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

Proyecto terminal

Control distribuido de los dispositivos eléctricos de un inmueble mediante TCP/IP

López Jarquín Aarón Israel
206202505

Trimestre 10-I

Dr. Rossen Petrov Popnikolov Potchinkov
Número económico: 12955

Agradecimientos

A Dios por permitirme cumplir una meta mas en mi vida, gracias a mis padres por el apoyo incondicional que me brindaron durante toda mi carrera, a mi asesor Dr.Rosen Petrov Popnikolov y al M. en C. Arturo Zúñiga López por haber participado no solo en este proyecto sino en mi desarrollo profesional ya que sin su ayuda y conocimientos no habría llegado hasta donde me encuentro ahora. Agradezco a ambos por su confianza, apoyo, comprensión, paciencia y sobre todo por su tiempo y enseñanza. También quiero agradecer a Marshall Lester | CEO, y Chief Technical Officer y a Ron Fienberg | Senior Software Engineer de Pulseworx por el material proporcionado, por su tiempo, interés y revisión de problemas durante el desarrollo y la implementación del proyecto terminal.
Y en general agradezco a todos los anteriores por su participación en dicho proyecto.

Con Dios está la sabiduría y el poder;
Suyo es el consejo y la inteligencia
Job 12:13

Contents

Objetivo general:

1- Diseñar y poner en operación un sistema que acople las tecnologías necesarias para controlar dispositivos eléctricos a través de la instalación eléctrica de un inmueble desde cualquier navegador, ubicado en cualquier punto del mundo.

Objetivos particulares:

1-. Integrar un microprocesador que pueda ser empleado como nodo de Internet, con el objetivo de controlar los dispositivos periféricos.
2-. Programar el microprocesador de manera tal que éste reciba los comandos remotos TCP/IP[1] y los transforme en señales comprensibles para los dispositivos periféricos acoplados a la red eléctrica.
3-. Integrar potenciómetros[2] a los dispositivos periféricos que emulen la actividad de sensores reales para ejemplificar la posible automatización de tareas.
4-. Diseñar e implementar una página WWW[3] básica para controlar los dispositivos periféricos remotamente, desde cualquier lugar del mundo.
5-. Elaborar la documentación correspondiente.

Introducción

Las nuevas tecnologías de la información y las comunicaciones por red eléctrica o por radiofrecuencia se han enlazado para permitir al hombre disfrutar de la comodidad y del tiempo de ocio sin que éste deba preocuparse del aseo, programar lavadoras, encender aparatos entre otras labores del hogar.
El confort es y seguirá siendo un punto importante para disminuir el incremento del stress que hemos experimentado al encontrarnos inmersos en el caos automovilístico; largas jornadas de trabajo; cambios ambientales etc. Todos estos factores nos han llevado a La búsqueda de comodidad en los hogares actuales y de automatización de grandes edificios comerciales, oficinas y residencias en diferentes países, tales como España, Japón y Estados Unidos, entre otros.

En la actualidad la domótica[4], es un concepto que se ha ido extendiendo a los pequeños negocios y a los entornos domésticos. En estos últimos ámbitos se pretende combinar un diseño razonable del edificio, utilizando técnicas arquitectónicas limpias y tradicionales, con el uso de tecnologías electrónicas que proporcionen los servicios más útiles a unos costos razonables.

La domótica puede definirse como la opción, integración y aplicación de las nuevas tecnologías informáticas y comunicativas al hogar. Incluye principalmente el uso de electricidad, dispositivos

---

[1] Protocolo de control de transmisión /Protocolo de Internet. Es un estándar de comunicaciones muy extendido, se utiliza a nivel mundial para realizar la conexión a internet y a servidores web.

[2] Resistor cuyo valor se puede ajustar para regular la intensidad de corriente en un circuito.

[3] Sistema de información con mecanismos de hipertexto o hipermedios enlazados y accesibles a través de Internet.
[4] Conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda.

electrónicos, sistemas informáticos y diferentes dispositivos de telecomunicaciones, incorporando la telefonía móvil e Internet. Algunas de sus principales características son: interacción, interrelación, facilidad de uso, teleoperación o manejo a distancia, fiabilidad y capacidad de programación y actualización. Su arquitectura puede ser centralizada o distribuida, aunque en realidad, por las ventajas de intercomunicación y ante los fallos, se emplea más la descentralizada. Los protocolos pueden ser estándar, es decir compatibles entre sí, y propietarios, que son los creados exclusivamente para un cliente o aplicación única. La configuración estándar cuenta con un sistema compuesto por ordenador u ordenadores, módem, tarjeta de sonido, dispositivos de ampliación de audio, baterías de emergencia, sondas de temperatura – exterior e interior -, detectores de humo, gas y agua, vídeo portero, sensores magnéticos para puertas y ventanas, detectores de presencia, mandos a distancia y emisores-receptores de señal.

Existen tres tipos de redes domóticas en el hogar según la infraestructura necesaria: las que utilizan nuevos cables, las que emplean los ya existentes – principalmente las redes eléctricas preexistentes – y las que se basan en sistemas inalámbricos o sin cables. Sus principales prestaciones o funciones son una mayor seguridad, la automatización y el telecontrol de los electrodomésticos y otros dispositivos, el acceso a los nuevos sistemas de telecomunicaciones y la superior disponibilidad de ocio y entretenimiento en casa. En todos los casos, existe una fuerte tendencia a hacer más cómoda y versátil la estancia en el lugar de la vivienda, al igual que se espera tener una mayor capacidad de gestión y monitoreo, tanto de los electrodomésticos como de los servicios públicos, donde destacan aspectos como el consumo, el gasto y el ahorro energético.

Se suele considerar que la domótica es una especie de disciplina emergente de interfase, en la que conjuntamente están implicados arquitectos, ingenieros eléctricos, electrónicos y civiles, programadores de sistemas y diseñadores. En su formación es recurrente que utilicen modelos de vivienda a escala, constituyéndose en un aspecto clave para aplicar y verificar las ventajas y posibilidades de los sistemas. En algunos casos, la formación en domótica dirigida a arquitectos, ingenieros y hasta promotores inmobiliarios, considera en su aplicación práctica las características sociodemográficas emergentes en términos de la estructura de la familia, incluyendo desde el cambio de papel de la mujer en el hogar hasta las condiciones económicas de sus ocupantes.
Es importante destacar que la automatización es parte de un sistema domótico, pero no lo es todo; un sistema domótico tiene asociado una inteligencia artificial, que analiza los estímulos (sensores) de la vivienda y toma una decisión, la más eficiente en cuestión de ahorro de recursos. [B0]

Últimamente se observa una tendencia pronunciada para realizar investigación en este campo y se integran nuevos objetivos que van mucho más allá de la climatización, del control de luces, persianas y audio.
La domótica genera diferentes tipos de servicios entre los cuales se encuentran la gestión de energía eléctrica, la ambientalización residencial, cuestiones de seguridad, audio distribuido, video, comunicación y, en general, todo el conjunto de elementos que favorecen la comodidad y la tranquilidad en el hogar.[B1]

Justificación

Actualmente existen empresas como HAI [B2] o Sirkom que se dedican a la automatización del hogar introduciendo *kits* completos que incluyen diversos dispositivos como son: sensores, cámaras, alarmas, y pantallas para controlar cada uno de los dispositivos con mayor facilidad, sin embargo los costos de los equipos y de instalación son elevados y el precio por la adquisición de pequeños *kits* no es inferior

a unos varios miles de dólares.

Por otro lado, la introducción de las computadoras y el impacto social que han tenido en la actualidad han provocado una evolución en el ser humano a nivel mundial. Las computadoras no sólo interactúan con nosotros acercándonos la información y por ende el conocimiento, sino que nos facilitan la resolución de varios problemas que involucran transportes, envíos de información, depósitos bancarios, reservaciones, compras, cotizaciones y, en general, la consulta de todo tipo de productos y servicios.
Por supuesto, la única forma de tener acceso a todos estos recursos es a través de la interconexión de diferentes redes. Internet representa una revolución informática en la que todos nosotros hemos participado. El crecimiento de Internet en los últimos años se ha dado de manera exponencial, al grado que las computadoras no sólo son un instrumento familiar sino que han emigrado al concepto de herramienta personal.

Las computadoras son unidades muy poderosas para propósitos generales, sin embargo es bueno mencionar que la gran mayoría de los dispositivos electrónicos no podrían tener una computadora con la potencia de procesamiento y el tamaño de una PC.
Por otro lado casi todo el equipo industrial, electrónico militar, médico entre algunos otros utiliza microprocesadores para llevar a cabo las tareas afines.
Cabe mencionar que desde la construcción del primer sistema digital a finales de 1940 se busco solucionar problemas científicos complejos pero no fue sino hasta 1950 cuando las aplicaciones computacionales se enfocaron en el negocio. La búsqueda de automatizar ciertas áreas con el fin de reducir el tiempo de operación de las mismas llevó a la investigación sobre nuevas tecnologías, protocolos, miniaturización etc.
En la actualidad existen microprocesadores que buscan cubrir estas necesidades, ya que se diseñan y desarrollan con el objetivo de reducir el consumo de energía y los costos de fabricación. Estos microprocesadores son circuitos integrados que se encargan eficientemente de tareas específicas y no de propósitos generales como las computadoras personales. Estos *chips* varían en procesador y memoria dependiendo la aplicación.

Actualmente, dentro de la Universidad Autónoma Metropolitana no se han realizado proyectos que involucren al protocolo UPB para controlar los dispositivos eléctricos debido a que ésta tecnología es una alternativa relativamente nueva, y por lo tanto no existía opción para elegir otro protocolo distinto a X.10. En cuanto a proyectos externos a la universidad, se encuentran los siguientes:
“Monitorización a distancia de una vivienda utilizando la red eléctrica” [B3], a diferencia de nuestro proyecto, éste se centra en la atenuación de la señal en función de la distancia, se analiza el protocolo *I2C*[5] y se propone un método para la conexión de sensores y actuadores dentro de una vivienda.
"El Protocolo x10: Una solución Antigua a Problemas actuales"[B4], a diferencia del proyecto propuesto, éste se concentra en la seguridad y envío de alertas usando tarjetas inteligentes para el envío de SMS[6].

Hoy en día, se busca reunir diversas tecnologías que nos permitan el control a distancia de dispositivos residenciales o industriales. La domótica, la computadora y el internet son algunas de las tecnologías que se acoplaron en este proyecto para tener mayor control sobre las instalaciones eléctricas de un inmueble y, sobre todo, los dispositivos que se conectan a ellas [B5] .

[5] Protocolo síncrono serial
[6] Acrónimo de envío de mensajes cortos.

Una de las muchas ventajas del acoplamiento mediante la instalación de la corriente alterna (AC)[7] existente es que no existen costos extras por modificaciones en la red eléctrica, lo cual hace de la automatización industrial o residencial una alternativa viable, de bajo costo y gran comodidad.[B6]

En el mercado se ofrecen muy pocos sistemas de control "centralizado" de los dispositivos eléctricos de un inmueble que reúnen estas características. Algunos, como el Stargate-IP [B7] no cumplen con la exigencia de tener un puerto Ethernet[8] y las funcionalidades del protocolo TCP/IP implementado [B8], por lo que se restringe el control del hogar solamente desde puntos internos del mismo.
Por otro lado, existe una central domótica llamada "Omnipro II" que reúne todas estas características pero también tiene la desventaja de que solamente el costo de la placa controladora es superior a los 1600 euros [B9], sin tomar en cuenta los módulos controladores de dispositivos periféricos, ni su instalación.

Debido a que la automatización del hogar está experimentando una gran expansión en las sociedades del primer mundo, es importante buscar otras alternativas menos costosas para ponerlas al alcance de las familias mexicanas.

A diferencia de la central domótica "Omnipro II", el proyecto terminal concluido muestra una alternativa de control accesible empleando otro tipo de controladores y dispositivos más económicos y con las mismas o mayores capacidades que su predecesor.

Se requirió que un ingeniero en computación o electrónica realizara el proyecto, ya que fueron necesarios conocimientos adquiridos en las aéreas de Arquitectura de computadoras Redes, microprocesadores, microcontroladores y lógica de programación. Todos estos conocimientos son adquiridos durante la carrera de ingeniería en computación.

Una posible continuación de este proyecto seria completar el sistema añadiendo un software para control interno en caso de no contar temporalmente son servicio de internet, intentando nuevamente reducir los costos que existen en la actualidad. Ampliar las capacidades del sistema agregando control de escenas, para aumentar la funcionalidad del mismo.
Otra posibilidad sería aumentar el número de dispositivos a controlar, agregar sensores para automatizar tareas relacionadas con el ambiente como la temperatura, precipitación, etc. Aumentar la funcionalidad agregando una bitácora o mecanismo de control de eventos.

Justificación de cambio del microprocesador

Debido al precio(55 dls el kit completo comparado con el rcm2200 de 85) y a la disponibilidad se halló un microprocesador viable, que nos convence de ser la mejor alternativa para el proyecto. El MCF51cn128 es un procesador de 32 bits de la familia Coldfire V1 de Freescale. A continuación se muestran las especificaciones técnicas del procesador.

En conclusión seleccionamos el TWR-MCF51CN128 debido a su constitución modular y expandible, la posible reconfiguración del hardware y la reutilización de módulos reducen el tiempo de diseño a bajo nivel, lo cual nos permite enfocarnos realmente en los objetivos del sistema; el bajo costo del microprocesador así como la flexibilidad del mismo hace posible la integración de hardware

[7] Corriente alterna, forma de corriente eléctrica donde la dirección de los electrones cambia en intervalos regulares
[8] Estándar de transmisión de datos para redes de área local.

prefabricado hecho a la medida física del TWR como es el caso de las pantallas LCD, memorias y sensores que son insertados en la propia torre.

Microprocesador MCF51CN128

Arquitectura de los microprocesadores ColdFire V1 de 32 bits

Las maquinas de 32 bits surgen como consecuencia de la saturación de las capacidades y desempeño de las antecesoras de 8 y 16 bits. Existen muchas empresas como Atmel y Freescale que están desarrollando soluciones de bajo costo y bajo consumo [B10a].

Características generales

Los tres principios importantes en el desarrollo de las máquinas ColdFire® V1 son:

- **Su núcleo:** De tamaño reducido, la más baja disipación de potencia y gran desempeño.
- **Periféricos y distribución de pines:** Compatibilidad total con las máquinas de 8 bits de la familia HCS08.
- **Segmentación (*Pipeline*):** Esta característica le proporciona al núcleo un alto desempeño.

La máquina ColdFire V1 viene implementada sobre una arquitectura de programación llamada ISA-C orientada al tratamiento de datos enteros de 32 bits, la cual exhibe entre sus atributos baja complejidad y costo.

Algunos aportes de la arquitectura ISA

- Soporta tratamiento de datos tipo byte, word y long, sobre instrucciones de movimiento y comparación.
- Posicionamiento de código independiente.
- Controlador de interrupciones
- Algunos tipos de operadores de manipulación de bits.
- Modelo simplificado del modo supervisor[9], conteniendo un manejo aparte de puntero a pila, un registro base para vectorización de eventos y un registro de configuración de la CPU.
- Soporta módulos opcionales como las de división de enteros, multiplicadoras/acumuladoras, unidad de aceleración criptográfica, entre otras.
- Respuesta programable ante la ejecución de código ilegal o decodificación de direcciones de memoria no implementadas. Estos eventos se pueden tratar como de excepción o de generación de RESET.
- Hasta 50 MHz de velocidad de procesamiento del núcleo sobre una tecnología de 0.25 micrones.
- Con una marca de 0.85 Dhrystones[10], 2.1 MIPS[11] cuando se ejecuta en FLASH y 1.05 DMIPS[12] cuando se ejecuta en RAM.

[9] Background mode (modo de puerta trasera).
[10] Dhrystone por segundo = reloj del procesador * número de pasadas / tiempo de ejecución
[11] Millones de instrucciones por segundo.

- FLASH de dos ciclos de acceso, con bajo consumo de energía.
- El controlador de la FLASH permite acceso por especulación como técnica de reducción de tiempos y eficiencia en la ejecución del programa del usuario.
- RAM con un ciclo de acceso, implementada en la plataforma del procesador sobre un bus de alta velocidad.

Arquitectura del núcleo

La característica más importante del núcleo del procesador ColdFire V1 es su *Pipeline*[13]. El núcleo está formado por dos estructuras independientes de *pipeline* con una interfaz de bus unificada, para maximización del desempeño con una reducción del *hardware* del núcleo,
lográndose una significativa reducción del costo.

Estructuras de segmentación

IFP[14]**:** Consiste de un cauce de dos niveles para la prebúsqueda de las instrucciones que luego son ingresadas al OEP, como se define a continuación.

OEP[15]**:** Consiste de un cauce de dos niveles, en el cual se decodifica la instrucción, se capturan los operandos desde el IFP y se ejecuta la función asociada a la instrucción.

Ambas estructuras están desacopladas mediante el empleo de un *buffer*, que opera como una lista tipo FIFO, en donde el IFP actúa como un dispensador en la pre-búsqueda de las instrucciones que serán servidas al OEP, minimizando la pérdida de tiempo por paros en la espera de instrucciones.

Etapas de la pre-búsqueda realizada por el IFP

- **IAG** [16] Etapa encargada de decodificar la instrucción a ser ejecutada.
- **IC**[17] Etapa de generación de los ciclos de reloj necesarios para la búsqueda de la instrucción.
- **IB**[18] Etapa para el almacenamiento en la lista FIFO de las instrucciones en espera de ejecución.

La estructura de ejecución OEP está implementada sobre un flujo tipo RISC[19], soportada por un registro de archivo de lectura dual conectado a la unidad aritmética y lógica.

Etapas OEP:

- **DSOC**[20] Ciclo de decodificación y selección de los operandos.

---

[12] Dhrystone MIPS
[13] Consiste en múltiples procesos ordenados de tal forma que el flujo de salida de un proceso alimenta la entrada del siguiente proceso.
[14] Instruction Fetch Pipeline (búsqueda de la instrucción).
[15] Operand Execution Pipeline (ejecucion de los operandos).
[16] Instruction Address Generation(generación de la dirección de la instrucción).
[17] Instruction fetch Cycle(ciclo de búsqueda de la instrucción)
[18] Instruction Buffer
[19] Reduced Instruction Set Computer
[20] Decode & Select Operand Cycle(ciclo de selección y decodificación de los operandos)

- **AGEX**[21] Ciclo de generación de la dirección después de ejecución. Determina la dirección destino, como resultado de la operación ejecutada.

Con el objetivo de minimizar el costo en la fabricación del circuito integrado, el bus de direcciones del núcleo de las máquinas V1 ha sido reducido de 32 bits a 24 bits. De lo anterior se deduce que la máxima capacidad direccionable por los procesadores ColdFire® V1 es de 16mb. La Figura 5.2 muestra un diagrama en bloques resumido de la arquitectura del núcleo de la máquina ColdFireV1 [B11a].

[21] Address Generation Execute Cycle(ciclo de ejecución de generación de la dirección)

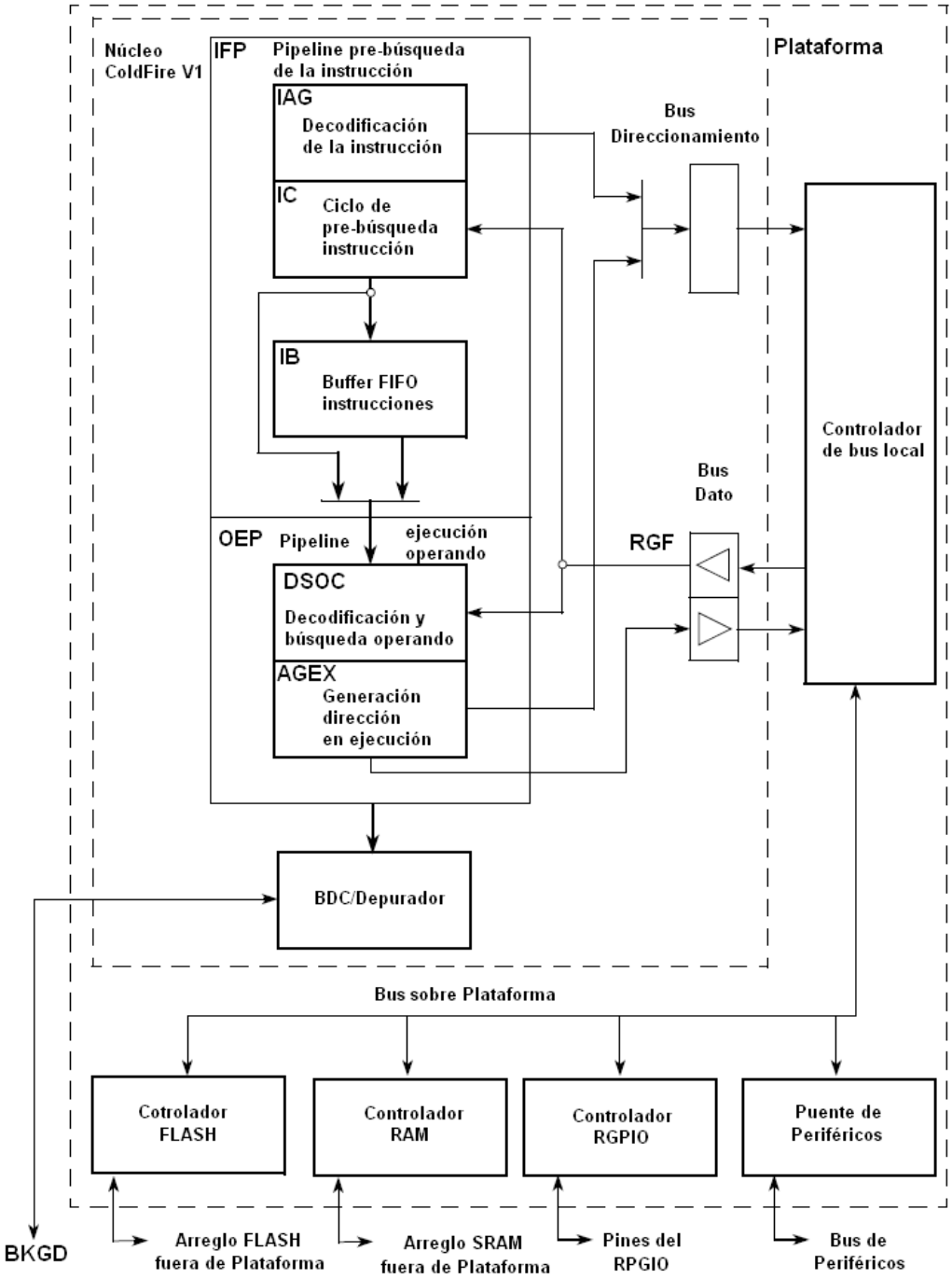


Figura 5.2. Núcleo de la familia ColdFire V1

MQX

Es importante destacar que el MCF51cn128 viene acoplado un sistema operativo llamado MQX.

MQX [12a], es un RTOS[22], este tipo de sistemas operativos administran el tiempo de un microprocesador o micro-controlador. Algunas de las características de un RTOS son las siguientes:

- Permite multitarea
- planificación tareas según prioridades
- Sincronización del acceso a los recursos
- Comunicación entre tareas
- Manejo de interrupciones

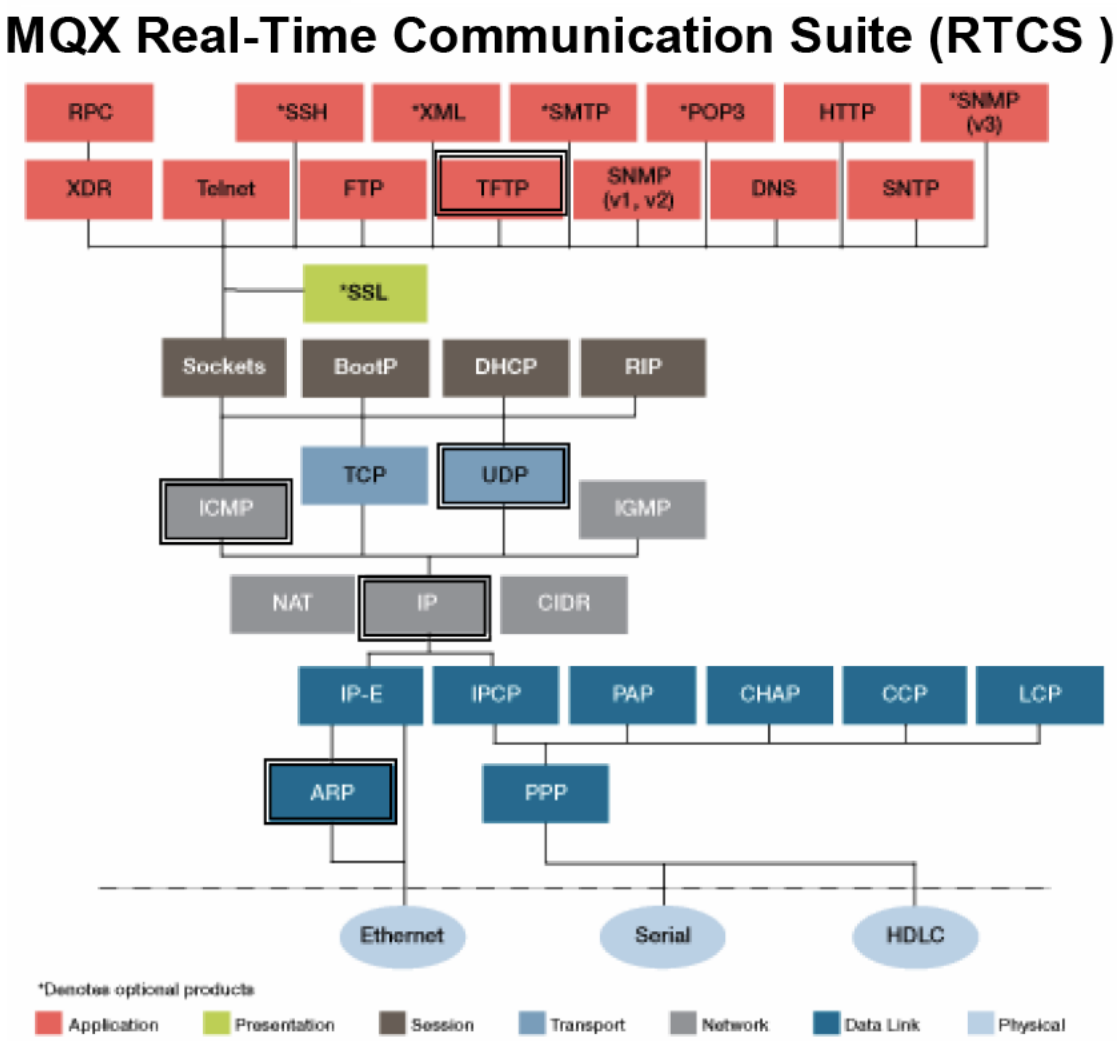


Figura 8.1 Suite de comunicación de tiempo real MQX

Como podemos observar en la figura 8.1[23], se muestra un esquema de las utilidades que posee nuestro S.O. en tiempo real dividido según colores conforme a las capas del modelo OSI. En la imagen

[22] Por sus siglas en ingles: real time operation system o sistema operativo en tiempo real.
[23] http://www.freescale.com/files/ftf_2010/Americas/FTF10_ENT_F0720.pdf

anterior se puede apreciar que el contenido de la Suite permite agregar servidores web, e-mail, administración de redes seguridad y ruteo entre otros.

El RTOS MQX fue diseñado específicamente para sistemas embebidos. Tomando en cuenta la memoria limitada y los requerimientos de respuesta en tiempo real de muchas de las aplicaciones embebidas. Su arquitectura provee al RTOS un conjunto completo de características lo suficientemente configurables para encajar en requerimientos de memoria muy pequeños.
Además el desempeño de las aplicaciones en tiempo real es una de las prioridades más importantes, el código es altamente optimizado dentro del contexto de manejo de interrupciones y switches y es usado para asegurar tiempos de respuesta rápidos.
MQX ofrece tecnología compatible con POSIX como es el caso del planificador de tareas, operación con hilos, y un moderno diseño de micro-kernel para paso de mensajes.

Este sistema operativo en tiempo real ha sido probado en arquitecturas de microprocesador embebidos como los ColdFire de Freescale y Power Architecture.
En la mayoría de sistemas embebidos se necesita que las aplicaciones respondan a eventos externos muy rápidamente, todos los servicios de tiempo critico han sido optimizados para conseguir velocidad y resultados deterministicos, las rutinas de servicio de interrupción han sido optimizadas para cada uno de los procesadores soportados y la secuencia de inicio asegura que la aplicación corre casi instantáneamente después de que el hardware ha sido reseteado.

MQX además provee drivers para manejo de periféricos en arquitecturas Coldfire y Power Architecture e incluye código ensamblado optimizado para aumentar el rendimiento de porciones específicas del RTOS. Dentro de MQX RTOS se encuentran muchas características importantes entre las cuales podemos destacar a las siguientes:

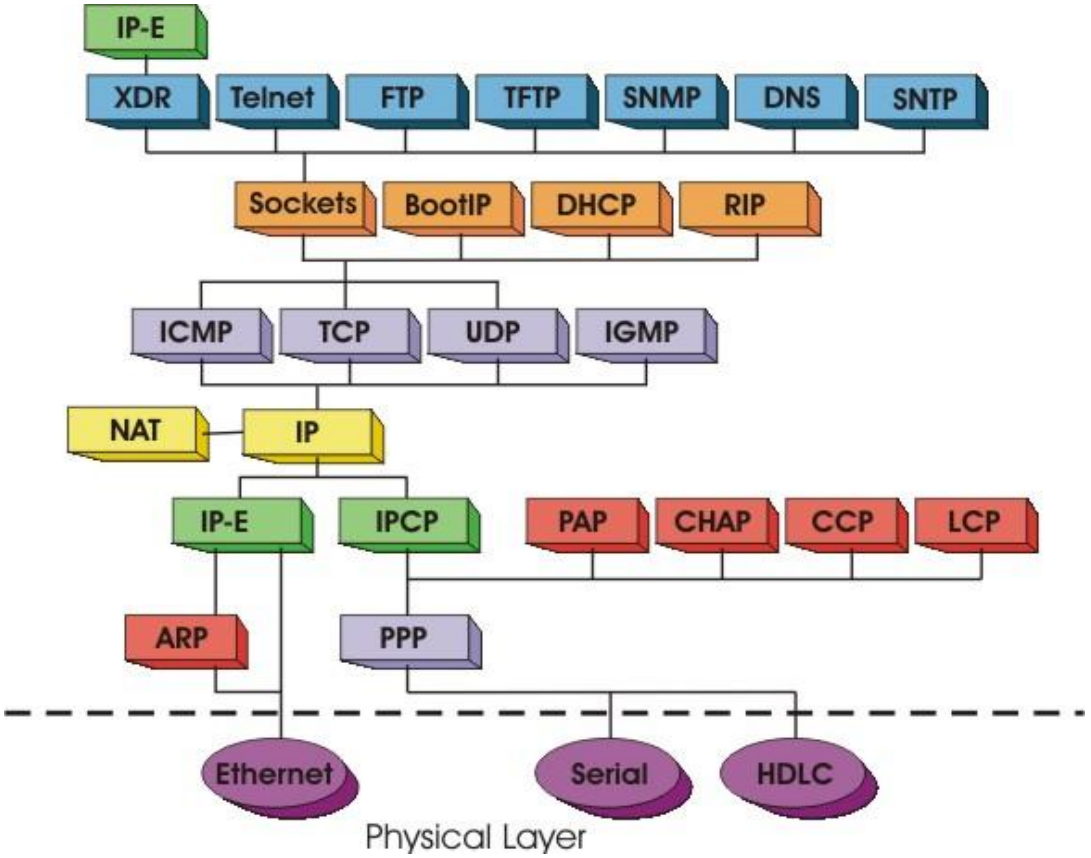
RTCS Stack de Internet embebido

Provee una gran cantidad de protocolos de aplicación para red TCP/IP y usa los drivers de MQX RTOS para llevar a cabo la conectividad a través del puerto serial o Ethernet. Esta altamente integrado con el sistema de archivos MFS, fue desarrollado y soportado por las herramientas de codewarrior. El RTCS fue implementado en ANSI C y el código fuente completo es proporcionado junto con el kit TWR.
El stack de internet soporta cualquier número de interfaces de hardware y cualquier número de direcciones IP en cada interfaz de hardware. Proporciona todas las características de configuración que podemos encontrar en PC y servidores pero adaptado a los confines de memoria de un dispositivo embebido. Al mismo tiempo se requiere que tengan un alto desempeño y eficiencia para asegurar que los sistemas embebidos pueden manejar grandes cantidades de tráfico en conexiones de banda ancha.

Protocolos soportados por RTCS

El stack es provisto con un extenso numero de protocolos estándar, esto permite a los desarrolladores instalar un solo producto que les permita desarrollar aplicaciones TCP/IP sin necesidad de adquirir

otros protocolos de nivel de aplicación haciendo a un lado la necesidad de perder tiempo integrando el protocolo adquirido a la base del stack TCP/IP.
El stack básico incluye soporte para los siguientes protocolos



Protocolos de Internet

- IP - Internet Protocol
- ARP - Address Resolution Protocol
- ICMP - Internet Control Message Protocol
- IGMP - Internet Group Management Protocol

Protocolos de acceso a la red

- Ethernet
- PPP - Point-to-Point Protocol
- HDLC - High-level Data Link Control

Protocolos de la capa de aplicación

- FTP - File Transfer Protocol
- TFTP - Trvial File Transfer Protocol
- Telnet - Telcommunication Network
- SNMP - Simple Network Management Protocol
- DHCP - Dynamic Host Configuration Protocol
- BootP - Bootstrap Protocol
- DNS - Domain Name System
- SNTP - Simple Network Time Protocol
- RPC- Remote Procedure Call

Protocolos de Transporte

- UDP - User Datagram Protocol
- TCP - Transmission Control Protocol
- RIP - Routing Information Protocol

Protocolos de red avanzados para RTCS

El stack RTCS puede ser extendido para soportar protocolos estándar de la industria con otros productos de red disponibles. Entre estos productos se encuentran aplicaciones que necesitan cumplir con requerimientos de conectividad específica.

- **Protocolos de seguridad:**
  - SSL - Secure Socket Layer
  - SSH - Secure Shell
- **Protocolos avanzados de ruteo y acceso a la red:**
  - NAT - Network Adress Translation
  - PPPoE - PPP over Etherent
- **Servidores Web embebidos y soporte para email:**
  - HTTP - Hypertext Transfer Protocol
  - SMTP - Simple Mail Transfer Protocol
  - POP3 - Post Office Protocol version 3
  - XML - Extensible Markup Language
- **Gestión de la red:**
  - SNMPv3 - Simple Network Management Proctocol version

HTTP

Usan un monto de recursos pequeño y son sumamente portables. Las herramientas incluyen el código fuente del servidor web y herramientas para la creación de páginas de internet.
El servidor web permite el almacenamiento en RAM, flash o en disco de páginas web embebidas, también tiene capacidades para cambiar paginas dinámicamente y mantener objetos dinamicos. Las páginas pueden ser protegidas por una contraseña de seguridad para acceso de lectura y escritura.

El servidor http básico es un simple motor de protocolo de transferencia de hipertexto que sirve contenido a los navegadores y clientes web, frecuentemente se utiliza para añadir conectividad a un dispositivo embebido. Es completamente compatible con las especificaciones HTTP 1.0/1.1, ofrece mejor desempeño y puede ser configurado para atender miles de solicitudes concurrentes.
El servidor proporciona acceso a una interfaz tipo CGI y sistema de archivos opcional para generar HTML dinámicamente.

El servidor http estándar además provee el servicio de objetos web HTML, GIF, JPEG, Applets, etc. Desde múltiples fuentes y sistemas de archivos agregando funcionalidad al anterior ofreciendo la inserción de variables dinámicamente en la pagina HTML. Dentro de las características avanzadas podemos encontrar otras como: Softpages la cual permite cambiar contenido HTML en las páginas de un dispositivo sin necesidad de recompilar la imagen del programa, host remotos el cual provee un proxy HTTP para extracción de objetos remotos desde otros servidores y java graphlets es una seria de applets de java que proporciona indicadores de control de gráficos para dispositivos embebidos.

MFS

MFS es un sistema de archives embebido tipo FAT compatible con Microsoft Windows y MS-DOS. El sistema de archivos tiene capacidades de formato, lectura, escritura e intercambio de archivos con los sistemas operativos compatibles que soporten los sistemas de archivos FAT-12, FAT-16 o FAT-32.

El sistema de archives MFS FAT fue agregado a las capacidades de MQX para añadir soporte a las aplicaciones. Tomando en cuenta la memoria limitada y los requerimientos de las aplicaciones embebidas. Este sistema de archivos es completamente compatible con MS-DOS y esta diseñado para trabajar en ambientes multihilo ya que proporciona todas las características que se encuentran en una PC convencional como puede ser soporte para nombres de archivos largos, múltiples volúmenes de disco, manejo de directorios, también es posible abrir múltiples archivos simultáneamente, provee una API estándar así como control para detección y recuperación de errores.

Protocolo UPB

Para poder enviar un mensaje UPB es necesario enviar un comando del Host (MCF51CN) al PIM (módulo UMCDB9. Ver imagen 2.1)
para solicitar la transmisión del mensaje UPB a la red eléctrica.
El PIM validará el mensaje y si pasa la prueba del checksum enviará una respuesta “PIM Accept” al host y transmitirá el mensaje a la corriente. En caso contrario enviará una respuesta “PIM Error”.

[B13a].

Para poder enviar un mensaje valido veremos cuál debe ser la estructura del mensaje UPB para que nuestros comandos solicitados sean transmitidos satisfactoriamente.

Estructura del mensaje

El paquete UPB está compuesto por un preámbulo (Preamble), un encabezado (Header), datos del mensaje (UPB Message) y un verificador de redundancia (Checksum). En la figura 4 podemos observar la composición del paquete de comunicación así como la cantidad de bytes que integran cada segmento.

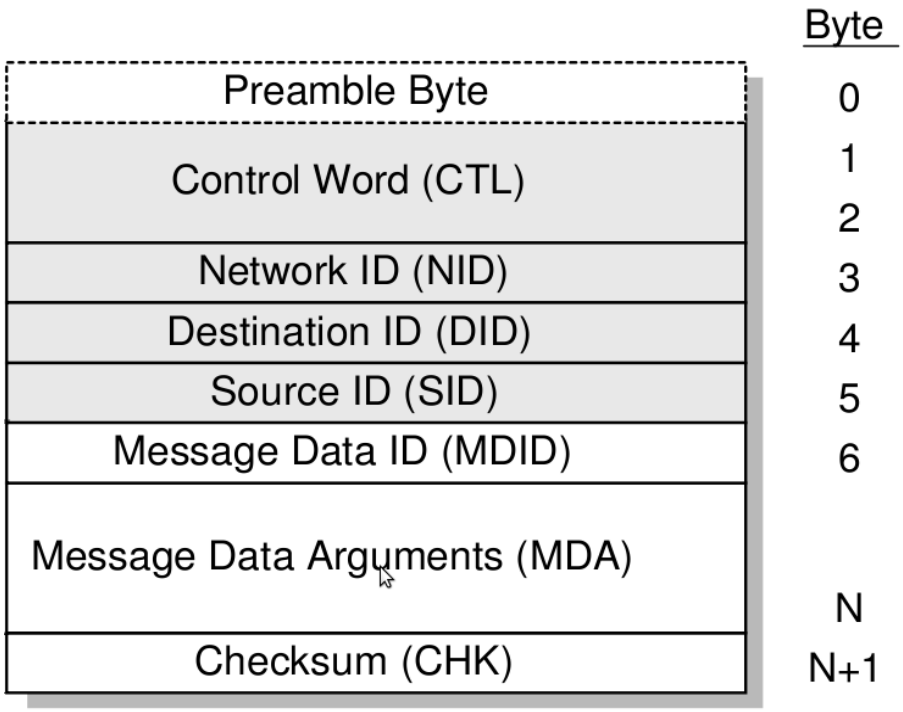


Figura 4. Paquete de comunicación UPB

Composición del paquete UPB

Preambulo:

El preambulo esta compuesto por una secuencia de 8 bits identificada por los pulsos 2,1,1,2 que se utilizan para brindar a los modulos UPB una mejor idea acerca del tamaño y las posiciones de los pulsos subsecuentes(pulsos que seran transmitidos a lo largo del paquete). Los pulsos ya estan preestablecidos y seran los mismos para indicar el inicio de cualquier mensaje transmitido.
Gracias a este modo de sincronización los modulos receptores pueden hacer ajustes de tiempo para llevar a cabo una mejor recepcion del mensaje. (ver figura 4.1)

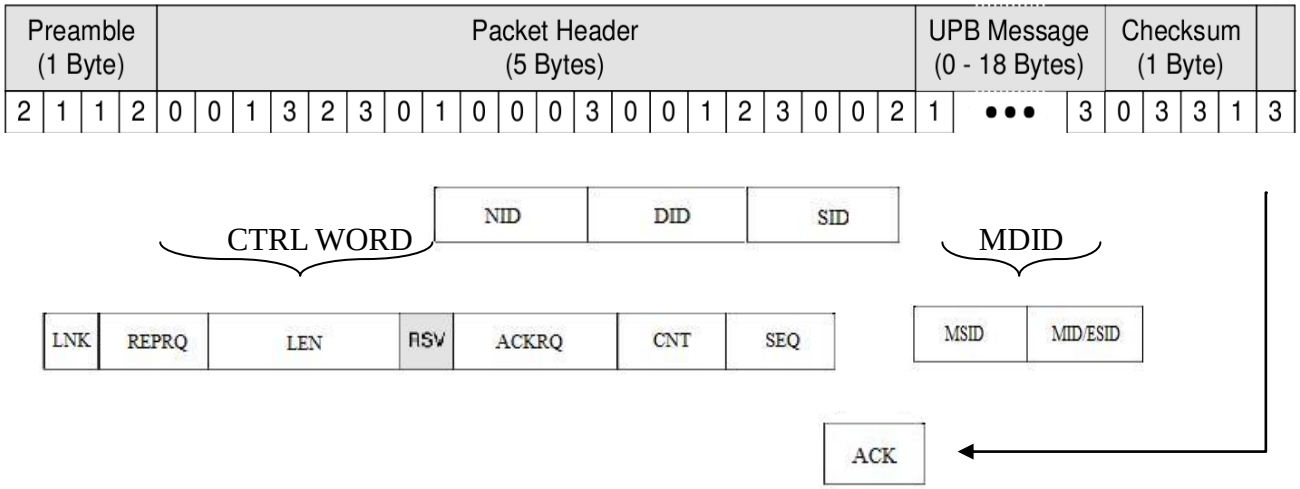


Figura 4.1

Encabezado del paquete:

Como se observa en la figura 4 y 4.1 el encabezado esta comprendido por 5 bytes. La informacion comprendida por el encabezado engloba detalles acerca de: quien envia el paquete, a quien va dirigido, en que red se encuentra y cual es el tamaño total del paquete. La composicion interna del Header o Encabezado esta organizada de la siguiente manera:

Palabra de control:

Está compuesta por 2 bytes y 7 subdivisiones en total como se muestra en la figura 4.2. explicaremos cada subdivision para identificar como funciona realmente la palabra de control del protocolo UPB

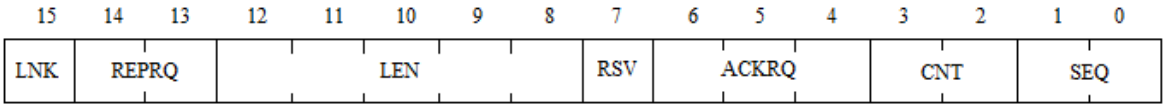


Figura 4.2 Palabra de control

LNK:

Longitud del campo: 1 bit.

Descripcion: si LNK=1 entonces el comando que queremos enviar tiene como objetivo comunicarse con un grupo de dispositivos que comparten un mismo identificador grupal o identificador de liga, en este caso el campo que corresponde al identificador de la direccion destino(ver figura 4.1 campo DID) es interpretado como identificador grupal. Cada dispositivo UPB puede tener asignado un ID de grupo con un rango entre 1 y 250. Por otro lado si LNK=0 implica que deseamos enviar un paquete directo, es decir buscamos la comunicación con un solo dispositivo y por lo tanto el campo DID es interpretado como ID del modulo.

REPRQ:
Longitud del campo: 2 bits
Este apartado sera puramente informativo ya que dentro del proyecto en cuestion siempre tomara el valor de 00 debido a que los repetidores no son usados comunmente en aplicaciones residenciales sino que tieneden a utilizarse mas en ambitos industriales y comerciales.
Descripcion: Es un campo de solicitud de repeticion que se usa para solicitar a un dispositivo especial llamado repetidor que reenvie el paquete cierta cantidad de veces según el nivel de repeticion deseado.
Los dispositivos repetidores solo reciben paquetes que contengan solicitud de repeticion, los retransmiten y los transforman en paquetes sin solicitud de repeticion. Los posibles valores que puede tomar el campo antes mencionado son los siguientes:

- 00 => 0 repeticiones
- 01 => 1 repeticion
- 10 => 2 repeticiones
- 11 => 4 repeticiones

LEN:
Longitud del campo: 5 bits.
Descripción: Indica el numero de bytes en todo el paquete sin incluir el preambulo.
Rango de la longitud del paquete: 6-24 bytes.
Nota: En caso de que la longitud este fuera del rango los receptores rechazaran el paquete.

RSV:
Longitud del campo: 1 bit.
Descripcion: siempre se establece un cero ya que esta reservado para un uso en el futuro

ACKRQ:
Longitud del campo: 3 bits.
Descripcion: se utiliza para solicitar confirmaciones de los paquetes enviados a los dispositivos.
El paquete de comunicación posee un campo opcional al final del mismo llamado campo de confirmacion en el cual el dispositivo emisor no genera ningun pulso.
El dispositivo receptor tiene disponible dicho espacio para generar un pulso especial llamado ACK que permite verificar la correcta recepcion del comando.
Existen 3 tipos de confirmaciones en este campo:

- Pulso ACK:
  La confirmacion implica generar un pulso aislado en la posicion 3 al final del paquete.
  Valor binario que recibe el campo para llevar a cabo la operación anterior: 001.
- Pulso ID:

  De la misma manera que en el anterior se genera un pulso en la posicion 3 que corresponde con el DID (identificador del dispositivo receptor, ver figura 4.1) es decir que si el dispositivo receptor tiene ID = 30 entonces generara su pulso ID=30 medios

ciclos
Valor binario que recibe el campo para llevar a cabo la operación anterior: 010.

- Mensaje ACK:
  El dispositivo que recibe el comando lleva a cabo la transmision de un mensaje completo despues de recibir el paquete.
  Valor binario que recibe el campo para llevar a cabo la operación anterior: 100.

Nota: el paquete enviado puede solicitar 0,1,2 o los 3 tipos de confirmacion

CNT:
Longitud del campo: 2 bits.
Descripcion: funciona como valor limite para indicar el numero de veces que el paquete sera transmitido, cada paquete se puede enviar un maximo de 4 veces, al llevar a cabo un reenvio del comando en multiples ocaciones incrementamos las posibilidades de que el dispositivo destino reciba el paquete de manera correcta. Los posibles valores que puede tomar el campo son los siguientes:

- 00 => 1 transmision
- 01 => 2 transmisiones
- 10 => 3 transmisiones
- 11 => 4 transmisiones

SEQ:
Longitud del campo: 2 bits.
Descripcion: el campo se utiliza para indicar el numero de transmision actual es decir en que numero de repeticion nos encontramos si es que el paquete se transmitira mas de una vez.

- 00 => 1ª transmision del paquete
- 01 => 2ª transmision del paquete
- 10 => 3ª transmision del paquete
- 11 => 4ª transmision del paquete.
  Para evitar confusiones con el reenvio del mismo paquete los campos SEQ/CNT proveen una forma de identificacion de multiples transmisiones que el dispositivo receptor puede interpretar de la siguiente manera:

  SEQ (transmision numero X) /CNT (de un total de Y transmisiones).
  Donde si X=Y indica que la siguiente transmision implica un mensaje nuevo.

- **NID** o Identificador de la red:
  Longitud: 1 byte
  Rango de posibles identificadores: (0-255)
  Descripcion: indica a que red UPB se trata de enviar el paquete.
  Existe un valor reservado para todas las redes, un valor global 0 que indica que el paquete esta dirigido a todos los dispositivos UPB.

- **DID** o Identificador del modulo receptor:
  Longitud: 1 byte
  Rango de posibles identificadores: (1-250)
  Valores reservados para tareas especificas: 0,251-255
  Descripcion: indica a que dispositivo(s) va dirigido el paquete. Este campo tiene dos interpretaciones distintas dependiendo del estado en el que se encuentre el bit LNK como se menciono anteriormente.
  En caso de que el bit LNK sea 1 el campo DID es grupal, en caso contrario el campo DID es simplemente el identificador de la unidad destino. Esta organización produce un crecimiento en el direccioamiento con un total de 512 direcciones, 256 para direccionamiento de conjuntos y 256 para direccionamiento directo.

- **SID** o Identificador del dispositivo fuente
  Esta compuesta por 1 byte
  Rango de posibles identificadores: (1-250)
  Descripcion: indica de donde o de quien proviene el paquete enviado.

Es importante observar que el numero maximo de dispositivos a direccionar en una red al utilizar el protocolo UPB es:
255 x 250 = 63750
Este rango de direccionamiento permite que podamos tener una gran cantidad de dispositivos electricos conectados mediante modulos UPB para poder automatizar grandes empresas y por ende llevar a cabo un ahorro de energia considerable.

Mensaje UPB (MDID):

Es un campo de longitud variable de 0-18 bytes .que se compone de un MDID (identificador de datos del mensaje) y un MDA(argumentos del mensaje). Algunos comandos no requieren de argumentos sin embargo algunos requieren directamente de ellos.

MDID

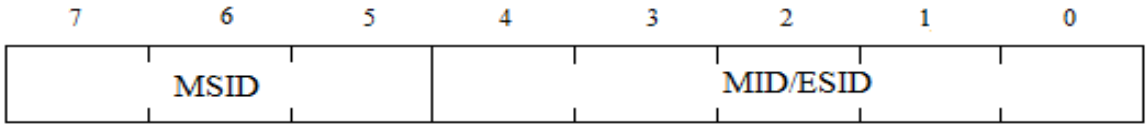


Figura 5.1 Byte de identificación de datos del mensaje

Longitud : 1 byte
El byte se subdivide en 2 campos como se muestra en la Tabla 5.1

- MSID
  Longitud:3 bits (7-5)
  Descripcion:funciona como identificador de una de las 8 diferentes categorias de mensajes que se han establecido(ver figura 5.2).
- MID/ESID
  Longitud:5 bits (4-0)
  Descripcion: MID se utiliza como identificador de un comando en particular dentro de la categoria seleccionada en el campo MSID. Mientras que ESID se utiliza en caso de que MSID tenga el valor “111” implicando asi la utilizacion de un conjunto de comandos extendido que puede ser definido por cualquier usuario que tenga necesidades especificas y que no se encuentren dentro de los comandos prestablecidos, su capacidad maxima es de 32 comandos extras que se pueden agregar posteriormente.

MSID	Descripcion
000	Conjunto de comandos generales para todos los dispositivos.
001	Conjunto de comandos utilizado para controlar la mayoria de los dispositivos.
010 y 011	Reservados para uso futuro.
100	Conjunto de reporte de mensajes generales para todos los dispositivos.
101 y 110	Reservados para uso futuro.
111	Subconjunto extendido que permite definir comandos según necesidades particulares.

Tabla 5.1 categorias de mensajes

MDA

Longitud : variable 0-17 bytes
Descripcion: campo opcional que se encuentra despues del mensaje y que contiene argumentos para especificar su ejecucion .

Checksum:

Longitud : 1 byte
Descripcion: se utiliza para verificar la integridad del paquete enviado. Se utilizan operaciones aritmeticas para coprobar que el envio es correcto, en este caso el procedimiento para formar el byte checksum se muestra a continuacion:

1-. Suma de los bytes del header + suma de los bytes del mensaje UPB, esto significa que se suman todos los bytes del paquete sin incluir el preambulo.
2-. Obtener el complemento a dos del resultado obtenido.
3-. Truncar la operación anterior a 8 bits.

Por lo tanto el dispositivo receptor puede comprobar la integridad del paquete recibido al hacer una suma de los bytes del header, del mensaje UPB y del checksum. Esta operación obtendra el resultado de cero si la transmision fue correcta y cualquier otro numero en caso contrario. La explicacion es muy sencilla ya que si sumamos un numero cualquiera con su representacion en complemento a 2, es como si restaramos el mismo numero es decir el resultado siempre sera cero.

CodeWarrior 6.2

CodeWarrior Development Studio es un entorno de desarrollo integrado completo (IDE) que provee un marco de trabajo visual y automatizado para acelerar el desarrollo de las aplicaciones embebidas más complejas. El IDE es una aplicación de software que integra la mayoría de las herramientas de programación en una única pieza de software.
Codewarrior incluye editor, ensamblador, compilador C/C++ optimizado, enlazador, simulador, programador, depurador y Processor Expert, una herramienta que automatiza la generación de código, configuración de periféricos, dispositivos externos y algoritmos dentro del ambiente para los microprocesadores Freescale.

Módulo PIM o UMC.

Funcionamiento

El modulo de interfaz de con la computadora, modelo UMC (Ver imagen 2.1) es usado para intercambiar comandos digitales entre una PC (computadora personal) o servidor y un dispositivo UPB dentro de un inmueble. El modulo comunica estos comandos sobre el cableado de la energía eléctrica. El modelo está disponible con interfaz UMC-DB9: Serial RS-232. Los dispositivos UPB pueden colocarse con libertad en cualquier parte de la casa, ningún cableado adicional es requerido y no se usan señales de radio frecuencia.

Operación

El modulo UMC intercambia mensajes entre la red eléctrica y la computadora o controlador conectado. El indicador del modulo se pone de color ámbar cuando esta encendido y se pone en rojo durante el envío de mensajes.
La interfaz tiene dos formas de operación: modo mensaje y modo pulso. El último modo de operación es reservado durante la interrupción de la energía.

El software UPStart automáticamente establece su interfaz en modo pulso, pero la mayoría de los controladores requieren modo mensaje para poder operar. Para restaurar el modo de operación de mensaje se realizan los siguientes pasos:

1. Colocar el modulo UMC en modo “Setup” presionando el *switch* que se localiza justo en la parte superior del Led indicador 5 veces consecutivas rápidamente usando un palillo no metálico . Posteriormente el Led parpadeara en rojo.
2. restablecer a modo mensaje presionando 10 veces el *switch*. En este caso el indicador parpadeara en color verde.

3. Presionar el *switch* dos veces para salir del modo SETUP.

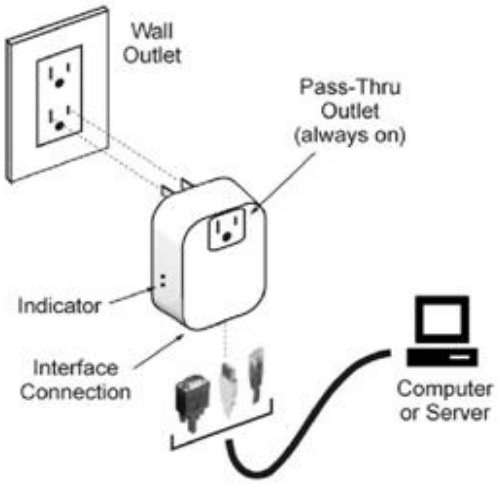


Imagen 2.1 Diagrama de conexión del módulo de interfaz domótico [24] (HOST/PIM)

Módulo UPB modelo UMA

Funcionamiento

El modulo eléctrico modelo UMA (Ver imagen 2.2) enciende o apaga los aparatos y lámparas conectadas a una toma de 120 VAC basándose en comandos digitales recibidos a través del cableado eléctrico y enviados por un controlador remoto UPB (PIM). Los controladores y módulos pueden ser localizados libremente sobre cualquier punto de la instalación eléctrica de igual manera no se usan señales de radiofrecuencia y no se necesita cableado adicional para la comunicación.

Instalación

El modulo fue diseñado para el uso de dispositivos que se conectan directamente a una toma de corriente de pared. Para instalar el modelo UMA se deben seguir los siguientes pasos:

1. Localizar el dispositivo o lámpara que se desea controlar y desenchufarlo de la toma. Conectar el modulo UMA a la toma de la pared (ver figura 2.2).
2. Enchufar el el dispositivo o lámpara en el socket que se encuentra en la parte inferor del modulo UMA tomando en cuenta que el dispositivo no debe exceder el nivel de 15 Amps.
3. Activar el *switch* del dispositivo lámpara en posición de encendido.
4. Si se desea se puede agregar un dispositivo a la parte frontal y enchufarlo al modulo UMA donde dicho modulo tendrá un acceso directo a la corriente y siempre estará encendido.

Configuración

Para poder configurar se requiere un software de configuración (UPStart) y un modulo PIM
La configuración es mediante un *wizard* y es necesario colocar el modulo UMA en modo "*setup*"
Para establecerlo es necesario presionar el *switch* 5 veces repetidamente con un instrumento no metálico. Para salir de este modo presionar una vez el *switch* o esperar 5 minutos.

Operación

El modulo opera de acuerdo con los comandos enviados por el controlador. El indicador del modulo se pone en rojo cuando la salida controlada está apagada y en verde cuando la salida está en encendido
La función de timer está disponible para le modelo UMA. Los comandos pueden encender el UMA para cargar el periodo de tiempo, automáticamente el modulo se apagara al final del periodo, es importante que esta característica no se va a explotar en el proyecto pero es interesante mencionarla.

[24] Módulo de interfaz UMC o módulo PIM imagen tomada de **www.simply-automated.com**

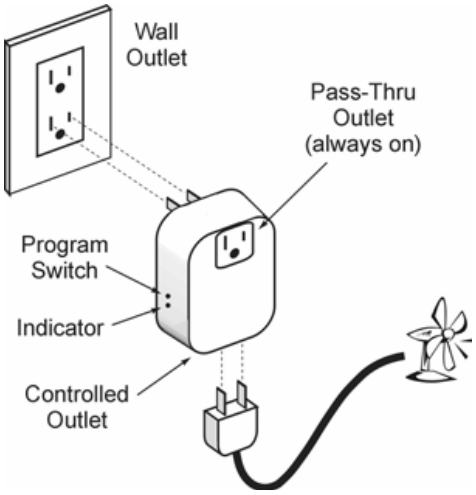


Imagen 2.2 Diagrama de conexión del modulo UMA

Interfaz serie RS232 con conector DB9

Es una interfaz (Ver imagen 3.1) que designa una norma para el intercambio serie de datos binarios entre un Equipo terminal de datos (DTE) y un Equipo de Comunicación de datos (DCE).
La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full dúplex.
Nota: en nuestro proyecto tanto el modulo PIM como el microprocesador son dispositivos DCE por lo tanto es necesario cruzar los cables para el caso de transmisión y recepción como se muestra en la figura 3.2

Número de clavija	Nombre
1	CD: Detector de transmisión
2	RXD: Recibir datos
3	TXD: Transmitir datos
4	DTR: Terminal de datos lista
5	GND: Señal de tierra
6	DSR: Ajuste de datos listo
7	RTS: Permiso para transmitir
8	CTS: Listo para enviar
9	RI: Indicador de llamada

Imagen 3.1 conector DB9 macho y hembra utilizado para conexión serie [25]

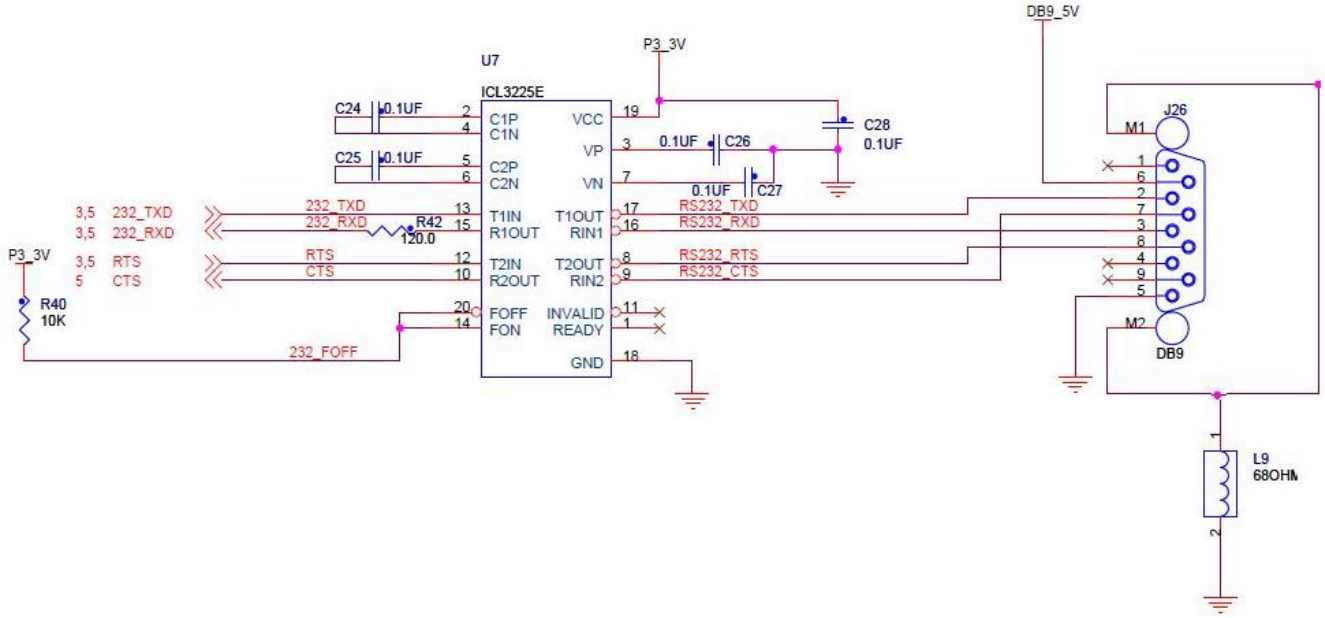
Crossover Cable DB9 to DB9

DCE Device **DB9** — **DCE to DCE Connections** — DCE Device **DB9**

Pin# DB9	RS-232 Signal Names		Signal Direction	Pin# DB9	RS-232 Signal Names	
#1	Carrier Detector (DCD)	CD		#1	Carrier Detector (DCD)	CD
#2	Receive Data (Rx)	RD		#2	Receive Data (Rx)	RD
#3	Transmit Data (Tx)	TD		#3	Transmit Data (Tx)	TD
#4	Data Terminal Ready	DTR		#4	Data Terminal Ready	DTR
#5	Signal Ground/Common (SG)	GND		#5	Signal Ground/Common (SG)	GND
#6	Data Set Ready	DSR		#6	Data Set Ready	DSR
#7	Request to Send	RTS		#7	Request to Send	RTS
#8	Clear to Send	CTS		#8	Clear to Send	CTS
#9	Ring Indicator	RI		#9	Ring Indicator	RI
Soldered to DB9 Metal - Shield		FGND		Soldered to DB9 Metal - Shield		FGND

Imagen 3.2 conexión entre dispositivos DCE/DCE

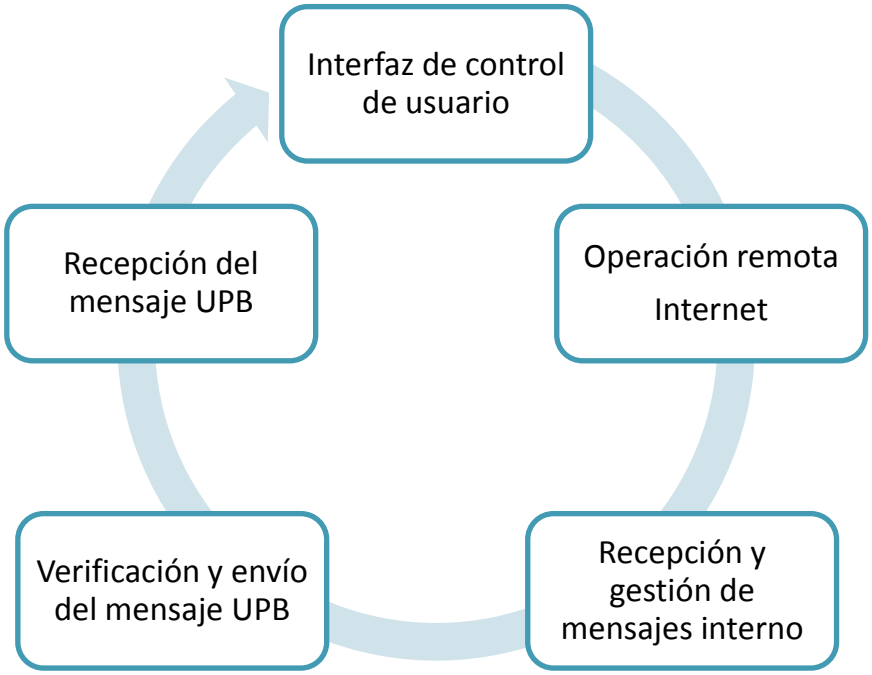
En la siguiente ilustración podemos distinguir como se encuentran las conexiones del modulo RS232 del microprocesador con lo que demostramos que el microprocesador es un dispositivo DCE al igual que el modulo PIM.
Para el armado del cable serial utilizamos el cable naranja como tierra (pin 5) y el naranja con blanco para la corriente en el pin 4 (DTR) al que le suministraremos energía a través de una pila de 9 volts.



[25] Imagen Conector DB9 puesto a disposición bajo la licencia Creative Commons http://es.kioskea.net

Desarrollo

Módulos involucrados en el proyecto



Interfaz de control de usuario
Las órdenes se envían a través de una página web (TCP/IP) remotamente.

Operación remota (Internet)
Los parámetros para la selección de tramas UPB son enviadas por la red hacia el módulo de gestión interno (MCF51CN128) utilizando el método GET.

Módulo de recepción y gestión de mensajes interno (MCF51CN128).
Recibe el flujo de información, la clasifica y solicita el envío del mensaje al módulo PIM (UMCdb9). Este módulo analiza el checksum y los demás campos que conforman la trama UPB para verificar la integridad del comando y determinar si es posible enviar el mensaje o simplemente rechazar la solicitud y enviar un mensaje de error al Host.

Módulo de verificación y envío del mensaje UPB (PIM)
Se realiza a través del puerto de comunicación en serie. El modulo PIM recibe y transmite a nivel de bits evitando la interpretación de los mismos a través de la interfaz RS232. Si el mensaje UPB es integro éste módulo monta el mensaje sobre la red eléctrica y lo envía a lo largo de la misma hasta que se encuentre el dispositivo destino.

Módulo de recepción del mensaje UPB (UMA)
Módulo conectado a la red eléctrica que interpreta el mensaje y ejecuta la función conforme lo establecido en la implementación del protocolo UPB.

Integración del sistema

El sistema integrado permite al usuario:

- Controlar los dispositivos desde cualquier ubicación a través de un navegador en Internet. (FIGURA 1.1)

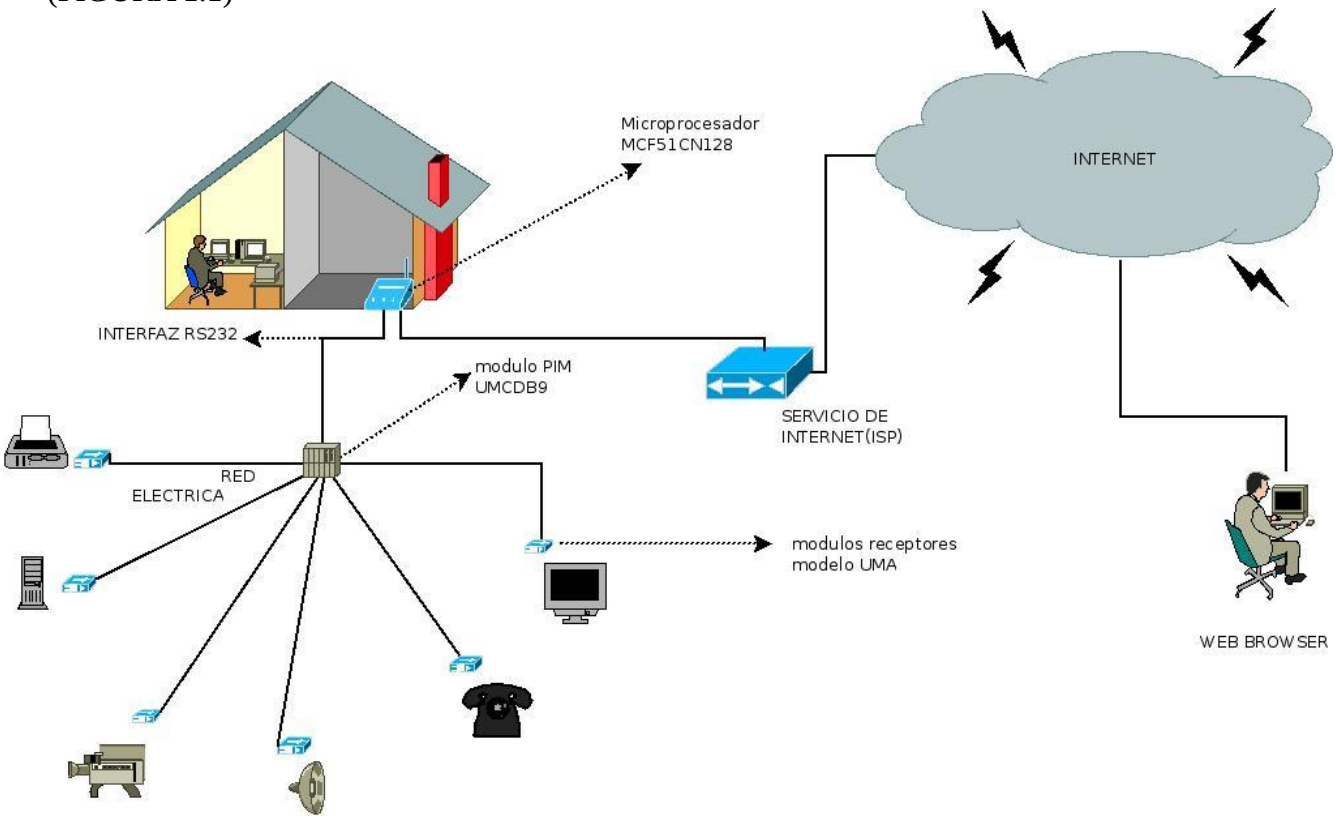


Figura 1.1 Control remoto de dispositivos desde un navegador.

- Microprocesador MCF51CN128 [B14a]
- Protocolo de comunicación UPB.
- Software: CodeWarrior 6.2
- Módulo PIM o UMC[26].
- Módulo receptor UPB modelo UMA por cada dispositivo a controlar.
- Interfaz serie RS232

[26] PIM o UMC (utilizaré cualquier termino indistintamente) módulo para intercambiar comandos digitales.

Especificaciones técnicas

El protocolo UPB fue seleccionado para llevar a cabo la comunicación entre dispositivos.
El método de comunicación del protocolo UPB es por medio de transmisión de información codificada a través de la línea eléctrica de corriente alterna.
Cada módulo receptor tiene su propio detector de cruce por cero para llevar a cabo la sincronización de envíos y recepciones de datos.

Microprocesador

Procesador: MCF51CN128 (64 pines)
Arquitectura: direccionamientos por byte, modo big endian, 32 bits
RTOS: MQX 3.2.1 con stack de USB, stack de TCP/IP, y sistema de archivos.
Máxima capacidad direccionable: 24 bits de memoria (16MB)
Interfaz de Programación: puerto mini-USB
Driver: software libre OSBDM-JM60
IDE: Codewarrior for microcontrollers 6.2
Lenguaje de programación: ANSI C
BAUD rate [3]: 115 200
Frecuencia: 50MHZ

UPB

Modulación: En el cruce por cero de la onda senoidal a velocidades de 50 o 60 Hz .
Ambiente de uso: Aplicaciones de control en sectores residenciales y comerciales.
El método de transmisión UPB: Pulsos sobrepuestos sobre AC.
Distancia promedio entre la central y los dispositivos: 20-25 metros (aproximadamente).
Máxima velocidad (tiempo de reacción al comando): .3 segundos.
Máxima cantidad de comandos enviados por segundo: 10.
Máxima cantidad de dispositivos conectados: 256.
Cantidad mínima de dispositivos a controlar: 2
La plataforma de desarrollo: Windows Xp, Windows Vista.
Plataforma de ejecución: Linux, Windows.
Lenguaje de programación: ANSI C.
Entorno de desarrollo: CodeWarrior6.2.
Control: mediante página de internet.
Navegadores probados: Firefox, IE7
Instrucciones de control: On/Off.
Convivencia entre X10/UPB: Si.
Limitado a sistemas eléctricos de 120 volts.
Módulos receptores: Enchufables con entrada para clavija trifásica.
Interfaz entre modulo PIM[27] y microprocesador: RS232.
Protocolos: ARP, RARP, ICMP, TCP, UDP, SMTP, HTTP, UPB etc.

Módulos

Conexión: plug-in

[27] The Powerline Interface Module (PIM) o módulo de interfaz con la instalación eléctrica.

Interfaz:RS232 *(requiere una constante positiva de 5 a 23 volts suministrada al pin DTR)
Corriente: AC
Velocidad: 4800 baudios
Canal: half duplex

Entregables

El sistema es capaz de controlar al menos dos dispositivos (lámparas) desde una página de Internet.
Es importante indicar que el sistema tiene capacidades superiores y que en este proyecto sólo se limita al control de dos dispositivos, aunque sabemos de antemano que es capaz de controlar más de 250 dispositivos y/o sensores entre otros.

Al finalizar los productos del trabajo que se incluirán en el reporte final serán los siguientes:

- Código fuente comentado.
- Manual de usuario.
- Guía rápida de instalación y uso de todos los dispositivos que intervienen en el sistema.
- Documentación y ficha técnica de cada dispositivo.

Debido a que MQX RTOS es una biblioteca de funciones para implementar aplicaciones multitarea en tiempo real explicare algunas de ellas a lo largo del documento para facilitar la comprensión del mismo.

EXISTEN 2 TAREAS PROGRAMADAS:
MAIN_TASK()
IO_TASK.

Para utilizar la “SUITE DE COMUNICACIONES EN TIEMPO REAL” (RTCS) se requiere la biblioteca <rtcs.h>

Para poder definir las tareas utilizamos la plantilla por default llamada MQX_template_list [ ] que se define de la siguiente manera:

```
const TASK_TEMPLATE_STRUCT  MQX_template_list [ ] =
{
        {Id de la tarea, nombre de la función, tamaño del stack, prioridad, nombre de la tarea,
atributos},
        {0,     0,      0,      0,      0,      0}
};
```

En nuestro caso es necesario declarar más de una tarea, lo cual podemos hacer dentro de la misma plantilla ya que MQX mantiene cada instancia (tarea) salvando su contexto (pc, registros y stack).

Es importante observar que una de las tareas necesita inicializarse automáticamente al conectar el microprocesador por lo tanto llevara la el atributo MQX_AUTO_START_TASK [B15] para que MQX pueda crear automáticamente la instancia y de esta manera correr la tarea principal.

```
const TASK_TEMPLATE_STRUCT  MQX_template_list [ ] =
{
        {1,     Main_Task,    1700,  8 ,     " Main ",      MQX_AUTO_START_TASK},
        {2,     io_task,      500,   7,      " IO poll ",   0},
        {0,     0,      0,    0,     0,      0}
};
```

Para poder iniciar una tarea declarada en la plantilla
const TASK_TEMPLATE_STRUCT MQX_template_list[]
Se utiliza la siguiente función para crear la tarea y dejar su estado = listo [B16].

_task_create("Numero del procesador donde se crea", "Id de la tarea", "algún parámetro de creación");

Si el primer parámetro es 0 se creara en el procesador local, además no incluiremos ningún parámetro para su creación por lo tanto los parámetros quedarían de la siguiente forma:

```
_task_create(0, 2, 0);
```

Ahora necesitamos definir las tareas que estamos creando para ello establecemos el punto de entrada a la tarea (inicio en la tarea dos por ser la más corta) de la siguiente manera:

```
void io_task(uint_32 temp)
{
        //código para tomar información del potenciómetro
}
```

Dentro de nuestra tarea es necesitamos el adc device driver para poder utilizar el potenciómetro (incluir el header bsp.h)
Para poder realizar la apertura del dispositivo ADC la función open requiere un apuntador al record de inicialización [B17].
El siguiente código de ejemplo muestra la implementación de lo descrito anteriormente:

```
/* ADC device init struct */
const ADC_INIT_STRUCT adc_init = {
   ADC_RESOLUTION_DEFAULT,     /* resolution */
};

f = fopen("adc:", (const char*)&adc_init);
```

El siguiente paso después de abrir el driver del dispositivo e inicializarlo es abrir el archivo del driver del canal como lo indica la sección 8.5.2 de la guía de usuario de entrada salida que menciona que la implementación puede ser de la siguiente manera [BD7].

```
const ADC_INIT_CHANNEL_STRUCT adc_channel_param1 = {
      ADC_SOURCE_AN1,        /* physical ADC channel */
      ADC_CHANNEL_MEASURE_ONCE | ADC_CHANNEL_START_NOW,
                        /* one sequence is sampled after fopen */
      10,                 /* number of samples in one run sequence */
      100000,               /* time offset from trigger point in us */
      500000,               /* period in us (=500ms) */
      0x10000,              /* scale range of result (not used now) */
      10,                 /* circular buffer size (sample count) */
      ADC_TRIGGER_2,         /* logical trigger ID that starts this ADC channel */
} ;
```

Adaptar el microprocesador para emplearlo como nodo de internet

El objetivo principal de adaptar el microprocesador es el de controlar los dispositivos periféricos.
Para poder utilizar el MCF51CN128 como servidor web utilizaremos las herramientas disponibles que han sido fuertemente integradas con MQX RTOS y el *stack* de internet embebido RTCS.
Utilizaremos el estándar HTTP que nos provee un generador de código en C a partir de código web. Esta herramienta nos ayudara a simplificar el proceso de conversión de objetos WEB a estructuras ANSI-C. De esta manera el código generado es compilado y posteriormente enlazado con los módulos HTTP de la aplicación [B18].

El cliente http o browser hace una solicitud (método GET) al servidor web, de un documento CGI, que en nuestro caso será el archivo secdata.cgi o archencendido.cgi los cuales están relacionados con las funciones cgi_sec_data () y encendido() respectivamente cuando los cgi's son solicitados automáticamente se ejecutan las funciones con las que se relacionan.
De este modo el servidor web recibe la información copiándola desde el stack TCP/IP a la RAM, esta solicitud al servidor es recibida por referencia y maneja como una lista ligada de segmentos de paquetes TCP/IP. De esta manera podemos observar que el protocolo TCP/IP es el responsable de dividir las respuestas Http (en paquetes tcp/ip) para enviarlas del servidor al cliente.

Cgi_sec_data()
Esta función CGI es la encargada de verificar el estado de la entrada física del potenciómetro, es decir lee el potenciómetro y envía una cadena con la lectura al archivo secdata.cgi para que la pagina web los pueda retener estos valores se pueden ver en el archivo <ip_address_webserver>/secdata.cgi. De este archivo es tomada la información para que el protocolo HTTP a través de la capa de presentación se encargue de plasmar dinámicamente los datos en HTML para el archivo mqx.html y towersystem.html respectivamente (ver imagen 1.2).

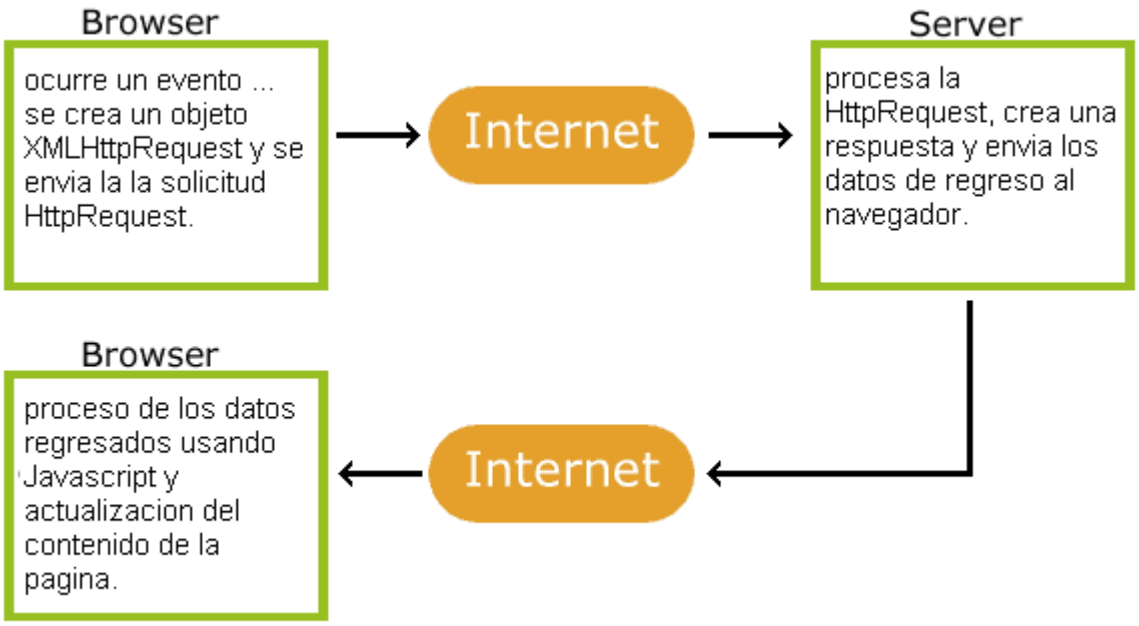


Imagen 1.2 implicaciones del objeto HttpRequest[28]

En resumen la ilustración muestra que una XMLHttpRequest es un objeto Javascript capaz de llamar al servidor y capturar su respuesta

Agregar contenido estático

Uno de los parámetros requeridos para inicializar el servidor http es un arreglo que describe el directorio raíz en nuestro caso lo pondremos en la raíz ("") del sistema de archivos tfs como se muestra a continuación.(el tfs se puede utilizar en aplicaciones tipo http que requieren un sistema de archivos de solo lectura)

```
const HTTPD_ROOT_DIR_STRUCT root_dir[] = {
        { "", "tfs:" },
        { 0, 0 }
};
```

También necesitamos definir una constante de tipo TFS_DIR_ENTRY, la cual es una estructura que almacena nombre, banderas ,datos y el tamaño de cada una de las páginas web asociadas.

[28] http://www.w3schools.com/ajax/ajax_intro.asp

const TFS_DIR_ENTRY tfs_data [ ];

Ahora levantaremos de la siguiente manera el trivial file system para poder dar servicio web y le enviaremos como segundo argumento la estructura que deseamos llenar.

```
io_ tfs_install("tfs:", tfs_data);
```

Después de esto basta con llamar un script de Perl llamado “mktfs” que genera las estructuras ANSI-C a partir de la pagina que implementamos con HTML y lo coloca en un archivo, en nuestro caso tfs_data.c. [B19]

NOTA:
El script se encuentra en la siguiente ruta: FreescaleMQX3.2\tools
y para ejecutarlo únicamente enviamos como argumento la ubicación del directorio donde se encuentran nuestras páginas HTML y como segundo argumento el nombre del archivo.c donde se guardara la conversión del código C.

Ahora debemos inicializar la tarea del servidor http en nuestro caso lo haremos con los parámetros por defecto con la función httpd_server_init:

```
HTTPD_STRUCT* httpd_server_init(
                HTTPD_ROOT_DIR_STRUCT *root_dir,
                const char *index_page);
```

Quedando nuestro código de la siguiente manera:

```
HTTPD_STRUCT* server;
server= httpd_server_init( root_dir, "index_page.html");
```

Programación del microprocesador de manera tal que éste reciba los comandos remotos TCP/IP y los transforme en señales comprensibles para los dispositivos periféricos UMA acoplados a la red eléctrica.

Por otro lado el diagrama de nuestra aplicación muestra que debemos ejecutar algunas funciones que serán llamadas desde el servidor así que es necesario establecer una tabla de funciones para llamadas tipo CGI en el servidor HTTP lo cual se hace con la función:

```
HTTPD_SET_PARAM_CGI_TBL(server, val)
```

Donde val es un apuntador a la tabla de funciones CGI.
De tal manera que podemos definir este bloque de la siguiente manera:

Primero definimos nuestra tabla de funciones o mapeo de asignación CGI que es la parte donde se asocia el nombre del archivo.CGI con la función es decir, todos los cambios en el potenciómetro se escribirán al vuelo (RAM) en un archivo llamado secdata.cgi y archenc.cgi, ambos se encuentran en el background y de estos archivos se tomara la información que se imprimirá en los documentos

web(HTML).

```
HTTPD_CGI_LINK_STRUCT cgi_lnk_tbl [ ] ={
        {“secdata”, cgi_sec_data},
        {“archenc”,encendido},
        {0,0}
};
```

Después registramos las llamadas a función CGI con sus respectivas paginas llamando a la función:
HTTPD_SET_PARAM_CGI_TBL(server,cgi_lnk_tbl);

Aun no hemos definido la función que será llamada cuando el cliente haga la solicitud al servidor web del archivo CGI [B20].
Ambas funciones se construirán de la siguiente manera:

```
static int cgi_sec_data(HTTPD_SESSION_STRUCT *session)
{
        //código comprendido en la función
}

static int encendido(HTTPD_SESSION_STRUCT *session)
{
        //código comprendido en la función
}
```

Por último para que la aplicación maneje el proceso del servidor se utiliza
httpd_server_poll (server, -1) periódicamente [B21].

Parámetros de Red

Para poder inicializar las funciones de red es necesario crear una instancia del RTCS que reserve recursos que RTCS necesita y crea una tarea:

```
int_32          error;
error = RTCS_create();
```

Para obtener la direccion MAC (ethernet) del dispositivo invocamos la siguiente función:

```
ENET_get_mac_address (BSP_DEFAULT_ENET_DEVICE, ENET_IPADDR, enet_address);
```

Ahora inicializamos el dispositivo con:

```
error = ipcfg_init_device (BSP_DEFAULT_ENET_DEVICE, enet_address);
```

Después hacemos una atadura (Bind) del dispositivo Ethernet a la red usando una constante estática (static), la dirección IP :

error = ipcfg_bind_staticip(BSP_DEFAULT_ENET_DEVICE, &ip_data);

Solución a la diferencia de velocidad de envío

Sabemos que el procesador trabaja a 115200 baudios mientras que el módulo PIM (UMCdb9) recibe el envío del procesador a 4800 baudios. Por lo tanto es necesario realizar el siguiente cambio dentro de las bibliotecas MQX

1-. Entramos al archivo user_config.h ubicado en la siguiente ruta:

C:\Program Files\Freescale\Freescale MQX 3.2\config\twrmcf51cn

Para verificar la configuración BSP y para checar que puerto está habilitado

```
/*
** BSP settings
*/
#define BSPCFG_ENABLE_TTYA      0
#define BSPCFG_ENABLE_TTYB      1
#define BSPCFG_ENABLE_ITTYB     0
#define BSPCFG_ENABLE_I2C0      0
#define BSPCFG_ENABLE_I2C1      0   // NOTE: Can't be enabled if ENET is enabled
#define BSPCFG_ENABLE_GPIODEV   1
#define BSPCFG_ENABLE_ADC       1
#define BSPCFG_ENABLE_SPI0      0
#define BSPCFG_ENABLE_SPI1      0   // NOTE: Shares wires with ADC.
#define BSPCFG_ENABLE_FLASHX    0
```

Después de observar que TTYB es el que está habilitado

2-. Vamos al archivo twrmcf51cn.h ubicado en la siguiente ruta:

C:\Program Files\Freescale\Freescale MQX 3.2\mqx\source\bsp\twrmcf51cn\twrmcf51cn.h

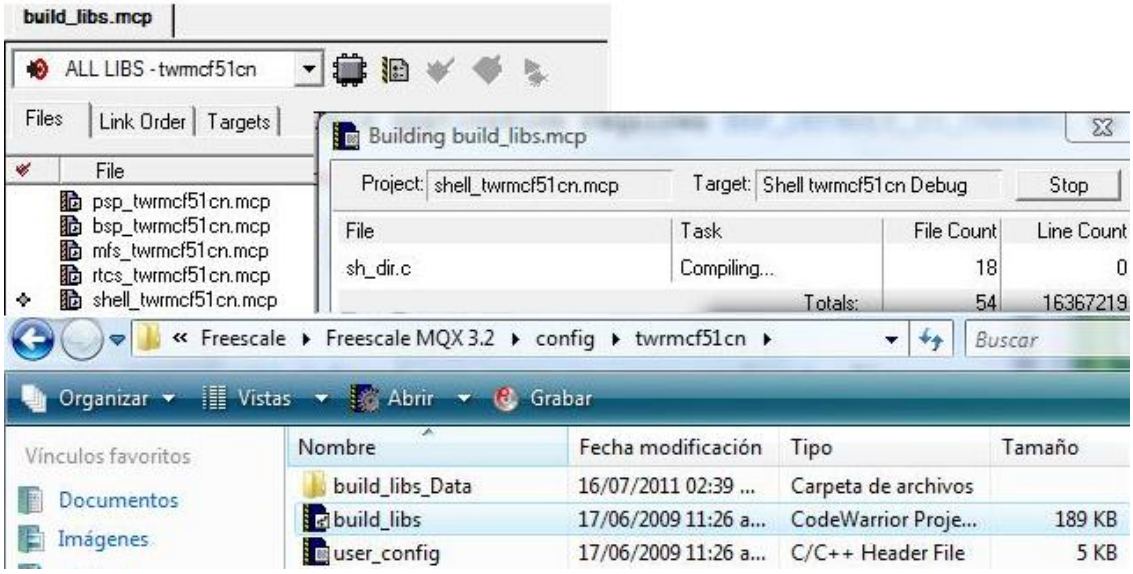
Y modificamos el archivo en la sección de configuración de baud rate como se muestra en la imagen inferior, la configuración por defecto trabaja con 115200 nosotros cambiamos a 4800 para soportar al modulo PIM

```
/* TTYA and ITTYA baud rate */
#ifndef BSPCFG_SCI0_BAUD_RATE
#define BSPCFG_SCI0_BAUD_RATE 115200
#endif

/* TTYB and ITTYB baud rate */
#ifndef BSPCFG_SCI1_BAUD_RATE
#define BSPCFG_SCI1_BAUD_RATE 4800 //#define BSPCFG_SCI1_BAUD_RATE 115200
#endif

/* TTYC and ITTYC baud rate */
#ifndef BSPCFG_SCI2_BAUD_RATE
#define BSPCFG_SCI2_BAUD_RATE 115200
#endif
```

3-. Recompilamos todas las bibliotecas del twr ubicadas en el proyecto build_libs ubicado en la siguiente ruta:

C:\Program Files\Freescale\Freescale MQX 3.2\config\twrmcf51cn\build_libs

Abrimos el proyecto con Codewarrior como se muestra en la imagen inferior



4-. En nuestro proyecto agregamos las líneas preventivas para asignación de la tasa de envio como se muestra en la parte sombreada (baud_rate) de la figura inferior.

```
#if !BSPCFG_ENABLE_IO_SUBSYSTEM
#error This application requires BSPCFG_ENABLE_IO_SUBSYSTEM defined non
#endif

#ifndef BSP_DEFAULT_IO_CHANNEL_DEFINED
#error This application requires BSP_DEFAULT_IO_CHANNEL to be not NULL.
#endif
   //para soportar al modulo PIM

#ifndef BSPCFG_SCI1_BAUD_RATE
#define BSPCFG_SCI1_BAUD_RATE 4800
#endif



#if !BSPCFG_ENABLE_ADC
#error This application requires BSPCFG_ENABLE_ADC defined non-zero in
#endif
```

Integracion del paquete para encender el dispositivo con ID=1

Una vez reconocidos los campos que componen el mensaje UPB es necesario distinguir en cuales nos vamos a enfocar durante el proyecto o cuales son los que necesitamos afectar para llevar a cabo el control requerido.

Nombre del	Lon	Hex	Binario	Descripcion

campo				
Preambulo	1	96	1001 0110	
Control word	2	08 10	0000 1000 0001 0000	LNK: paquete directo= 0 REPRQ:sin uso de repetidor = 00 LEN: tamaño 8 bytes = 01000 RSV:siempre es cero = 0 ACKRQ:con solicitud de confirmacion=001 CNT:solo se envia una vez el mensaje = 00 SEQ: se transmite por primera vez = 00
Network ID **Destination ID** **Source ID**	1 1 1	26 01 FF	0010 0110 0000 0001 1111 1111	NID: para la red numero ‘38’ DID: el destino es el dispositivo numero ‘1’ SID: lo manda el modulo con ID= 128
Message Data ID	1	22	0010 0010	MSID:se usa el conjunto de comandos de control de dispositivos= 001 MID: comando de control de dispositivos = 00010 “GOTO”
Message data argument **Level** **Rate** **Channel**	0-17 1 -- --	 64 -- --	 0110 0100 ---------- ----------	 Nivel: 100% = 64 Velocidad: 0 Canal: al omitirse el argumento se aplica a todos los canales del dispositivo
Checksum	1	3C	0011 1100	Resultado del Verificador de redundancia

Integracion del paquete para apagar un dispositivo

Nombre del campo	Lon	Hex	binario	Descripcion
Preambulo	1	96	1001 0110	
Control word	2	08 10	0000 1000 0001 0000	LNK: paquete directo= 0 REPRQ:sin uso de repetidor = 00 LEN: tamaño 8 bytes = 01000 RSV:siempre es cero = 0 ACKRQ:con solicitud de confirmacion=001 CNT:solo se envia una vez el mensaje = 00 SEQ: se transmite por primera vez = 00
Network ID **Destination ID** **Source ID**	1 1 1	26 01 FF	0010 0110 0000 0001 1111 1111	NID: para la red numero ‘38’ DID: el destino es el dispositivo numero ‘1’ SID: lo manda el modulo con ID= 128
Message Data	1	22	0010 0010	MSID:se usa el conjunto de comandos de

ID				control de dispositivos= 001 MID: comando de control de dispositivos = 00010 “GOTO”
Message data argument **Level** **Rate** **Channel**	0-17 1 1 --	 00 -- --	 0000 0000 ---------- ----------	 Nivel: 0% Velocidad: 0 Canal: al omitirse el argumento se aplica a todos los canales del dispositivo
Checksum	1	A0	1010 0000	Resultado del Verificador de redundancia

Nota: Durante el armado del paquete podemos notar que el byte de preambulo no es contado en el campo LEN por lo tanto este preambulo es unicamente una referencia y no se envia a través del puerto serie. En caso de incluirlo obtendremos una negativa del modulo PIM.

De este segmento podemos deducir que serán 4 tramas diferentes las que enviaremos al modulo PIM

Trama No	Dispositivo	Descripción	Composición
1	1	Encender	08102601FF22643C
2	1	Apagar	08102601FF2200A0
3	2	Encender	08102601FF22643B
4	2	Apagar	08102601FF22009F

Es importante recalcar que los identificadores de dispositivo y de red son parámetros que podemos definir enviando tramas de configuración al módulo PIM. Sin embargo en este proyecto se realizó la configuración con un software como se muestra a continuación y por lo tanto los identificadores nos fueron asignados arbitrariamente.

Configuración de la red y los módulos UPB usando UPSTART

Ya que la programación de la interfaz para configuración de la red y los módulos UMCDB9 está fuera del alcance de este proyecto utilizaremos una herramienta gratuita (UPSTART) para poder realizar dicha configuración.
Nota: Durante la instalación y configuración de la nueva red existe la posibilidad de usar la interfaz serial o un adaptador USB a serial en cualquier caso se debe seleccionar la opción “Powerline interface module”.
Para tener una referencia completa adjunto “UPStart_ Quick_Start_Guide_070813.pdf” donde encontraremos la configuración de los dispositivos y de la red con mayor detalle.

Comunicación Serial PIM/Host

El modulo PIM o UMC puede ser usado tanto para transmitir información a la corriente eléctrica como para recibir información de la misma. El PIM utiliza interfaz serial RS-232 para la comunicación entre

él y el dispositivo host tanto para recibir comandos del host como para enviar respuestas y reportes al host[29] (ver figura 20).

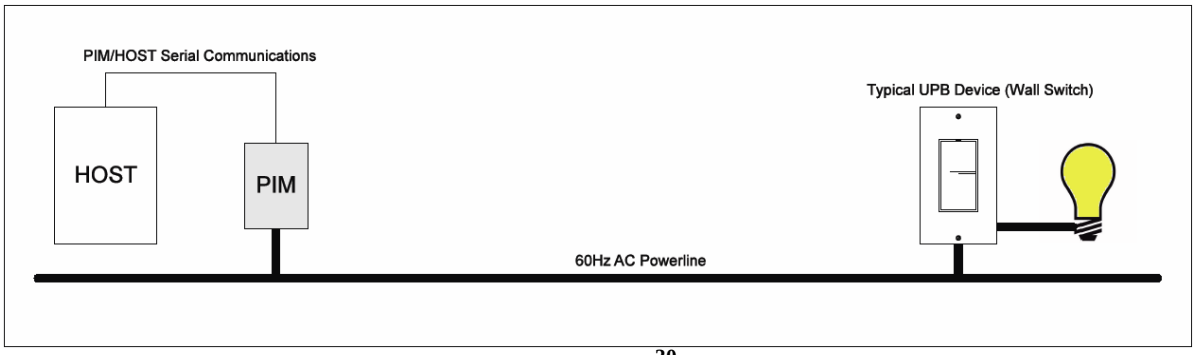


Figura 20 Diagrama de comunicación PIM/Host[30]

Todas las comunicaciones PIM/Host (ver imagen 20.1)se hacen en serie a 4800 baudios en el formato "N-8-1"[31] y todas las comunicaciones consisten en caracteres ASCII, además es necesario terminar con un carácter de retorno de carro (<cr>).

Imagen 20.1 comunicación PIM/host mediante RS232.

NOTA:
Dentro del formato de comunicación del puerto serial 8N1 no se agrego control de flujo por hardware ni software y se mantuvo la señal DTR con una constante positiva de voltaje entre 5 y 23 volts (ver imagen 20.2) ya que el fabricante dio estos requisitos para lograr una comunicación exitosa entre los

[29] En nuestro caso será el microprocesador MCF51CN128
[30] Dibujo tomado de http://www.simply-automated.com/tech_specs/PimComm1.5a.pdf que muestra la comunicación de los dispositivos sobre la corriente eléctrica
[31] Parámetros de configuración del puerto serie (sin bit de paridad, 8 bits de datos, 1 bit de paro)

módulos. Cualquiera de estos puntos que no se cubran hacen imposible la comunicación entre el host y el modulo PIM. (Esta información no se encuentra en los manuales ni en la descripción técnica y fue necesario solicitarla al fabricante para poder concluir con el proyecto).

Para proveer el voltaje al pin DTR se realizaron algunas pruebas para poder hacer algún puente en el microprocesador ya que las funciones relacionadas con los manuales y bibliotecas de mqx no están implementadas para nuestro modulo en particular y por lo tanto fue necesario buscar una alternativa externa. Debido a que encontramos algunas discrepancias en el diagrama del MCF51CN 128 decidimos no realizar ningún puente para proveer los 5 volts desde el mismo y evitar así cualquier problema con la circuitería.

Figura 20.2 suministro de constante positiva de voltaje al pin DTR (pin 4) mediante pila de 9 volts.

Conclusión:

Podemos destacar que este proyecto es una alternativa de control de dispositivos remotos con costo moderado. Esta solución es aplicable a diversos campos como es el caso de la industria donde es posible utilizarlo para la automatización de tareas, programación de eventos, etc.; seguridad pública para la activación de alarmas, bloqueo de puertas, monitoreo del estado de alguna unidad delicada, monitoreo de presión temperatura, humedad, movimiento, etc.; el entretenimiento donde es posible cubrir aspectos como la ambientalización del hogar y confort durante el control de casas o edificios inteligentes entre otros. Además ha tomado mucha fuerza en los últimos años debido, no solo a la cantidad de vertientes

en las que puede ser aplicada, sino a la confiabilidad del protocolo manejado durante el envío de comandos.
Es importante mencionar que la tecnología antes mencionada no requiere infraestructura adicional ya que se utiliza la instalación eléctrica existente como bus para el control de los dispositivos sin requerir cambio alguno en la instalación.
Todo esto aunado a la simplificación y programación de tareas en intervalos preestablecidos así como la disminución en costos energéticos, al limitar el flujo de la corriente eléctrica únicamente en instantes o intervalos de tiempo específicos.

Bibliografía

[B0] Domingo Solans Campo, Las Nuevas Tecnologías Al Servicio De Los Mayores. Domótica., Castellón de la Plana, Mayo 2005

[B1] Huidobro Moya J.M., Millán Tejedor R.J., *Domótica Edificios Inteligentes*, 1ª ed. México, Limusa, 2007, pp. 5-22.

[B2] HAI http://www.sesdi.com/ci/index.html consultada el 8 de junio de 2010

[B3] Serrano V., Villena M, Hoyos D., "Monitorización a distancia de una vivienda utilizando la red electrica", *Avances en Energías Renovables y Medio Ambiente*, Vol. 11, 2007.

[B4] Cuevas J. C.,Martínez J. , Merino P.,"El Protocolo x10: Una solución Antigua a Problemas actuales", Dpto. Lenguajes y Ciencias de la Computación Universidad de Málaga, Copyright (C) 2002 http://www.lcc.uma.es/~pedro/publications/566_art.pdf consultada el 3 de junio de 2010

[B5] http://www.casainteligente.com/ consultada el 3 de junio de 2010

[B6] Bingham Jared, "A study of home builder advertising
for smart home technologies", A thesis submitted to the faculty of
Brigham Young University for the degree of Master of Science, Agosto, 2006
http://contentdm.lib.byu.edu/ETD/image/etd1421.pdf consultada el 5 de junio de 2010

[B7] Interfacing Stargate-IP with UPB networks
http://www.jdstechnologies.com/download/appnotes/upb.pdf consultada el 8 de junio de 2010

[B8] TCP/IP en la Domótica
http://www.casadomo.com/noticiasDetalle.aspx?c=27&idm=34 consultada el 19 de mayo de 2010

[B9] Costo “Omnipro II”
http://www.sirkom.com/es/familias.php?id=3&Id_categoria=1&Id_subcategoria=1
Consultada el 20 de mayo de 2010

[B10a]Versión 1 ColdFire® White paper. Rev 0. 07/2006. Freescale Semiconductor.

[B11a] Microcontroladores De 32 Bits Coldfire V1/ Familia JM Diego Alejandro Múnera Hoyos Revisión 0.1

[B12a] https://www.freescale.com/lms/auth/repo/go?rid=2228228&flag=Launch

[B13a]http://www.simply-automated.com/tech_specs/PimComm1.5a.pdf

[B14a] MCF51CN128

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MCF51CN128
Consultada el 20 de mayo de 2010

[B15]**Freescale MQXTM Real-Time Operating System User Guide, Rev. 0 , pg 24**

[B16]**Freescale MQXTM Reference Manual, Rev. 0 , pg 318**

[B17]**Freescale MQX (TM) I/O Drivers User's Guide, Rev. 1**

[B18] **http://www.embedded-access.com/products/web_server.html**

[B19]**Freescale MQXTM RTCSTM User's Guide, Rev. 1 , pg 72**

[B20]**Freescale MQXTM RTCSTM User's Guide, Rev. 6 , pg 76**

[B21]**Freescale MQXTM RTCSTM User's Guide, Rev. 1 , pg 154**