzados.

El **"Capítulo II"**, brinda el marco teórico general en que se sustenta todo el proyecto.

El **"Capítulo III"**, abarca todos los procedimientos realizados durante el desarrollo, programación y las diferentes pruebas realizadas para el logro de este proyecto.

El **"Capítulo IV"**, engloba todos los resultados finales obtenidos en la implementación de este proyecto.

En **"Conclusiones y recomendaciones"**, se presentan las conclusiones finales a la que se ha llegado respecto de este proyecto, así también se indican algunas recomendaciones.

La implementación y el éxito de este proyecto se debieron gracias al esfuerzo proporcionado por el departamento de investigación y desarrollo de la compañía "C y Q Ingeniería e Instrumentación S.A.C.".

CAPITULO I
NECESIDAD

1.1 Descripción de la necesidad.

Hoy en día las oficinas bancarias y/o gubernamentales por política corporativa requieren optimizar el sistema de encendido de iluminación, aire acondicionado, grupo electrógeno y monitoreo de temperatura, esta optimización conlleva a una reducción de sus costos energéticos, monitoreo de sus sistemas y confort en el desempeño de las labores de sus empleados, pero a la vez la optimización de dichos sistemas deben ir acompañados de costos relativamente bajos en cuanto a los dispositivos de monitoreo y control en forma remota, es decir hardware y software de bajo costo.

En este sentido existe la necesidad para este rubro de implementar hardware y desarrollar software localmente y de bajo costo, si bien es cierto en el mercado existen sistemas de monitoreo y control remoto de fabricantes tradicionales y reconocidos mundialmente pero son demasiado costosos, lo que conlleva a retrasar los proyectos o implementarlos solamente en las oficinas más importantes y emblemáticas dejando de lado a las otras.

Por otro lado esta misma necesidad existe por ejemplo en las constructoras que construyen edificios multifamiliares, condominios y centros comerciales, donde darle un grado de automatización con sistemas y equipos de fabricantes tradicionales y reconocidos hacen que el costo por departamento resulte demasiado costoso para el usuario final y no pueda pagarlo.

1.2 Objetivos del diseño y la implementación.

Los objetivos principales respecto de la implementación de este proyecto son:

✓ Permite reducir costos en cuanto a la implementación de sistemas de control (Hardware) aplicadas a oficinas bancarias, edificaciones gubernamentales y que pueda extenderse a hogares, centros comerciales y también para la pequeña y mediana industria.

✓ Permite reducir costos en cuanto a desarrollo de software de monitoreo y control en forma remota mediante la utilización de software libre.

✓ Permite reducir en general costos en la realización de un proyecto de monitoreo y control para una edificación inclusive para la mediana y pequeña industria. Como consecuencia de esta reducción de costos es posible que más clientes o usuarios finales puedan implementar estos sistemas y de esta manera lograr la tan ansiada eficiencia en los procesos productivos, ahorro en el consumo energético, seguridad y confort.

1.3 Alcances del proyecto.

El alcance de este proyecto involucra comandar (manualmente, no automáticamente) y monitorear señales de proceso desde cualquier dispositivo remoto conectado a internet por medio de una interfaz web gráfica o pantalla de monitoreo y control en forma remota, las señales que se comandarán son:

- Señal de salida digital "Encendido de sistema de iluminación"
- Señal de salida digital "Encendido de aire acondicionado".
- Señal de salida digital "Encendido de grupo electrógeno".

Las señales que se deben monitorear son:

- Señal de entrada digital "Alarma de intrusión".
- Señal de entrada analógica 4-20mA "Sensado de temperatura en oficina".
- Señal de entrada analógica 4-20mA "Sensado de nivel de cisterna de agua".

1.4 Limitaciones del proyecto.

Las limitaciones que se tienen con respecto al proyecto en general y específicamente con la implementación es que no se tienen rutinas de control o no se ha implementado técnicas de control ni control avanzado dado que no es el objetivo (a lo mas existe un control manual).

CAPÍTULO II
MARCO TEORICO GENERAL

2.1 Introducción a la automatización de edificios.

La automatización de edificios e infraestructuras es análogo a la automatización de procesos industriales y en ambos casos se requiere de una serie de sistemas y sub-sistemas tanto en hardware como software para poder monitorear y/o controlar alguna variable de campo o de proceso en forma centralizada o en forma remota y todo el sistema de automatización se clasifica en varios segmentos a la cual al conjunto se le denomina "La pirámide de la automatización", en la figura siguiente (FIG. 2.1) se muestra los diferentes segmentos en los cuales se sub-divide "La pirámide de la automatización", en el caso particular de este informe y para el caso de la implementación solo se ha basado en los tres primeros segmentos, es decir instrumentación de campo (sensores y actuadores), controladores (PLC´s, microcontrolador) y sistema de supervisión y control (SCADA), estos conceptos fueron extraídos de los libros referenciados en [1] y [2].

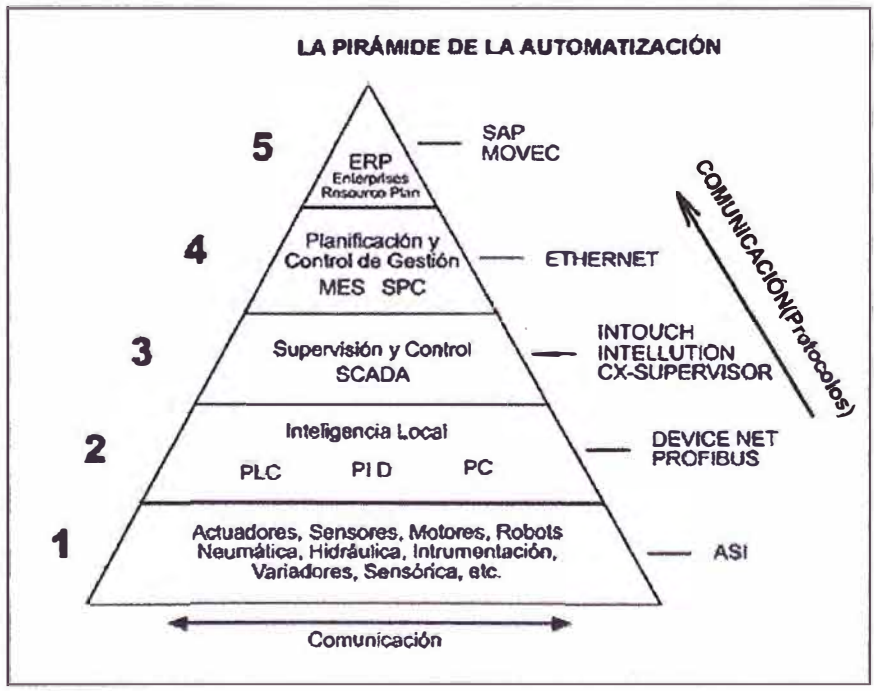


FIG. 2.1, Clasificación en cinco niveles de la pirámide de automatización, Fuente: Manual/Diapositivas curso de instrumentación industrial UNI-FIIS.

2.2 Sensores y actuadores.

Los sensores, son parte fundamental de todo sistema de automatización si bien es cierto se ubican en el segmento más bajo de "La pirámide de la automatización" pero no por ello deja de ser hasta cierto punto el segmento más importante, dado que el control óptimo de un proceso cualquiera depende de una señal correcta del proceso por lo que el sensor debe ser adecuadamente seleccionado para el proceso y debidamente calibrado a los parámetros que requiere el proceso, en el mercado encontramos sensores para edificios, oficinas y hogares como por ejemplo:

✓ Sensores de nivel, para monitorear y controlar el nivel de agua de la cisterna del sistema contra-incendio de un edificio, por ejemplo una bolla, o una sonda capacitiva.

✓ Sensores de presión, para monitorear la presión en los ductos del sistema de aire acondicionado, para evitar que los ductos se sobre-presionen.

✓ Sensores de temperatura, para monitorear temperatura ambiente en oficinas o habitaciones, por ejemplo un sensor tipo termistor.

También en el mercado encontramos actuadores como por ejemplo:

✓ Válvulas con actuadores neumáticos, para dosificar el caudal de agua helada hacia las manejadoras de aire acondicionado en los edificios.

✓ Válvulas (Damper) con actuadores eléctricos, para modular por ejemplo el flujo de aire

Por otro lado todos los sensores e instrumentos de proceso en general deben ser necesariamente calibrados/ajustados a sus parámetros de operación normal, para que puedan dar lecturas correctas hacia los dispositivos de control y visualización, todos estos pasos son descritos en los libros, manuales y guías referenciados en este informe [1] y [2].

2.3 Controlador lógico programable (PLC).

Un controlador lógico programable (PLC), es un dispositivo de control que van desde los MICRO-PLC hasta los PLC`s altamente sofisticados y que operan como DCS´s (sistemas de control distribuido) para aplicaciones sencillas como control de iluminación en plantas y/o edificios, apertura y cierre de puertas, monitoreo y control de variables de grupo electrógeno hasta control de intensidad de iluminación, temperatura, producción de cemento, gas, petróleo, gases criogénicos, etc. Este tipo de dispositivos contienen técnicas de control pre-establecidas y bastante avanzadas que lo hacen ideal para aplicaciones de la gran industria y aplicaciones bastante complejas, hoy en día hasta cierto punto hay plantas en las cuales se ha dotado a los PLC´s de técnica de control de red neural es decir que los controladores aprenden de los eventos para que puedan tomar la decisión más acertada ante cualquier eventualidad.

Obsérvese el cuadro 2.1, en donde se aprecia la comparación que existe entre las diferentes gamas de controladores en este caso para el fabricante "Siemens", quien provee soluciones en automatización y control para casi cualquier aplicación industrial, médica, aeroespacial, naval, etc. Y obsérvese también que varían desde los controladores más simples hasta los controladores más complejos y sofisticados como el S7-400, que puede ser empleado hoy en día en la industria como controlador que hace posible que aprendan de los eventos que en la planta se presenten.

2.4 Microcontrolador.

Un microcontrolador [10] es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora:

- ✓ Unidad central de procesamiento.
- ✓ Memoria.
- ✓ Periféricos de entrada y salida.

Son diseñados para reducir costos y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control de un electrodoméstico sencillo como una batidora utilizará un procesador muy pequeño (4 u 8 bits), en cambio, un reproductor de música y/o vídeo digital (mp3 o mp4) requerirá de un procesador de 32 bits o de 64 bits y de uno o más Códecs de señal digital (audio y/o vídeo), el control de un sistema de frenos ABS (Antilock Brake System, por sus siglas en ingles) se basa normalmente en un microcontrolador de 16 bits, al igual que el sistema de control electrónico del motor en un automóvil.

Los microcontroladores representan la inmensa mayoría de los chips de computadoras vendidos, un 50 % son controladores "simples" y los restantes corresponden a DSP´s (Procesadores digitales de señales) más especializados, por otro lado por ejemplo pueden encontrarse en casi cualquier dispositivo electrónico aplicado a automóviles, lavadoras, hornos microondas, teléfonos celulares, etc.

Por ejemplo, un microcontrolador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria de acceso aleatorio y/o ROM/EPROM/EEPROM/FLASH, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos programas de control y un cristal de sincronización, los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidores analógicos a digital, temporizadores, UART´s y buses de interfaz serie especializados, como I^2C y CAN, frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados.

CUADRO 2.1, cuadro comparativo entre dispositivos de control desde la gama baja hasta la gama alta

INDUSTRIA	APLICACION	CONTROLADOR
CASA/HOGAR	CONTROL DE ILUMINACION, APERTURA DE PUERTAS, TEMPERATURA.	MICRO-PLC, LOGO, FABRICANTE: SIEME 
EDIFICIOS/CENTROS COMERCIALES/INDUSTRIA DE ALIMENTOS, PAPELERA, ETC. DE MEDIANA CAPACIDAD	CONTROL DE BOMBAS DE GUA, ILUMINACION, FLUJO, TEMPERATURA PROCESAMIENTO DE CHOCOLATE, PAPEL, ETC.	MINI-PLC, S7-1200, FABRICANTE: SIEMENS 
EDIFICIOS SUPER-COMPLEJOS, GAS CRIOGENICO, GAS NATURAL, PETROLEO	CONTROL DE VARIABLES DE GRUPO ELECTROGENO, CALDERO, AGUA HELADA, CONTROL DE PRESION, FLUJO DE GAS NATURAL VEHICULAR.	PLC DE GAMA MEDIA, S7-300, FABRICANTE: SIEMENS 
CEMENTERA, GASIFERA, PETROLERA, QUIMICA	CONTROL DE VARIABLES COMPLEJAS, PRODUCCION DE CEMENTO, REFINACION DE GAS, PETROLEO Y REACCIONES QUIMICAS	PLC DE GAMA ALTA, S7-400, FABRICANTE: SIEMENS 

Por ejemplo en la imagen (FIG. 2.2) se puede apreciar toda la gama de microcontroladores que provee la firma "Microchip" incluso se puede apreciar a cuantos bit´s opera cada familia de microcontroladores, se coloca como ejemplo en este informe los microcontroladores de la marca Microchip por que la implementación del sistema se ha basado en un microcontrolador de la serie 18 específicamente el microcontrolador PIC 18F67J60, para mayores detalles respecto de estas familias de microcontroladores referirse al apartado [10] de la bibliografía.

2.5 Tarjeta acondicionadora de señal.

Una tarjeta de I/O´s digitales y analógicas es una interfaz de acondicionamiento de señales de proceso hacia los puertos de entrada/salida de un microcontrolador (podría ser incluso también cualquier PLC), por ejemplo en las FIG. 2.3 y FIG. 2.4 se muestra una tarjeta de I/O´s digitales y analógicas de un registrador de variables en general, en este caso un registrador muy utilizado en la industria por ser universal, obsérvese la gran cantidad de canales con los que dispone, en la FIG. 2.5 se observa la pantalla del registrador donde se puede apreciar los canales de entrada/salida, por ejemplo unidades de tiempo, presión y voltaje.

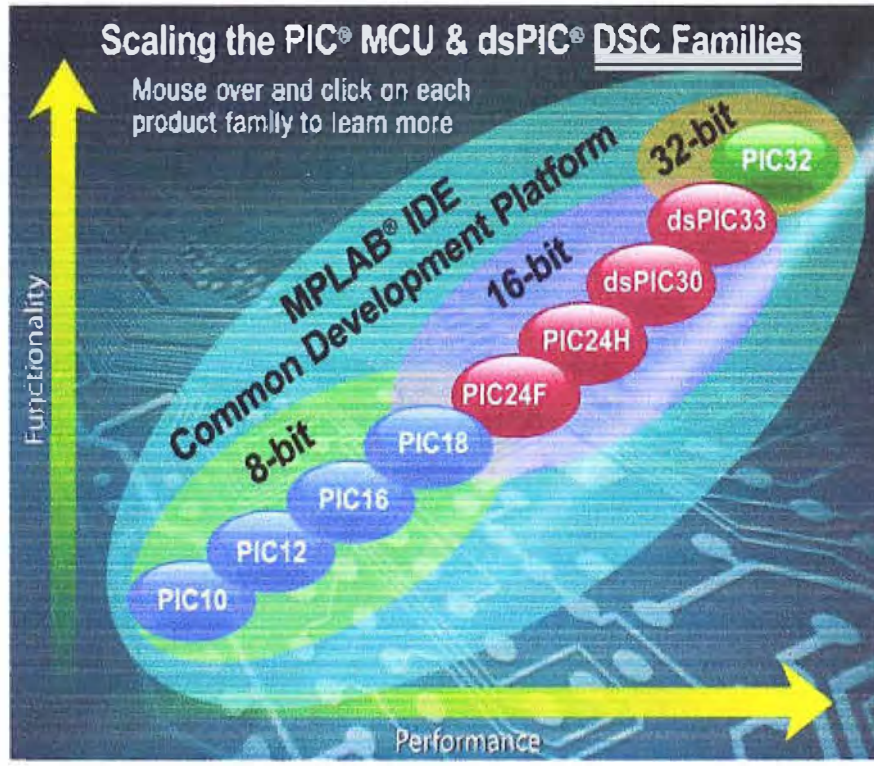


FIG. 2.2, Familia de microcontroladores del fabricante Microchip, Fuente: gráfico extraído de la página web de Fabricante Microchip [10]

2.6 Protocolos y redes de comunicación de edificios.

Se pueden definir las comunicaciones en automatización y control como el "Área de la tecnología que estudia la transmisión de información entre circuitos y sistemas electrónicos utilizados para llevar a cabo tareas de control y gestión en un proceso productivo", inicialmente en la década de 1980, las comunicaciones comenzaron a realizarse mediante comunicaciones digitales punto a punto para, posteriormente evolucionar hacia la aplicación de redes multipunto, todo este desarrollo empezó en el rubro industrial y que después se extendió al rubro domótico e inmótico.

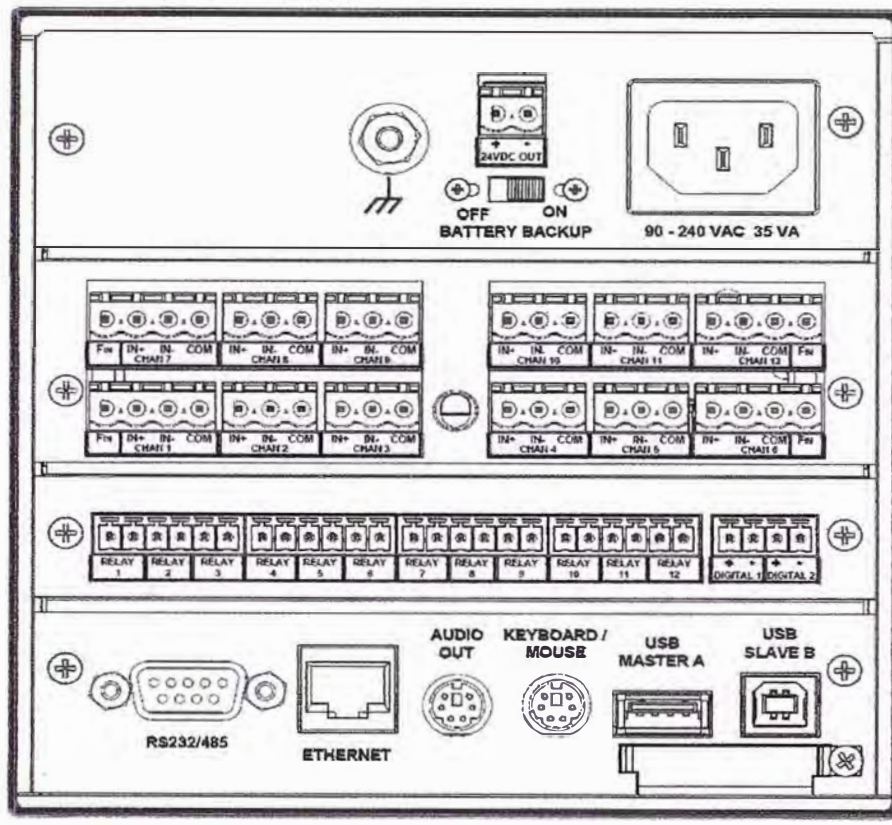


FIG. 2.3, Tarjetas acondicionadoras de señal y puertos de comunicación, Fuente: Manual Datachart 6000, fabricante: Monarch Instrument [11].

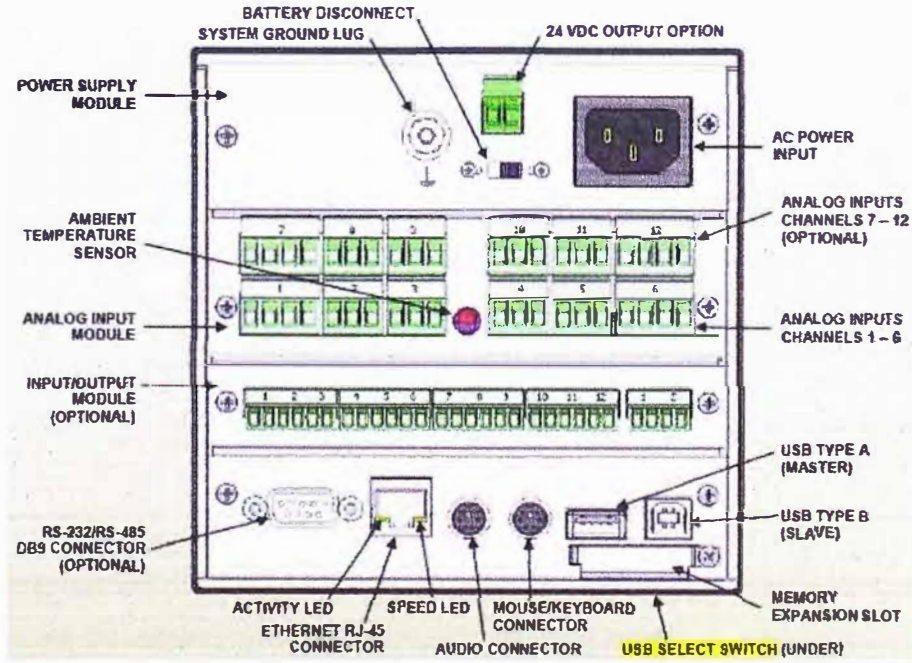


FIG. 2.4, Detalle del tipo de señales que puede aceptar la tarjeta acondicionadora de señal, Fuente: Manual Datachart 6000, fabricante: Monarch Instrument [11].

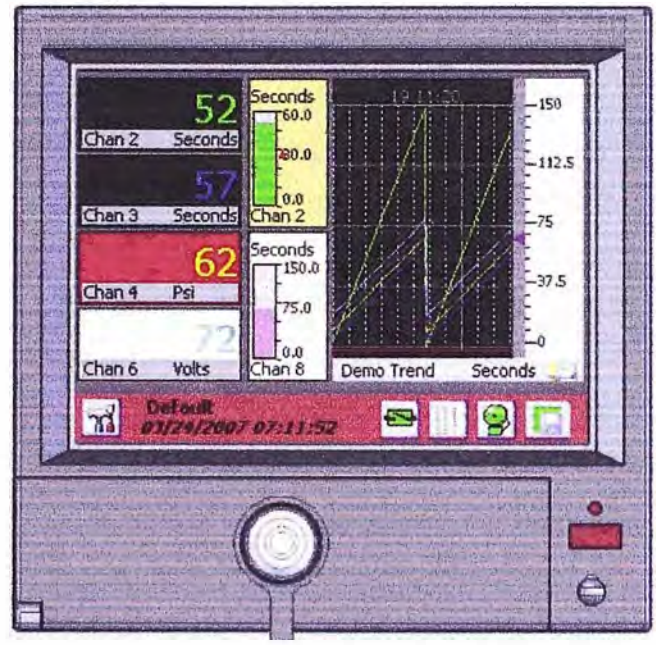


FIG. 2.5, Cuantificación y visualización de las señales que ingresan por los puertos de la tarjeta acondicionadora de señal, por los diferentes canales (en esta imagen se aprecia la presión, el tiempo y voltaje), Fuente: Manual Datachart 6000, fabricante: Monarch Instrument [11].

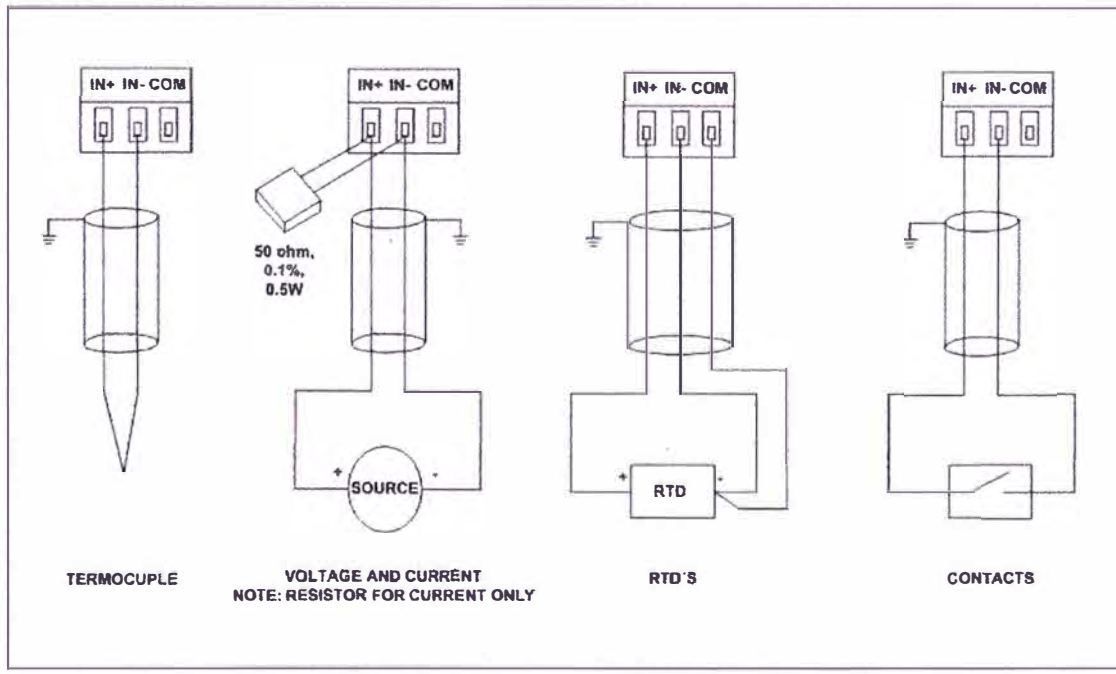


FIG. 2.6, Diferentes formas de conexiones y tipos de señales que puede aceptar la tarjeta acondicionadora de señales (por ejemplo en este caso, corriente, resistencia y contactos secos), Fuente: Manual Datachart 6000, fabricante: Monarch Instrument [11].

Conforme se iba evolucionando en el tiempo estas redes fueron expandiéndose así como los protocolos de comunicación que se desarrollaron fueron según la aplicación que se

tenía, por ejemplo en la industria existen interfaz con protocolos especializados para operar en dicha área sin embargo estos no necesariamente pueden ser aplicados en el rubro de las casas y edificios.

Conforme se fue evolucionando en el tiempo estas redes fueron expandiéndose así como los protocolos de comunicación que se desarrollaron fueron según la aplicación que se tenía, por ejemplo en la industria existen interfaz con protocolos especializados para operar en dicha área sin embargo estos no necesariamente pueden ser aplicados en el rubro de las casas y edificios.

En las siguientes imágenes se puede apreciar la evolución de las redes industriales en estas últimas décadas, obsérvese cuan beneficioso en costo y tiempo es la utilización de un bus de campo, hoy muchos de los integradores de sistemas de automatización o proveedores de servicios emplean estos sistemas como parte fundamental en la ejecución de sus proyectos.

Por otro lado estas redes son muy importantes porque permiten y para la aplicación que se tenga obtener mejores rendimientos con respecto a velocidad y capacidad de transferencia de datos o por ejemplo en cuanto a seguridad de las comunicaciones en atmósferas explosivas (ambiente industrial, pero también existen áreas clasificadas en una edificación, por ejemplo en la zona donde se almacena el combustible para alimentar a los grupos electrógenos), todas estas consideraciones se tienen que tener en cuenta al momento de seleccionar o elegir un sistema de comunicación.

Un bus de campo es un sistema de transmisión de información que simplifica enormemente cualquier instalación o sistema que se quiera automatizar.

Por otro lado si bien es cierto en la industria existen una gran cantidad de protocolos como por ejemplo Profibus, Fieldbus, Hart, Asi-interface, etc. En lo que respecta a este informe solamente tomaremos en cuenta los protocolos más empleados en la industria de automatización de casas y edificios (domótica e inmótica) tales como Bacnet, Lonkwork, Modbus, KNX y los diferentes protocolos que soporta la red Ethernet.

2.6.1 Protocolo Modbus.

Es un protocolo de comunicación ampliamente difundido, la designación Modbus Modicon corresponde a una marca registrada por Gould Inc. Como en tantos otros casos, la designación no corresponde propiamente al estándar de red, incluyendo todos los aspectos desde el nivel físico hasta el de aplicación, sino a un protocolo de enlace (nivel OSI 2). Puede, por tanto, implementarse con diversos tipos de conexión física y cada fabricante suele suministrar un software de aplicación propio, que permite parametrizar sus productos. El medio físico de conexión puede ser un bus semi-duplex (half duplex) (RS-485 o fibra óptica) o duplex (full duplex), (RS-422, BC 0-20mA o fibra óptica).

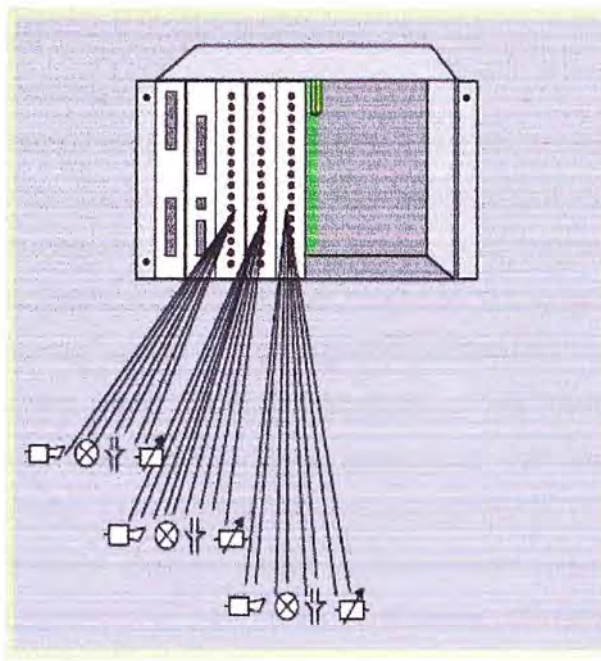


FIG 2.7, Instalación de hace 20 años

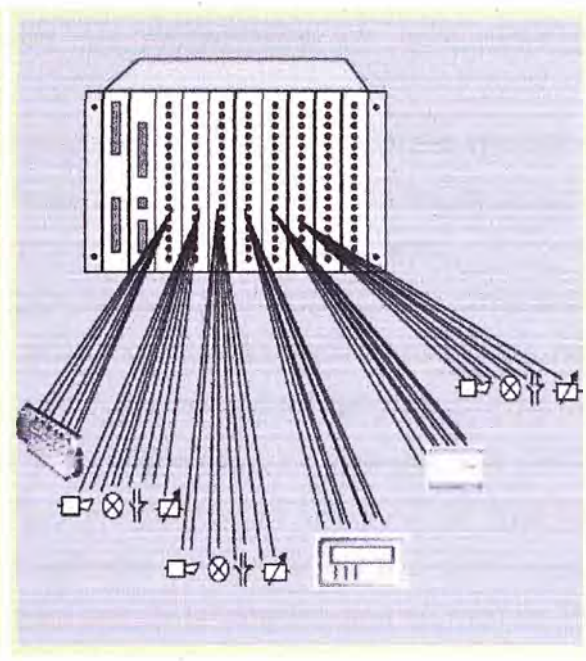


FIG 2.8, Instalación de hace diez años

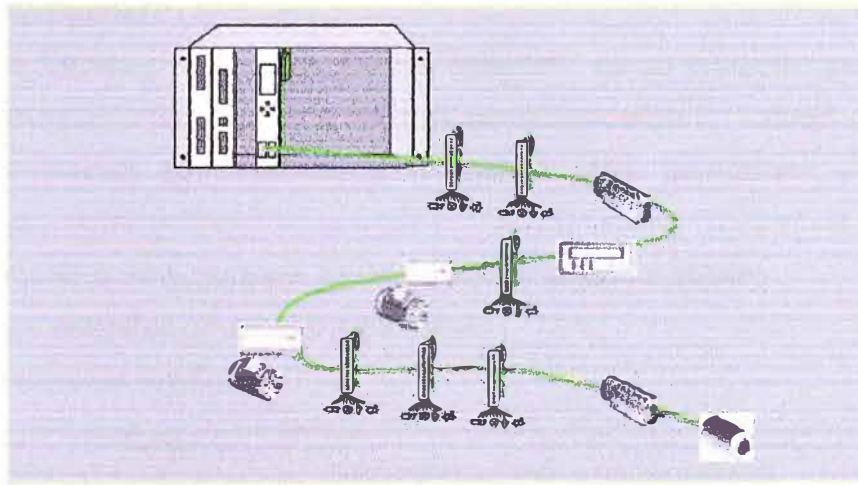


FIG 2.9, Instalación con red de comunicación, en la actualidad.

Fuente: gráficos extraídos de la guía de automatización industrial del dpto. De ingeniería de sistemas y automática de la Universidad de Vigo - España.

La comunicación es asíncrona y las velocidades de transmisión previstas van desde los 75 baudios a 19.200 baudios.

La máxima distancia entre estaciones depende del nivel físico, pudiendo alcanzar hasta 1200 m sin repetidores.

La estructura lógica es del tipo maestro-esclavo, con acceso al medio controlado por el maestro. El número máximo de estaciones previsto es de 63 esclavos más una estación maestra.

Los intercambios de mensajes pueden ser de dos tipos:

Intercambios punto a punto, que soportan siempre dos mensajes: una demanda del maestro y una respuesta del esclavo (puede ser simplemente un reconocimiento «acknowledge»).

Mensajes difundidos. Estos consisten en una comunicación unidireccional del maestro a todos los esclavos. Este tipo de mensajes no tiene respuesta por parte de los esclavos y se suelen emplear para mandar datos comunes de configuración, reinicio, etc.

La codificación de datos dentro de la trama puede hacerse en modo ASCII o puramente binario, según el estándar RTU (Remote Transmission Unit). En cualquiera de los dos casos, cada mensaje obedece a una trama que contiene cuatro campos principales, según se muestra en la figura 2.11. La única diferencia estriba en que la trama ASCII incluye un carácter de encabezamiento (3AH) y los caracteres CR y LF al final del mensaje. Pueden existir también diferencias en la forma de calcular el CRC, puesto que el formato RTU emplea una fórmula polinómica en vez de la simple suma en módulo 16. Con independencia de estos pequeños detalles, a continuación se da una breve descripción de cada uno de los campos del mensaje, obsérvese la imagen (FIG. 2.11).

2.6.2 Protocolo Bacnet.

Abreviación de "Building Automation Control Network", es un protocolo padrón definido por ANSI/ASHRAE/ISO Standard 135-2004. El protocolo define un modelo de sistema de automatización predial, que describe la interacción entre dispositivos y sistemas, el protocolo define:

- ✓ Datos y comandos estructurados en un modelo orientado a objeto;
- ✓ Servicios que describen el acceso a los datos;
- ✓ Una arquitectura de red flexible.

: (3AH)	Nº Esclavo (00-3F$_H$)	Código de Operación	Subfunciones, Datos	LRC(16) H L	CR (0D$_H$)	LF (0A$_H$)

Codificación ASCII

Nº Esclavo (00-3F$_H$)	Código de Operación	Subfunciones, Datos	CRC(P16) H L

Codificación RTU

FIG 2.11, Campos de la trama del mensaje tanto en ASCII como en RTU, Fuente: Dpto. de Tecnología electrónica Universidad Politécnica de Cartagena [4]:

El padrón BACnet define seis tipos de redes de comunicación para transporte de mensajes BACnet, como presenta la FIG 2.15. El tipo de red define la capa física y de enlace. Los seis tipos de redes son:

- ➢ BACnet ARCnet.

- BACnet Ethernet.
- BACnet Lontalk.
- BACnet MS/TP.
- BACnet IP.

Un equipo BACnet posee una colección de informaciones definida como objetos y propiedades.

Un objeto BACnet representa una información física o virtual del equipo, como una entrada o salida digital o analógica, variables de control y parámetros. La norma BACnet define 25 tipos de objetos. Cada objeto es identificado por una propiedad llamada Identificador de objeto que codifica la instancia y el tipo del objeto en un número binario de 32 bits.

Una propiedad BACnet representa características o informaciones de un objeto BACnet. Es a través de las propiedades que los otros elementos pueden acceder a las informaciones del equipo. El acceso a la propiedad puede ser definido como solamente lectura o escrita/lectura. La especificación BACnet define servicios que son agrupados en cinco categorías:

- ✓ Acceso a objetos.
- ✓ Gestión del equipo.
- ✓ Alarmas y eventos.
- ✓ Transferencia de archivos.
- ✓ Terminal virtual.

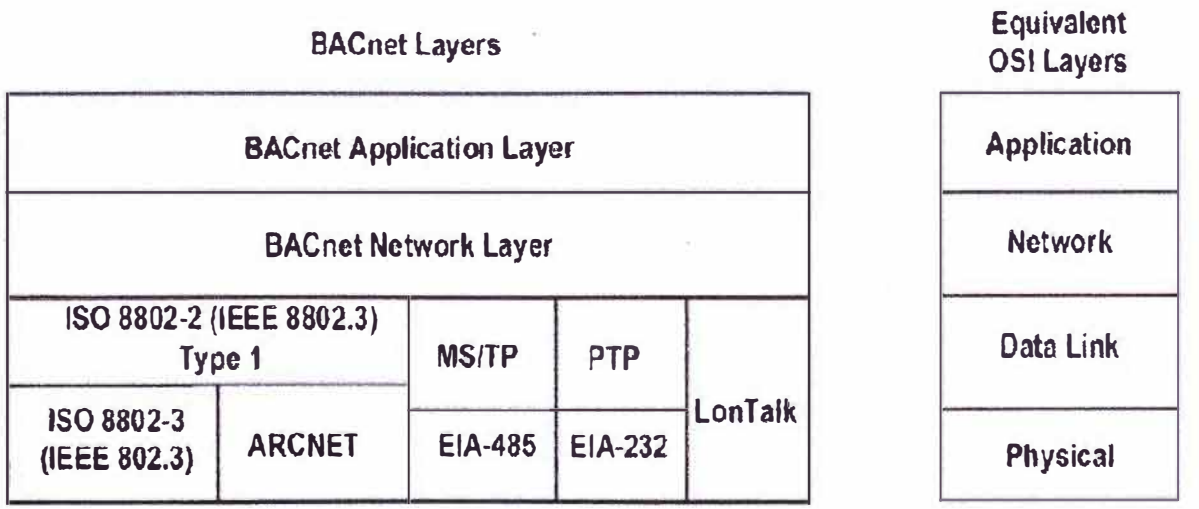


FIG 2.12, diferentes capas protocolo BACnet, Fuente: gráficos extraídos del estándar internacional de protocolos de comunicación BACnet

Conforme el conjunto de servicios ofrecidos en el equipo se puede clasificar los dispositivos BACnet en seis diferentes perfiles:

- ✓ BACnet Operator Workstation (B-OWS).
- ✓ BACnet Building Controller (B-BC).
- ✓ BACnet Advanced Application Controller (B-AAC).
- ✓ BACnet Application Specific Controller (B-ASC).
- ✓ BACnet Smart Actuator (B-AS).
- ✓ BACnet Smart Sensor (B-SS).

2.6.3 Protocolo Lonwork.

Es una plataforma de control creada por la compañía norteamericana Echelon. Las redes Lonworks describen de una manera efectiva una solución completa a los problemas de sistemas de control. Al igual que la industria informática, la industria del control fue creada, y en muchos casos todavía lo es, basada en soluciones centralizadas de control punto-a-punto. Esto significa que existe un "maestro" o controlador principal similar a un ordenador, físicamente cableado a cada punto de control particular, como actuadores o sensores, denominados "esclavos". El resultado final es funcional, pero es caro y difícil para mantener, ampliar y gestionar. Igualmente, es menos fiable frente a fallos, ya que la caída del controlador principal provoca la caída de todo el sistema. El comienzo de las redes Lonworks se basó en conceptos muy simples: 1) los sistemas de control son fundamentalmente idénticos, independientemente de la aplicación final; 2) un sistema de control distribuido es significativamente más potente, flexible, y ampliable que un sistema de control centralizado; y 3) la empresas ahorran más dinero a largo plazo instalando redes distribuidas que instalando redes centralizadas. La tecnología Lonworks proporciona una solución a los múltiples problemas de diseño, construcción, instalación, y mantenimiento de redes de control; redes que pueden variar en tamaño desde dos a 32,000 dispositivos y se pueden usar en cualquier aplicación desde supermercados a plantas petrolíferas, desde aviones hasta ferrocarriles, etc .

Actualmente, en casi todas las industrias hay una tendencia a evitar los sistemas propietarios o los esquemas de control basados en sistemas centralizados. Los fabricantes están utilizando sistemas abiertos, chips estándar, sistemas operativos estándar y componentes para construir productos que mejoren la flexibilidad, el costo del sistema y su instalación. La tecnología Lonworks está acelerando la tendencia a evitar los sistemas propietarios o los sistemas centralizados, proporcionando interoperabilidad, una tecnología robusta, desarrollos más rápidos y ahorro económico.

Más de 4000 empresas utilizan las redes Lonworks hoy en dia, y el número está creciendo rápidamente. Todas las áreas del campo de control están plenamente cubiertas por productos compatibles con Lonworks incluyendo sistemas de detección de

incendios, sistemas de climatización, sistemas de seguridad, sistemas de gestión de energía, sistemas de alumbrado, etc.

2.6.4 Protocolo KNX

"KNX Association" es el creador y propietario de la tecnología KNX, el único estándar abierto para todas las aplicaciones de control de la vivienda y el edificio, como por ejemplo el control de la iluminación y las persianas, así como variados sistemas de seguridad, calefacción, ventilación, aire acondicionado, monitorización, alarma, control de agua, gestión de energía, contador, así como electrodomésticos del hogar, audio/video y mucho más. Esta tecnología puede ser usada tanto en viviendas y edificios modernos.

Para los miembros de la "KNX Association" el sistema está a su disposición gratis, además puede ser implementado sobre cualquier plataforma de micropocesadores.

KNX está aprobado como:

- ✓ Estándar Europeo (CENELEC EN 50090 y CEN EN 13321-1).
- ✓ Estándar Internacional (ISO/IEC 14543-3).
- ✓ Estándar Chino (GB/Z 20965).
- ✓ Estándar Norteamericano (ANSI/ASHRAE 135).

Este estándar se fundamenta en más de 20 años de experiencia en el mercado, que incluye a sus predecesores EIB, EHS y BatiBUS.

Por otro lado KNX soporta diferentes medios de comunicación como por ejemplo:

- ✓ **Par trenzado (KNX TP),** KNX es transmitido a través de un cable bus separado, con una estructura jerarquizada en líneas y áreas.
- ✓ **Corrientes portadoras (KNX PL),** KNX es transmitido sobre la red eléctrica existente.
- ✓ **Radio frecuencia (KNX RF),** KNX es transmitido por señales de radio. Estos dispositivos pueden ser unidireccionales o bidireccionales.
- ✓ **IP/Ethernet (IP KNX),** Este conocido medio de comunicación puede ser usado en conjunto con las especificaciones "KNXnet/IP", que permiten enviar tramas KNX encapsuladas en tramas IP.

2.7 La red Ethernet

También conocido como estándar IEEE 802.3, es un estándar de transmisión de datos para redes de área local que se basa en el siguiente principio:

Todos los equipos en una red Ethernet están conectados a la misma línea de comunicación compuesta por cables cilíndricos.

Se distinguen diferentes variantes de tecnología Ethernet respecto del medio físico, en la tabla siguiente (cuadro 2.1) se puede apreciar las diferentes tecnologías existentes así como la respectiva velocidad a la que pueden transferir los datos, también se aprecia la distancia máxima a la cual se puede conectar un dispositivo desde un router o switch.

CUADRO 2.1, Diferentes tecnologías de medio de comunicación en Ethernet según el estándar IEEE 802.3.

Tecnología	Velocidad de transmisión	Tipo de cable	Distancia máxima	Topología
10Base2	10 Mbps	Coaxial	185 m	Bus (Conector T)
10BaseT	10 Mbps	Par Trenzado	100 m	Estrella (Hub o Switch)
10BaseF	10 Mbps	Fibra óptica	2000 m	Estrella (Hub o Switch)
100BaseT4	100Mbps	Par Trenzado (categoría 3UTP)	100 m	Estrella. Half Duplex (hub) y Full Duplex (switch)
100BaseTX	100Mbps	Par Trenzado (categoría 5UTP)	100 m	Estrella. Half Duplex (hub) y Full Duplex (switch)
100BaseFX	100Mbps	Fibra óptica	2000 m	No permite el uso de hubs
1000BaseT	1000Mbps	4 pares trenzado (categoría 5e ó 6UTP)	100 m	Estrella. Full Duplex (switch)
1000BaseSX	1000Mbps	Fibra óptica (multimodo)	550 m	Estrella. Full Duplex (switch)
1000BaseLX	1000Mbps	Fibra óptica (monomodo)	5000 m	Estrella. Full Duplex (switch)

El protocolo TCP/IP, se relaciona automáticamente como el protocolo sobre el que funciona la red Internet. Esto, en cierta forma es cierto, ya que se le llama TCP/IP, a la familia de protocolos que nos permite estar conectados a la red Internet.

El protocolo TCP, funciona en el nivel de transporte del modelo de referencia OSI, proporcionando un transporte fiable de datos. El protocolo IP, funciona en el nivel de red del modelo OSI, que nos permite encaminar nuestros datos hacia otras maquinas, pero un protocolo de comunicaciones debe solucionar una serie de problemas relacionados con la comunicación entre computadoras, además de los que proporciona los protocolos TCP e IP.

Internet no es un nuevo tipo de red física, sino un conjunto de tecnologías que permiten interconectar redes muy distintas entre sí. Internet no es dependiente de la máquina ni del sistema operativo utilizado. De esta manera, podemos transmitir información entre un servidor Unix y una computadora que utilice Windows 98. O entre plataformas completamente distintas como Macintosh, Alpha o Intel. Es más, entre una máquina y otra generalmente existirán redes distintas: redes Ethernet, redes Token Ring e incluso enlaces vía satélite. Como vemos, está claro que no podemos utilizar ningún protocolo que dependa de una arquitectura en particular. Lo que estamos buscando es un método de interconexión general que sea válido para cualquier plataforma, sistema operativo y

tipo de red. La familia de protocolos que se eligieron para permitir que Internet sea una Red de redes es TCP/IP. Nótese aquí que hablamos de familia de protocolos ya que son muchos los protocolos que la integran, aunque en ocasiones para simplificar hablemos sencillamente del protocolo TCP/IP.

El protocolo TCP/IP, tiene que estar a un nivel superior del tipo de red empleado y funcionar de forma transparente en cualquier tipo de red. Y a un nivel inferior de los programas de aplicación (páginas WEB, correo electrónico, etc) particulares de cada sistema operativo. Todo esto nos sugiere el siguiente modelo de referencia ver (FIG. 2.16), donde se aprecia la capa de aplicación por ejemplo HTTP, SMTP, FTP, para el caso de este informe se empleará HTTP, en la capa de transporte se tienen UDP y TCP, para el caso de este informe se empleará "TCP" y en la capa de red se tienen "IP".

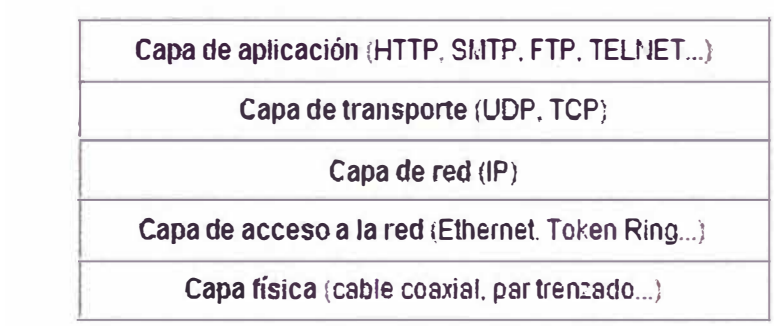


FIG 2.13, Diferentes protocolos según modelo TCP/IP, Fuente: Modelo OSI TCP/IP.

En la imagen (FIG. 2.14) se indica el modelo de referencia TCP/IP y el modelo que tiene implementado la pila de microchip correspondiente a la tarjeta PIC WEB, que se ha utilizado en este proyecto.

2.7.1 Protocolo TCP

(Transmission Control Protocol, protocolo de control de transmisión), el protocolo TCP es orientado a conexión, es necesario establecer una conexión previa entre las dos máquinas antes de poder transmitir algún dato. A través de esta conexión los datos llegarán siempre a la aplicación destino de forma ordenada y sin duplicados. Finalmente, es necesario cerrar la conexión.

El protocolo TCP permite una comunicación confiable entre dos aplicaciones. De esta forma, las aplicaciones que lo utilicen no tienen que preocuparse de la integridad de la información, dan por hecho que todo lo que reciben es correcto.

El flujo de datos entre una aplicación y otra viajan por un circuito virtual. Sabemos que los datagramas IP pueden seguir rutas distintas, dependiendo del estado de los enrutadores intermedios, para llegar a un mismo sitio. Esto significa que los datagramas IP que transportan los mensajes siguen rutas diferentes aunque el protocolo TCP logre la ilusión de que existe un único circuito por el que viajan todos los bytes uno detrás de otro (algo

así como una tubería entre el origen y el destino). Para que esta comunicación pueda ser posible es necesario abrir previamente una conexión. Esta conexión garantiza que todos los datos lleguen correctamente de forma ordenada y sin duplicados. La unidad de datos del protocolo es el byte, de tal forma que la aplicación origen envía bytes y la aplicación destino recibe estos bytes.

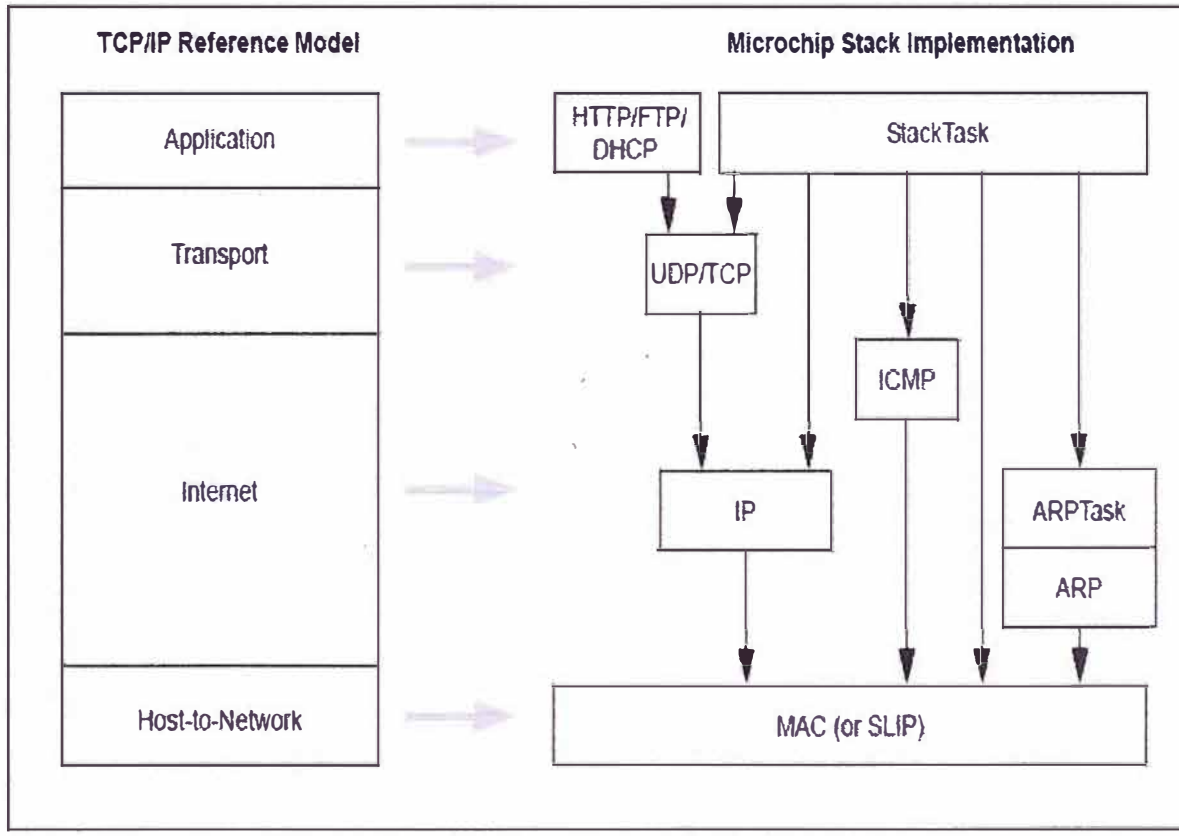


FIG 2.14, Equivalencia entre el modelo TCP/IP y el modelo de la pila implementada en la PIC WEB, Fuente: Datasheet PIC WEB [9].

Sin embargo, cada byte no se envía inmediatamente después de ser generado por la aplicación, sino que se espera a que haya una cierta cantidad de bytes, se agrupan en un segmento y se envía el segmento completo. Para ello son necesarias unas memorias intermedias o buffers. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP. Si el segmento es muy grande será necesario fragmentar el datagrama, con la consiguiente pérdida de rendimiento y si es muy pequeño, se estarán enviando más cabeceras que datos. Por consiguiente, es importante elegir el mayor tamaño de segmento posible que no provoque fragmentación.

El protocolo TCP envía un flujo de información no estructurado. Esto significa que los datos no tienen ningún formato, son únicamente los bytes que una aplicación envía a otra. Ambas aplicaciones deberán ponerse de acuerdo para comprender la información que se están enviando.

Cada vez que se abre una conexión, se crea un canal de comunicación bidireccional en el que ambas aplicaciones pueden enviar y recibir información, es decir, una conexión es full-duplex.

Cada vez que llega un mensaje se devuelve una confirmación (acknowledgement) para que el emisor sepa que ha llegado correctamente. Si no le llega esta confirmación pasado un cierto tiempo, el emisor reenvía el mensaje.

Veamos a continuación la manera más sencilla (aunque ineficiente) de proporcionar una comunicación confiable. El emisor envía un dato, arranca su temporizador y espera su confirmación (ACK). Si recibe su ACK antes de agotar el temporizador, envía el siguiente dato. Si se agota el temporizador antes de recibir el ACK, reenvía el mensaje.

Los siguientes esquemas representan este comportamiento (FIG. 2.15).

Esta técnica (FIG. 2.15) es perfectamente válido aunque muy ineficiente debido a que sólo se utiliza un sentido de la comunicación a la vez y el canal está desaprovechado la mayor parte del tiempo. Para solucionar este problema se utiliza un protocolo de ventana deslizante, que se resume en la siguiente imagen (FIG. 2.16). Los mensajes y las confirmaciones van numerados y el emisor puede enviar más de un mensaje antes de haber recibido todas las confirmaciones anteriores.

Una conexión son dos pares dirección IP puerto. No puede haber dos conexiones iguales en un mismo instante en toda la red. Aunque bien es posible que un mismo ordenador tenga dos conexiones distintas y simultáneas utilizando un mismo puerto. El protocolo TCP utiliza el concepto de conexión para identificar las transmisiones. En el siguiente ejemplo (FIG. 2.17) se han creado tres conexiones. Las dos primeras son al mismo servidor web (puerto 80) y la tercera a un servidor de FTP (puerto 21).

Las estaciones clientes salen por los puertos asignados como son el puerto 1256, 1305 y 1323 con las direcciones IP debidamente asignadas según su configuración. Ya hemos comentado que el flujo de bytes que produce una determinada aplicación se divide en uno o más segmentos TCP para su transmisión. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP. Para facilitar el control de flujo de la información los bytes de la aplicación se numeran, de esta manera, cada segmento indica en su cabecera el primer byte que transporta. Las confirmaciones o acuses de recibo (ACK) representan el siguiente byte que se espera recibir (y no el número de segmento recibido, ya que éste no existe) en la siguiente figura (FIG. 2.18) se aprecia una trama TCP.

- **Puerto fuente** (16 bits). Puerto de la máquina origen. Al igual que el puerto destino es necesario para identificar la conexión actual.
- **Puerto destino** (16 bits). Puerto de la máquina destino.

- **Número de secuencia** (32 bits). Indica el número de secuencia del primer byte que trasporta el segmento.
- **Número de acuse de recibo** (32 bits). Indica el número de secuencia del siguiente byte que se espera recibir. Con este campo se indica al otro extremo de la conexión que los bytes anteriores se han recibido correctamente.

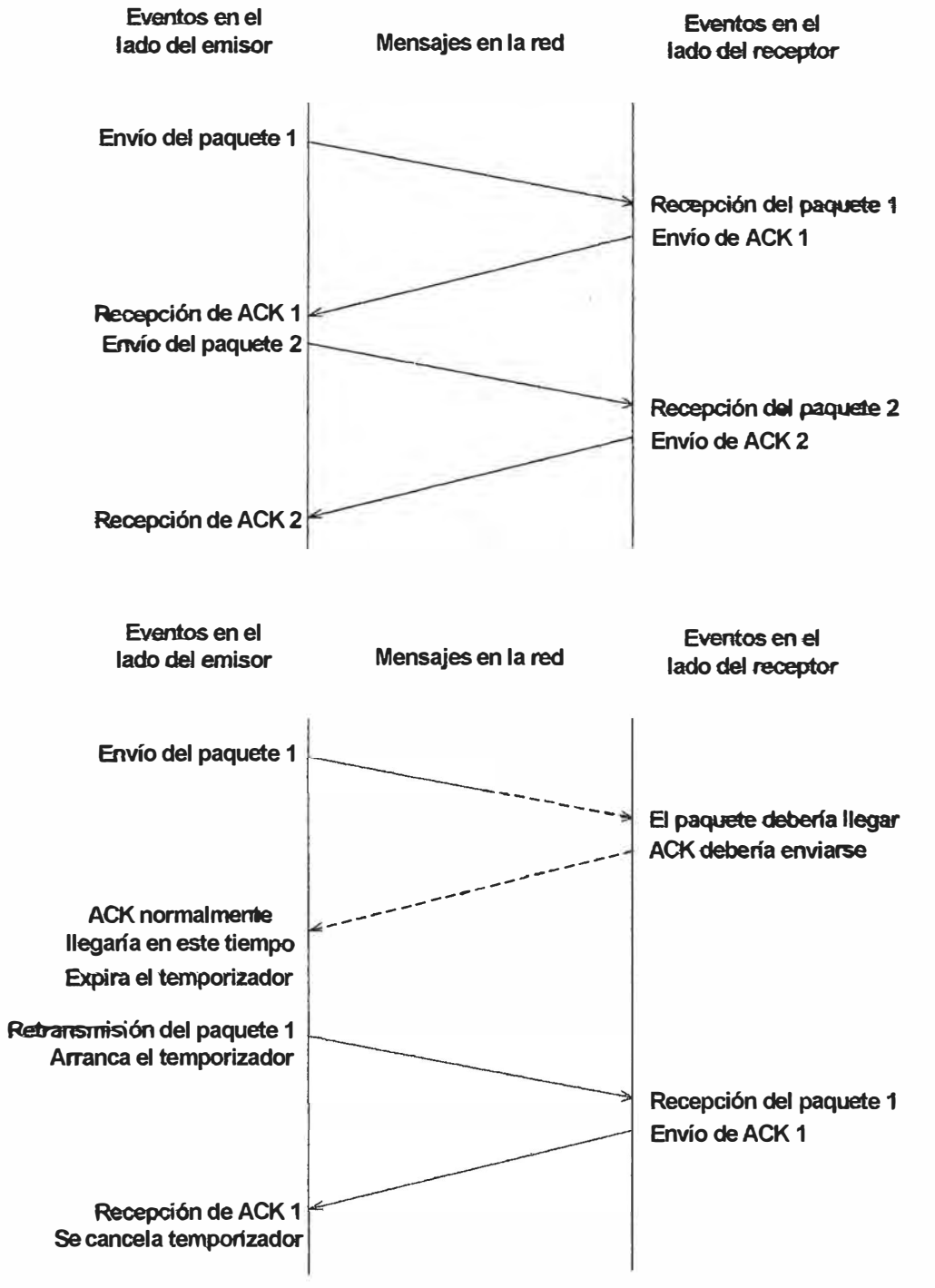


FIG 2.15, Esquema de envío de paquetes, pérdida de paquetes y reenvío de paquetes, Fuente: Libro "TCP/IP entorno windows 2000, Philippe Mathon.

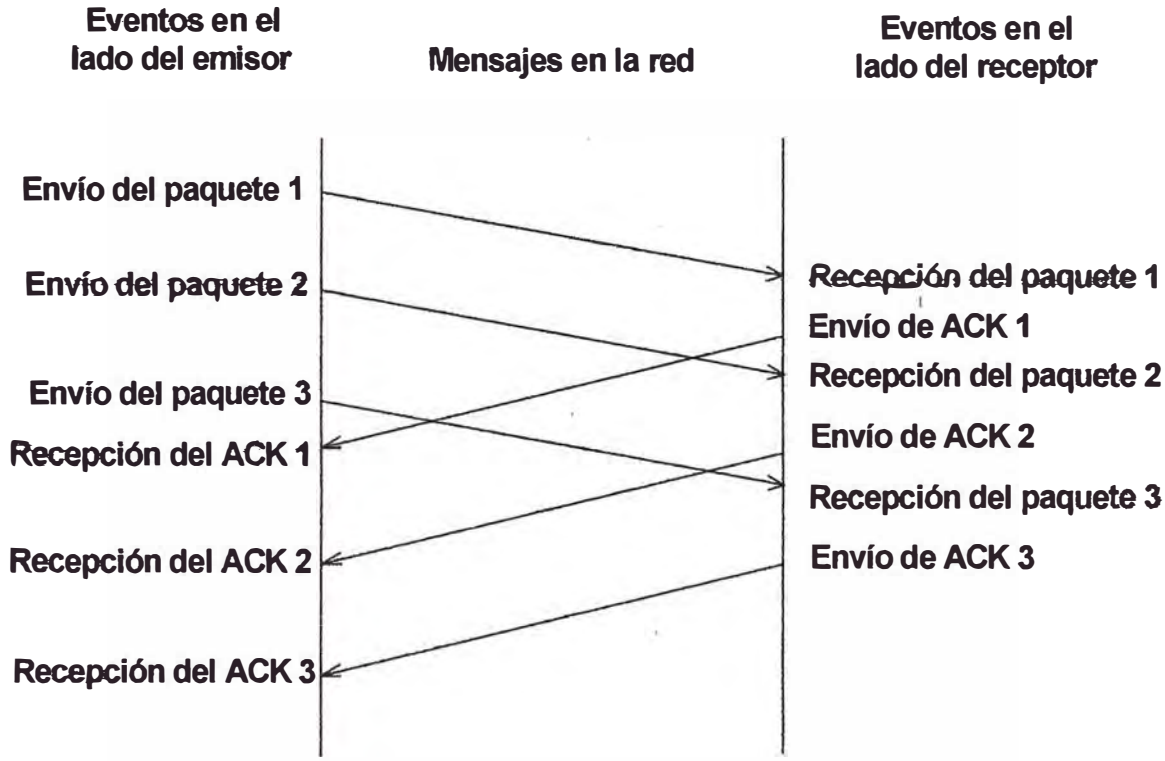


FIG 2.16, técnica de envío de paquetes mucho más eficiente, Fuente: Libro "TCP/IP entorno windows 2000, Philippe Mathon.

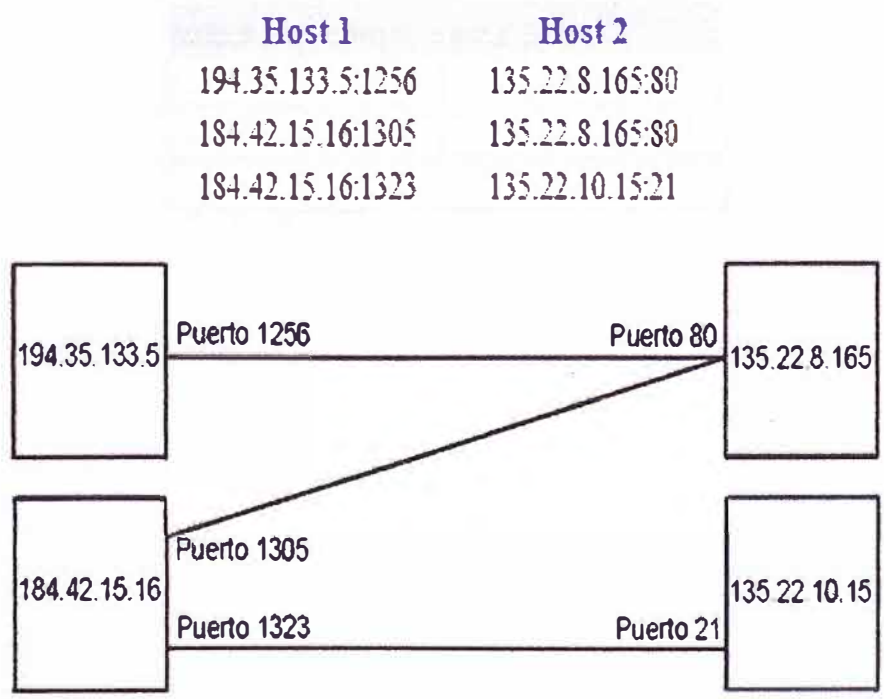


FIG 2.17, Ejemplo de conexiones y puertos asignado, Fuente: Libro "TCP/IP entorno Windows 2000, Philippe Mathon.

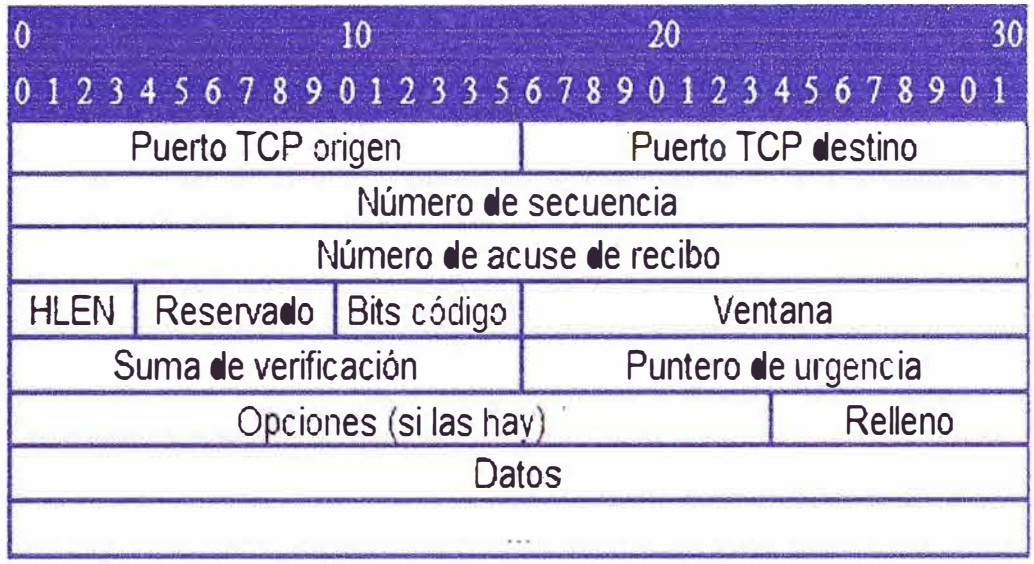


FIG 2.18, Conformación de una trama TCP, Fuente: Libro "TCP/IP entorno Windows 2000, Philippe Mathon.

- **HLEN** (4 bits). Longitud de la cabecera medida en múltiplos de 32 bits (4 bytes). El valor mínimo de este campo es 5, que corresponde a un segmento sin datos (20 bytes).
- **Reservado** (6 bits). Bits reservados para un posible uso futuro.
- **Bits de código** o indicadores (6 bits). Los bits de código determinan el propósito y contenido del segmento. A continuación se explica el significado de cada uno de estos bits (mostrados de izquierda a derecha) si está a 1.
- **URG,** el campo puntero de urgencia contiene información válida.
- **ACK,** el campo Número de acuse de recibo contiene información válida, es decir, el segmento actual lleva un ACK. Observemos que un mismo segmento puede transportar los datos de un sentido y las confirmaciones del otro sentido de la comunicación.
- **PSH,** la aplicación ha solicitado una operación push (enviar los datos existentes en la memoria temporal sin esperar a completar el segmento).
- **RST,** interrupción de la conexión actual.
- **SYN,** parámetro de sincronización de los números de secuencia en el cual se transmitirá. Se utiliza al crear una conexión para indicar al otro extremo cual va a ser el primer número de secuencia con el que va a comenzar a transmitir (veremos que no tiene porqué ser el cero).
- **FIN,** indica al otro extremo que la aplicación ya no tiene más datos para enviar. Se utiliza para solicitar el cierre de la conexión actual.
- **Ventana** (16 bits), número de bytes que el emisor del segmento está dispuesto a aceptar por parte del destino.

- **Suma de verificación** (24 bits), suma de comprobación de errores del segmento actual. Para su cálculo se utiliza una pseudo-cabecera que también incluye las direcciones IP origen y destino.
- **Puntero de urgencia** (8 bits), se utiliza cuando se están enviando datos urgentes que tienen preferencia sobre todos los demás e indica el siguiente byte del campo Datos que sigue a los datos urgentes. Esto le permite al destino identificar donde terminan los datos urgentes. Nótese que un mismo segmento puede contener tanto datos urgentes (al principio) como normales (después de los urgentes).
- **Opciones,** si está presente únicamente se define una opción: el tamaño máximo de segmento que será aceptado.
- **Relleno,** se utiliza para que la longitud de la cabecera sea múltiplo de 32 bits.
- **Datos.** Información que envía la aplicación.

2.7.2 Protocolo UDP

(User Datagram Protocol, protocolo de datagrama de usuario) proporciona una comunicación muy sencilla entre las aplicaciones de dos ordenadores. Al igual que el protocolo IP, UDP es:

- No orientado a conexión, No se establece una conexión previa con el otro extremo para transmitir un mensaje UDP. Los mensajes se envían solamente y éstos pueden duplicarse o llegar desordenados al destino.
- No fiable, Los mensajes UDP se pueden perder o llegar dañados.
- UDP utiliza el protocolo IP para transportar sus mensajes. Como vemos, no añade ninguna mejora en la calidad de la transferencia; aunque sí incorpora los puertos origen y destino en su formato de mensaje. Las aplicaciones (y no el protocolo UDP) deberán programarse teniendo en cuenta que la información puede no llegar de forma correcta.

- **Puerto UDP de origen** (16 bits, opcional). Número de puerto de la máquina origen.
- **Puerto UDP de destino** (16 bits). Número de puerto de la máquina destino que se encuentra en la red.
- **Longitud del mensaje UDP** (16 bits). Especifica la longitud medida en bytes del mensaje UDP incluyendo la cabecera. La longitud mínima es de 8 bytes.
- **Suma de verificación UDP** (16 bits, opcional). Suma de comprobación de errores del mensaje. Para su cálculo se utiliza una pseudo-cabecera que también incluye las direcciones IP origen y destino. Para conocer estos datos, el protocolo UDP debe interactuar con el protocolo IP.

- **Datos**. Aquí viajan los datos que se envían las aplicaciones. Los mismos datos que envía la aplicación origen son recibidos por la aplicación destino después de atravesar toda la red.

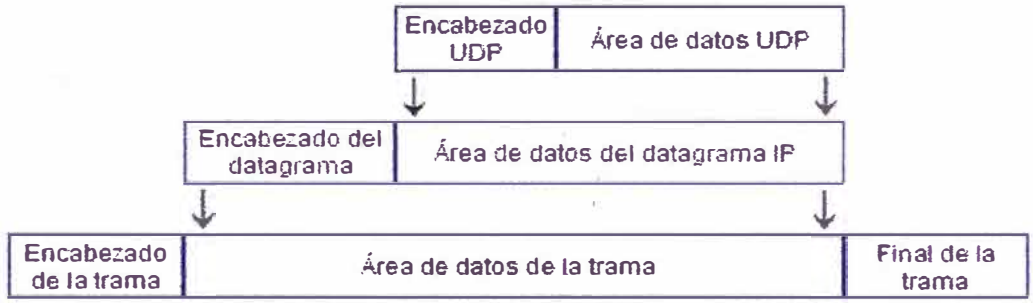


FIG 2.20, Formato de mensajes UDP, Fuente: Libro "TCP/IP entorno Windows 2000, Philippe Mathon.

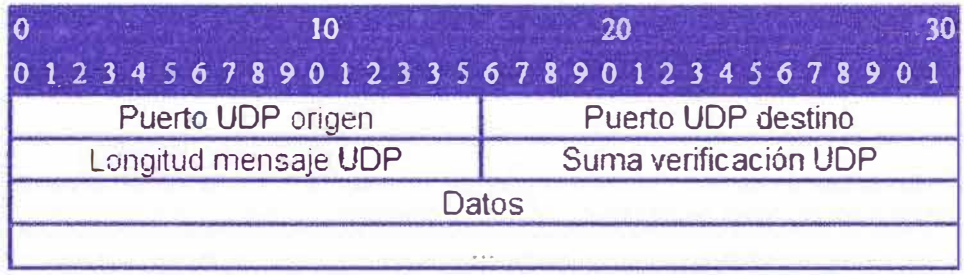


FIG 2.21, Conformación de una trama UDP, Fuente: Libro "TCP/IP entorno Windows 2000, Philippe Mathon.

En el cuadro 2.2, se aprecia las características principales de los dos protocolos mencionados TCP y UDP.

CUADRO 2.2, Características principales entre protocolos de comunicación en la capa de transporte.

Características del protocolo	TCP	UDP
Crea una conexión	Si	No
Transferencia confiable de los datos	Si	No
Control de congestion	Si	No
Delimitación de límites en los mensajes	No	Si
Fragmentación e integración de la información	Si	No
Multiplexación de información del paquete	Si	No
Envío de datos desordenado	No	Si

2.7.3 Protocolo IP

Es parte de la capa de Internet del conjunto de protocolos TCP/IP. Es uno de los protocolos de Internet más importantes ya que permite el desarrollo y transporte de datagramas IP (paquetes de datos), aunque sin garantizar su "entrega". En realidad, el protocolo IP procesa datagramas IP de manera independiente al definir su representación, ruta y envío.

El protocolo IP determina el destinatario del mensaje mediante 3 campos:

✓ El campo de dirección IP.

✓ El campo de máscara de subred, una máscara de subred le permite al protocolo IP establecer la parte de la dirección IP que se relaciona con la red.

✓ El campo de pasarela predeterminada, le permite al protocolo de Internet saber a qué equipo enviar un datagrama, los datos circulan en Internet en forma de datagramas (también conocidos como paquetes). Los datagramas son datos encapsulados, es decir, datos a los que se les agrega un encabezado que contiene información sobre su transporte (como la dirección IP de destino). Los ruteadores analizan (y eventualmente modifican) los datos contenidos en un datagrama para que puedan transitar.

A continuación en la imagen (FIG. 2.19) se indica cómo es un datagrama IP.

✓ **Versión** (4 bits): es la versión del protocolo IP que se está utilizando (actualmente se utiliza la versión IPv4) para verificar la validez del datagrama. Está codificado en 4 bits.

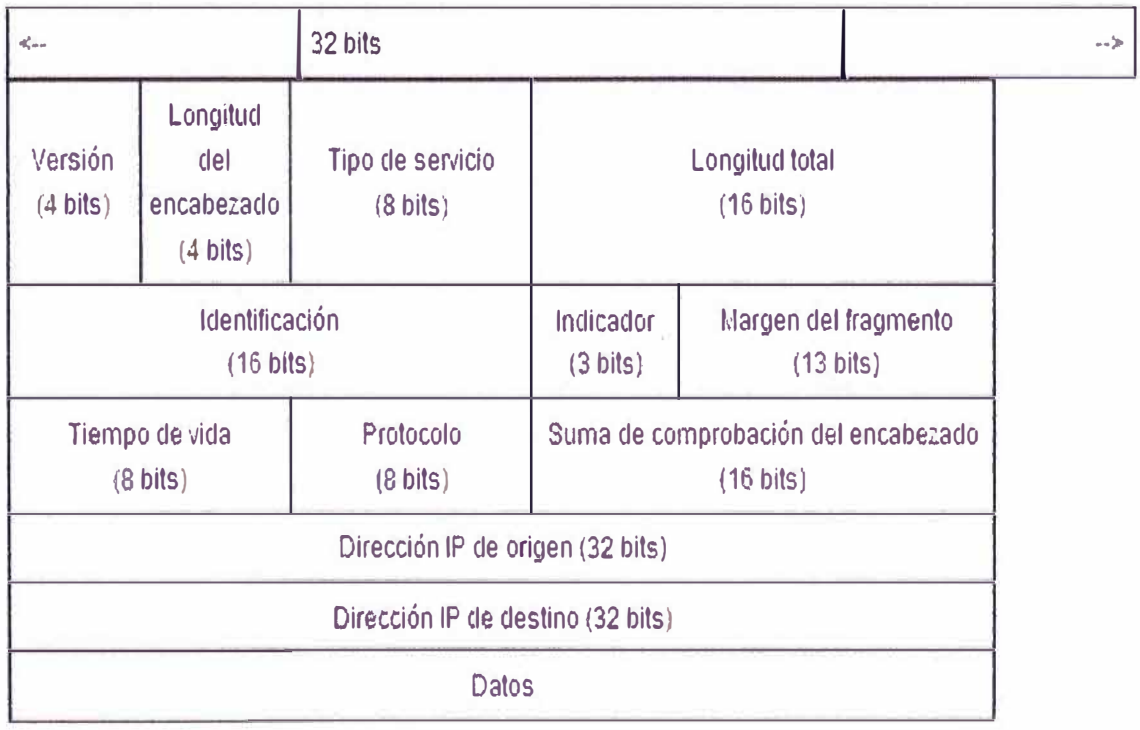


FIG 2.19, Datagrama IP, Fuente: Imágenes extraídas del libro "TCP/IP entorno Windows 2000, Philippe Mathon.

✓ **Longitud del encabezado** o IHL por Internet Header Length (Longitud del encabezado de Internet) (4 bits): es la cantidad de palabras de 32 bits que componen el encabezado (Importante: el valor mínimo es 5). Este campo está codificado en 4 bits.

✓ **Tipo de servicio** (8 bits): indica la forma en la que se debe procesar el datagrama.

✓ **Longitud total** (16 bits): indica el tamaño total del datagrama en bytes. El tamaño de este campo es de 2 bytes, por lo tanto el tamaño total del datagrama no puede exceder los 65536 bytes. Si se lo utiliza junto con el tamaño del encabezado, este campo permite determinar dónde se encuentran los datos.

✓ **Identificación, indicadores y margen del fragmento** son campos que permiten la fragmentación de datagramas. Esto se explica a continuación.

✓ **TTL** o **Tiempo de vida** (8 bits): este campo especifica el número máximo de ruteadores por los que puede pasar un datagrama. Por lo tanto, este campo disminuye con cada paso por un ruteador y cuando alcanza el valor crítico de 0, el ruteador destruye el datagrama. Esto evita que la red se sobrecargue de datagramas perdidos.

✓ **Protocolo** (8 bits): este campo, en notación decimal, permite saber de qué protocolo proviene el datagrama.

- ICMP 1
- IGMP: 2
- TCP: 6
- UDP: 17

✓ **Suma de comprobación del encabezado (16 bits)**: este campo contiene un valor codificado en 16 bits que permite controlar la integridad del encabezado para establecer si se ha modificado durante la transmisión. La suma de comprobación es la suma de todas las palabras de 16 bits del encabezado (se excluye el campo suma de comprobación). Esto se realiza de tal modo que cuando se suman los campos de encabezado (suma de comprobación inclusive), se obtenga un número con todos los bits en 1.

✓ **Dirección IP de origen** (32 bits): Este campo representa la dirección IP del equipo remitente y permite que el destinatario responda.

✓ **Dirección IP de destino** (32 bits): dirección IP del destinatario del mensaje.

2.7.4 Protocolo HTTP

Protocolo de Transferencia de HiperTexto es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP/1.0 está recogida en el RFC 1945. Fue

propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores. HTTP se basa en sencillas operaciones de solicitud/respuesta.

Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

2.7.5 Protocolo FTP.

Protocolo de transferencia de archivos (en inglés File Transfer Protocol).

2.7.6 Protocolo SMTP.

Protocolo de transferencia de correo (en inglés Simple Mail Transfer Protocol).

2.8 Sistema de monitoreo y control remoto (SCADA).

SCADA es un acrónimo de Supervisory Control And Data Acquisition (control y adquisición de datos de supervisión). Los sistemas SCADA utilizan la computadora y tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales y edificaciones. Estos sistemas son parte integral de la mayoría de los ambientes industriales complejos o muy geográficamente dispersos ya que pueden recoger la información de una gran cantidad de fuentes muy rápidamente, y la presentan a un operador en una forma amigable. Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas.

Los primeros sistemas automatizados SCADA fueron altamente modificados con programas de aplicación específicos para atender a requisitos de algún proyecto particular. Como ingenieros de varias industrias asistieron al diseño de estos sistemas, su percepción de SCADA adquirió las características de su propia industria. Proveedores de sistemas de software SCADA, deseando reutilizar su trabajo previo sobre los nuevos proyectos, perpetuaron esta imagen de industria-específicos por su propia visión de los ambientes de control con los cuales tenían experiencia. Solamente cuando nuevos proyectos requirieron funciones y aplicaciones adicionales, hizo que los desarrolladores

de sistemas SCADA tuvieran la oportunidad de desarrollar experiencia en otras industrias.

2.9 Software libre.

Software Libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software, é aquí algunas plataformas de software libres:

- ✓ Netbeans, para desarrollo de aplicaciones en java.
- ✓ Apache, para implementación de servidores.
- ✓ Tomcat, para aplicación en servidores.
- ✓ MySQL, para aplicaciones de base de datos.

2.9.1 Plataforma de desarrollo Java.

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de "Sun Microsystems", capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual (JVM) encargada de la ejecución de las aplicaciones, y un conjunto de bibliotecas estándar que ofrecen una funcionalidad común.

La plataforma es así llamada la plataforma Java e incluye:

- ✓ Plataforma Java, Edición Estándar (Java Platform, Standard Edition), o Java SE (antes J2SE).
- ✓ Plataforma Java, Edición Empresa (Java Platform, Enterprise Edition), o Java EE (antes J2EE).
- ✓ Plataforma Java, Edición Micro (Java Platform, Micro Edition), o Java ME (antes J2ME).

El entorno de desarrollo, donde se realiza toda la programación en java es "NetBeans", esto entorno de desarrollo es soportado por "Sun Microsystems".

2.9.2 Servidor web.

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados. En este punto es necesario aclarar lo siguiente: mientras que comúnmente se utiliza la palabra servidor para referirnos a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones descritas.

Gracias a los avances en conectividad y la gran disponibilidad de banda ancha, hoy en día es muy común establecer los servidores web dentro de la propia empresa, sin tener

que recurrir a caros alojamientos en proveedores externos. Esto es posible gracias a Apache, uno de los mejores y el más utilizado entre los servidores web que existen. Apache ha construido una gran reputación entre los servidores web gracias a su gran estabilidad, confiabilidad y el gran aporte del grupo de voluntarios que planean y desarrollan todo lo relativo a esta plataforma, desde la documentación hasta el mismo código en sí.

Entre las ventajas que presenta un servidor como "Apache" se encuentran las siguientes:

✓ Es personalizable, la arquitectura modular de "Apache" permite construir un servidor hecho a la medida.

✓ Permite la implementación de los últimos y más nuevos protocolos.

✓ En cuanto a la administración los archivos de configuración de "Apache" están en ASCII, por lo que tiene un formato simple, y pueden ser editados tan solo con un editor de texto. Estos son transferibles, lo que permite la clonación efectiva de un servidor. El servidor puede ser administrado vía línea de comandos, lo que hace la administración remota muy conveniente.

✓ Por otra parte se trata de un servidor muy eficiente. Mucho esfuerzo se ha puesto en optimizar el rendimiento del código "C" de "Apache". Como resultado, este corre rápido y consume menos recursos de sistema en comparación a otros servidores. Además, Apache corre en una amplia variedad de sistemas operativos, incluyendo varias versiones de UNIX, Windows9x/NT, MacOS (Sobre Power PC), y varios otros.

El soporte de Apache es provisto por "The Apache Group" o "La Fundación Apache".

2.9.3 Aplicaciones web.

Las aplicaciones web, son parte importante para la visualización de páginas web que nos permiten brindar dinamismo, dichas aplicaciones pueden realizarse en PHP (Hypertext Pre-processor [8]) que es un software libre y es compatible con el servidor "Apache".

2.9.4 Servidor de base de datos.

Una base de datos es una colección de información organizada de forma que un programa de computadora pueda seleccionar rápidamente los fragmentos de datos que necesite. Una base de datos es un sistema de archivos electrónico.

Las bases de datos tradicionales se organizan por campos, registros y archivos. Un campo es una pieza única de información; un registro es un sistema completo de campos; y un archivo es una colección de registros. Por ejemplo, una guía de teléfono es análoga a un archivo. Contiene una lista de registros, cada uno de los cuales consiste en tres campos: nombre, dirección, y número de teléfono.

CAPÍTULO III
HARDWARE Y SOFTWARE IMPLEMENTADO PARA SATISFACER LA NECESIDAD

3.1 Arquitectura de monitoreo y control del sistema implementado.

El punto más crítico en esta implementación fue la elección de la arquitectura de monitoreo y control en forma remota de la variable de proceso, esta elección se basó en las siguientes consideraciones, en la FIG. 3.5 se aprecia la arquitectura empleada:

✓ El monitoreo y control de la variable de proceso debía ser visualizada y controlada (control manual) desde cualquier dispositivo electrónico tales como computadoras de escritorio, Laptop´s, Notebook´s, teléfonos celulares con tecnología 3G, PC´s tipo tabletas y cualquier dispositivo que pueda soportar una conexión a internet.

✓ Se debía emplear una red de comunicación universal.

✓ La tarjeta electrónica que debía interactuar con la variable de proceso tenía que ser de bajo costo y sencilla de programar.

✓ El software y las pantallas gráficas de monitoreo y control debían desarrollarse e implementarse utilizando software libre.

3.1.1 Elección de la red y protocolos utilizada.

Partiendo de que la red mas empleada en la actualidad es la red Ethernet con una gran cantidad de dispositivos conectadas a ellas, por la tendencia de que esta red crece a una taza mucho mayor que cualquier otra red y por que los equipos portátiles como las Laptop´s, teléfonos inteligentes las PC tipo tableta soportan y pueden conectarse a dicha red con suma facilidad es que se optó por emplear Ethernet como la red principal para este proyecto.

Por otro lado si bien es cierto se había elegido la red Ethernet, ahora había que elegir el protocolo que se utilizaría para transportar nuestros datos y mostrar aplicaciones web, en este sentido se necesitaba un protocolo de enlace con la tarjeta o dispositivo de control ya que ellas cuentan con un identificador (nombre asignado identificador IP) para lograr ello se eligió el protocolo IP, luego se requería un protocolo que transportara los datos por medio de la red, dicho transporte tenía que ser orientado a conexión es decir debíamos

garantizar que los datos llegasen a su destino no importando mucho el tiempo que se demorase, para lograr ello se optó por emplear el protocolo TCP obsérvese las ventajas en el cuadro 2.2, y luego para mostrar las aplicaciones desde cualquier dispositivo conectado remotamente se eligió el protocolo HTTP.

Entonces dentro de esta plataforma de red se eligieron los siguientes protocolos:

- ✓ TCP, como protocolo de transporte.
- ✓ IP como protocolo de enlace.
- ✓ HTTP como protocolo de aplicación.

El medio físico que permitió la comunicación sin involucrar costo adicional fue la tecnología 10/100BASET con cable UTP de par trenzado categoría 5 ó 6 con velocidades de transferencia de 10/100Mbps, según como se indica en el cuadro 2.1.

3.1.2 Elección de la tarjeta de monitoreo y control (hardware).

En primer lugar la red elegida fue la red Ethernet por lo tanto se debía adquirir un equipo o tarjeta electrónica con las siguientes características:

- ✓ El equipo o tarjeta debía tener un puerto Ethernet y que soportara protocolos como TCP, IP Y HTTP.
- ✓ El equipo o tarjeta electrónica debía tener puertos de entradas/salidas digitales y entradas analógicas.
- ✓ Debía ser el de más bajo costo en el mercado ó de costo moderado.

En función de estos requerimientos, en el mercado existen controladores y PLC´s que contienen puertos de comunicación Ethernet, y puertos de entradas/salidas que cumplían parte de los requerimientos iniciales de este proyecto pero el inconveniente con este tipo de controladores es que son de un costo bastante elevado por lo que se debía elegir una tarjeta electrónica menos costosa, en el mercado se encontró el fabricante de tarjetas de control "Olimex", dichas tarjetas tienen incorporado microcontroladores tan familiares como el de "Microchip" y soportan los protocolos de comunicación requeridos, entre las diferentes tarjetas desarrolladas por "Olimex" se tienen por ejemplo las tarjetas "PIC MINI-WEB", "PIC WEB" y "PIC MAXI-WEB", en donde las características principales de dichas tarjetas son como se indica en el cuadro 3.1.

En función del cuadro 3.1 y luego de realizar una comparación entre las diferentes tarjetas se eligió la tarjeta "PIC WEB" porque tiene:

- ✓ Puerto Ethernet.
- ✓ Soporta la pila de Microchip TCP – IP (Soporta protocolos TCP, IP y HTTP).
- ✓ Puertos de entradas/salidas digitales y analógicas.
- ✓ Costo moderado/bajo.

CUADRO 3.1, Características de las diferentes tarjetas de monitoreo y control desarrolladas por el fabricante "Olimex".

COMPARACIONES TÉCNICAS ENTRE TARJETAS PARA MONITOREO Y CONTROL			
CARACTERISTICAS	PIC MINI-WEB	PIC WEB	PIC MAXI-WEB
Tipo de microconntrolador utilizado por la tarjeta	PIC 18F25J10	PIC 18F67J60	PIC18F97J60
Memoria de programa del microcntrolador	32 Kbytes	128 Kbytes	128 Kbytes
Espacio de memoria para alojar página web	1Mbit	1Mbit	1Mbit
Conector para comunicación serial	No	Si	Si
Servidor web	Si	Si	Si
Protocolos soportados	Pila Microchip TCP IP	Pila Microchip TCP IP	Pila Microchip TCP IP
Voltaje de operación	3.3 V	3.3 V	3.3 V
Cantidad de entradas digitales	3	8	15
Cantidad de salidas digitales	4	8	13
Cantidad de entradas analógicas	3	6	12
Pantalla de visualización LCD	No	No	Si
Costos	Bajo	Medio	Alto

Todas estas características cumplen con los requerimientos iniciales del proyecto.

Se descartó la tarjeta "PIC MINI WEB" porque aún teniendo un costo bastante bajo está limitado en cuanto a entradas/salidas digitales y entradas analógicas y también por el hecho de no tener un conector serial de configuración, esta tarjeta podría emplearse para aplicaciones puntuales.

Por otro lado se descartó la tarjeta "PIC MAXI WEB" por el hecho de tener un costo bastante elevado y también por tener una pantalla de visualización LCD innecesarios para esta aplicación, esta tarjeta podría ser empleada principalmente como una tarjeta de entrenamiento.

Los datos técnicos de la tarjeta "PIC WEB" se detallan en el ANEXO I y del microcontrolador en el ANEXO J.

En la imagen (FIG. 3.1), se aprecia los componentes principales de la tarjeta electrónica que se eligió, el diagrama del circuito electrónico de dicha tarjeta se encuentra detallado en el ANEXO M, que acompaña a este informe.

3.1.3 Implementación de la tarjeta acondicionadora de señal.

Según la arquitectura de monitoreo y control los sistemas que son monitoreados y controlados como sistemas de iluminación, aire acondicionado, grupo electrógeno, temperatura y nivel de cisterna de agua son entradas/salidas del tipo digitales así como entradas del tipo analógicas, entonces la tarjeta acondicionadora diseñada debe soportar los siguientes tipos de señales:

- ✓ Puertos de salida digital (DO), salida con relé.

✓ Puertos de entrada digital (DI), 2.4 - 3.5 volt como máximo en las patillas del microcontrolador (uno lógico), 0 volt como mínimo en las patillas del microcontrolador (cero lógico).

✓ Puerto de entrada de señal analógica (AI), y que por el mismo puerto alimente al sensor (transmisor pasivo) con 24VDC y a la vez obtenga señal de 4-20mA proveniente del sensor (transmisor).

El diseño de los puertos de la tarjeta acondicionadora de señal se basó principalmente en las tensiones y corrientes máximas que puede aceptar la tarjeta electrónica PIC WEB.

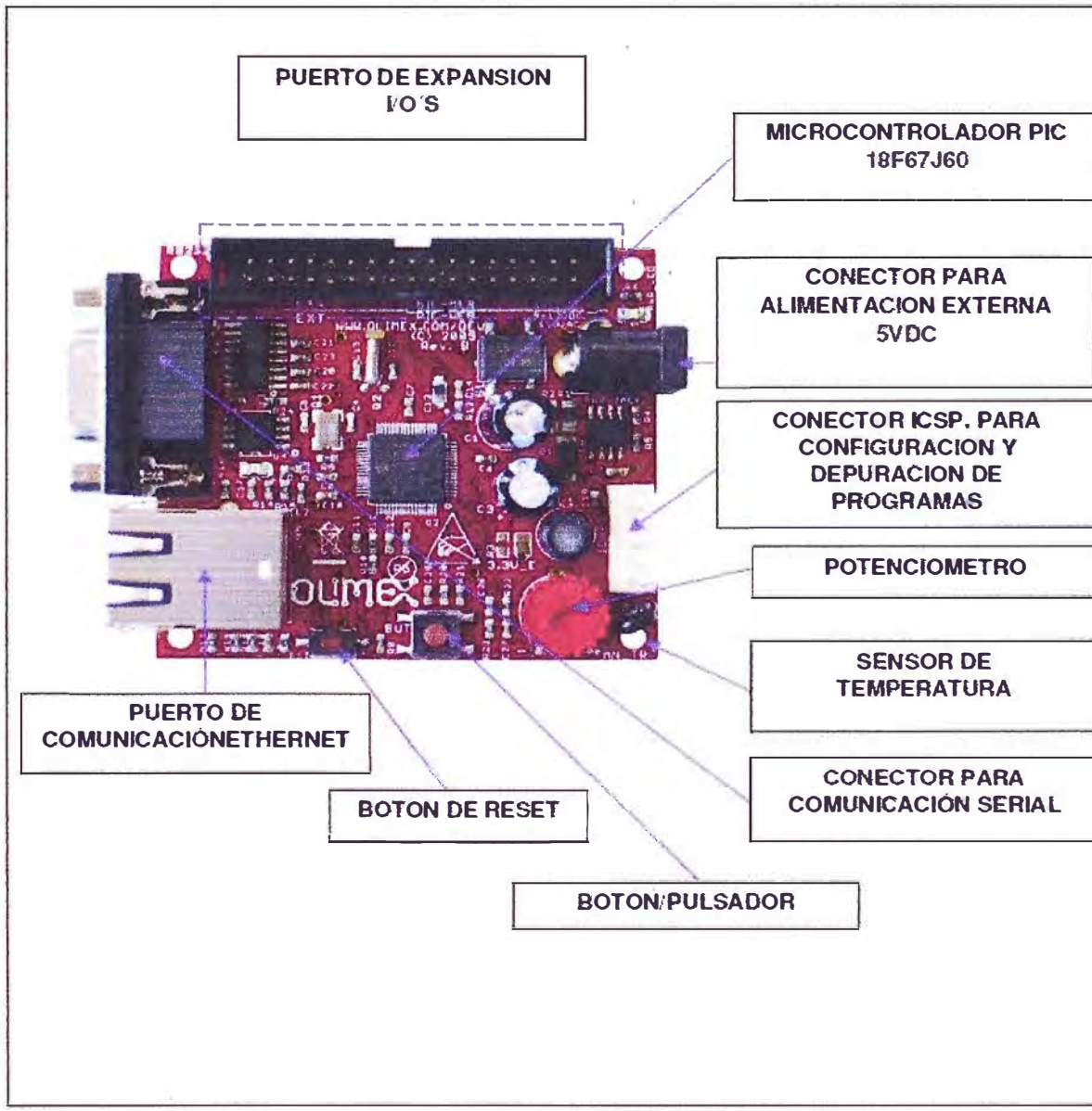


FIG 3.1, Componentes físicos a bordo de la tarjeta PIC WEB, Fuente: Especificaciones técnicas de la tarjeta PIC WEB, fabricante OLIMEX.

Según el datasheet de la tarjeta electrónica (VER ANEXO I) se tiene lo siguiente:

➢ Las entradas digitales operan con una tensión de 2.4 voltios, obsérvese la imagen (FIG. 3.2).

➢ Las salidas digitales operan con una tensión de 3.3 voltios, obsérvese la imagen (FIG. 3.3).

➢ Las entradas analógicas deben operar como máximo con una tensión de 3.3 volt, el convertidor A/D es de 10bit´s (FIG. 3.2).

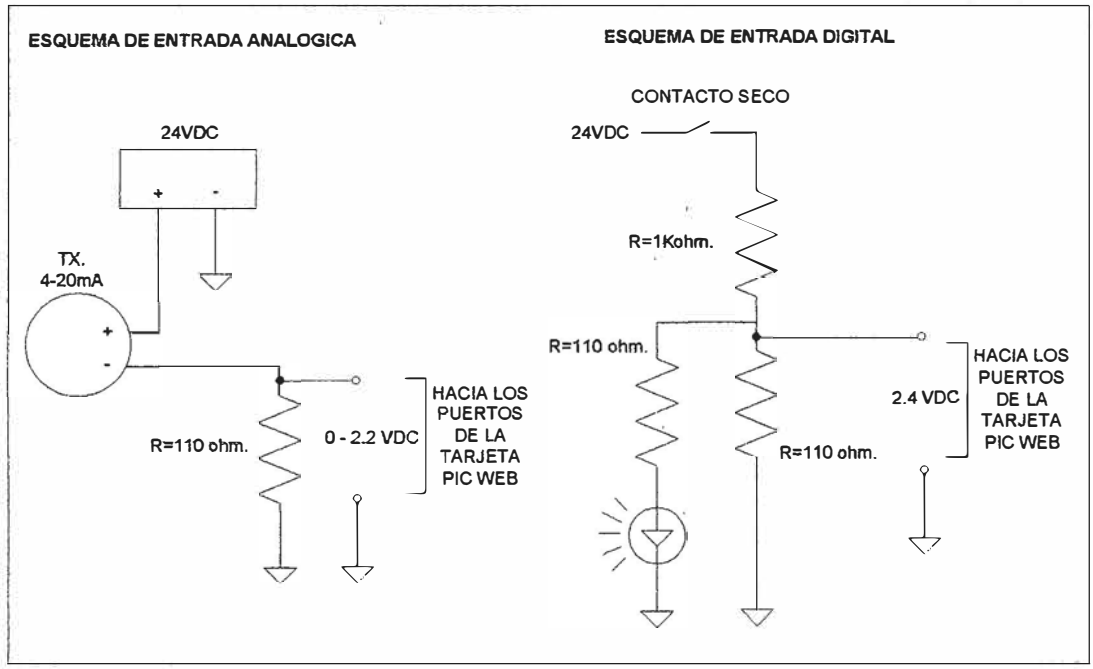


FIG 3.2, Diagrama de entradas digitales y analógicas de la tarjeta diseñada, Fuente: Archivos compañía "C y Q Ingeniería e Instrumentación S.A.C.".

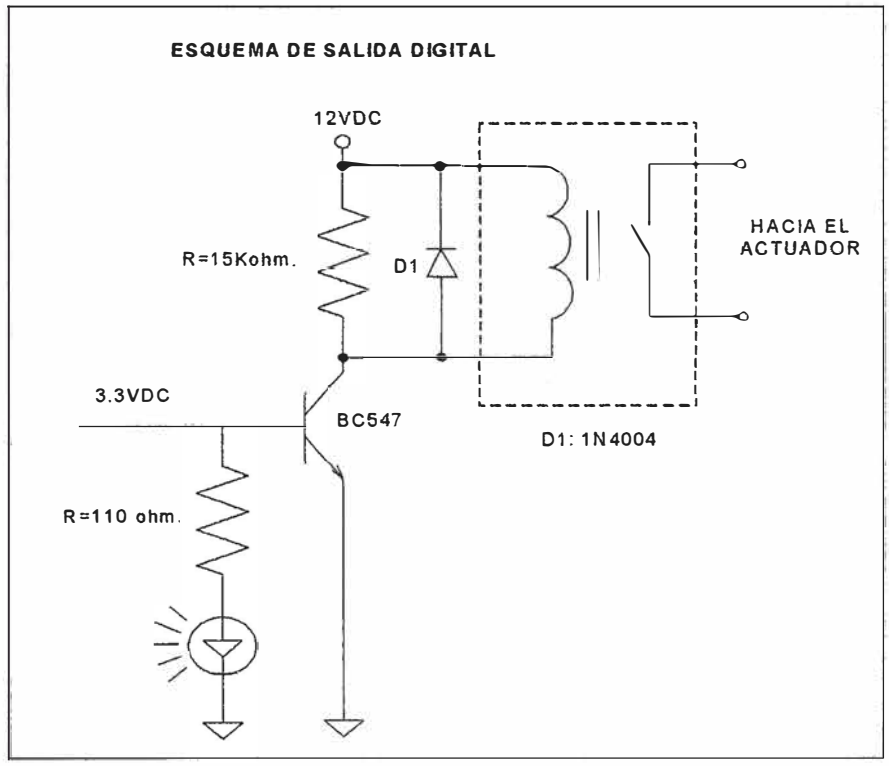


FIG 3.3, Diagrama de salida digital de la tarjeta diseñada, Fuente: Archivos de la compañía "C y Q Ingeniería e Instrumentación S.A.C.".

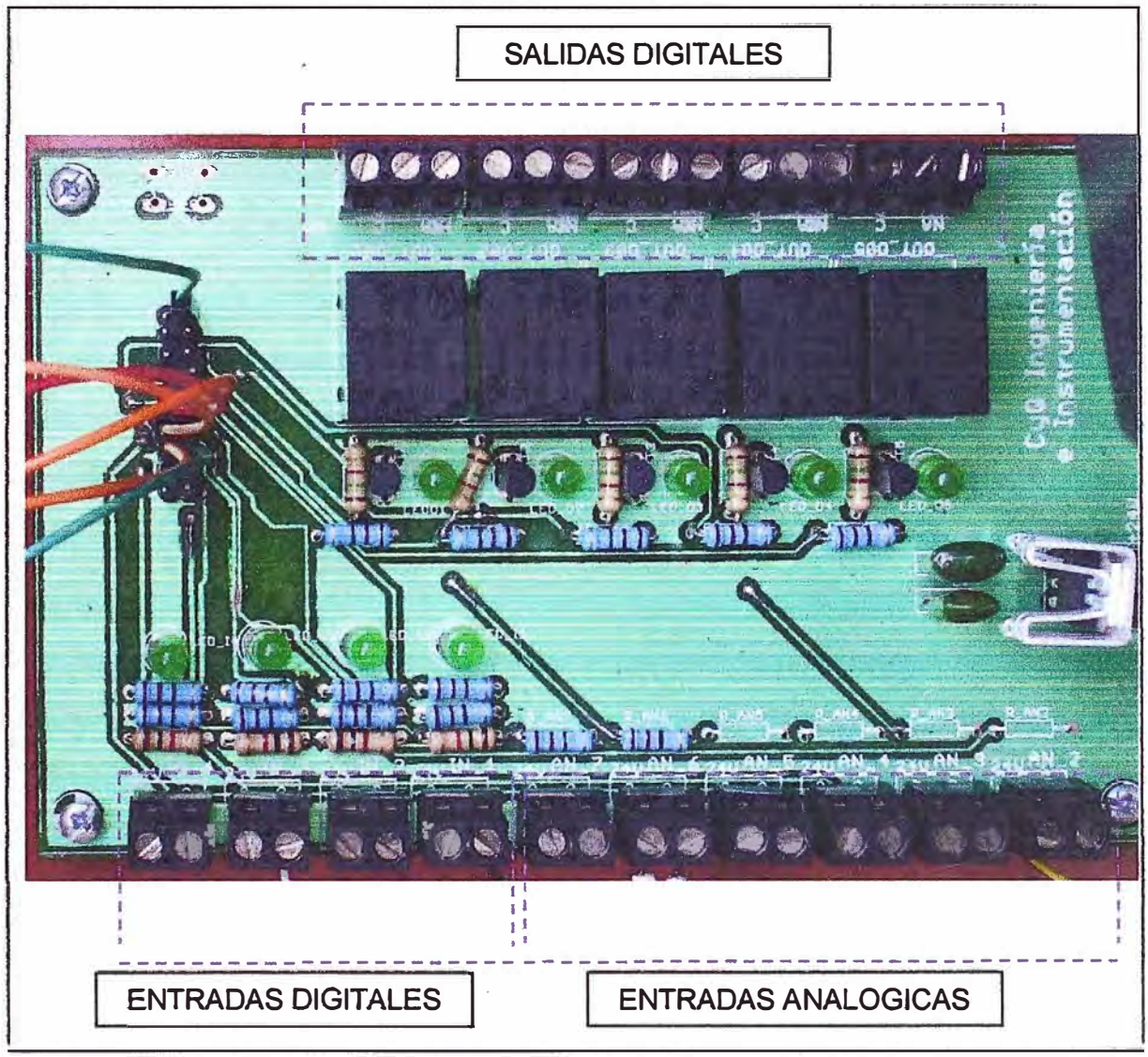


FIG 3.4, Tarjeta acondicionadora de señal, se observa puertos de entrada/salida digital y analógica, Fuente: Archivos compañía "C y Q Ingeniería e Instrumentación S.A.C.".

La tarjeta acondicionadora de señal diseñada e implementada es como la que se muestra en la imagen FIG. 3.4, esta tarjeta como se puede apreciar está dimensionada para soportar cuatro entradas digitales, seis entradas analógicas y cinco salidas digitales.

3.1.4 Elección de la plataforma de software que se comunica con la tarjeta de control.

Si bien es cierto la tarjeta electrónica "PIC WEB", es un pequeño servidor y soporta una página web (véase aplicaciones demostrativas en el "ANEXO K"), esta es limitada en cuanto a la memoria en el microcontrolador donde se aloja dicha página, este espacio es de 1Mbit de memoria flash tal como se especifica en el "ANEXO I" de este informe.

En consecuencia la pantalla que puede cargarse en la tarjeta electrónica "PIC WEB" sería como la que se indica en el "ANEXO K", esta pantalla no soporta sub-pantallas,

animaciones, dinamismos, alta intensidad de gráficos lo que lo hizo limitado para nuestras aplicaciones de implementar un sistema SCADA convencional.

En función de las limitaciones del espacio de memoria del microcontrolador a bordo de la tarjeta electrónica "PIC WEB", de la falencia de dinamismos y por la necesidad de implementar una plataforma elaborada y general se optó por lo siguiente:

✓ Las pantallas de monitoreo y control debían alojarse en una estación servidora y no en el espacio de memoria de la tarjeta electrónica "PIC WEB".

✓ Las pantallas de monitoreo y control que se alojan en la estación servidora debían desarrollarse en plataformas de software libre.

Como consecuencia de estos requerimientos y para lograr dicho cometido se debía desarrollar un conector que establezca la comunicación entre la tarjeta electrónica "PIC WEB" y la estación servidora para poder intercambiar información, dicho conector por el lado del servidor debía cumplir los requerimientos siguientes:

✓ Debe ser robusto

✓ Que pueda operar independiente sobre cualquier sistema operativo.

✓ El entorno de desarrollo sea libre.

Para satisfacer estos requerimientos del proyecto existen dos plataformas bien desarrolladas como es el caso la plataforma "Visual Basic .NET" y la plataforma "Java", en el cuadro 3.2 se indican las características principales de cada plataforma.

CUADRO 3.2, Características principales de la plataforma de desarrollo .NET y JAVA.

CARACTERÍSTICAS ENTRE PLATAFORMAS DE DESARROLLO		
CARACTERISTICAS	**PLATAFORMA VISUAL BASIC .NET**	**PLATAFORMA JAVA**
Compañía propietaria	Microsoft Corp.	Sun Microsystem
Lenguajes de programación compatibles	C #, Visual Basic.NET, ASP.NET, C++, C.	Java
Programación orientada a objetos	Si	Si
Requieren de entorno de jecución	Si	Si
Entorno de desarrollo	Visual studio.NET	Eclipse. Netbeans.
licenciamiento	Comercial, Educacional	GNU (General public license)
Implementaciones más relevantes en el API	Interacción con dispositivos periféricos, Manejo de datos, gestion de memoria, Transmisión por XML y TCP/IP, etc.	Conexión con base de datos, sockets, servicios web, etc.
Compatible con Windows	Si	Si
Compatible con Linux	No	Si

Ambas plataformas nos permiten crear pantallas de supervisión y control, permiten crear conectores de comunicación con la tarjeta electrónica PIC WEB y son robustos, pero del cuadro 3.2 se observa que la plataforma "Java" con respecto a la plataforma "Visual Basic .Net" tiene dos particularidades bien definidas, tiene licenciamiento libre y además es compatible con sistemas operativos Windows y Linux por lo que "Java" es la plataforma que se eligió para este proyecto y como herramienta de programación se utilizó NetBeans 6.9.1., del fabricante y desarrollador Sun Microsystems, se utilizó Netbeans a diferencia

"Oracle" es una plataforma propietaria, está bastante desarrollada, es bastante robusta, pero a la vez bastante compleja de implementar por lo que solo grandes corporaciones las tienen implementadas, por otro lado "MySQL" es una plataforma con bastantes bondades técnicas como las apreciadas en el cuadro 3.3, bastante sencilla de implementar, robusta y la ventaja principal es que es una base de datos libre, por lo que para los objetivos de este proyecto la plataforma "MySQL" fue elegida.

3.2.2 Elección del servidor web.

En un sentido muy estricto un "Servidor web" no es lo mismo que "Servidor de aplicaciones web", en este proyecto inicialmente se tenía que elegir un "Servidor web" que cumpla con lo siguiente:

- ✓ Debía ser software libre.
- ✓ Debía ofrecer instalaciones sencillas para pequeños sitios web.
- ✓ Posibilidad de expandirse y tener un nivel como otros servidores.
- ✓ Debía ser robusto.
- ✓ Debía conectarse directamente a una base de datos.
- ✓ Debía ser capaz de soportar/utilizar intérpretes y aplicaciones de servidores como PHP.
- ✓ Debía soportar gran cantidad de usuarios conectados.

En el cuadro 3.4, se presentan las características principales de dos servidores web libres por un lado se tiene "Apache" perteneciente a la fundación Apache y por el otro "AOL Server" perteneciente a AOL Inc. Compañía que liberó el servidor "AOL server", se puede observar que ambas son robustas y soportan conectividad con base de datos, sin embargo "AOL Server" se desarrolla en Tcl (Tool Command Languaje, por sus siglas en inglés), es un lenguaje de script creado por John Ousterhout, que tiene cierta complejidad al implementarse una página web, sin embargo "Apache", es un servidor que cumple los requerimientos del proyecto, es el más utilizado hoy en día y es conocida la sintaxis en PHP que nos permite implementar páginas web dinámicas.

Bajo estas consideraciones se eligió el servidor "Apache 2.2", este es un servidor libre y cumple con los requerimientos iniciales.

Las aplicaciones web que interactúan con el usuario son soportadas por "Apache" por medio de PHP (lenguaje de programación soportado por Apache y que permite crear páginas web), en este caso se utilizó PHP versión 5.

3.3 Arquitectura de comunicación entre las diferentes plataformas.

En función de lo descrito en los apartados 3.1 y 3.2 de este informe, en la FIG. 3.6 se muestra la arquitectura empleada en esta aplicación de la forma como se comunica las diferentes plataformas utilizadas en esta implementación.

CUADRO 3.4, Características principales de "Apache" y "AOL Server"

CARACTERÍSTICAS PRINCIPALES DE SERVIDOR APACHE Y AOL	
APACHE	**AOL SERVER**
Pertenece a la fundación Apache ORG	Pertenece a AOL Inc.
Apoyo fuerte para proveedores de Servicios de Internet (ISP's).	Deficiente para dar apoyo a proveedores de Servicios de Internet (ISP's) que requieren miles de sitios pequeños con páginas estáticas
No existe uniformidad para conectarse a bases de datos, cada programador puede utilizar diferentes módulos o modificaciones para conectarse con una base de datos.	Excelente conectividad para Base de Datos , su diseño fue con esta intención.
Existen amplias librerías disponibles, especialmente en Perl y PHP.	Existen algunas librerías escritas en (Tcl) para facilitar el desarrollo de aplicaciones.
Soporta una gran gamma de lenguajes y debido a esto cada programador difiere de las funciones que utiliza (ya que muy pocas funciones fueron construidas internamente al servidor)	Todo su desarrollo se realiza en el mismo lenguaje (Tcl) y generalmente solo requiere de las funciones (API) internas con las que fue construido.

3.4 Declaración de variables en la tarjeta electrónica (PIC WEB).

Si bien es cierto la tarjeta electrónica contiene diversos protocolos de comunicación en Ethernet como por ejemplo TCP, IP, UDP, HTTP, SMTP, de nada serviría esto si es que no se declaran o se acondicionan las variables adecuadamente en la tarjeta electrónica PIC WEB, dado que no se podrían visualizar o comandar desde una plataforma de monitoreo y control como es el caso de esta aplicación para lograr ello se tienen que realizar ciertos pasos fundamentales.

3.4.1 Declaración de variables en la pila del microcontrolador.

La declaración de las variables internas del microcontrolador correspondientes a entradas digitales, salidas digitales y entradas analógicas es fundamental para poder direccionar o apuntar en forma remota a dicha variable, en los "ANEXOS A y B" de este informe, se muestra parte del código fuente escrito en lenguaje "C" (Lenguaje requerido para programación de microcontroladores de la familia 18 del fabricante "Microchip"), programa que viene embebido en el microcontrolador al que se le modificado ciertas partes del código.

En el "ANEXO A", de este informe se aprecia la definición de los puertos de entrada/salida digitales del microcontrolador estas son definidas en el archivo "hardwareprofile.h", archivo de cabecera, obsérvese que se han definido los puertos utilizados, en el caso de salidas digitales se le ha asignado como "LEDn", donde "n" denota número entero, en el caso de entradas digitales se le ha asignado "BUTTONn", donde "n" en este caso también denota un número entero.

En el "ANEXO B", se ha declarado los puertos de entrada analógica y se le ha configurado el procesamiento a cada tipo de señal digital y analógica, todas estas

declaraciones y procesamientos se encuentran configuradas en el archivo "CustomHTTPApp.c", obsérvese que incluso se muestra definida las entradas analógicas "pot" y "temp", que son variables que vienen por defecto definidas dentro del microcontrolador, entonces del mismo modo que "pot" y "temp" se declararon "an2" y "an3", entradas analógicas requeridas en esta implementación, las definiciones para "an2" y "an3" fueron realizadas para aceptar cualquier señal analógica del tipo 4-20mA.

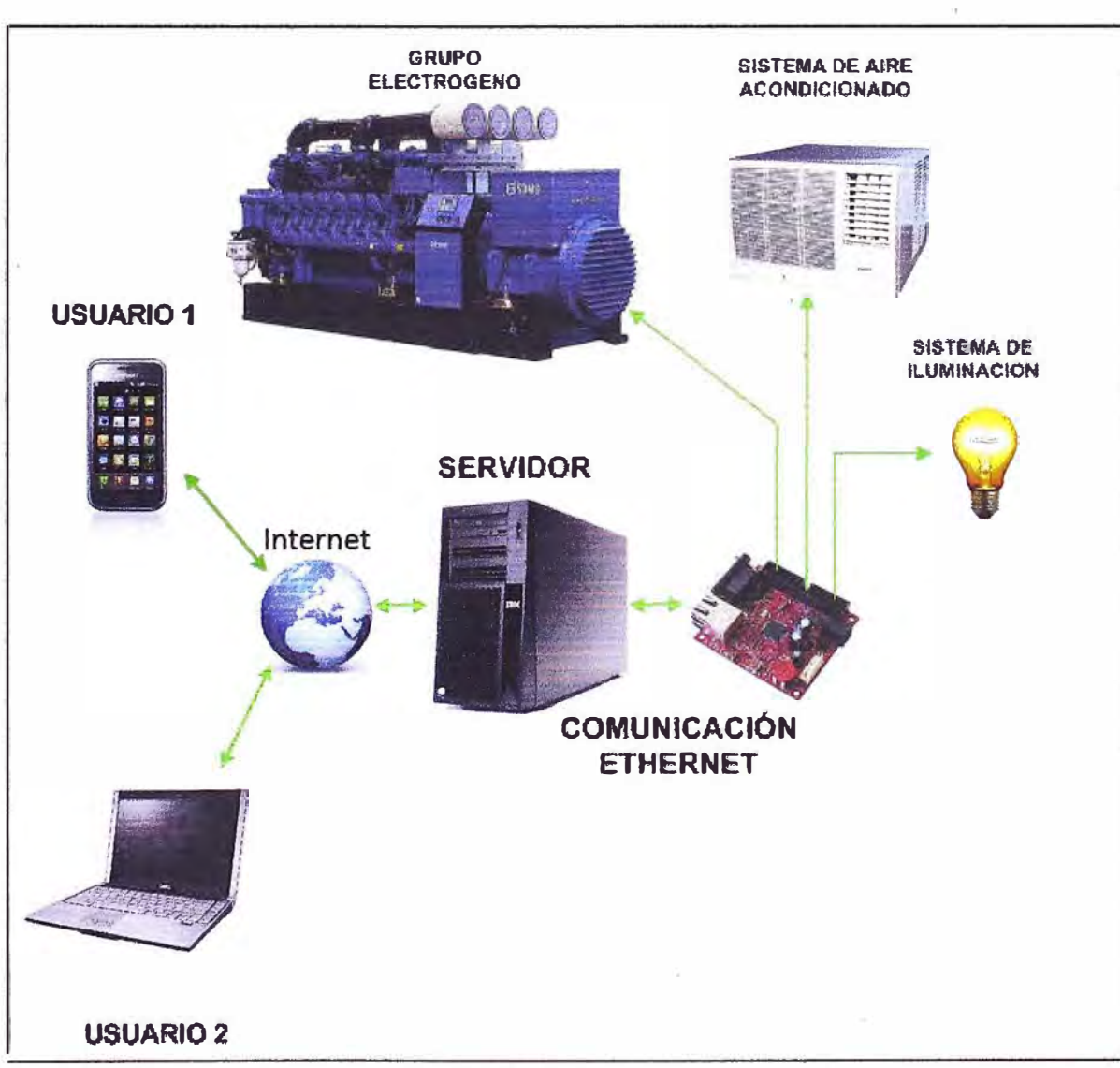


FIG 3.5, Arquitectura de monitoreo y control para un solo cliente, Fuente: Archivos del proyecto, compañía "C y Q Ingeniería e Instrumentación S.A.C."

3.5 Conexión con la tarjeta electrónica PIC WEB.

Para realizar la conexión entre la tarjeta PIC WEB y la plataforma Java instalado en el servidor, se requiere de un puerto y el respectivo conector (implementado en Java), en la figura siguiente se muestra el diagrama de funcionamiento del sistema, se puede apreciar que se emplea "Stream´s" obsérvese la FIG. 3.6, el

3.6 Conexión entre el conector en java y base de datos.

En el apartado 3.5, se realiza la conexión entre la tarjeta electrónica y la plataforma Java, simultáneamente esta plataforma debe interactuar almacenando y extrayendo información desde la base de datos, entonces la plataforma Java en esta aplicación cumple dos funciones principales:

- ✓ Realiza la conexión con la tarjeta electrónica.
- ✓ Realiza la conexión con la base de datos.

En el "ANEXO D" se muestra el código fuente que permite la conexión con la base de datos, se observa que aún cuando se pierda la comunicación por algún motivo la información como está almacenada en una base de datos, no se pierde el usuario conectado remotamente visualizará el último dato almacenado.

3.7 Conexión entre el servidor Apache y la base de datos

En el "ANEXO E", se muestra el código fuente escrito en PHP (lenguaje de programación para aplicaciones con servidores, nos permite crear páginas web dinámicas), dicho código fuente nos permite por medio del servidor "Apache" interactuar directamente con la base de datos, hay que tener en cuenta que el servidor "Apache", y las aplicaciones web que se generaron no interactúan directamente con la tarjeta electrónica PIC WEB sino únicamente con la base de datos.

El código fuente mostrado permite que un usuario conectado remotamente pueda acceder al sistema para lo cual debe ingresar el "usuario" y "contraseña" asignado, si es que lo ingresado no coincide con lo pre-configurado en la base de datos o si existe un problema de comunicación el sistema automáticamente muestra un error.

El formato en el cual se guardan tanto el "Usuario" y la "Contraseña" son encriptados debido a un criterio netamente de seguridad de esta manera dotamos al sistema con un grado de seguridad que lo hace bastante confiable.

El grado de encriptación se da únicamente para la forma de acceso mas no así para la información almacenada.

CAPÍTULO IV
ANÁLISIS Y PRESENTACION DE RESULTADOS

4.1 Edificación a escala para las pruebas e implementación.

En la siguiente imagen se aprecia una construcción fabricada a escala especialmente para la realización de las pruebas e implementación del sistema, nótese que se cuenta con un sistema de iluminación, un sistema de ventilación (que simularán un sistema de aire acondicionado), y también se le ha dotado de un sistema de accionamiento de persianas o ventanas.

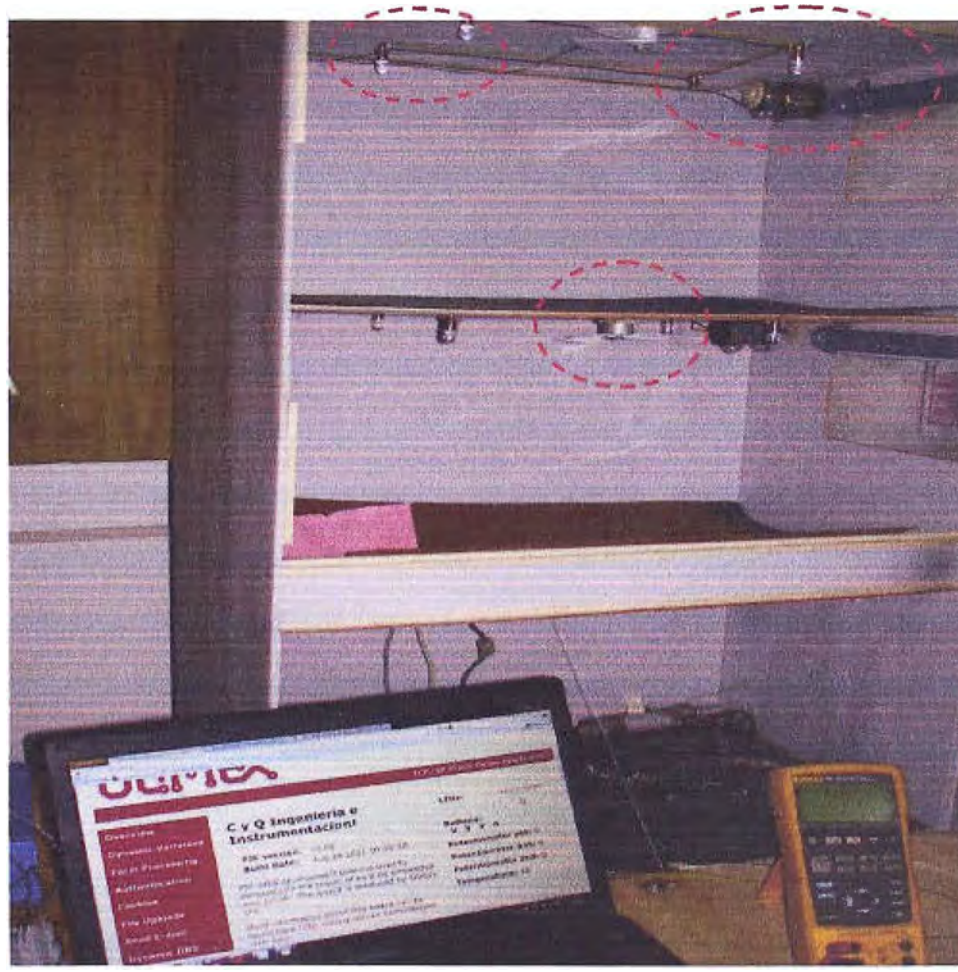


FIG 4.1, Imagen panorámica de la maqueta en donde se realizaron las pruebas, fuente: Archivos Compañía "C y Q Ingeniería e Instrumentación S.A.C.".

4.2 Pruebas iniciales de comunicación con la tarjeta electrónica.

En las siguientes imágenes FIG. 4.2 y FIG. 4.3, se muestra el aplicativo web de la pequeña pagina que se configuró y se descargó en la tarjeta electrónica esto con la finalidad de verificar que las variables que se declararon y configuraron en el ANEXO A respondan satisfactoriamente.

Para lograr visualizar esta mini-página web apuntamos en una ventana de "Internet explorer" por ejemplo la siguiente dirección "192.168.0.30" e inmediatamente se carga la mini-página web se puede apreciar en estas imágenes que se tienen los parámetros "Leds", "Buttons", "Potentiometer pot", "Potentiometer an2, an3" y "Temperature" todos ellos indicando sus valores actuales, en los círculos con líneas punteadas se observa los cambios de estado ante entradas digitales físicas véase imágenes FIG. 4.2 y FIG. 4.3 (flecha arriba/abajo ^).

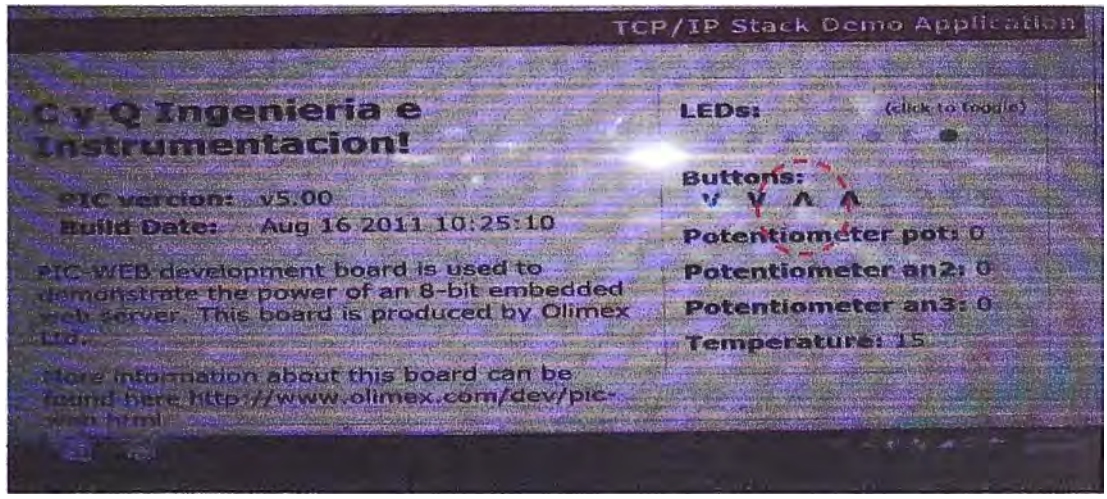


FIG 4.2, Pruebas de entradas discretas "Boton 1", Fuente: Archivos del proyecto, Compañía "C y Q Ingeniería e Instrumentación S.A.C."

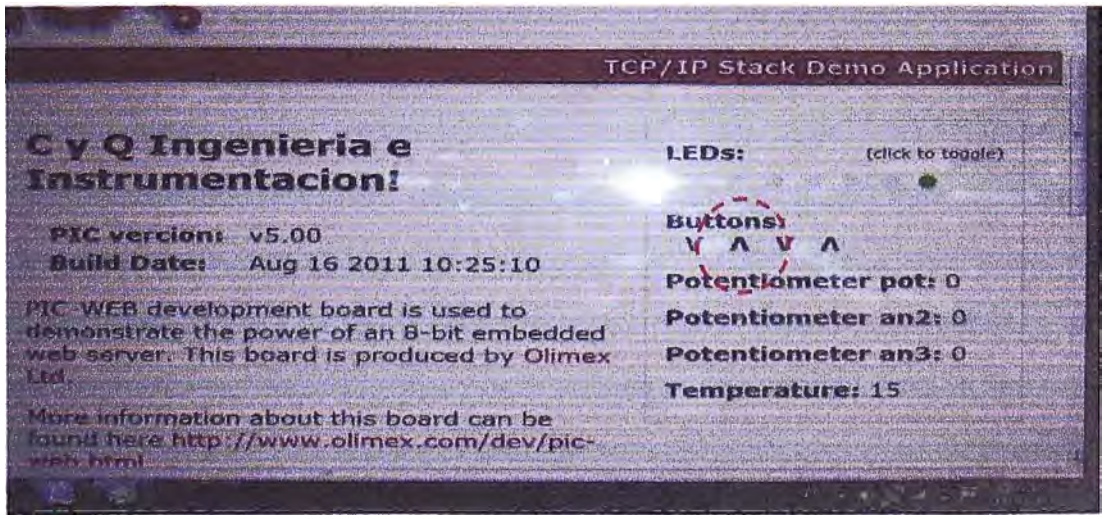


FIG 4.3, Pruebas de entradas discretas "Boton 2", Fuente: Archivos del proyecto, Compañía "C y Q Ingeniería e Instrumentación S.A.C.

En las imágenes FIG. 4.4, FIG. 4.5 y FIG. 4.6 se muestran los valores obtenidos en el puerto de entrada asignada a una de variable analógica por ejemplo "Potentiometer an2", la variable analógica se generó con un calibrador de procesos, equipo que genera una señal de 4-20mA, los valores obtenidos son como se indican en la tabla de la FIG. 4.4, estos valores obtenidos en formato decimal fueron necesarios para poder hacer el escalamiento respectivo por ejemplo la lectura en decimal "141-704" debido a una señal proveniente de un transmisor de 4-20mA sería equivalente en una pantalla de visualización remota como "0-100%", "0-100ºC", "0-20bar", etc.

SEÑAL GENERADA	VALOR OBTENIDO
0mA	0
4mA	141
8mA	282
20mA	704

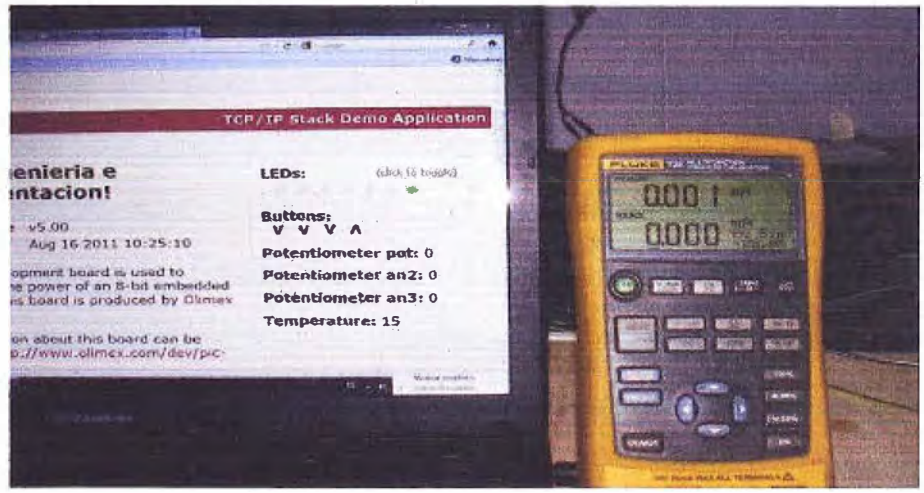


FIG 4.4, Pruebas de entrada analógica "AN2", Fuente: Archivos del proyecto, Compañía "C y Q Ingeniería e Instrumentación S.A.C."

FIG 4.5, Pruebas de entrada analógica "AN2", Fuente: Archivos del proyecto, Compañía "C y Q Ingeniería e Instrumentación S.A.C."

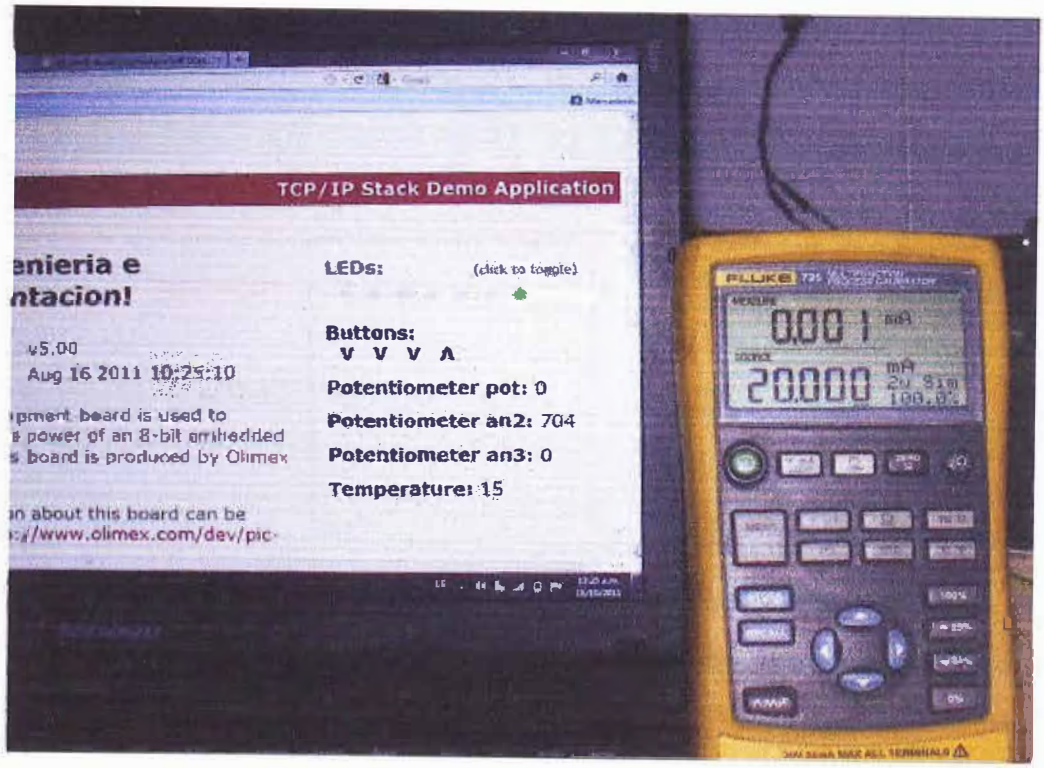


FIG 4.6, Pruebas de entrada analógica "AN2", Fuente: Archivos del proyecto, Compañía "C y Q Ingeniería e Instrumentación S.A.C."

Se sabe que para un convertidor análogo/digital de 10 bits se tienen un rango de 0 a 1023 en formato decimal para un voltaje de entrada de 0 a 3.3 voltios, sin embargo para una señal máxima generada de 20mA con un calibrador de procesos (equipo empleado

en este proyecto para generar las señales analógicas de 4-20mA [5]) se tiene un equivalente a 704 en formato decimal esta no equivalencia entre la señal obtenida de 704 (valor obtenido según las pruebas véase FIG. 4.6) y la señal máxima teórica que debería obtenerse obedece al arreglo de resistencias realizadas en la tarjeta acondicionadora de señal (Véase FIG 3.2), en ella se colocó una resistencia de 110 ohmios lo que equivaldría a una caída de tensión de 0.44 a 2.2 voltios para un rango de señal de 4-20mA respectivamente lo que equivaldría un rango en formato decimal de 136.4 a 682, estos valores son calculados teóricamente en función de los voltajes a la entrada del microcontrolador y son semejantes a los obtenidos en las imágenes FIG. 4.4, 4.5 y 4.6.

"Potentiometer an2", El ligero error cometido entre los cálculos y el obtenido experimentalmente en las imágenes FIG. 4.4, 4.5 y 4.6 se compensan por software, es decir en la plataforma de aplicaciones web.

Tanto la entrada analógica "Potentiometer pot", "Potentiometer an2", "potentiometer an3" y "Temperature" fueron configuradas en el microcontrolador refiérase a los "ANEXOS A y B", los puertos que se emplearon para efectos de prueba en este proyecto fueron "Potentiometer an2", "Potentiometer an3" y "Temperature" para las señales de nivel de cisterna de agua, presión en ducto de sistema de aire acondicionado y temperatura interna del edificio (habitación dentro del edificio), los transmisores pasivos empleados y que son compatibles con los puertos configurados son como lo indica el datasheet del "ANEXO L" .

La temperatura que se obtuvo en estas pruebas obedecieron a la señal proveniente de un termistor que se encuentra embebido en la tarjeta electrónica PIC WEB, obsérvese la FIG. 3.1.

4.3 Conexión con la plataforma Java.

Luego de la declaración de variables realizada en el microcontrolador se tuvo que probar la conectividad con la plataforma Java, dado que como se había previsto inicialmente debe existir una conectividad con la tarjeta electrónica "PIC WEB", para la respectiva transferencia de información parte del código fuente escrito en Java se encuentra en el "ANEXO D", dicho código nos permite hacer la conectividad con la tarjeta electrónica conforme a las variables declaradas en los "ANEXOS A y B", el compilado y la ejecución de dicho código en Netbeans 6.9.1 nos permitió obtener una salida en el visualizador de ejecución del programa los valores actuales de la variable declarada tal y como se indica en la FIG. 4.7.

Estos resultados muestran que efectivamente existe una comunicación con la tarjeta electrónica además de leer los valores de cada variable declarada luego de obtener estos resultados satisfactoriamente se tendrá que almacenar los datos en otra plataforma.

Dicha plataforma es una base de datos.

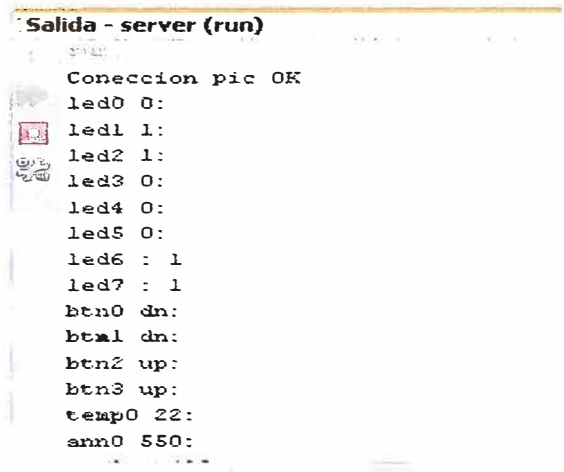


FIG 4.7, Resultado del compilado y ejecución del código desarrollado en Java, donde se aprecia los valores actuales de las variables declaradas en la tarjeta electrónica, Fuente: Compañía "C y Q Ingeniería e Instrumentación S.A.C."

4.4 Conexión remota entre el usuario y la base de datos/ingreso al sistema.

Para poder ingresar remotamente al sistema de monitoreo y control desde cualquier dispositivo/equipo con conexión a internet en primera instancia se debe abrir una aplicación que permita acceder al sistema por ejemplo el "Internet explorer" aplicativo de Windows o el "Mozilla Firefox" y desde allí direccionar lo siguiente "**cyq.dyndns.org**",

FIG 4.8, Pantalla de acceso al sistema de monitoreo y control en forma remota, Fuente: Compañía "C y Q Ingeniería e Instrumentación S.A.C."

Esta dirección apunta al servidor obsérvese la FIG. 3.6, este servidor contiene la aplicación web, en primera instancia aparece la siguiente ventana FIG. 4.8, donde se aprecia un campo en el cual se tienen que validar el "usuario" y "contraseña", este usuario y contraseña está pre-determinado y almacenado en la base de datos ver "ANEXO E".

Si el "usuario" y "contraseña" es ingresado incorrectamente el sistema mostrará una advertencia de error y no se podrá acceder al sistema.

Enseguida se tiene la siguiente pantalla obsérvese la FIG. 4.9, en ella se puede apreciar la pestaña para poder acceder al sistema, sin embargo también esta pantalla tienen la posibilidad de reconfigurar el nuevo usuario, contraseña y nivel de acceso, esto es importante porque por ejemplo el nivel de acceso permitirá limitar las acciones de los usuarios, es decir a monitorear solamente otros a monitorear ó configurar el sistema, etc.

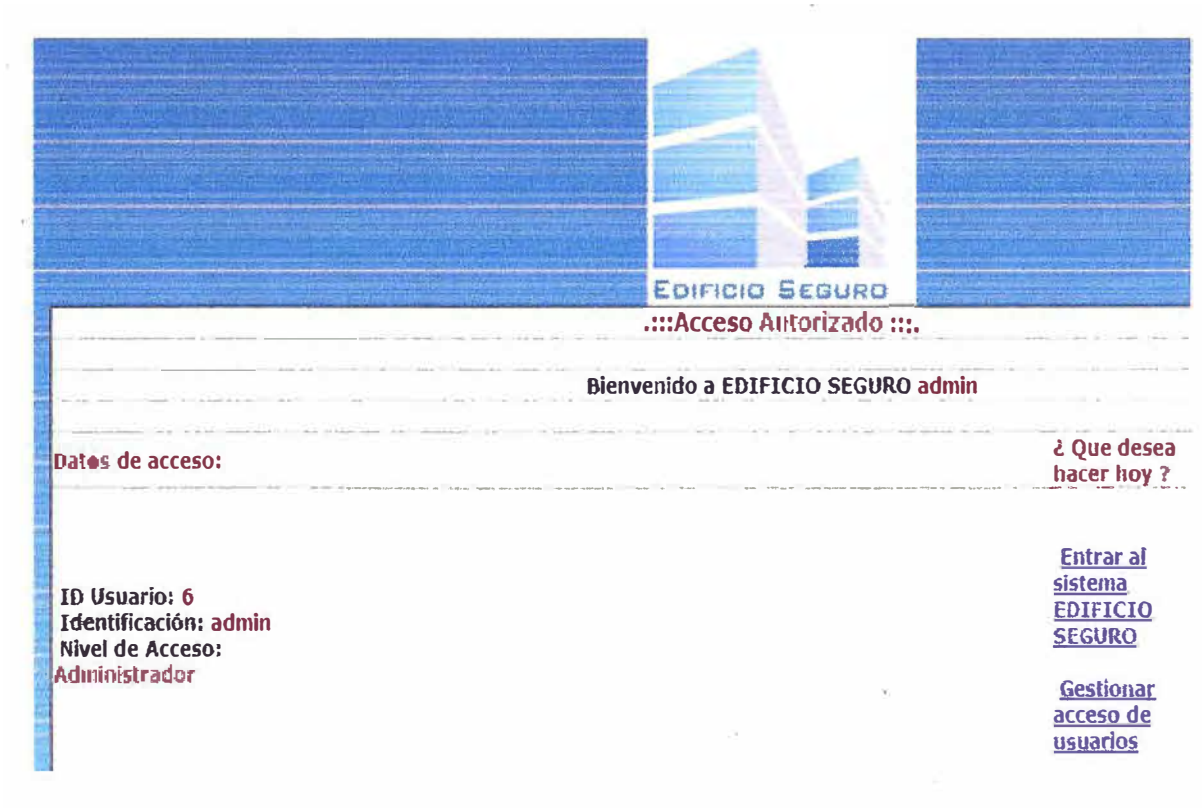


FIG 4.9, Pantalla para acceder al sistema, configurar el usuario, contraseña y nivel de acceso, Fuente: Compañía "C y Q Ingeniería e Instrumentación S.A.C."

4.5 Pantallas principales.

En las imágenes FIG. 4.10 y FIG. 4.11, se muestran las pantallas principales del sistema desde el cual se pueden monitorear y controlar las variables de proceso, obsérvese que en la pantalla FIG. 4.10, se ha colocado una plano (vista de planta), de esta manera el usuario puede en todo momento saber la ubicación de cada sistema a monitorear y controlar, esto con la finalidad de hacer más intuitiva la navegación, también se ha dotado

a la pantalla de pestañas de comando o botones, también existen botones para acceder a otros pisos esto es importante si se tienen edificaciones con varios pisos.

En la imagen FIG. 4.11, se aprecia por ejemplo sistemas como tanques de almacenamiento de combustible para los grupos electrógenos y también los ductos de aire acondicionado y las variables que normalmente se requieren monitorear son por ejemplo nivel y presión respectivamente, entonces el nivel se ha estipulado en "%" y la presión en "bar".

Se observa que los gráficos y planos dentro de la pantalla no son tan refinados, esto obedece a que no se ha utilizado software especializado para el diseño sino más bien se ha utilizado imágenes y recortes que se han tomado de otros archivos no tan especializados.

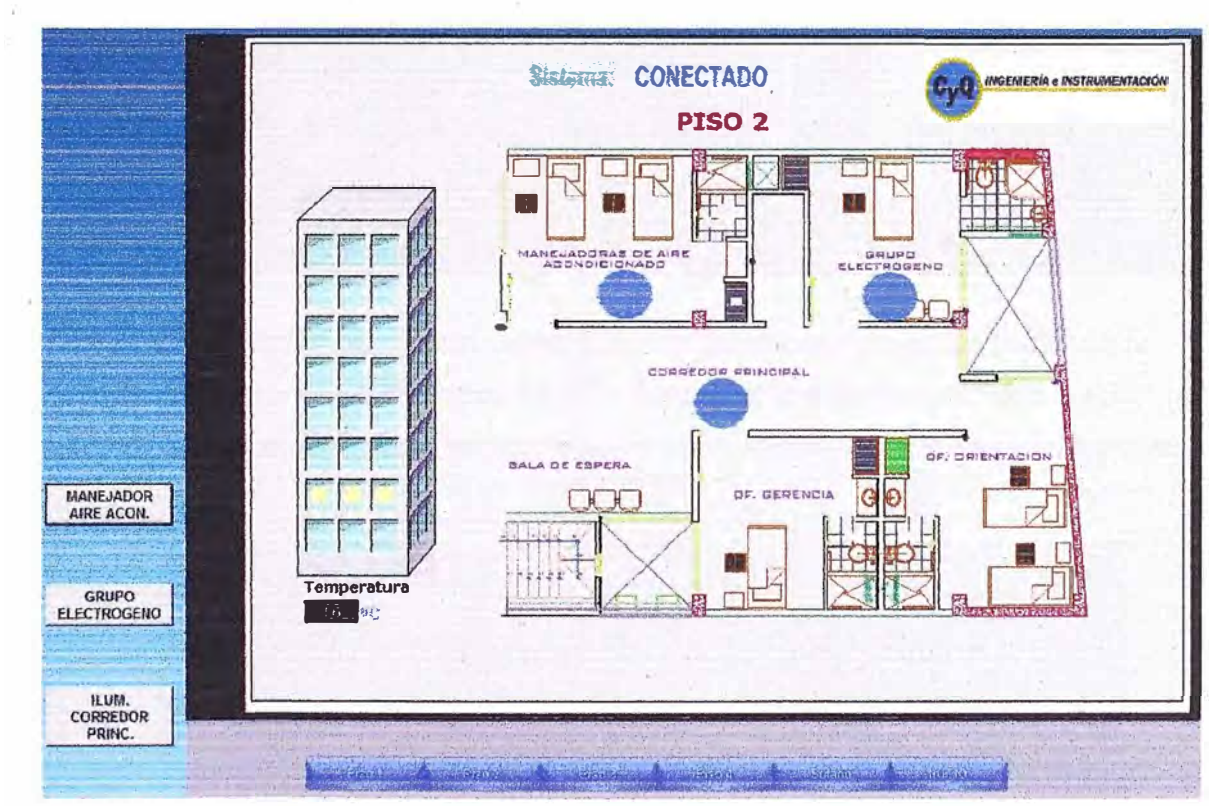


FIG 4.10, Pantalla principal del piso 2, vista de planta de la infraestructura, Fuente: Compañía "CyQ Ingeniería e Instrumentación S.A.C."

En la imagen FIG. 4.12 se aprecia los valores obtenidos producto de las simulaciones con el calibrador de procesos, ambas señales son analógicas de 4-20mA.

Ambas señales fueron escaladas como sigue:

- ✓ Visualización de nivel: 0-100%.
- ✓ Visualización de presión: 0-100bar.

Estas pruebas sirvieron para dejar configurado el escalamiento de las señales de 4-20mA que se conectaran a los puertos analógicos predefinidos. Entonces cualquier transmisor

que se instale y se conecte a estos puertos con una señal de 4-20mA será fácilmente leída por el sistema, en caso la señal fuese de otro tipo el escalamiento tendrá que reconfigurarse necesariamente.

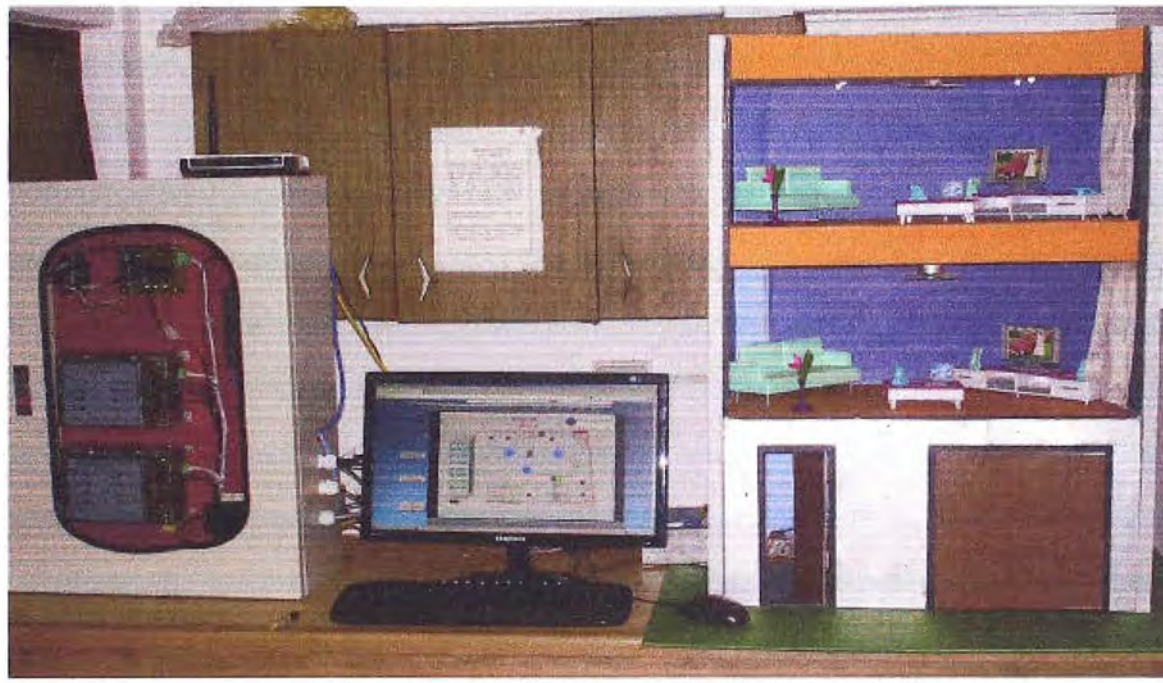


FIG 4.11, Tablero de control, pantalla principal del piso 2, vista de planta de la infraestructura, Fuente: Compañía "C y Q Ingeniería e Instrumentación S.A.C."

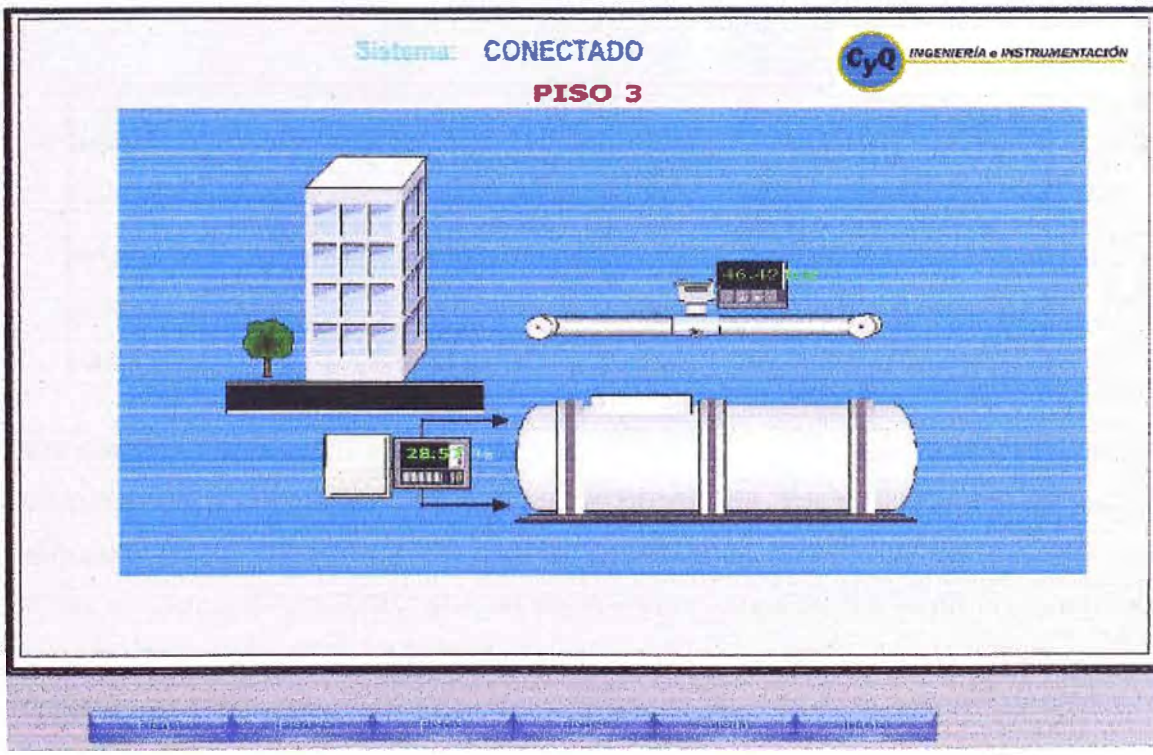


FIG 4.12, Pantalla principal del piso 3, vista de planta de la infraestructura, Fuente: Compañía "C y Q Ingeniería e Instrumentación S.A.C."

En las Figuras FIG. 4.10 y FIG. 4.12 se aprecian las pantallas finales como resultado de la implementación de este proyecto y esta interactúa con la periferia como se muestra en la FIG. 4.11, se observa que para una aplicación real el tablero de control debe reducirse considerablemente puesto que el espacio es un factor muy importante, las pantallas de supervisión en esta aplicación son funcionales pero estas deben mejorarse para tener una apariencia mucho más elaborada, para ello se tendría que emplear software especializado para la creación de imágenes, el tener una vista de planta del edificio, la casa o el hogar es una práctica común (ver FIG. 4.10) partiendo del supuesto que las pantallas deben ser intuitivas para el usuario que navegue por ellas.

Todas estas pantallas se alojan en el servidor, entonces para que el sistema no falle por ejemplo por pérdida de energía en la red eléctrica ésta debe de dotarse de un sistema ininterrumpido de energía e implementar todas las protecciones eléctricas necesarias.

4.6 Participantes del proyecto.

Para el diseño e implementación de este proyecto participaron un equipo de trabajo con especialistas en cada área como se indica en el cuadro 4.1, en donde se me asignó la tarea de dirigir el proyecto y de colaborar activamente en las tareas de programación del hardware, en este sentido mi labor fue importante al igual que todos los involucrados para el éxito de este proyecto.

CUADRO 4.1, Especialistas que intervinieron en el desarrollo y ejecución del proyecto compañía "C y Q Ingeniería e Instrumentación S.A.C."

ITEM	ESPECIALISTAS	CANTIDAD
1	DIRECTOR/ENCARGADO DEL PROYECTO	1
2	PROGRAMADOR DE MICROCONTROLADORES	1
3	DESARROLLADOR DE SOFTWARE	1
4	DISEÑADOR DE TARJETAS ELECTRÓNICAS	1

4.7 Cronograma de ejecución del proyecto.

El proyecto en su totalidad fue culminado dentro de los seis meses proyectados inicialmente, en el cuadro 4.2, se puede apreciar los tiempos empleados por cada actividad en donde la actividad que demandó mayor atención fue la de implementación de las pantallas de monitoreo y control.

4.8 Costos del proyecto.

En el cuadro 4.3, se detalla los costos de los equipos y desarrollos que se hubieran necesitado para la realización de este proyecto empleando equipos y software de un fabricante tradicional en este caso del fabricante Siemens.

Por otro lado en el cuadro 4.4, se detalla los costos de los equipos y desarrollos que se necesitaron en la implementación de este proyecto empleando hardware de fabricantes no tradicionales y desarrollo de software en plataformas libres, se observa pues entonces haciendo una comparación entre ambos cuadros que resulta 83% menos costoso realizar una implementación de este tipo con hardware como la PIC WEB y plataformas de software libre que utilizando hardware y software de fabricantes como Siemens, ahora esta reducción es solo del 83% para esta aplicación en caso se tenga por ejemplo una agencia bancaria con 100 sucursales (es decir 100 oficinas distribuidas), la diferencia de costos es a tener muy en cuenta.

CUADRO 4.2, Tiempos de ejecución del proyecto según las actividades, Fuente: Archivos de la compañía "C y Q Ingeniería e Instrumentación S.A.C.".

		AÑO 2011					
ITEM	ACTIVIDADES	MAY	JUN	JUL	AGO	SET	OCT
1	Conformación del equipo de trabajo, diseñadores de tarjetas, programadores y desarrolladores.	X					
2	Elección del hardware y software a utilizar.	X					
3	Estudio y análisis del funcionamiento de la tarjeta PIC WEB, análisis de señales a utilizar, selección de protocolos de comunicación a emplear.	X	X				
4	Programación de la tarjeta PIC WEB.		X	X	X		
5	Implementación de la tarjeta acondicionadora de señal.		X	X			
6	Desarrollo de la plataforma de software y pantallas de monitoreo y control.		X	X	X	X	X
7	Implementación de un edificio a escala para las respectivas pruebas.				X	X	
8	Pruebas generales tanto de hardware como de software.				X	X	X

CUADRO 4.3, Relación de costos de dispositivos y equipos de control de un fabricante tradicional, fuente: Centro de costos compañía "C y Q Ingeniería e Instrumentación S.A.C."

DISPOSITIVO/EQUIPO DE CONTROL	COSTO EN NUEVOS SOLES (S/.)
Fuente DC Sitop y sistemas de protección – fabricante Siemens.	950.00
PLC con comunicación Ethernet – fabricante Siemens.	2254.70
Módulo con entradas analógica – fabricante Siemens.	810.78
Software de monitoreo y control Win - CC - fabricante Siemens.	10821.16
Servicio de programación de PLC y programación de pantallas de supervisión.	3000.00
COSTO TOTAL DE SUMNISITROS Y DESARROLLO	**17836.64**

Los costos en este cuadro son unitarios y fueron tomadas de la lista de precios 2011, sector industria – Fabricante Siemens.

CUADRO 4.4, Relación de costos de dispositivos y equipos de control de un fabricante no tradicional, Fuente: Centro de costos compañía "C y Q Ingeniería e Instrumentación S.A.C."

DISPOSITIVO/EQUIPO DE CONTROL	COSTO EN NUEVOS SOLES (S/.)
Fuente de 24VDC.	190.00
Tarjeta acondicionadora de señal.	300.00
Tarjeta electrónica PIC WEB (Incluye impuestos y costos de importación).	440.00
Desarrollo de software, programación de pantallas y programación de tarjeta electrónica PIC WEB.	2500.00
COSTO TOTAL DE SUMNISTROS Y DESARROLLO	**3430.00**

Los costos en este cuadro son unitarios y fueron extraídos del centro de costos de la compañía "C y Q Ingeniería e Instrumentación S.A.C.".

Nota:

Los costos de la estación servidora, licencia por sistema operativo y sensores no se ha incluido en los cuadros 4.3 y 4.4 debido a que estos obedecen a un estudio previo de la

planta o proceso que se requiera monitorear y controlar, esto no es parte del alcance de esta implementación.

CONCLUSIONES Y RECOMENDACIONES

- La implementación de sistemas que involucren monitoreo y control de variables de proceso en forma remota (ya sea de procesos industriales u edificaciones) en la actualidad resulta bastante costoso a tal punto que solo algunas empresas como hoteles cinco estrellas, bancos (oficinas principales), grandes residencias y la gran industria puedan acceder a ella sin embargo los hogares comunes, los edificios multifamiliares, inclusive algunas oficinas de bancos conocidos, oficinas gubernamentales y la pequeña y mediana industria no tienen las posibilidades económicas de implementar un sistema de monitoreo y control en forma remota con sistemas de fabricantes tradicionales, entonces es en función de esta necesidad que la compañía "C y Q Ingeniería e Instrumentación S.A.C.", decidió implementar una alternativa de monitoreo y control en forma remota de variables de proceso con la consigna de que debería ser lo más barato, confiable y que esté en funcionamiento lo más pronto posible con el objeto de dar una solución alternativa a los sistemas tradicionales. En esta primera etapa del proyecto involucró el monitoreo de variables y un control (manual) en forma remota (que es justamente lo expuesto en este informe), posteriormente y en una segunda etapa se trabajará netamente en control.

Entonces la elección y utilización de un hardware como es la tarjeta electrónica PIC WEB, que lleva embebido un microcontrolador con bastantes bondades técnicas como llevar embebido una gran cantidad de protocolos de comunicación en Ethernet, por la gran cantidad de entradas/salidas digitales y analógicas, por una diversidad de aspectos técnicos y también por su bajo consumo energético mucho menos que los microcontroladores tradicionales de uso general, lo hicieron idóneo para cumplir con los objetivos, en sí esta tarjeta en realidad es un pequeño servidor, que incluso si la aplicación fuese pequeña y limitada como por ejemplo encender y apagar el sistema de iluminación de una casa no sería necesario desarrollar software adicional, suficiente con una pequeña reprogramación de la tarjeta, sin embargo para cumplir los objetivos encomendados por la compañía no fue suficiente el espacio de memoria que aloja la página web en el pequeño servidor, había la necesidad entonces de buscar desarrollar e

implementar una plataforma aplicado a cada usuario basándose en plataformas de software libre como es el caso de "Java" para realizar la conexión entre las variables de la tarjeta PIC WEB y la base de datos "MySQL", por otro lado no bastó solamente desarrollar en estas plataformas sino que se debió implementar un servidor web como es "Apache" y una plataforma de aplicaciones web, entonces para lograr el objetivo de que un usuario pueda acceder a su planta, edificio u hogar remotamente es necesario complementar todas esta plataformas y todas estas plataformas por supuesto son libres, de esta manera y con la integración de todas ellas se logró implementar el sistema satisfactoriamente.

- Si bien es cierto la aplicación expuesta en este informe fue realizada en una pequeña maqueta a escala y con variables de encendido de iluminación, aire acondicionado, monitoreo de temperatura de oficina y nivel de cisterna de agua, la misma aplicación fue instalada a manera demostrativa en una planta industrial de gases criogénicos para monitorear nivel y presión en tanques de oxígeno y nitrógeno líquido, donde se puso a prueba la confiabilidad del sistema en un entorno real y agresivo y esta resultó exitosa desde todo punto de vista.
- El microcontrolador de la tarjeta electrónica si bien es cierto tiene embebido un código fuente que viene por defecto, este en particular tuvo que modificarse según los requerimientos, y fundamentalmente lo que se modificó es el código para acondicionar señales de entrada digital, señales de salida digital y también para acondicionar señales de entrada analógica, específicamente tuvo relevancia la señal analógica convencional de 4-20mA, que es un tipo de señal que todos los dispositivos de campo la soportan, con respecto al código que gestionan los protocolos de comunicación como es el caso de "TCP" e "IP", no se realizó modificación alguna esto permaneció tal y como el fabricante Microchip lo diseñó.
- La implementación de una base de datos fue muy importante porque evita problemas de saturación o lentitud en el sistema, por ejemplo cuando se tienen instaladas varias tarjetas electrónicas distribuidas o cuando las variables de proceso se incrementan sustancialmente, por otro lado muchos de los clientes requieren obtener reportes e históricos de alguna variable en particular entonces esto es muy necesario, y es uno de los requerimientos principales de los clientes, así que no enlazar una base de datos en estos sistemas es sumamente crítico.
- Un punto muy importante fue la elección de la plataforma que se empleó para dotar al sistema de un "servidor web" lo suficientemente robusto como para no saturar el sistema, debido a una gran cantidad de usuarios conectados a la vez, en este sentido "Apache",

fue la plataforma empleada, esta plataforma fue elegida no sólo para esta aplicación pequeña sino que se proyecto para soportar más aplicaciones con más tarjetas electrónicas PIC WEB.

- En los cuadros 4.3 y 4.4, se visualiza una lista de costos por un lado un conjunto de equipos, dispositivos y desarrollo de un fabricante tradicional por ejemplo "Siemens" y por el otro los costos de equipos, dispositivos y desarrollo de un proveedor local no tradicional "C y Q Ingeniería e Instrumentación S.A.C", se observa que los costos de hardware, software y desarrollo de un sistema no tradicional representa un 83% menos respecto de los costos de hardware, software y desarrollo de un sistema tradicional, si bien es cierto las potencialidades en cuanto a capacidad de procesamiento, modularidad y soporte de técnicas de control implementadas en un equipo de fabricantes tradicionales con respecto a la tarjeta electrónica elegida para este proyecto "PIC WEB" son bastante amplias no es razón suficiente cuando el costo es elevado o se torna elevado por lo tanto en función de lo expuesto este tipo de aplicaciones que emplean dispositivos y equipos de fabricantes no tradicionales o desarrollados localmente son una alternativa adecuada y confiable para aplicaciones en hogares, edificios, entidades gubernamentales y en la pequeña y mediana industria.
- La implementación de este proyecto se realizó como consecuencia de que pequeños y medianos clientes de la compañía "C y Q Ingeniería e Instrumentación S.A.C." (donde soy encargado de supervisar y ejecutar los proyectos de automatización), rechazan las propuestas económicas para sistemas de automatización de hogares, edificios y entidades bancarias e inclusive algunas industrias por sus altos costos, bajo estos inconvenientes es que se optó por desarrollar un sistema que involucre tanto hardware como desarrollo de software que permita a nuestros clientes tener el monitoreo y control (manual) de sus variables de proceso desde cualquier tipo de dispositivo fijo o móvil que esté conectado a internet, gracias entonces a esta alternativa los presupuestos son bastante aceptados por nuestros clientes y ven en este tipo de implementaciones y desarrollos un sistema robusto y confiable, dado que si bien en lo que respecta a esta implementación solamente se ha implementado en laboratorio y a escala ya se tuvo una experiencia con respecto a una demostración en el rubro industrial donde los resultado fueron óptimos.
- La importancia de esta implementación radica fundamentalmente en que es una alternativa no tradicional para aplicaciones de monitoreo y control en forma remota donde el costo es bastante asequible para cualquier cliente y tanto el desarrollo, implementación y soporte técnico pos venta es local.

- Este sistema es recomendable para pequeñas aplicaciones donde las variables son no críticas por ejemplo en casas y hogares, en este caso no se cuenta con demasiados recursos económicos como para instalar un sistema tradicional o en el caso de edificios, oficinas bancarias y gubernamentales donde se administra una gran cantidad de sucursales y por lo tanto realizar un proyecto de monitoreo y control con sistemas tradicionales suponen un costo bastante elevado y que acaban por no implementarse.

ANEXO A
PROGRAMA EN C-DECLARACION DE VARIABLES DIGITALES EN EL MICROCONTROLADOR

```
#define OLIMEX_WEB
//#define   OLIMEX MINI      // based on PIC18F25J10
//#define   OLIMEX P67J60       // based on PIC18F67J60
}//#define YOUR BOARD

#if defined(OLIMEX_MAXI) || defined(OLIMEX_WEB) || defined(OLIMEX_MINI) || \
        defined(OLIMEX_MICRO) || defined(OLIMEX_P67J60)
    #define OLIMEX_HW
#endif

#elif defined(OLIMEX_WEB) && defined(HI TECH C)
    #error "HI TECH not supported."
#elif defined(OLIMEX_WEB) && !defined(HI_TECH_C)
// PICDEM.net 2 (PIC18F97J60 + ENC28J60)
    // We have an early revision sillicon for now - do we really need it?
    #define ETH 18Fx7J60 ERRATA

    // I/O pins
    #define LED0_TRIS           (TRISBbits.TRISB4)/* parpadeo, Sal. digital LED0, propia tarjet.
    #define LED0_IO             (LATBbits.LATB4)

    #define LED1 TRIS           (TRISCbits.TRISC2)/* Salida digital LED1, puerto C2, EXT-8*/
    #define LED1 IO             (LATCbits.LATC2)

    #define LED2_TRIS           (TRISEbits.TRISE1)/* Salida digital LED2, puerto E1, EXT-6*/
    #define LED2 IO             (LATEbits.LATE1)
    #define LED3_TRIS           (PRODL)           /* NO DEFINIDO*/
    #define LED3_IO             (PRODL)
    #define LED4_TRIS           TRISEbits.TRISE4  /* Salida digital LED4, puerto E4, EXT-22*/
    #define LED4_IO             LATEbits.LATE4
    #define LED5_TRIS           (PRODL)           /* NO DEFINIDO*/
    #define LED5_IO             (PRODL)
    #define LED6_TRIS           (PRODL)           /* NO DEFINIDO*/
    #define LED6_IO             (PRODL)
    #define LED7_TRIS           (PRODL)           /* NO DEFINIDO*/
    #define LED7 IO             (PRODL)


    //#define LED_GET()         (LED1_IO)   //NOT USED
    #define LED_PUT(a)          (LED1_IO = (a))

    #define BUTTON0_TRIS       (TRISBbits.TRISB0) /*Entrada digital debida al boton de la tarjeta */
    #define BUTTON0_IO         (PORTBbits.RB0)
    #define BUTTON1_TRIS       (TRISBbits.TRISB1) /*Entrada digital BTN1, puerto B1, EXT-12 */
    #define BUTTON1_IO         (PORTBbits.RB1)
    #define BUTTON2_TRIS       (TRISBbits.TRISB2) /*Entrada digital BTN2, puerto B2, EXT-13 */
    #define BUTTON2_IO         (PORTBbits.RB2)
    #define BUTTON3_TRIS       (TRISBbits.TRISB3) /*Entrada digital BTN3, puerto B3, EXT-14 */
    #define BUTTON3_IO         (PORTBbits.RB3)
```

ANEXO B
PROGRAMA EN C-DECLARACION DE VARIABLES ANALOGICAS Y PROCESAMIENTO DE LAS SEÑALES EN EL MICROCONTROLADOR

```
/*****************************************************************************
  Section:
    Dynamic Variable Callback Functions
  *****************************************************************************/

/*****************************************************************************
  Function:
    void HTTPPrint_varname(void)

  Internal:
    See documentation in the TCP/IP Stack API or HTTP2.h for details.
  *****************************************************************************/

void HTTPPrint_builddate(void)
{
    curHTTP.callbackPos = 0x01;
    if(TCPIsPutReady(sktHTTP) < strlenpgm((ROM char*)  DATE  " "  TIME  ))
        return;

    curHTTP.callbackPos = 0x00;
    TCPPutROMString(sktHTTP, (ROM void*)  DATE  " "  TIME  );


void HTTPPrint_version(void)
{
    TCPPutROMString(sktHTTP, (ROM void*)VERSION);
}



ROM BYTE HTML_UP_ARROW[] = "up";
ROM BYTE HTML_DOWN_ARROW[] = "dn";

void HTTPPrint_btn(WORD num)
{
    // Determine which button
    switch(num)
    {
        case 0:
            num = BUTTON0_IO;
            break;
        case 1:
            num = BUTTON1_IO;
            break;
        case 2:
            num = BUTTON2_IO;
            break;
        case 3:
            num = BUTTON3_IO;
            break;
        default:
            num = 0;
```

```
void HTTPPrint_ledSelected(WORD num, WORD state)
{
    // Determine which LED to check
    switch(num)
    {
        case 0:
            num = LED0_IO;
            break;
        case 1:
            num = !LED1_IO;
            break;
        case 2:
            num = LED2_IO;
            break;
        case 3:
            num = LED3_IO;
            break;
        case 4:
            num = LED4_IO;
            break;
        case 5:
            num = LED5_IO;
            break;
        case 6:
            num = LED6_IO;
            break;
        case 7:
            num = LED7_IO;
            break;

        default:
            num = 0;


    // Print output if TRUE and ON or if FALSE and OFF
    if((state && num) || (!state && !num))
        TCPPutROMString(sktHTTP, (ROM BYTE*)"SELECTED");
    return;
}
```

```
void HTTPPrint_an2(){

    BYTE AN0String[8];
    WORD ADval;

#if defined(__18CXX)
    // ADC chanel 2
    ADCON0bits.CHS0 = 0;
    ADCON0bits.CHS1 = 1;
    ADCON0bits.CHS2   0;
    ADCON0bits.CHS3 = 0;
    DelayMs(1);
    // Wait until A/D conversion is done
    ADCON0bits.GO = 1;
    while(ADCON0bits.GO);

    // Convert 10-bit value into ASCII string
    ADval = (WORD)ADRES;
    //ADval *= (WORD)10;
    //ADval /= (WORD)102;
    uitoa(ADval, AN0String);

#else
     ADval = (WORD)ADC1BUF0;
     //ADval *= (WORD)10;
     //ADval /= (WORD)102;
     uitoa(ADval, (BYTE*)AN0String);
#endif

     TCPPutString(sktHTTP, AN0String);

}
 void HTTPPrint_temp(void)
 {
      BYTE TempString[8];
      int temperature, tmp;

 #if defined(__18CXX)
      // ADC chanel9
      ADCON0bits.CHS0 = 1;
      ADCON0bits.CHS1 = 0;
      ADCON0bits.CHS2 = 0;
      ADCON0bits.CHS3 = 1;
      // Wait until A/D conversion is done
      ADCON0bits.GO = 1;
      while(ADCON0bits.GO);

      // Convert 10-bit value into ASCII string
      tmp = (WORD)ADRES;
      if(tmp>512) {
           tmp = tmp - 512;
           tmp/=9;
           temperature = 20 + tmp;
```

```
    else {
        tmp = 512 - tmp;
        tmp/=9;
        temperature = 20 - tmp;
    }
    //ADval *= (WORD)10;
    //ADval /= (WORD)102;
    itoa(temperature, TempString);
#else
    ADval = (WORD)ADC1BUF0;
    //ADval *= (WORD)10;
    //ADval /= (WORD)102;
    uitoa(ADval, (BYTE*)AN0String);
#endif

    TCPPutString(sktHTTP, TempString);
}
void HTTPPrint_pot(void)
{
    BYTE AN0String[8];
    WORD ADval;

#if defined(__18CXX)
    // ADC chanel8
    ADCON0bits.CHS0 = 0;
    ADCON0bits.CHS1 = 0;
    ADCON0bits.CHS2 = 0;
    ADCON0bits.CHS3   1;
    DelayMs(1);
    // Wait until A/D conversion is done
    ADCON0bits.GO = 1;
    while(ADCON0bits.GO);

    // Convert 10-bit value into ASCII string
    ADval = (WORD)ADRES;
    //ADval *= (WORD)10;
    //ADval /= (WORD)102;
    uitoa(ADval, AN0String);

#else
    ADval = (WORD)ADC1BUF0;
    //ADval *= (WORD)10;
    //ADval /= (WORD)102;
    uitoa(ADval, (BYTE*)AN0String);
#endif

    TCPPutString(sktHTTP, AN0String);
```

ANEXO C
PROGRAMA CONECTOR ENTRE JAVA Y LA TARJETA PIC WEB

```
public class StAXStreamTreeViewer extends JFrame {
  private JTree jTree;

  DefaultTreeModel defaultTreeModel;

  public StAXStreamTreeViewer()throws Exception {
    StAXStreamTreeViewer viewer = new StAXStreamTreeViewer();
    File file = new File("Pic-Web");
    DefaultMutableTreeNode base = new DefaultMutableTreeNode("Static port: "
        + file.getAbsolutePath());

    defaultTreeModel = new DefaultTreeModel(base);
    jTree = new JTree(defaultTreeModel);

    buildTree(defaultTreeModel, base, file);

    getContentPane().add(new JScrollPane(jTree), BorderLayout.CENTER);
    viewer.setVisible(true);
    setSize(800, 450);
  }
  public static void main(String[] args)throws Exception {
    new StAXStreamTreeViewer();
  }
```

ANEXO D
PROGRAMA CONECTOR CON LA BASE DE DATOS

```
46  String entrada;
47  String cadena="";
48
49  while ((entrada = br.readLine()) != null){
50          cadena = cadena + entrada;
51  }
52
53  DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
54  DocumentBuilder db = dbf.newDocumentBuilder();
55
56  InputSource archivo = new InputSource();
57  archivo.setCharacterStream(new StringReader(cadena));
58
59  Document documento = db.parse(archivo);
60  documento.getDocumentElement().normalize();
61
62  NodeList nodeLista = documento.getElementsByTagName("response");
63  System.out.println("Informacion de PIC");
64
65  for (int s = 0; s < nodeLista.getLength(); s++) {
66
67          Node primerNodo = nodeLista.item(s);
68
69
70          String  led0;
71         String   led1;
72         String   led2;
73         String   led3;
74        String    led4;
75       String    led5;
76      String    led6;
77      String   led7;
78      String    btn0;
79      String   btn1;
80      String    btn2;
81      String  btn3;
82      String pot0;
83       String  temp0;
```

ANEXO E
CONECTOR ENTRE EL SERVIDOR WEB Y BASE DE DATOS

```php
// Chequeamos si se está autentificandose un usuario por medio del formulario
if (isset($_POST['user']) && isset($_POST['pass'])) {

// Conexión base de datos.
// si no se puede conectar a la BD salimos del scrip con error 0 y
// redireccionamos a la pagina de error.
$db_conexion= mysql_connect("$Servidor", "$Usuario", "$Password") or die(header ("Location:  $redir?error_login=0"));
mysql_select_db("$BaseDeDatos");

// realizamos la consulta a la BD para chequear datos del Usuario.
$usuario_consulta = mysql_query("SELECT ID,usuario,pass,nivel_acceso FROM $sql_tabla WHERE usuario='".$_POST['user']."'")

 // miramos el total de resultado de la consulta (si es distinto de 0 es que existe el usuario)
 if (mysql_num_rows($usuario_consulta) != 0) {

    // eliminamos barras invertidas y dobles en sencillas
    $login = stripslashes($_POST['user']);
    // encriptamos el password en formato md5 irreversible.
    $password = md5($_POST['pass']);

    // almacenamos datos del Usuario en un array para empezar a chequear.
    $usuario_datos = mysql_fetch_array($usuario_consulta);

    // liberamos la memoria usada por la consulta, ya que tenemos estos datos en el Array.
    mysql_free_result($usuario_consulta);
    // cerramos la Base de dtos.
    mysql_close($db_conexion);

    // chequeamos el nombre del usuario otra vez contrastandolo con la BD
```

ANEXO F
PROGRAMA EN MICROCONTROLADOR PIC 18F87J60 SERVIDOR TCP

```
/*********************************************************************
 *
 *  Generic TCP Server Example Application
    Module for Microchip TCP/IP Stack
     -Implements an example "ToUpper" TCP server on port 9760 and
      should be used as a basis for creating new TCP server
      applications
     -Reference: None.  Hopefully AN833 in the future.
 *********************************************************************/
#define __GENERICTCPSERVER_C

#include "TCPIPConfig.h"

#if defined(STACK_USE_GENERIC_TCP_SERVER_EXAMPLE)

#include "TCPIP Stack/TCPIP.h"


// Defines which port the server will listen on
#define SERVER_PORT 9760


/*****************************************************************************
  Function:
    void GenericTCPServer(void)

  Summary:
    Implements a simple ToUpper TCP Server.

    Description:
      This function implements a simple TCP server.  The function is invoked
      periodically by the stack to listen for incoming connections.  When a
      connection is made, the server reads all incoming data, transforms it
      to uppercase, and echos it back.

      This example can be used as a model for many TCP server applications.

    Precondition:
      TCP is initialized.

    Parameters:
      None

    Returns:
      None
    ***************************************************************************/
  void GenericTCPServer(void)
  {
      BYTE i;
      WORD w, w2;
      BYTE AppBuffer[32];
      WORD wMaxGet, wMaxPut, wCurrentChunk;
      static TCP_SOCKET   MySocket;
      static enum _TCPServerState
      {
          SM_HOME = 0,
          SM_LISTENING,
        TCPServerState = SM_HOME;
```

```
    switch(TCPServerState)
    {
        case SM_HOME:
            // Allocate a socket for this server to listen and accept connections on
            MySocket = TCPOpen(0, TCP_OPEN_SERVER, SERVER_PORT,
TCP_PURPOSE_GENERIC_TCP_SERVER);
            if(MySocket == INVALID_SOCKET)
                return;

            TCPServerState   SM_LISTENING;
            break;

        case SM_LISTENING:
            // See if anyone is connected to us
            if(!TCPIsConnected(MySocket))
                return;


            // Figure out how many bytes have been received and how many we can
transmit.
            wMaxGet   TCPIsGetReady(MySocket);  // Get TCP RX FIFO byte count
            wMaxPut = TCPIsPutReady(MySocket);  // Get TCP TX FIFO free space

            // Make sure we don't take more bytes out of the RX FIFO than we can put
into the TX FIFO
            if(wMaxPut < wMaxGet)
                wMaxGet = wMaxPut;

            // Process all bytes that we can
            // This is implemented as a loop, processing up to sizeof(AppBuffer)
bytes at a time.
            // This limits memory usage while maximizing performance.  Single byte
Gets and Puts are a lot slower than multibyte GetArrays and PutArrays.
            wCurrentChunk = sizeof(AppBuffer);
            for(w = 0; w < wMaxGet; w += sizeof(AppBuffer))
            {
                // Make sure the last chunk, which will likely be smaller than
sizeof(AppBuffer), is treated correctly.
                if(w + sizeof(AppBuffer) > wMaxGet)
                    wCurrentChunk = wMaxGet - w;

                // Transfer the data out of the TCP RX FIFO and into our local
processing buffer.
                TCPGetArray(MySocket, AppBuffer, wCurrentChunk);

                // Perform the "ToUpper" operation on each data byte
                for(w2 = 0; w2 < wCurrentChunk; w2++)
                {
                    i = AppBuffer[w2];
                    if(i >= 'a' && i <= 'z')

                        i -= ('a' - 'A');
                        AppBuffer[w2] = i;


                // Transfer the data out of our local processing buffer and into the
```

```
TCP TX FIFO.
                TCPPutArray(MySocket, AppBuffer, wCurrentChunk);


            // No need to perform any flush.  TCP data in TX FIFO will automatically
transmit itself after it accumulates for a while.  If you want to decrease latency
(at the expense of wasting network bandwidth on TCP overhead), perform and explicit
flush via the TCPFlush() API.

            break;
    }


#endif //#if defined(STACK_USE_GENERIC_TCP_SERVER_EXAMPLE)
```

ANEXO G
PROGRAMA EN MICROCONTROLADOR PIC 18F67J60 CLIENTE TCP

```
/*********************************************************************
 *
    Generic TCP Client Example Application
 *  Module for Microchip TCP/IP Stack
     -Implements an example HTTP client and should be used as a basis
 *    for creating new TCP client applications
 *   -Reference: None.  Hopefully AN833 in the future.

 /*********************************************************************
#define __GENERICTCPCLIENT_C

#include "TCPIPConfig.h"

#if defined(STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE)

#include "TCPIP Stack/TCPIP.h"


// Defines the server to be accessed for this application
static BYTE ServerName[]    "www.google.com";

// Defines the port to be accessed for this application
static WORD ServerPort   80;

// Defines the URL to be requested by this HTTP client
static ROM BYTE RemoteURL[] = "/search?as_q=Microchip&as_sitesearch=microchip.com";


/*********************************************************************

 Function:
   void GenericTCPClient(void)

 Summary:
   Implements a simple HTTP client (over TCP).

 Description:
   This function implements a simple HTTP client, which operates over TCP.
   The function is called periodically by the stack, and waits for BUTTON1
   to be pressed.  When the button is pressed, the application opens a TCP
   connection to an Internet search engine, performs a search for the word
   "Microchip" on "microchip.com", and prints the resulting HTML page to
   the UART.

   This example can be used as a model for many TCP and HTTP client
   applications.

 Precondition:
   TCP is initialized.

 Parameters:
   None

 Returns:
   None
  ********************************************************************/
void GenericTCPClient(void)
{
    BYTE                i;
```

```
    WORD                w;
    BYTE                vBuffer[9];
    static TICK         Timer;
    static TCP_SOCKET   MySocket = INVALID_SOCKET;
    static enum _GenericTCPExampleState
    {
        SM_HOME = 0,
        SM_SOCKET_OBTAINED,
        SM_PROCESS_RESPONSE,
        SM_DISCONNECT,
        SM_DONE
      GenericTCPExampleState = SM_DONE;

    switch(GenericTCPExampleState)
    {
        case SM_HOME:
            // Connect a socket to the remote TCP server
            MySocket = TCPOpen((DWORD)&ServerName[0], TCP_OPEN_RAM_HOST, ServerPort,
TCP_PURPOSE_GENERIC_TCP_CLIENT);

            // Abort operation if no TCP socket of type
TCP PURPOSE_GENERIC_TCP_CLIENT is available
            // If this ever happens, you need to go add one to TCPIPConfig.h
            if(MySocket == INVALID_SOCKET)
                break;

            #if defined(STACK_USE_UART)
            putrsUART((ROM char*)"\r\n\r\nConnecting using Microchip TCP API...\r\n");
            #endif


            GenericTCPExampleState++;
            Timer = TickGet();
            break;

        case SM_SOCKET_OBTAINED:
            // Wait for the remote server to accept our connection request
            if(!TCPIsConnected(MySocket))
            {
                // Time out if too much time is spent in this state
                if(TickGet()-Timer > 5*TICK_SECOND)
                {
                    // Close the socket so it can be used by other modules
                    TCPDisconnect(MySocket);
                    MySocket = INVALID_SOCKET;
                    GenericTCPExampleState--;

                break;


            Timer   TickGet();

            // Make certain the socket can be written to
            if(TCPIsPutReady(MySocket) < 125u)
                break;

            // Place the application protocol data into the transmit buffer.  For
this example, we are connected to an HTTP server, so we'll send an HTTP GET request.
            TCPPutROMString(MySocket, (ROM BYTE*)"GET ");
```

```
                TCPPutROMString(MySocket, RemoteURL);
                TCPPutROMString(MySocket, (ROM BYTE*)" HTTP/1.0\r\nHost: ");
                TCPPutString(MySocket, ServerName);
                TCPPutROMString(MySocket, (ROM BYTE*)"\r\nConnection: close\r\n\r\n");

                // Send the packet
                TCPFlush(MySocket);
                GenericTCPExampleState++;
                break;

            case SM_PROCESS_RESPONSE:
                // Check to see if the remote node has disconnected from us or sent us
    any application data
                // If application data is available, write it to the UART
                if(!TCPIsConnected(MySocket))
                {
                    GenericTCPExampleState = SM_DISCONNECT;
                    // Do not break;  We might still have data in the TCP RX FIFO waiting
    for us


                // Get count of RX bytes waiting
                w = TCPIsGetReady(MySocket);

                // Obtian and print the server reply
                i = sizeof(vBuffer)-1;
                vBuffer[i] = '\0';
                while(w)
                {


                if(w < i)
                {
                    i = w;
                    vBuffer[i] = '\0';

                w -= TCPGetArray(MySocket, vBuffer, i);
                #if defined(STACK_USE_UART)
                putsUART((char*)vBuffer);
                #endif

                // putsUART is a blocking call which will slow down the rest of the
stack
                // if we shovel the whole TCP RX FIFO into the serial port all at
once.
                // Therefore, let's break out after only one chunk most of the time.
The
                // only exception is when the remote node disconncets from us and we
need to
                // use up all the data before changing states.
                if(GenericTCPExampleState == SM_PROCESS_RESPONSE)
                    break;


            break;

        case SM_DISCONNECT:
            // Close the socket so it can be used by other modules
            // For this application, we wish to stay connected, but this state will
still get entered if the remote server decides to disconnect
```

```
            TCPDisconnect(MySocket);
            MySocket = INVALID_SOCKET;
            GenericTCPExampleState = SM_DONE;
            break;

        case SM_DONE:
            // Do nothing unless the user pushes BUTTON1 and wants to restart the
whole connection/download process
            if(BUTTON1_IO == 0u)
                GenericTCPExampleState = SM_HOME;
            break;


#endif  //#if defined(STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE)
```

ANEXO H
PROGRAMA EN MICROCONTROLADOR PIC 18F67J60 APLICACIÓN HTTP

```
/*****************************************************************
 * HTTPPrint.h
 * Provides callback headers and resolution for user's custom
 * HTTP Application.
 *
 * This file is automatically generated by the MPFS Utility
 * ALL MODIFICATIONS WILL BE OVERWRITTEN BY THE MPFS GENERATOR
 *****************************************************************/

#ifndef __HTTPPRINT_H
#define __HTTPPRINT_H

#include "TCPIP Stack/TCPIP.h"

#if defined(STACK_USE_HTTP2_SERVER)

extern HTTP_STUB httpStubs[MAX_HTTP_CONNECTIONS];
extern BYTE curHTTPID;

void HTTPPrint(DWORD callbackID);
void HTTPPrint_hellomsg(void);
void HTTPPrint_cookiename(void);
void HTTPPrint_(void);
void HTTPPrint_builddate(void);
void HTTPPrint_led(WORD);
void HTTPPrint_lcdtext(void);
void HTTPPrint_ledSelected(WORD,WORD);
void HTTPPrint_version(void);
void HTTPPrint_btn(WORD);
void HTTPPrint_pot(void);
void HTTPPrint_an2(void);
void HTTPPrint_an3(void);
void HTTPPrint_temp(void);
void HTTPPrint_olimex_ext_test(void);
void HTTPPrint_uploadedmd5(void);
void HTTPPrint_status_ok(void);
void HTTPPrint_ddns_status(void);
void HTTPPrint_ddns_status_msg(void);
void HTTPPrint_ddns_service(WORD);
void HTTPPrint_ddns_user(void);
void HTTPPrint_ddns_pass(void);
void HTTPPrint_ddns_host(void);
void HTTPPrint_status_fail(void);
void HTTPPrint_smtps_en(void);
void HTTPPrint_config_mac(void);
void HTTPPrint_config_hostname(void);
void HTTPPrint_config_dhcpchecked(void);
void HTTPPrint_config_ip(void);
void HTTPPrint_config_gw(void);
void HTTPPrint_config_subnet(void);
void HTTPPrint_config_dns1(void);
void HTTPPrint_config_dns2(void);
void HTTPPrint_reboot(void);
void HTTPPrint_rebootaddr(void);
void HTTPPrint_snmp_en(void);
void HTTPPrint_read_comm(WORD);
void HTTPPrint_write_comm(WORD);
```

```
void HTTPPrint(DWORD callbackID)
{
    switch(callbackID)
    {
        case 0x00000000:
            HTTPIncFile((ROM BYTE*)"header.inc");
            break;
        case 0x00000001:
            HTTPPrint_hellomsg();
            break;
        case 0x00000002:
            HTTPIncFile((ROM BYTE*)"footer.inc");
            break;
        case 0x00000003:
            HTTPPrint_cookiename();
            break;
        case 0x00000004:
            HTTPPrint_();
            break;
        case 0x00000005:
            HTTPPrint_builddate();
            break;
        case 0x00000006:
            HTTPPrint_led(7);
            break;
        case 0x00000007:
            HTTPPrint_led(6);
            break;
        case 0x00000008:
            HTTPPrint_led(5);
            break;
        case 0x00000009:
            HTTPPrint_led(4);
            break;
        case 0x0000000a:
            HTTPPrint_led(3);
            break;
        case 0x0000000b:
            HTTPPrint_led(2);
            break;
        case 0x0000000c:
            HTTPPrint_led(1);
            break;
        case 0x0000000d:
            HTTPPrint_lcdtext();
            break;
        case 0x0000000e:
            HTTPPrint_ledSelected(4,TRUE);
            break;
        case 0x0000000f:
            HTTPPrint_ledSelected(4,FALSE);
            break;
        case 0x00000010:
            HTTPPrint_ledSelected(3,TRUE);
            break;
        case 0x00000011:
            HTTPPrint_ledSelected(3,FALSE);
            break;
```

```
case 0x00000012:
    HTTPPrint_ledSe ected(2,TRUE);
    break;
case 0x00000013:
    HTTPPrint_ledSelected(2,FALSE);
    break;
case 0x00000014:
    HTTPPrint_ledSelected(1,TR JE);
    break;
case 0x00000015:
    HTTPPrint_ledSelected(1,FALSE);
    break;
case 0x00000016:
    HTTPPrint_version();
    break;
case 0x00000017:
    HTTPPrint_led(0);
    break;
case 0x000000 8:
    HTTPPrint_btn(0);
    break;
case 0x00000019:
    HT PPrint_btn(1);
    break;
case 0x0000001a:
    HTTPPrint_btn(2);
    break;
case 0x000000 b:
    HTTPPrint_btn(3);
    break;
case 0x0000001c:
    HTTPPrint_pot();
    break;
case 0x000000 d:
    HTTPPrint_an2();
    break;
case 0x0000001e:
    HTTPPrint_an3();
    break;
case 0x0000001f:
    HTTPPrint_temp();
    break;
case 0x00000020:
    HTTPPrint_olimex_ext_test();
    break;
case 0x00000021:
    HTTPPrint_uploadedmd5();
    break;
case 0x00000022:
    HTTPPrint_status_ok();
    break;
case 0x00000023:
    HT PPrint_ddns_status();
    break;
case 0x00000024:
    HTTPPrin._ddns_status_msg();
    break;
case 0x00000025:
```

```
        HTTPPrint_ddns_service(0);
        break;
    case 0x00000026:
        HTTPPrint_ddns_service(1);
        break;
    case 0x00000027:
        HTTPPrint_ddns_service(2);
        break;
    case 0x00000028:
        HTTPPrint_ddns_user();
        break;
    case 0x00000029:
        HTTPPrint_ddns_pass();
        break;
    case 0x0000002a:
        HTTPPrint_ddns_host();
        break;
    case 0x0000002b:
        HTTPPrint_status_fail();
        break;
    case 0x0000002c:
        HTTPPrint_smtps_en();
        break;
    case 0x0000002d:
        HTTPPrint_config_mac();
        break;
    case 0x0000002e:
        HTTPPrint_config_hostname();
        break;
    case 0x0000002f:
        HTTPPrint_config_dhcpchecked();
        break;
    case 0x00000030:
        HTTPPrint_config_ip();
        break;
    case 0x00000031:
        HTTPPrint_config_gw();
        break;
    case 0x00000032:
        HTTPPrint_config_subnet();
        break;
    case 0x00000033:
        HTTPPrint_config_dns1();
        break;
    case 0x00000034:
        HTTPPrint_config_dns2();
        break;
    case 0x00000035:
        HTTPPrint_reboot();
        break;
    case 0x00000036:
        HTTPPrint_rebootaddr();
        break;
    case 0x00000037:
        HTTPPrint_snmp_en();
        break;
    case 0x00000038:
        HTTPPrint_read_comm(0);
```

```
            break;
        case 0x00000039:
            HTTPPrint_read_comm(1);
            break;
        case 0x0000003a:
            HTTPPrint_read_comm(2);
            break;
        case 0x0000003b:
            HTTPPrint_write_comm(0);
            break;
        case 0x0000003c:
            HTTPPrint_write_comm(1);
            break;
        case 0x0000003d:
            HTTPPrint_write_comm(2);
            break;
        default:
            // Output notification for undefined values
            TCPPutROMArray(sktHTTP, (ROM BYTE*)"!DEF", 4);
    }

    return;
}

void HTTPPrint_(void)
{
    TCPPut(sktHTTP, '~');
    return;
}

#endif

#endif
```

ANEXO I
ESPECIFICACIONES TÉCNICAS DE LA TARJETA PIC WEB

- PIC18F67J60 microcontroller
- 1Mbit on board serial flash for web pages storage
- ICSP/ICD connector for programming and debugging with PIC-ICD2, PIC-ICD2-POCKET and PIC-ICD2-TINY.
- Reset button
- User event button
- Analogue trimmer potentiometer
- Thermistor for temperature monitoring
- RS232 driver and connector
- Complete web server and TCP-IP stack support as per Microchip's open source TCP-IP stack
- Power plug-in jack for +5VDC power supply
- Voltage regulator +3.3V and filtering capacitors
- status LED
- Extension header to connect to other boards
- PCB: FR-4, 1.5 mm (0,062"), solder mask, silkscreen component print
- Dimensions 60x65 mm (2.36x2.55")

ANEXO J
ESPECIFICACIONES TÉCNICAS DEL MICROCONTROLADOR PIC 18F67J60

MICROCHIP

PIC18F97J60 FAMILY

64/80/100-Pin High-Performance, 1-Mbit Flash Microcontrollers with Ethernet

Ethernet Features:

- IEEE 802.3™ Compatible Ethernet Controller
- Fully Compatible with 10/100/1000Base-T Networks
- Integrated MAC and 10Base-T PHY
- 8-Kbyte Transmit/Receive Packet Buffer SRAM Supports One 10Base-T Port
- Programmable Automatic Retransmit on Collision
- Programmable Padding and CRC Generation
- Programmable Automatic Rejection of Erroneous Packets
- Activity Outputs for 2 LED Indicators
- Buffer:
  - Configurable transmit/receive buffer size
  - Hardware-managed circular receive FIFO
  - Byte-wide random and sequential access
  - Internal DMA for fast memory copying
  - Hardware assisted checksum calculation for various protocols
- MAC:
  - Support for Unicast, Multicast and Broadcast packets
  - Programmable Pattern Match of up to 64 bytes within packet at user-defined offset
  - Programmable wake-up on multiple packet formats
- PHY:
  - Wave shaping output filter

Flexible Oscillator Structure:

- Selectable System Clock derived from Single 25 MHz External Source:
  - 2.778 to 41.667 MHz
- Internal 31 kHz Oscillator
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if oscillator stops
- Two-Speed Oscillator Start-up

External Memory Bus (100-pin devices only):

- Address Capability of up to 2 Mbytes
- 8-Bit or 16-Bit Interface
- 12-Bit, 16-Bit and 20-Bit Addressing modes

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA on PORTB and PORTC
- Five Timer modules (Timer0 to Timer4)
- Four External Interrupt pins
- Two Capture/Compare/PWM (CCP) modules
- Three Enhanced Capture/Compare/PWM (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Up to Two Master Synchronous Serial Port (MSSP) modules supporting SPI (all 4 modes) and I^2C™ Master and Slave modes
- Up to Two Enhanced USART modules:
  - Supports RS-485, RS-232 and LIN/J2602
  - Auto-wake-up on Start bit
  - Auto-Baud Detect (ABD)
- 10-Bit, Up to 16-Channel Analog-to-Digital Converter module (A/D):
  - Auto-acquisition capability
  - Conversion available during Sleep
- Dual Analog Comparators with Input Multiplexing
- Parallel Slave Port (PSP) module (100-pin devices only)

Special Microcontroller Features:

- 5.5V Tolerant Inputs (digital-only pins)
- Low-Power, High-Speed CMOS Flash Technology:
  - Self-reprogrammable under software control
- C compiler Optimized Architecture for Reentrant Code
- Power Management Features:
  - Run: CPU on, peripherals on
  - Idle: CPU off, peripherals on
  - Sleep: CPU off, peripherals off
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 134s
- Single-Supply 3.3V In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with 3 Breakpoints via Two Pins
- Operating Voltage Range of 2.35V to 3.6V (3.1V to 3.6V using Ethernet module)
- On-Chip 2.5V Regulator

PIC18F97J60 FAMILY

Device	Flash Program Memory (bytes)	SRAM Data Memory (bytes)	Ethernet TX/RX Buffer (bytes)	I/O	10-Bit A/D (ch)	CCP/ ECCP	MSSP			EUSART	Comparators	Timers 8/16-Bit	PSP	External Memory Bus
								SPI	Master I^2C™					
PIC18F66J60	64K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N
PIC18F66J65	96K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N
PIC18F67J60	128K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N
PIC18F86J60	64K	3808	8192	55	15	2/3	1	Y	Y	2	2	2/3	N	N
PIC18F86J65	96K	3808	8192	55	15	2/3	1	Y	Y	2	2	2/3	N	N
PIC18F87J60	128K	3808	8192	55	15	2/3	1	Y	Y	2	2	2/3	N	N
PIC18F96J60	64K	3808	8192	70	16	2/3	2	Y	Y	2	2	2/3	Y	Y
PIC18F96J65	96K	3808	8192	70	16	2/3	2	Y	Y	2	2	2/3	Y	Y
PIC18F97J60	128K	3808	8192	70	16	2/3	2	Y	Y	2	2	2/3	Y	Y

ANEXO K
PANTALLA DE PRUEBA PIC WEB

C y Q Ingenieria e Instrumentacion!

PIC vercion: v5.00
Build Date: Aug 16 2011 10:25:10

PIC-WEB development board is used to demonstrate the power of an 8-bit embedded web server. This board is produced by Olimex Ltd.

More information about this board can be found here http://www.olimex.com/dev/pic-web.html.

LEDs: {click to toggle}

Buttons:
V V V ^

Potentiometer pot: 0

Potentiometer an2: 0

Potentiometer an3: 0

Temperature: 15

On the right you'll see the current status of the demo board. For a quick example, click the LEDs to toggle the lights on the board. Press the push buttons (except MCLR!) or turn the potentiometer and you'll see the status update immediately. This examples uses AJAX techniques to provide real-time feedback.

This site is provided as a tutorial for the various features of the HTTP web server, including:

- **Dynamic Variable Substitution** - display real-time data
- **Form Processing** - handle input from the client
- **Authentication** - require a user name and password
- **Cookies** - store session state information for richer applications
- **File Uploads** - parse files for configuration settings and more

ANEXO L
DATASHEET DEL TRANSMISOR DE PRESION DIFERENCIAL

Section 5 Specifications

FUNCTIONAL SPECIFICATIONS

Service
Liquid, gas, and vapor.

Range

Code
2:0–50 to 0–250 inH_2O (0–12.4 to 0–62.2 kPa).
3:0–200 to 0–1,000 inH_2O (0–49.7 to 0–248.6 kPa).

Output

Code
A:4–20 mA dc.
M:1–5 V dc, low power.

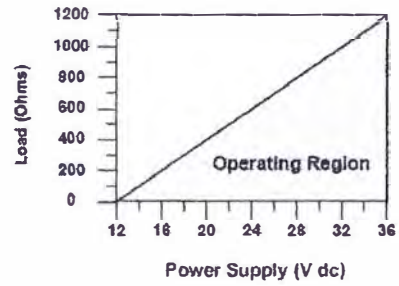
Power Supply
External power supply required.

Output Code A
Operates on 12 to 36 V dc, with no load.

Output Code M
Operates on 6 to 14 V dc.

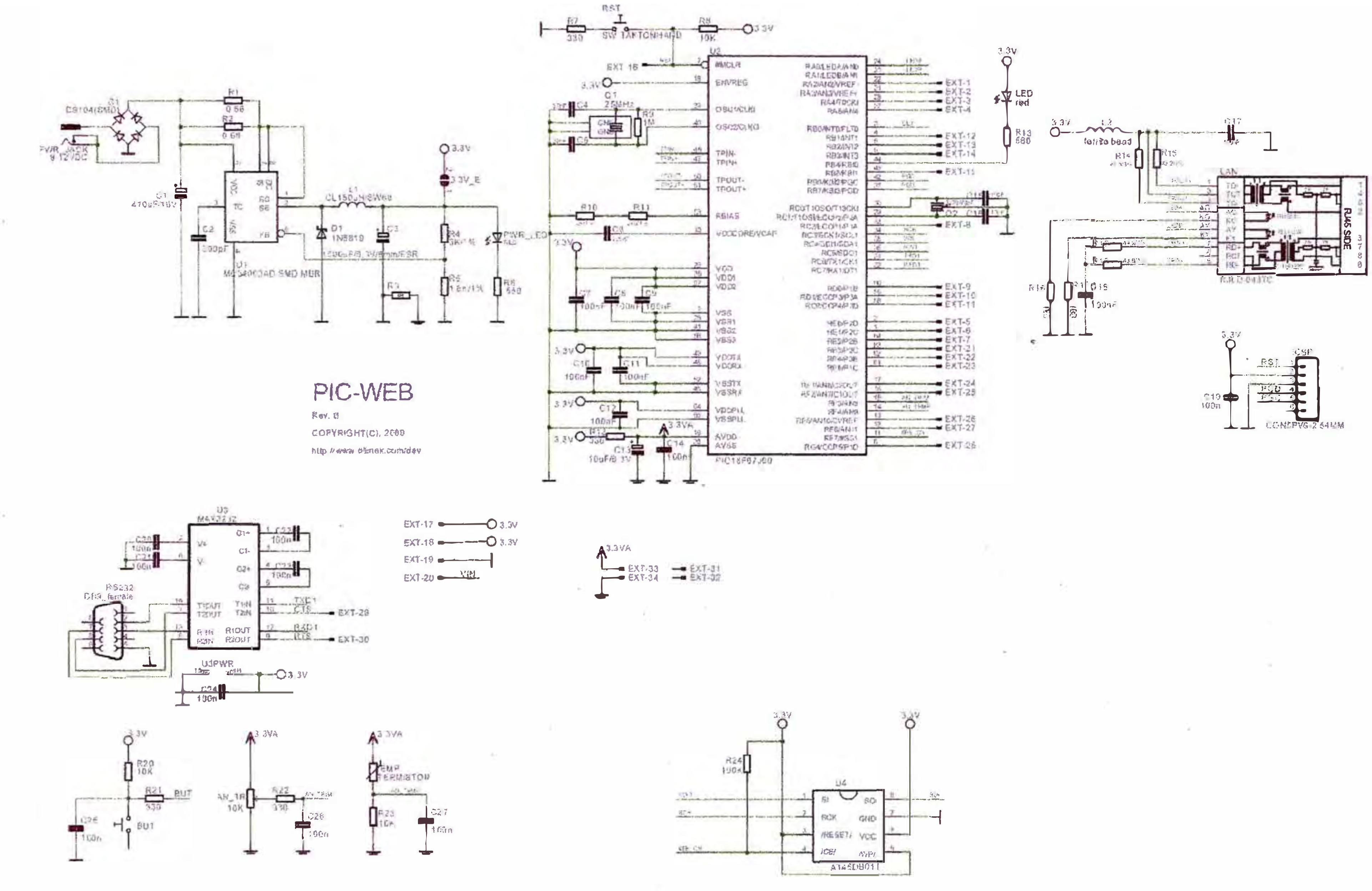
Load Limitations
Output Code A

Maximum Load = 50 × (Power Supply Voltage – 12)



Output Code M
Requires a minimum load impedance of 1 MΩ

ANEXO M
DIAGRAMA CIRCUITAL DE LA TARJETA PIC WEB



PIC-WEB
Rev. B
COPYRIGHT(C), 2000
http://www.olimex.com/dev
RST
R7
330
R8
10K
3.3V
EXT 16
ENVREG
OSC1/CLKI
OSC2/CLKO
TPIN-
TPIN+
TPOUT-
TPOUT+
RBIAS
VDDCORE/VCAP
VDD
VSS
VDDTX
VDDRX
VSSTX
VSSRX
VDDPLL
VSSPLL
AVDD
AVSS
PIC18F97J60
EXT-1
EXT-2
EXT-3
EXT-4
EXT-12
EXT-13
EXT-14
EXT-8
EXT-9
EXT-10
EXT-11
EXT-5
EXT-6
EXT-7
EXT-21
EXT-22
EXT-23
EXT-24
EXT-25
EXT-26
EXT-27
EXT-28
LED
red
R13
560
ferrite bead
LAN
RJ45 SIDE
ICSP
D1
1N5819
PWR_LED
MAX3232
RS232
DB9_female
EXT-29
EXT-30
U3PWR
EXT-17
EXT-18
EXT-19
EXT-20
3.3VA
EXT-33
EXT-34
EXT-31
EXT-32
R20
10K
R21
330
BUT
R22
330
C26
100n
R23
10K
C27
100n
R24
100K
U4
SI
SO
SCK
GND
/RESET/
VCC
/CS/
/WP/

BIBLIOGRAFÍA

[1] Antonio Creus Sole, "Instrumentación Industrial", séptima edición, Editorial Marcombo - España, 2005.

[2]José Acedo Sánchez, "Control avanzado de procesos, teoría y práctica", Editorial Diaz de Santos S.A. – Madrid, Espaňa, 2003.

[3] Philippe Mathon, "TCP/IP entorno Windows 2000", Editorial Eni. – Barcelona, España, 2001.

[4] Prof. Manuel Jimenez Buendía, "Comunicaciones Industriales", Dpto. de Tecnología electrónica, Universidad Politécnica de Cartagena.

[5] Fluke Corporation, "Multifunction Process Calibrator", Manual de uso Rev. 3, Washington, USA, 1998.

[6] James gosling, Bill Joy, Guy Steele, Gilad Bracha, "The java language specification second edition", sun Microsystems, Inc. – California, USA, 1996.

[7] Netbeans, "Entorno de desarrollo, ejemplos y aplicaciones", Web site: http://netbeans.org/

[8] Ángel cobo, Patricia Gómez. Daniel Pérez, Rocío rocha, "PHP y MySQL, tecnologías para el desarrollo de aplicaciones web", Editorial Diaz de Santos S.A. – Madrid, España, 2005.

[9] PIC WEB, "Manuales, guias, programas y aplicaciones", Web Site: http://www.olimex.com/dev/pic-web.html

[10] Microcontrolador PIC18F67J60, "Datasheet, aplicaciones y demos", Web site: http://www.microchip.com/

[11] Fabricante de registradores de procesos Monarch Instrument, "Datasheet, aplicaciones y demos", Web site: www.monarch instrument.com