TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: Diseño de nodos inteligentes para instalaciones domóticas basadas en bus

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad en Sistemas de Telecomunicación

AUTOR: Pablo Fernández Molina

DIRECTOR: Marcos Quílez Figuerola

FECHA: 21 de enero de 2008

Título: Diseño de nodos inteligentes para instalaciones domóticas basadas en bus

Autor: Pablo Fernández Molina

Director: Marcos Quílez Figuerola

Fecha: 21 de enero de 2008

Resumen

Los dos objetivos planteados en este trabajo consisten en desarrollar nodos domóticos capaces de funcionar de forma independiente o en red y estudiar las posibilidades que ofrece el bus CAN en el ámbito de la domótica.

Para su realización se ha estudiado el bus CAN, se ha diseñado y construido el hardware de los nodos basados en un microcontrolador, se han programado los microcontroladores en lenguaje C, se ha programado una aplicación para PC mediante LabVIEW de forma que el PC se integre en la red como un nodo más y se ha definido un protocolo para la comunicación de los nodos mediante bus CAN. Finalmente, se han realizado pruebas para verificar el correcto funcionamiento de todos y cada uno de los elementos del sistema.

Como resultado se ha obtenido el prototipo de una red domótica con tres nodos, ampliable hasta 112. De los nodos construidos, dos son estándar y el tercero es una interfaz que permite acciones de monitorización y control. Los nodos se comunican mediante un bus CAN, utilizando el protocolo definido en este trabajo.

Title: Design of intelligent nodes for bus-based home automation installations

Author: Pablo Fernández Molina

Director: Marcos Quílez Figuerola

Date: January, 21th 2008

Overview

The aim of this work consist of developing domotic nodes to work in a stand-alone way or as a part of a network, as well as to study the possible use of the CAN bus in home automation applications.

For its accomplishment we studied the CAN bus, we designed and build the hardware of the nodes based on a microcontroller, we programmed the microcontrollers using C language, we developed a PC application using LabVIEW so that the PC is integrated in the network as a node and we defined a protocol for the communication of the nodes. Finally, tests have been run to verify the correct operation of each element in the system.

As a result we obtained the prototype of a domotic network with three nodes, expandable up to 112. Two of the constructed nodes are standard and the third one is an interface that allows monitoring and control actions. The nodes communicate by means of a CAN bus, using the protocol defined in this work.

Agradecimientos

Con las siguientes palabras me gustaría agradecer a todas aquellas personas que han hecho posible, en mayor o menor medida, que este trabajo se lleve a cabo; ya que como se suele decir esto no es solo cosa de uno.

"En el plano personal me gustaría agradecer a mis padres el apoyo, la paciencia, la compresión y en especial, las facilidades que me han dado. Sin ellos no le podría haber dado la gran dedicación que le he proporcionado a este proyecto.

En el ámbito técnico agradecer en primer lugar al departamento de electrónica de la EPSC por permitirme utilizar el laboratorio, así como también sus recursos y materiales. Y en este apartado hacer una mención especial por la colaboración del técnico del laboratorio, Francis López, sobre todo por la ayuda de última hora.

Agradecerles también a todos los compañeros de la universidad, principalmente a aquellos con los que he compartido el laboratorio durante el transcurso del proyecto, desde Meri, Laura o Lara hasta Marcos, Fran y muchos otros. Ellos también han aportado su granito de arena.

Y por último, y no por ello en menor medida, a mi director del proyecto, Marcos Quílez, no tan solo ya por su ayuda, sino por confiar en mí a la hora de ofrecerme la posibilidad de realizar este proyecto.

Gracias a todos."

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN Y OBJETIVOS

La evolución de la automatización y control de edificios, la domótica, ofrece numerosas posibilidades tanto al usuario como al diseñador de estos sistemas. Hace años, lo habitual era encontrar una única unidad central “inteligente” capaz de activar y desactivar cargas eléctricas. Esta central podía, además, ofrecer la posibilidad de ser accionada a distancia mediante un módem telefónico. Con el tiempo, la tendencia ha sido distribuir la inteligencia del sistema domótico entre los distintos elementos que lo conforman. Esta circunstancia ha dado lugar a distintos tipos de arquitecturas o topologías.

En este contexto, los **objetivos** de este trabajo final de carrera son dos. Por una parte implementar unos nodos domóticos que puedan funcionar de forma independiente, pero que a la vez puedan conectarse entre ellos formando una red. Es decir, cada nodo ha de ser capaz gestionar un conjunto de sensores y actuadores conectados a él, pero también ha de poder integrarse en una red para enviar y recibir información a través de ella. El segundo objetivo consiste en estudiar las posibilidades que ofrece el bus CAN (Controller Area Network) para ser aplicado en redes domóticas.

La memoria se ha organizado en 7 capítulos, seguidos de un apartado de conclusiones y dos anexos. En el capítulo 1 se presenta el estado del arte en instalaciones domóticas y del uso de CAN en este ámbito, y se plantean los requisitos exigidos a los nodos que se querían desarrollar. El capítulo 2 ofrece una visión general de los sistemas domóticos y de las distintas arquitecturas que se pueden encontrar. El tercer capítulo describe las características principales del bus CAN. Los dos siguientes capítulos explican el diseño de la red y de sus nodos. A continuación, el capítulo 6 se define el protocolo desarrollado a medida para la aplicación. El último capítulo describe el sistema completo y las pruebas que se han realizado para verificar su correcto funcionamiento.

En los anexos se ha incluido aquella información que, siendo necesaria para una documentación completa del proyecto, podía entorpecer la lectura de este documento.

CAPÍTULO 1. PLANTEAMIENTO INICIAL

1.1 Estado del arte y antecedentes

Actualmente, los productos más novedosos en el sector de la domótica u hogar digital (como también le denominan muchos fabricantes), son dispositivos orientados a ofrecer mayor seguridad y confort, reducir el consumo de energía, gestionar eficientemente los recursos, y plantear nuevas actividades de ocio en el hogar.

Indomo, por ejemplo, una empresa puntera dedicada a la integración tecnológica, dispone de aparatos muy avanzados que incluyen simuladores de presencia o modos predeterminados de funcionamiento. El simulador de presencia memoriza la actividad habitual en la vivienda y la reproduce de forma automática cuando la vivienda está vacía. El creador de modos de funcionamiento reproduce funciones especiales para activar en momentos determinados. El modo noche, por ejemplo, permite activar la detección de intrusión, ajustar la climatización y limitar la iluminación al 20% con sólo pulsar un botón cercano a la cabecera de la cama.

Simon, otra empresa dedicada al mundo de la domótica, proporciona como novedad del 2007 una unidad llamada SimonVOX. Se trata de un sistema sencillo de telecontrol, seguridad y gestión de la vivienda a través de teléfono o pantalla táctil. SimonVOX está destinado para todo tipo de viviendas y pequeños negocios.

Por otra parte, en el mercado existe un número considerable de productos innovadores relacionados con el bus CAN. Por ejemplo, *National Instruments* suministra un simulador de dispositivos CAN capaz de transmitir y recibir mensajes. Asimismo incluye una interfaz de alta velocidad, una entrada adicional para un monitor y una interfaz integrada a hardware de adquisición de datos o control programable de botones para parámetros de entrada.

Hasta el momento, en la *Escola Politècnica Superior de Castelldefels* (EPSC) se han realizado numerosos proyectos relacionados con la domótica. Entre otros, "*Casas domóticas*" (1996), "*Aplicaciones domóticas vía WAP, HTML y SMS*" (2004), o "*Instalación domótica e ICT para un edificio de viviendas*" (2007).

En cambio, los trabajos que tratan el bus CAN son más escasos. En general estos trabajos no se centran en el estudio de este bus y menos todavía en su aplicación a la domótica. Uno de los proyectos más directamente relacionados con el CAN fue el de "*Interconnexió del bus CAN via ràdio per a automoció*" (2003).

La principal innovación que aporta este trabajo de final de carrera consiste en la combinación entre estas dos temáticas: la utilización del bus CAN en la domótica aplicada a la vivienda. Hasta la fecha, el bus de comunicaciones CAN

es un protocolo que aún no se ha hecho un hueco en dicho sector. Esta situación justifica la realización de este proyecto, ya que se pretende explorar las posibilidades que ofrece el bus CAN para ser implementado en una instalación domótica con nodos independientes e inteligentes.

1.2 Requisitos

A la hora de elegir qué servicios iba ofrecer la instalación domótica, se ha tenido en cuenta aquellas aplicaciones que actualmente predominan en la mayoría de viviendas domóticas y que son fundamentales en los ámbitos que se engloban:

Tabla 1.1 Servicios que ofrecerá la instalación domótica propuesta.

Ámbito	Aplicación más común	Servicio
Ahorro Energético	Climatización	Control, automatización y zonificación tanto de la climatización como de la refrigeración
Confort	Iluminación	Apagado general de todas las luces de la vivienda. Automatización del apagado y encendido en cada punto de luz. Regulación de la luz según el nivel de luminosidad ambiente.
Protección	Presencia	Simulación de la presencia por zonas.
Comunicación	Control Remoto	Presencia en el control tanto interno como externo a través de una interfaz gráfica (PC, PDA, etc.). Transmisión de alarmas.

A modo de muestra, se puede ver un vídeo demo [1] creado por la empresa Domótica Viva para promocionar sus productos, en el cual se puede observar muy claramente aquellos servicios que ofrece actualmente cualquier hogar domótico avanzado. Es un pequeño botón que sirve para hacerse una idea del estilo de servicios que busca implementar la instalación domótica de este proyecto, sin llegar a ser tan ambicioso, sino más bien con las funcionalidades básicas indicadas en la Tabla 1.1.

Los nodos que incorporará esta red también habrán de cumplir con unas especificaciones y prestaciones las cuales se describen a continuación:

- **Entradas analógicas:** dos entradas para conectar sensores analógicos. Se prevé medir nivel de iluminación y temperatura.
- **Entradas digitales**: conexiones para señales de entrada con dos estados, como por ejemplo pulsadores o sensores con salida del tipo todo/nada. Se contempla un mínimo de 3 entradas de este tipo.
- **Salidas digitales**: conexiones para activar y desactivar actuadores, como por ejemplo relés, triacs o dimmers. Se prevé un mínimo de 3.
- **Puerto para bus CAN:** comunicación entre nodos.
- **Puerto serie:** comunicación opcional a través del puerto RS232.

CAPÍTULO 2. LA DOMÓTICA

Según el diccionario de la Real Academia Española (RAE) el término domótica proviene de la unión de la palabra "*domus"* (que significa "casa" en latín) y el sufijo *"-tica"* (esta terminación remite a "automática", y hoy en general induce al significado de "gestión por medios informáticos"). Etimológicamente lo que se entiende por domótica es el conjunto de sistemas capaces de automatizar y controlar las diferentes instalaciones de una vivienda. Los conceptos de automatización y control se asocian al encendido, apagado, apertura, cierre y regulación de aparatos y sistemas de instalaciones eléctricas. En resumen se podría definir como la integración de la tecnología en el diseño inteligente de un recinto, ya sea una vivienda o no.

2.1 La domótica aplicada a la vivienda

Normalmente, el objetivo principal de implantar una red domótica en una vivienda es el de aportar servicios de gestión energética y de comunicación, mejorar la seguridad tanto personal como patrimonial y aumentar el bienestar. Por tanto, los servicios que ofrece la domótica se pueden agrupar según cuatro aspectos principales: ahorro energético, nivel de confort, protección patrimonial y comunicaciones.

2.1.1 Dispositivos

La extensión de una solución domótica puede variar desde un único dispositivo que realiza una sola acción, hasta amplios sistemas que controlan prácticamente todas las instalaciones dentro de la vivienda. Los distintos dispositivos de un sistema domótico se clasifican en los siguientes grupos:

1. **Controlador:** gestiona el sistema según la programación y la información que recibe, como si de un microprocesador se tratara.
2. **Actuador:** es un dispositivo capaz de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema concreto (encendido/apagado, subida/bajada, apertura/cierre, etc.).
3. **Sensor:** es el dispositivo que monitoriza el entorno captando información que transmite al sistema (sensores de agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc.).
4. **Bus:** es el medio de transmisión que transporta la información entre los distintos dispositivos ya sea por un cableado propio, por la redes de otros sistemas (red eléctrica, red telefónica, etc.) o de forma inalámbrica.
5. **Interfaz:** se refiere a los dispositivos (interruptores, conectores, pantallas, etc.) en los que se muestra la información del sistema para los usuarios y/o donde los mismos pueden interactuar con el sistema.

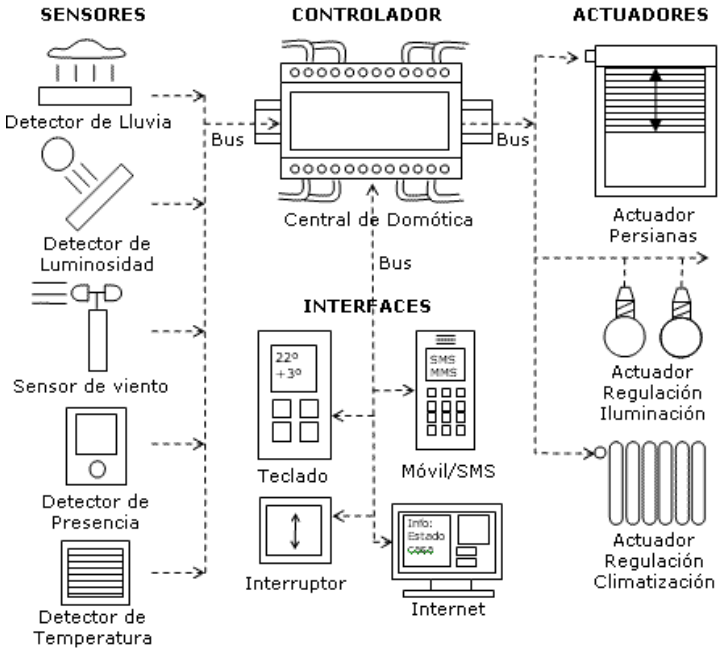

Fig. 2.1 Ejemplos de dispositivos de sistemas de domótica

Hay que señalar que todos los dispositivos de un sistema domótico no tienen porqué estar físicamente separados, sino que varios de ellos pueden estar combinados en un mismo nodo o controlador. Por ejemplo, un nodo puede estar compuesto por un controlador, varios actuadores y sensores, y diferentes interfaces (ver Fig. 2.1). Tampoco es estrictamente necesario que los cinco tipos de dispositivos se incluyan en una misma instalación domótica. Hay algunas arquitecturas, como se verá en el siguiente apartado, que no requieren de un bus. Lo mismo se puede decir por el resto de tipos de dispositivos.

2.1.2 Modelos de arquitecturas

La arquitectura de los sistemas de domótica hace referencia a la estructura de su red. Dependiendo de donde reside la inteligencia del sistema domótico, hay cuatro arquitecturas diferentes:

Arquitectura Centralizada: un controlador centralizado, envía la información a los actuadores e interfaces según el programa, la configuración y la información que recibe de los sensores, sistemas interconectados y usuarios.

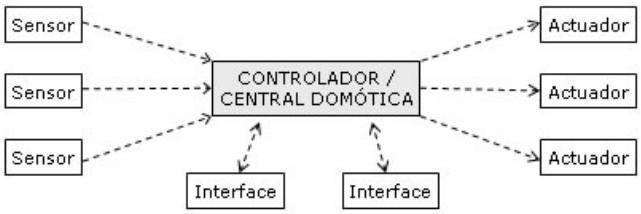

Fig. 2.2 Arquitectura domótica centralizada

Arquitectura Descentralizada: existen varios controladores, interconectados por un bus, que envían información entre ellos y a los actuadores e interfaces conectados a los controladores. Todo ello también según el programa, la configuración y la información que reciben de los sensores, sistemas interconectados y usuarios.

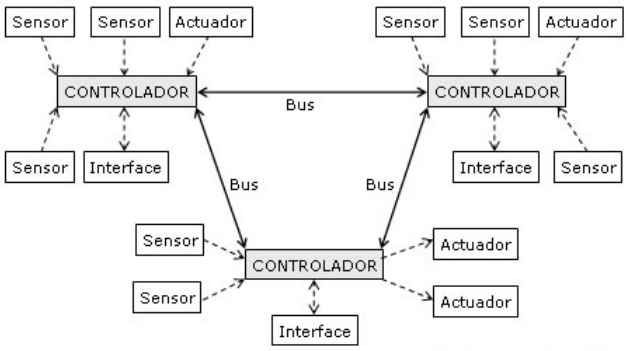

Fig. 2.3 Arquitectura domótica descentralizada

Arquitectura Distribuida: cada sensor y actuador es también un controlador capaz de actuar y enviar información al sistema según el programa, la configuración, la información que capta por sí mismo y la que recibe de los otros dispositivos del sistema.

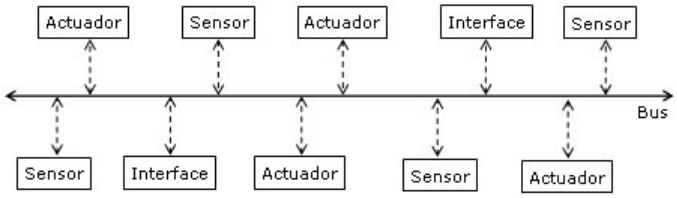

Fig. 2.4 Arquitectura domótica distribuida

Arquitectura Mixta (Híbrida): se combinan las arquitecturas de los sistemas centralizadas, descentralizadas y distribuidas. A la vez que puede disponer de un controlador central y varios controladores descentralizados, los dispositivos de interfaces, sensores y actuadores pueden también ser controladores y procesar la información según el programa, la configuración, la información que capta por sí mismo, y tanto actuar como enviarla a otros dispositivos de la red, sin que necesariamente pase por otro controlador.

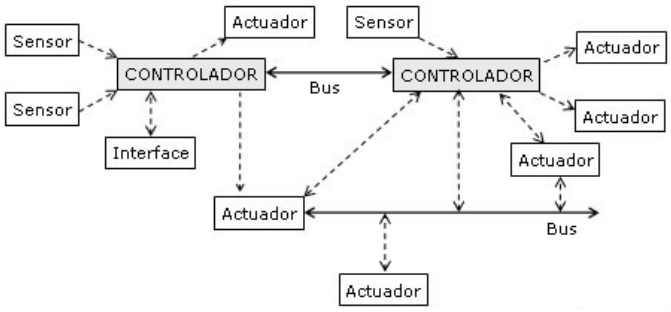

Fig. 2.5 Arquitectura domótica mixta o híbrida

2.1.3 Buses de transmisión

El medio de transmisión de la información entre los distintos dispositivos de un sistema domótico puede ser de varios tipos. Cuando el medio de transmisión se utiliza para transmitir información entre los dispositivos denominados controladores, también se le nombra Bus. Como se ha visto, algunas arquitecturas requieren de un bus de transmisión. Por ello, se va a detallar a continuación cuales son los principales buses domóticos que se pueden encontrar en el mercado:

EIB (European Installation Bus): Bus pensado para ser utilizado como un sistema de gestión de la instalación eléctrica de un edificio. Su propósito comprende la monitorización y control de sistemas tales como el alumbrado, la calefacción, el aire-acondicionado, ventilación, persianas y alarmas de un edificio. Se puede definir como un sistema descentralizado en el que cada uno de los dispositivos conectados tiene control propio. Utiliza su propio cableado, con lo cual se ha de proceder a instalar las conducciones adecuadas en el hogar para poder montar el sistema. Lleva más de 20 años en el mercado de la automatización penetrando lentamente en el campo de la construcción, a pesar de que, es un sistema muy robusto y fiable.

X10: Este bus es uno de los más utilizados en el ámbito de la domótica. Es un protocolo de comunicación que permite controlar aparatos eléctricos a través de la instalación de la red eléctrica. Utiliza la línea eléctrica (220 V o 110 V) para transmitir señales de control entre equipos de automatización del hogar en formato digital. El estándar surgió hace 20 años y lleva más de 15 funcionando a nivel comercial. Actualmente es un protocolo que está presente en el mercado mundial, sobre todo en Norteamérica y Europa (España y Gran Bretaña fundamentalmente).

CEBus (Consumer Electronic Bus): En 1992 fue presentada la primera especificación. Se trata de un protocolo, para entornos distribuidos de control, que está definido en un conjunto de documentos (en total unas 1000 páginas). Como es una especificación abierta cualquier empresa puede conseguir estos documentos y fabricar productos que implementen este estándar. En estos documentos se recogen detalles tanto del nivel físico como de protocolo. En lo que respecta a las características técnicas es interesante resaltar su gran versatilidad, admitiendo múltiples medios físicos, lo que le habilita para ser instalado incluso en edificaciones no equipadas con un cableado de datos por separado.

LonWorks/LonTalk: Es un protocolo diseñado para cubrir los requisitos de la mayoría de las aplicaciones de control: edificios de oficinas, hoteles, transporte, industrias, monitorización de contadores de energía, vivienda, etc. La utilización de LonWorks por más de 1000 fabricantes y el éxito que ha tenido en instalaciones en las que impera la fiabilidad y robustez, se debe a que desde su origen ofrece una solución con arquitectura descentralizada, extremo a extremo, que permite distribuir la inteligencia entre los sensores y los actuadores instalados en la vivienda. Cubre desde el nivel físico hasta el nivel de aplicación de la mayoría de los proyectos de redes de control.

ZigBee: En este caso ya se trata de un protocolo inalámbrico, comparado con los anteriores que eran físicamente cableados. Es una especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal. Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías. En principio, el ámbito donde se preveía que esta tecnología cobrara más fuerza es en domótica debido a diversas características que lo diferencian de otras tecnologías: su bajo consumo, su topología de red en malla y su fácil integración (se pueden fabricar nodos con muy poca electrónica).

A continuación se comparan algunas de las prestaciones básicas de estos buses:

Tabla 2.1 Comparativa de los diferentes buses citados

Tecnología	Medio de transmisión	Velocidad de transmisión	Distancia máxima al dispositivo
EIB	TP, Cable eléctrico, RF o Infrarrojos.	9.600 bps 1.200/2.400 bps	1.000 m. 600 m. 300 m.
X10	Cable eléctrico.	60 bps en EEUU 50 bps en Europa	185 m^2
CEBus	TP, Cable eléctrico, Radio, Coaxial o Infrarrojos.	10.000 bit/s	En función de las características del medio
LonWorks	TP, Cable eléctrico, Radio o Coaxial	78 Kbps - 1,28 Mbps 54 Kbps	1.500 m. - 2.700 m.
ZigBee	Inalámbrico	20 Kbps - 250 Kbps	10 m. - 75 m.

CAPÍTULO 3. BUS DE COMUNICACIONES CAN

En el apartado 2.1.3 no se menciona el bus CAN entre los más utilizados en redes domóticas. Sin embargo las características de este bus podrían justificar su uso en este ámbito, tal y como se verá en el capítulo 4. En este capítulo se presenta una breve descripción sobre el protocolo de CAN.

Para averiguar si el CAN era un bus aplicable a una red domótica, ha sido necesario llevar a cabo una investigación profunda sobre este protocolo. Para agilizar la lectura, en este capítulo sólo se presentan los conceptos básicos necesarios para poder abordar el resto de secciones. El resto de información recopilada se presenta en el anexo A para ofrecer una documentación más completa.

3.1 Presentación y especificaciones

El protocolo de comunicación CAN fue originalmente desarrollado y especificado en los años 80 por la compañía alemana Robert Bosch en un intento por resolver los problemas de comunicación entre los diferentes sistemas de control de los coches. No es necesario decir que la idea no tardó en pasar de los vehículos al sector de la automatización y control. Si se desea conocer más sobre los orígenes del CAN se invita a leer el apartado A.1 del anexo.

CAN está estructurado de acuerdo con el modelo OSI en una arquitectura colapsada de dos capas, esto es, la capa física y la capa de enlace de datos. Como el resto de capas del modelo OSI (red, transporte, sesión, presentación y aplicación) no están especificadas por el protocolo, es necesario diseñarlas. Sobre todo es de vital importancia especificar un protocolo de alto nivel (HLP). En el apartado A.1 del anexo se puede consultar más información sobre la arquitectura y capas de este protocolo.

La especificación original de CAN, publicada por Robert BSCH GMBH, se llama Especificación de CAN 2.0-A. La razón por la cual existe una parte A es porque una especificación actualizada de CAN se publicó después por la misma compañía, y contenía tanto la Parte A, como una nueva versión de CAN llamada Parte B. La Especificación CAN 2.0-B es compatible con la Parte A. La diferencia se localiza básicamente en el formato del encabezado del mensaje del identificador. La especificación CAN 2.0-A define sistemas CAN con un estándar de 11 bits del identificador. En cambio, CAN 2.0-B especifica la trama extendida con 29 bits en el identificador. La Especificación de CAN 2.0 es de dominio público. La mayoría de las implementaciones actuales de hardware CAN usan la nueva versión de la especificación, incluso aunque sólo sea de forma pasiva (solo transmite mensajes estándar, pero comprueba si recibe mensajes estándar y mensajes extendidos). Para entender mejor la composición de las tramas (lo que incluye el campo del identificador) se pude examinar el apartado A.2.

3.2 Características principales

A continuación se mostraran algunas de las propiedades fundamentales del CAN. Prácticamente todas ellas suponen una ventaja respecto otros tipos de buses convencionales:

- ✓ La comunicación está basada en mensajes y no en direcciones.
- ✓ Un mensaje es diferenciado por el campo llamado identificador, que no indica el destino del mensaje, pero sí describe el contenido del mismo.
- ✓ No hay un sistema de direccionamiento de los nodos en el sentido convencional. Los mensajes se envían según su prioridad.
- ✓ La prioridad entre los mensajes la define el identificador. Se trata de una prioridad para el acceso al bus.
- ✓ Es un sistema multimaestro. Cuando el bus está libre, cualquier nodo puede empezar la transmisión de un mensaje, y el mensaje con mayor prioridad gana la arbitración del bus.
- ✓ Todos los nodos CAN son capaces de transmitir y recibir datos y varios pueden acceder al bus de datos simultáneamente.
- ✓ Un nodo emisor envía el mensaje a todos los nodos de la red, cada nodo, según el identificador del mensaje, lo filtra y decide si debe procesarlo inmediatamente o descartarlo. Como consecuencia el sistema se convierte en multicast en el cual un mensaje puede estar dirigido a varios nodos al mismo tiempo (Ver Fig. 3.1).
- ✓ Gran fiabilidad y robustez en la transmisión. Detecta errores, los señaliza, envía mensaje de error y reenviará el mensaje corrupto una vez que el bus vuelva a estar activo. Además puede operar en ambientes con condiciones extremas de ruido e interferencias gracias a que es un bus diferencial. Para más información al respecto consultad el apartado A.4 del anexo.
- ✓ Es un protocolo de comunicaciones normalizado, con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común o bus.
- ✓ Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto.
- ✓ El procesador principal delega la carga de comunicaciones a un periférico inteligente (controlador), por lo tanto el procesador principal dispone de mayor tiempo para ejecutar sus propias tareas.

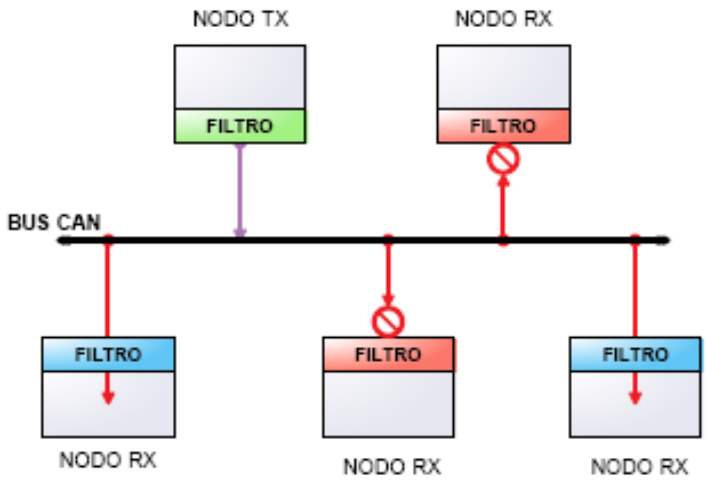

Fig. 3.1 Ejemplo de sistema multicast

También tiene algún que otro inconveniente, como son los siguientes:

- ✓ Velocidad limitada por la longitud de la red. Cada red puede alcanzar como máximo los 1000 m. de longitud (50 kbps) o una velocidad máxima de 1 Mbps (25 m.). Para más información consultad el apartado 6.3.

- ✓ La cantidad de nodos abonados a la red es variable pero limitada. Esta limitación viene marcada según el hardware utilizado. Para más información al respecto consultad el apartado 6.3.

- ✓ Algunos HLP's pueden resultar complejos.

Como se ha comentado anteriormente, una de las grandes virtudes del CAN es su característica de arbitraje para dar acceso al bus. Esta técnica se conoce también con el nombre de arbitramiento tipo "bit-wise". Se basa en la observación constante, por parte de todos los nodos que conforman la red, del nivel de señal presente en el bus. Si uno o más nodos empiezan una transmisión al mismo tiempo, pierde la arbitración del bus aquel que trata de transmitir un nivel recesivo y observa que algún otro transmite un nivel dominante. La lucha por el bus se realiza únicamente durante la transmisión del identificador. Una vez que el bus esté libre, los nodos que no hayan podido transmitir su mensaje, iniciaran una nueva competencia y de esta forma, un nuevo proceso de arbitramiento se llevará a cabo.

Un ejemplo de este proceso se puede observar en la Fig. 3.2. Los nodos del 1 al 3 empiezan a transmitir un mensaje a la vez. El nodo número 1, es el primero que pasa a ser receptor del mensaje al tratar de transmitir un bit recesivo mientras los otros dos transmitían un bit dominante. Lo mismo sucede con el nodo número 3 un tiempo después, así que por último el que logra transmitir el mensaje exitosamente es el nodo número 2. El resto de nodos se convierten en receptores. Como se aprecia, es una técnica muy sencilla pero a la vez inteligente y eficaz, ya que los mensajes permanecen intactos a pesar de detectar colisiones y no conlleva retrasos en los mensajes de alta prioridad.

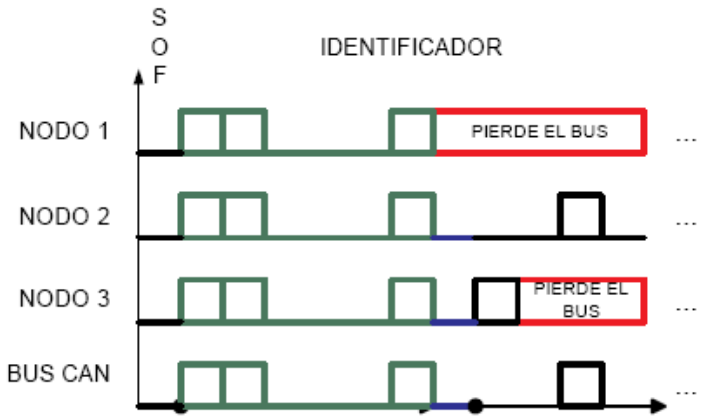

Fig. 3.2 Ejemplo de la arbitración del bus CAN

3.3 Máscaras y filtros

Hay que recordar que CAN es un protocolo que se basa en tipos de mensajes, no en direcciones. Esto significa que los mensajes no son enviados de un nodo a otro según una dirección destino, sino que todos los nodos reciben todos los mensajes que se transmiten al bus. Por tanto, no hay forma de enviar un mensaje exclusivamente a un nodo en concreto, ya que todos los nodos recogen el tráfico de la red. Pero una vez recibido el mensaje correctamente, los nodos realizan un test de aceptación para determinar si lo procesan o no. Este proceso se denomina filtrado de mensajes. El campo de la trama que se "testea o filtra" es el campo de arbitraje (identificador).

El funcionamiento de este proceso es bien sencillo. Los filtros guardan el valor del identificador, el cual se ha tenido que configurar previamente, de los mensajes de interés para el nodo en cuestión. Las máscaras se utilizan para indicar cuales bits de los 29 o 11 del identificador serán revisados al ejecutar el proceso de filtrado, es decir, cuales serán comparados. La máscara permite realizar el proceso de filtrado de forma más rápida. Si un nodo tiene distintos tipos de mensajes de interés, con la máscara podrá aceptarlos más rápido sin necesidad de revisar todo el identificador. El proceso descrito se expresa en la Tabla 3.1 y un ejemplo del mismo se observa en la Fig. 3.3:

Tabla 3.1 Tabla de la verdad de filtros y máscaras

Bit n de la máscara	Bit n del filtro	Bit n del identificador	¿Aceptado o rechazado?
0	X	X	Aceptado
1	0	0	Aceptado
1	0	1	Rechazado
1	1	0	Rechazado
1	1	1	Aceptado

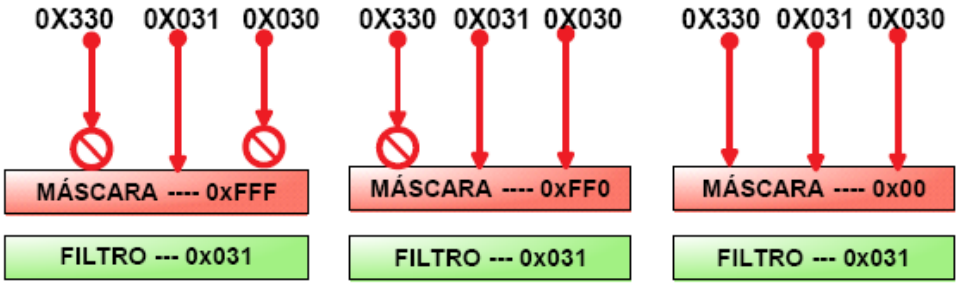

Fig. 3.3 Ejemplo del manejo de filtros y máscaras

3.4 Los componentes del medio físico

Los elementos básicos que conforman un sistema basado en el bus CAN son los siguientes: cableado, terminadores, controlador y transceptor (transmisor-receptor).

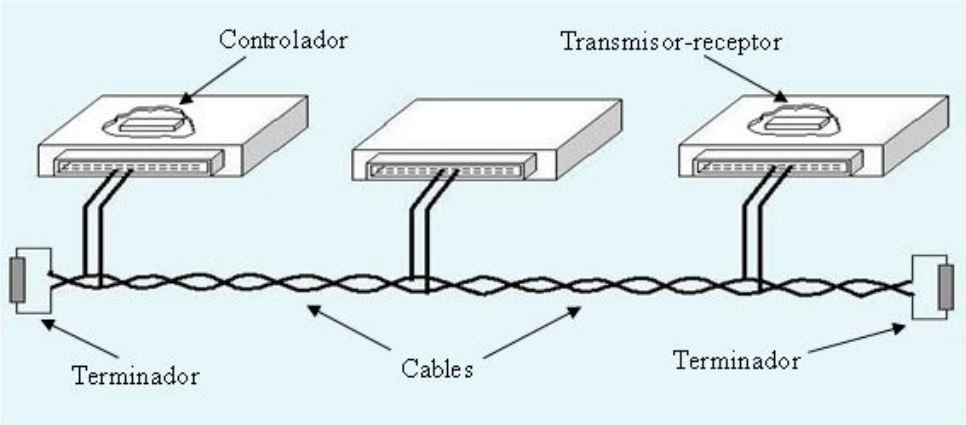

Fig. 3.4 Los componentes que forman una red basada en el bus CAN

Cableado
Los cables lo constituyen dos hilos paralelos, trenzados y/o blindados, según los requerimientos electromagnéticos. Los cables se suelen llamar CAN_H y CAN_L, de acuerdo con la polaridad de la diferencia del voltaje entre ellos.

Terminadores
También conocidos como elementos de cierre. Son resistencias conectadas a los extremos de los cables para así conseguir un bus cerrado. Permiten adecuar el funcionamiento del sistema a diferentes longitudes de cables y número de unidades de control abonadas, ya que impiden fenómenos de reflexión que pueden provocar que no se pueda alcanzar las pendientes adecuadas en los flancos de la señal. De esta forma no se producen errores en la transmisión. Para conseguir esto, estas resistencias deben ser de 120Ω.

Controlador
Es el elemento encargado de la comunicación entre el microprocesador de la unidad de control y el transceptor. Trabaja acondicionando la información que

entra y sale entre ambos componentes. En el fondo, es quien, de una manera u otra, gestiona el protocolo CAN. El controlador está situado en la unidad de control, por lo que existen tantos como unidades estén conectadas al sistema.

Transceptor (Transmisor-receptor)
El transceptor, más conocido popularmente como transceiver, es el elemento que une el controlador (por los terminales R_x y T_x) con el bus (por medio de sus terminales CAN_H y CAN_L). Tiene la misión de recibir y de trasmitir los datos, además de acondicionar y preparar la información para que pueda ser utilizada por los controladores.

3.5 Tipos de implementación

Teniendo en cuenta los elementos básicos que conforman el bus CAN, existen tres tipos de implementación donde la comunicación es igual para todas. La diferencia radica en los filtros de aceptación, en la capacidad de almacenamiento de las tramas, en la responsabilidad que asume el microcontrolador o el controlador CAN, etc., es decir, en el hardware del nodo. Las tres implementaciones incorporan un microcontrolador, ya que este componente representa una herramienta de hardware ideal para el desarrollo de aplicaciones con conexión CAN. Las tres distintas implementaciones son:

Basic CAN
En esta implementación existe un vínculo muy fuerte entre el controlador CAN y su microcontrolador asociado. El microcontrolador será interrumpido para tratar cada mensaje CAN que reciba. Aquí, el controlador CAN está restringido a un único buffer de mensajes. El microcontrolador es quién lleva el peso de las tareas haciendo así que el controlador CAN sea más simple y por tanto más barato. Este método es bueno para nodos encargados de manejar informaciones esporádicas, disminuyendo la ocupación del bus. Es la arquitectura más simple.

Full CAN
En este caso, el controlador CAN tiene varios buffers. Además tiene la capacidad para filtrar los tipos de mensaje que desee y puede transmitir y recibir mensajes sin ayuda del microcontrolador. En definitiva, el controlador le reduce la carga al microcontrolador. También se pueden habilitar interrupciones en el microcontrolador para notificarle la llegada de un mensaje. Este tipo de arquitectura consiste en un microcontrolador que incluya, no sólo sus características propias sino además un módulo CAN con las características de un microcontrolador CAN. El transceiver se sitúa de manera separada.

Serial Linked I/O
Los dispositivos Link Input/Output (SLIOS) son dispositivos de bajo coste y baja inteligencia. Son controladores sin capacidad de programación. Son interfaces preconfigurados que requieren de un nodo CAN programable para controlarlo y son usados para salidas y entradas lejanas del bus. Son dispositivos esclavos físicamente direccionados con jumpers o con switches DIP.

CAPÍTULO 4. DISEÑO DE LA INSTALACIÓN DOMÓTICA

En este capítulo se van a describir los compromisos de diseño de la instalación domótica que se han adoptado. En primer lugar se elegirá un modelo de arquitectura para la red, teniendo en cuenta solamente los conceptos relacionados con la domótica. Seguidamente se darán a conocer los resultados extraídos del estudio del protocolo CAN como posible bus para integrar en una red domótica. Y por último se citarán otras comunicaciones utilizadas en este sistema domótico como por ejemplo el puerto serie (RS232).

4.1 Modelo de arquitectura domótica

A la hora de definir un sistema domótico, hay dos aspectos que son especialmente relevantes: su tipología y sus funcionalidades. Por lo que hace referencia a las funcionalidades, se darán más detalles en el siguiente capítulo, donde se describirán los las funciones que desempeñará cada nodo de la red.

En cuanto a la arquitectura, hay que elegirla en base a los cuatro tipos que se dieron a conocer en el apartado 2.1.2. En esta ocasión se ha escogido la arquitectura descentralizada como modelo a implementar. Esta arquitectura era aquella que disponía de varios y pequeños dispositivos o controladores, a partir de ahora serán etiquetados como nodos, capaces de adquirir y procesar la información de múltiples sensores y transmitirlos al resto de dispositivos distribuidos por la vivienda.

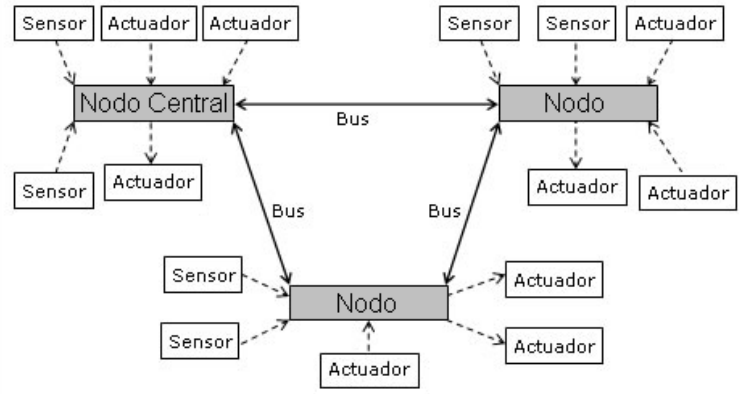

Fig. 4.1 Ejemplo de arquitectura para la instalación domótica propuesta

Resumiendo un poco, la instalación domótica que se propone (ver Fig. 4.1) estará compuesta por un número definido de nodos inteligentes. Cada nodo abarcará una dependencia distinta de la vivienda para así conseguir que ésta quede controlada por zonas. Y de cada nodo colgarán sus respectivos actuadores y sensores para así poder completar la instalación. Aunque es una instalación basada en una arquitectura descentralizada, lo que implica que existan muchos nodos y no un solo nodo, sí que habrá un nodo que tendrá la

categoría de nodo central. Será necesario que la instalación disponga de un único nodo que tenga esta categoría porque sin él, el sistema no se podría iniciar ni se podrían llevar a cabo las tareas de gestión y control de la red. Esta pequeña variación viene adoptada de la arquitectura mixta, la cual sí contempla que es uno de los nodos pueda desempeñar el papel de nodo central.

Este modelo de arquitectura seleccionado requiere de un bus para que los nodos se puedan comunicar entre ellos. Por ello el siguiente paso a realizar es elegir un bus de comunicaciones para la red de la instalación domótica.

4.2 Valoración del CAN como bus para una red domótica

En la actualidad el CAN es bus de comunicación extremadamente interesante. Lo que lo hace tan atrayente es su sencillez. Esta sencillez, junto con el resto de características descritas en el apartado 3.2, es lo que sugiere explorar el uso de CAN en redes domóticas. Las conclusiones extraídas son:

- Es un sistema que se ha ido adaptando rápidamente a otros campos gracias a su flexibilidad, lo que ha hecho que su uso se fuera extendiendo. Esto provoca que su coste sea bajo frente al coste de otros posibles sistemas.
- Posibilidad de implementación con dispositivos relativamente simples y baratos, junto con una gran cantidad de fabricantes en el mercado de dispositivos.
- Óptimo para sistemas que necesitan transmitir y recibir mensajes de realmente pequeña cantidad de información.
- Un sistema basado en tipo de mensajes en vez de en direcciones, permitiendo al bus trabajar en modo multicast.
- Rápido y robusto sistema de transmisión de mensajes con un mecanismo de tratamiento de errores muy eficaz. Es un protocolo muy robusto frente a los problemas de ruido.

Todo ello apoya la idea del CAN como un bus adecuado para redes domóticas. Pero el razonamiento decisivo ha sido que el CAN es un bus sencillo (gran facilidad de desarrollo) e inteligente, y eso es lo que hace muy interesante sobre todo desde el punto de vista del ingeniero a la hora de diseñarlo e incorporarlo como bus domótico.

Es curioso como entre los buses que se citaban en el apartado 2.1.3 no aparecía el CAN. Aunque se puede encontrar algún que otro fabricante o empresa que proporciona aplicaciones domóticas basadas en CAN, especialmente en Estados Unidos, es cierto que tales soluciones escasean. Es extraño que este protocolo no se haya implementado aún en el campo de la domótica, más después de haber llegado a la conclusión de que el CAN es un bus que se adapta perfectamente a las características demandadas por una instalación domótica. Sin ir más lejos, inicialmente CAN fue definido como un bus orientado a trabajar con dispositivos inteligentes, sensores y actuadores, dentro de diversos sistemas distribuidos y esto, es totalmente predominante en sistemas domóticos.

4.2.1 Tipo de implementación escogida

Una vez visto que el CAN reúne las prestaciones adecuadas para ser implementado como bus domótico, se ha de proceder a seleccionar una de las tres implementaciones que ofrece dicho protocolo (ver apartado 3.5).

La arquitectura que se ha decidido utilizar en esta instalación domótica es la del Full CAN. Su elección se debe más que nada a la restricción que nos impone el material que se va a utilizar para dicha instalación. Ya se verá en el siguiente capítulo que la composición de los nodos se basa en una pequeña placa comercial la cual proporciona una conexión vía CAN. Pero para conseguir ese bus de comunicación con tal placa es necesario que la arquitectura esté formada por un microcontrolador con controlador de CAN incorporado, y por un transceiver externo. La elección de estas placas se realizó teniendo en cuenta el tipo de implementación que exigían, ya que es la más compacta y la que mejor se adapta a las necesidades que se buscan para este sistema. Para más detalles sobre los componentes seleccionados consultad los capítulos 5 y 6.

4.3 Otras comunicaciones del sistema

A parte del bus CAN hay dos tipos de comunicaciones más que se utilizarán en el sistema final. Uno de ellos es el puerto serie (RS232), y el otro es un puerto de programación para microcontroladores. Si se desea conocer más detalles acerca del protocolo RS232 así como del puerto de programación para microcontroladores, puede consultarse el anexo B.

Cada uno de los nodos que se han diseñado incorpora una conexión de puerto serie para mandar información vía ordenador. Es decir, en la comunicación RS232 los transmisores son todos los nodos de la red y la función de receptor la realiza un ordenador. Solo se utilizará la transmisión en ese sentido, es decir, no habrá respuesta por parte del el ordenador a través del puerto serie. Aún así, físicamente se dejará preparado el hardware del nodo para que también se pueda realizar la comunicación en el otro sentido.

El motivo de la utilización del puerto serie viene más impuesto por un requisito del ingeniero y no como una utilidad para el usuario, ya que su uso ha estado ligado a la etapa de desarrollo y verificación de los nodos. Su función era la de comprobar que todas las rutinas y acciones realizadas por el nodo eran correctas, así como las transmisión y recepción de mensajes CAN y sus contenidos. Aunque posteriormente se ha diseñado una aplicación para PC que permite monitorizar el bus, seguía siendo necesario el uso del puerto serie pues es la única comunicación que nos informa de forma directa y segura lo que sucede. De esta manera también se le daba validez a esta aplicación. Por lo tanto, no se le ha especificado ni asignado una función concreta al RS232 para la aplicación final una vez finalizada la etapa de desarrollo del sistema. Por eso quizás puede parecer coherente prescindir de tal comunicación. Pero a pesar de esto, se ha decido mantener este protocolo en el diseño final del nodo, ya que pensando solamente en términos de mantenimiento posteriores de cada nodo, puede resultar una comunicación muy válida e útil.

CAPÍTULO 5. DISEÑO Y DESARROLLO DE LOS NODOS

En esta parte de la memoria se va a dar a conocer todas las tareas y explicaciones relacionadas con el diseño previo de los nodos y su implementación posterior, para así poder poner en marcha un prototipo de instalación domótica al interconectar ese conjunto de nodos creados. Al tratarse de nodos serán inteligentes e independientes han de ser capaces de ejercer las acciones pertinentes por sí mismo, o dicho en otras palabras, capaces de controlar su entorno inmediato, a la vez que permanecen conectados con el resto de nodos a través de la red o bus CAN. Concretamente se han diseñado y desarrollado tres nodos, suficientes para crear una instalación domótica sencilla y básica.

5.1 Tipos de nodos y descripción de los servicios

La instalación domótica que se propone contempla diferentes tipos de nodos. Lo que distingue un tipo de otro es por un lado su composición hardware y por otro lado su programación (software). La primera distinción se detalla en el apartado 5.1.1 y la segunda en el apartado 5.1.2.

5.1.1 Según el hardware

El hardware de cada nodo va asociado directamente a la función principal que está destinado a ejercer. Según esto se han definido dos tipos: el nodo estándar y el nodo interfaz. El nodo estándar se puede catalogar como el nodo “normal” o predominante. En cambio, el nodo interfaz es un nodo “especial” que permite una interacción más intuitiva y directa. Ha de quedar claro que este nodo no es más que un “extra” que se ha decido añadir al sistema, pero su uso podría ser totalmente omisible desde el punto de vista del buen funcionamiento de la red. Lo que es el grueso de la instalación domótica estará formado por los nodos estándar. Una red mínima ya se podría formar simplemente con dos nodos estándar sin necesidad del nodo interfaz, formando una instalación domótica muy sencilla pero viable. Es más, gran parte de las funcionalidades que ofrece el nodo interfaz podrían ser implementadas perfectamente en cualquier nodo estándar, aunque de forma más rudimentaria, pero sería totalmente posible.

Por tanto, como la diferencia entre los dos tipos de nodos radica en su hardware, a continuación se va a describir la composición de cada uno.

La idea final de un nodo estándar no es nada más que un dispositivo con unas determinadas entradas y salidas para poder conectarle sensores, actuadores, etc. Concretamente dispondrá de entradas para sensores analógicos y digitales, entradas para pulsadores y salidas para accionar relés de dos

contactos o dimmers. El número de estas salidas y entradas dependerá de la zona que abarque el nodo, es decir, de los servicios que incorpore.

El hardware de los nodos estándar se puede dividir en dos partes. Por un lado, como el controlador de los nodos estándar estará fundamentado en un microcontrolador, más concretamente por los PIC's (modelo fabricado por *Microchip*), será necesario un circuito mínimo para su correcto funcionamiento. Esta parte la proporcionará unas placas comerciales que se han adquirido para que formen la base del hardware de un nodo estándar. Esta placa comercial (o board), es la SBC28PC de *Modtronix* [6]:

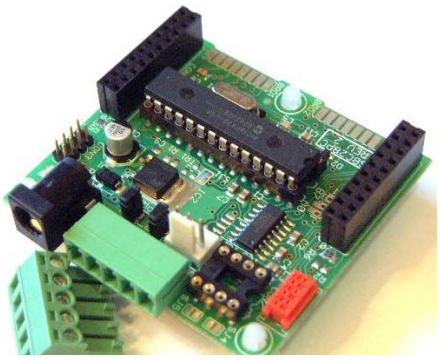

Fig. 5.1 Board SBC28PC

Esta placa está diseñada para ser utilizada con cualquier microcontrolador de 28 pines. Como se pretendía, incluye todo el acondicionamiento necesario para que el PIC funcione correctamente, como un reloj de 20 MHz, un regulador para la alimentación, reset (MCLR), etc. No solo incorpora eso, sino que también dispone de tres interfaces o comunicaciones: puerto serie (RS232), I2C y CAN BUS (no se pueden utilizar simultáneamente). Para poder utilizar el CAN es necesario montar un PIC que incluya un controlador de CAN (se ha escogido la familia PIC18FXX8 de 28 pines: 18F248 ó 18F258 [7]). También incorpora un zócalo para insertar el transceiver de CAN. Mediante un jumper se activa o desactiva la resistencia de terminación que lleva incluida. Además, la placa proporciona una serie de entradas y salidas (puertos I/O) que comunican con los distintos pines del PIC. La alimentación puede oscilar entre 7 y 30 V. En este caso, los nodos estándar estarán alimentados a 24 V, la alimentación típica en domótica.

La segunda parte que completa el hardware del nodo estándar está formada por los componentes necesarios para darle las funcionalidades requeridas a cada una de las entradas y salidas del nodo, así como otros servicios. En el sistema final esta parte del hardware se implementará en una placa PCB, pero durante el proceso de pruebas se montará sobre una "protobard". Ambas soluciones se conectarán a la placa comercial a través de los puertos I/O que comunican con el PIC, formando el conjunto del hardware para todo nodo estándar.

En cuanto al nodo interfaz, éste estará compuesto por el módulo USB-CAN (ver apartado B.3 del anexo) y por una computadora (PC, UMPC, PocketPC, PDA,

etc.) capaz de poder ejecutar el software de la aplicación de alto nivel que ha sido diseñada. La función de este nodo es la de interactuar o comunicar a una persona (diseñador o residente) con la instalación domótica. Desde este nodo se podrán dar órdenes, acciones, instrucciones, etc. genéricas sobre toda la red. Además permitirá observar y controlar (monitorizar) toda la información de la vivienda que circula a través del bus CAN.

Los tres nodos que se han desarrollado son: dos estándar (lo mínimo necesario para formar una red) y uno interfaz (con uno por red será siempre suficiente).

5.1.2 Según el software

La segunda distinción, de más bajo nivel, está sujeta a la programación que lleva cada nodo, es decir, a los servicios que ofrecen ya que el software se distinguirá por las funciones que incorporen. Que un nodo tenga una programación con unas funciones concretas y no otras, dependerá de la zona de la vivienda a la cual esté destinado a controlar. Es necesario diferenciar entre los nodos estándar a nivel de programación porque de esta forma se agrupan nodos que requieren de necesidades muy similares, ya que si son del mismo tipo, sus softwares serán idénticos. Esto es de gran ayuda, especialmente para el protocolo, para por ejemplo conseguir que los nodos de un específico tipo procesen mensajes de tipo “Broadcast” que van dirigidos para ellos. Es evidente que esta segunda categoría de tipos de nodos solo diferenciará entre sí a los nodos estándar. El nodo interfaz no necesita de ninguna distinción porque en caso de que haya alguno en la red, sería único y por lo tanto su software también.

En resumidas cuentas se puede decir que los nodos estándar a su vez también están diferenciados en varios tipos, los cuales solamente se distinguen en la red a nivel de programación. Estos tipos son los siguientes:

- **Nodo Interior:** aquellos nodos que controlan zonas del interior de la vivienda. Éstos a la vez se diversifican en:

    1. **Zona de paso:** aquellos nodos situados en zonas de tránsito, como un pasillo, un recibidor, etc.
    2. **Zona no de paso:** los nodos que no están situados en zonas de paso, o dicho de otra manera, el resto de nodos interiores. Aquí ya se podría distinguir en función de la zona que controlan: habitación, salón, cocina, etc.

- **Nodo Exterior:** aquellos nodos que controlan zonas del exterior de la vivienda, como por ejemplo, una terraza o un jardín.

Existe también otra categoría de nodo a nivel de programación que hay que tener en cuenta. Esta otra clase de nodo se denomina **central**. No se ha descrito en el listado anterior porque no identifica un tipo de nodo concreto, sino más bien especifica unas funcionalidades extras de las cuales dispone el nodo. Cualquier tipo de nodo de los descritos con anterioridad (interior de paso,

interior no de paso o exterior) puede desarrollar las funciones de central. La idea es aprovechar un nodo ya existente en la red para que realice esas funciones en vez de crear uno expresamente para ello. Un nodo de categoría central viene a hacer las funciones de nodo principal de la red. Su función es la de llevar todo el control de la red de forma genérica así como mantener una comunicación directa con el nodo interfaz. Es necesario que de todos los nodos estándar que forman la red domótica uno tenga la categoría de central.

5.2 Explicación del software para Nodos Estándar

Los nodos estándar se han programado en lenguaje C para microcontroladores PIC. Esta programación se ha realizado con el compilador de la casa CCS, denominado "*PCWH Compiler*" [8]. El software de cada nodo estándar consiste en un proyecto, creado con dicho compilador, que consta de varios archivos en lenguaje C relacionados entre sí. Para entenderlo mejor, se han agrupado los archivos que contiene este proyecto según su contenido o temática. En la Fig. 5.2, aparte de poder observar los módulos (agrupaciones) y sus denominaciones, también se aprecian las relaciones entre ellos (flechas).

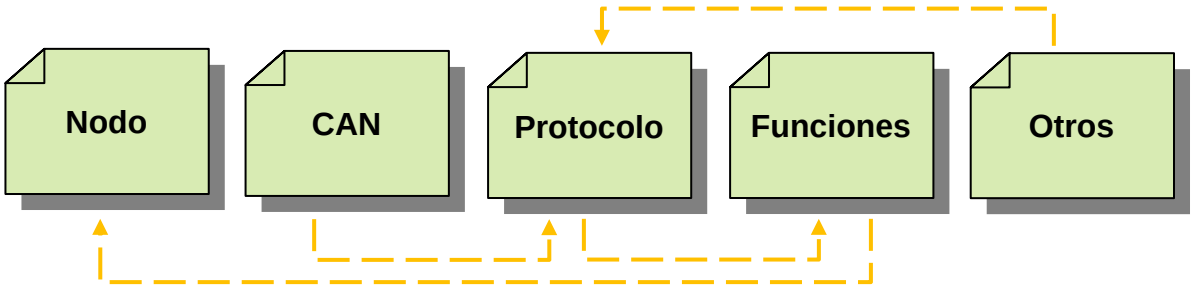

Fig. 5.2 Distribución de los archivos del software para un Nodo Estándar

Todo el contenido formado por los módulos "CAN, Protocolo, Funciones y Otros" es común (idéntico) para todos los nodos estándar independientemente del tipo nodo que se trate (interior, exterior, etc.). La única distinción entre los software de cada nodo se halla en el módulo "Nodo".

En el módulo "Nodo" es donde se especifica el tipo de nodo según el software, es decir, si es interior de paso, interior de no paso o exterior, así como si ejerce las funciones de central o no. Dependiendo del tipo que especifique, lógicamente este módulo llevará una programación basada en unas funciones u otras para dar el tipo de servicios, rutinas, automatismos, etc. que se adapten al tipo de nodo. Para nodos que sean del mismo tipo, este módulo será idéntico por lo que todo el software será el mismo. Este módulo está formado por un archivo con el código fuente en C y una librería (aquellos archivos que contienen las cabeceras utilizadas para definir los pines, los registros, los registros de los bits, las funciones y las directivas de preprocesador, etc.). Para que este módulo pueda funcionar correctamente necesita del contenido del resto de módulos, los cuales están entrelazados a través del módulo "Funciones", que será por tanto al que invocará este módulo (ver Fig. 5.2). Como el módulo "Nodo" es distinto según su tipo, el contenido del mismo se

detallará más adelante para los ejemplos de nodos diseñados (ver apartados 5.3.3 y 5.4.3). En cambio el resto de módulos, al ser siempre idéntico su contenido para todo tipo de nodo, se explicará a continuación a grandes rasgos en qué consisten.

Por un lado está el módulo “CAN”. Este módulo es el que incluye las funciones básicas para configurar el controlador de CAN, así como enviar y recibir una trama de datos o remota. Está pensado para funcionar en un controlador CAN de los que van incorporados en la familia de microcontroladores PIC18FXX8. El módulo “CAN” está compuesto de dos archivos: uno con el código fuente en C, y una librería. Los dos archivos que contiene ese módulo de “CAN” ya estaban creados e incluidos entre las librerías que proporciona CCS con su compilador. Aún así, ambos han sido bastante modificados para poder adaptarlos posteriormente al módulo “Protocolo”.

El módulo “Otros” contiene varios archivos, todos ellos extraídos de los que también proporciona CCS, necesarios para realizar algunas funciones especiales o para poder utilizar algún dispositivo en especial. En la mayoría de casos han sido ligeramente modificados. Por ejemplo, uno de ellos contiene las directrices necesarias para utilizar un temporizador (reloj en tiempo real).

El módulo “Protocolo” es aquél que contiene el protocolo de alto nivel (HLP) creado en este proyecto. Básicamente su función es la de interpretar los mensajes del bus CAN y así poder operar con ellos de forma sencilla y semiautomática (para más información al respecto ver el capítulo 6). Está formado únicamente por un archivo con el código fuente en C, el cual necesita de los módulos “CAN” y “Otros” para funcionar correctamente.

El módulo “Funciones” dispone de las rutinas básicas y comunes para el correcto funcionamiento de un nodo cualquiera de la red. Nuevamente está compuesto de un archivo con el código fuente en C y una librería. Es necesario llamar al módulo “Protocolo” para que pueda funcionar.

5.3 Nodo Estándar: Interior Zona de Paso (Central)

5.3.1 Funcionalidad

El primer nodo estándar de los dos que se han diseñado es de tipo interior, más concretamente de Zona de Paso. Además se ha escogido como el nodo que a la vez cumplirá con la categoría de nodo central. El nodo controlará en esta ocasión la entrada de una vivienda (recibidor) con un pasillo adjunto. Los automatismos principales que llevará a cabo este nodo serán dos:

1.- Automatización de la iluminación. Encendido de las luces si el nivel de luz interior detectado está por debajo de cierto umbral mínimo. En caso contrario la iluminación permanecerá apagada. Esta automatización será gestionada en función de si se detecta a alguien o no en la zona controlada.

2.- Control del acceso a la vivienda. La puerta de entrada a la vivienda incorporará una cerradura electrónica. Abrir la puerta tanto desde fuera como desde dentro de la vivienda se llevará a cabo a través de un sistema eléctrico gestionado por este nodo. El doble cierre de la cerradura será automático.

Seguidamente se describen las salidas y entradas que tiene el nodo (tipo y cantidad), y qué dispositivos se les puede conectar. También se da a conocer la función de cada uno de esos dispositivos:

Tabla 5.1 Prestaciones del nodo interior zona de paso (central)

Dispositivo	Tipo de puerto	Nro. de puertos	Función
Sensor de luz	Entrada analógica	1	Controlar la luminosidad de la zona, necesaria para realizar la automatización de la iluminación.
Sensor digital	Entrada digital	2	Un sensor de presencia para controlar la automatización de la iluminación, y un sensor de posición para controlar el estado de la puerta necesario para el control de acceso a la vivienda.
Alumbrado (luces)	Salida digital	2	Controlar un sistema de iluminación, formado, por ejemplo, por dos bombillas.
Alarma	Salida digital	1	Avisador lumínico o sonoro. Se activa cuando llega una petición de evento especial (ver apartado 6.1).
Relé de 2 contactos	Salida digital	1	Control de la cerradura electrónica de la puerta de entrada a la vivienda.
Pulsador de doble pulsación	Entrada digital	1	La pulsación corta es para contestar a una petición de actividad de evento especial (cuando se activa la alarma) y la pulsación larga para abrir la puerta de entrada desde el interior de la vivienda.

Internamente el nodo incluirá más características y prestaciones. Por ejemplo, el hardware interno del nodo incorpora un temporizador (o reloj en tiempo real) para poder conocer en todo instante que lo requiera la situación los datos sobre la fecha y hora (día de la semana, mes, año, hora, minutos, etc.). Este dispositivo solo lo incorporará el nodo que haga las funcionalidades de central. El resto de nodos si necesitan conocer la fecha u hora harán una petición al nodo central para que se los proporcione a través del bus CAN.

5.3.2 Hardware

En este apartado se explicará y detallará la parte del hardware del nodo que aporta las configuraciones necesarias para acondicionar las entradas y salidas del nodo, así como otros componentes que proporcionan otras funcionalidades.

En el proceso de desarrollo de los nodos hay que diferenciar dos etapas. La primera etapa es la de pruebas, en la cual el montaje se realiza sobre una "protoboard". Una vez realizadas las verificaciones y modificaciones oportunas, se implementa la etapa final, donde esta parte del hardware del nodo se monta sobre una PCB para que quede totalmente anclado y compacto sobre la placa comercial. Todo junto formará el hardware del nodo.

En el montaje sobre la "protoboard", no solamente se ha introducido la parte del hardware propia del nodo, sino también aquellos dispositivos que van conectados a las entradas y salidas que necesita para funcionar: pulsadores, leds (imitando el funcionamiento de unas bombillas), sensores analógicos y jumpers (imitando la respuesta de los sensores digitales todo/nada). Esto era necesario para hacer las pruebas pertinentes, pero estos dispositivos no forman parte del hardware del nodo. Las siguientes figuras muestran el esquemático del circuito realizado y su montaje, todo sobre la "protoboard":

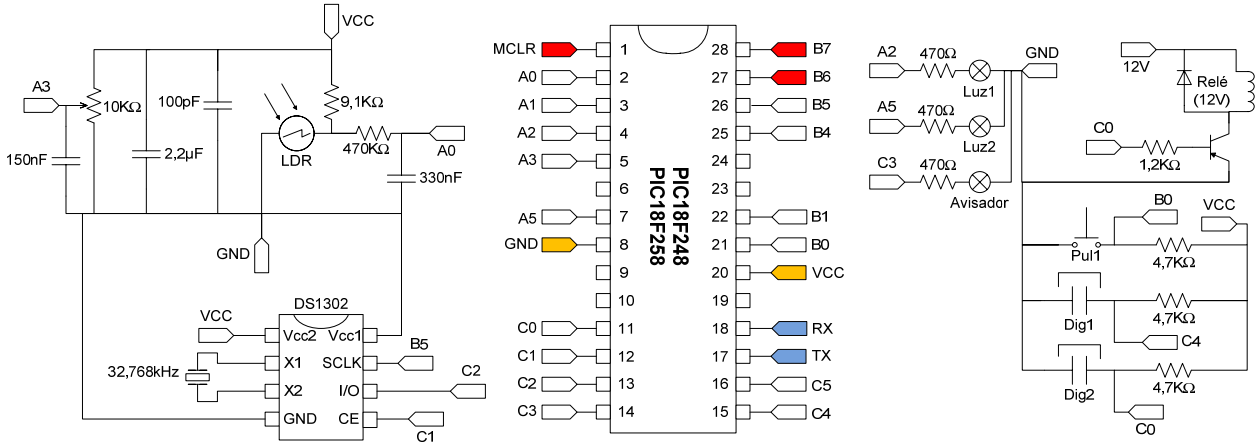

Fig. 5.3 Esquemático de una parte del hardware para el Nodo Interior Zona de Paso (Central)

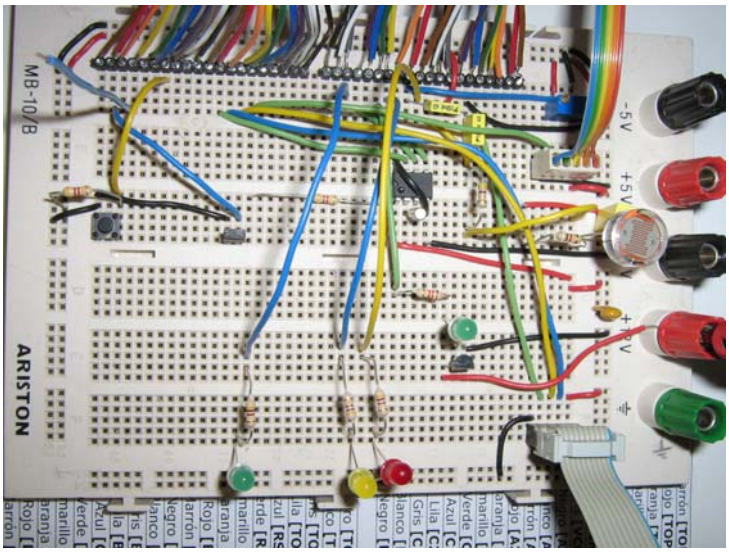

Fig. 5.4 Resultado del montaje sobre "protoboard" de una parte del hardware para el Nodo Interior Zona de Paso (Central)

El microcontrolador utilizado para este nodo será el PIC18F258 (al realizar las tareas de nodo central es necesario de la mayor memoria tanto RAM como ROM disponible, de ahí que se haya escogido este PIC y no el 248), el cual va insertado en la placa comercial SBC28PC.

En la Fig. 5.3 se aprecia que el circuito diseñado es muy simple. Los pines del PIC que están marcados con colores marcan aquellos pines utilizados para la comunicación. Los de color rojo junto con los naranja (alimentación y masa) conjuntan las cinco conexiones ICD del programador PICFLASH. Por otro lado los pines azules más los naranja forman las cuatro conexiones necesarias para la conexión del puerto serie. Los pines de la comunicación CAN no se han marcado pues ya se encarga la placa SBC28PC de realizar dicha conexión.

Las entradas digitales se han acondicionado con una resistencia de "Pull-up" que conecta un pin de entrada digital con V_{cc}. Por lo tanto, tanto el pulsador como los jumpers serán normalmente cerrados, ya que si el pulsador no está presionado, el valor del pin será cercano a 5 V (nivel alto), en caso contrario será cercano a 0 V (nivel bajo).

La salida prevista para conectar un relé, ha de ser uno cuya bobina funcione con 12 V y consuma 50 mA. Como el PIC no puede suministrar esa corriente se pone un transistor. Entre el pin del PIC (que es el que da la señal de conduce o no conduce) y el transistor se coloca una resistencia encargada de ajustar la corriente que debe suministrar a la Base del transistor para que entre el Colector y el Emisor conduzca corriente. El PIC solo debe suministrar esa corriente necesaria, luego ya se encarga el transistor de proporcionar la corriente al relé, con una intensidad que depende de lo que aguante el transistor y no de la que proporcione el pin del PIC. El transistor elegido es el BC183 [14]. Al necesitar el relé una I_C de 50 mA y tener el transistor una ganancia mínima de 125 (β), el suministro de corriente a la base mínimo sería de:

$$I_B \geq I_C / \beta = 50 / 125 = 0{,}4 \text{ [mA]} \quad \textbf{(5.1)}$$

Por tanto la resistencia máxima que se pude poner a la entrada del pin es de:

$$R_{máx} \leq (5 - 0{,}7) / 0{,}4 = 10{,}75 \text{ [k}\Omega\text{]} \quad \textbf{(5.2)}$$

Los 5 V son la tensión que proporciona el pin del PIC y 0,7 V es la V_{BE} (la caída de tensión que sufre el voltaje que se aplica a la Base al atravesar el transistor por la unión Base - Emisor). Por tanto se ha de utilizar una resistencia que esté por debajo de 10,75 kΩ. La elegida es de 1,2 kΩ.

Otro dispositivo utilizado ha sido el temporizador o reloj en tiempo real, en concreto, el modelo DS1302 de MAXIM [15]. Este simplemente necesita de tres pines digitales del PIC, más la alimentación y masa correspondientes junto con un reloj de cristal de 32.768 kHz.

El circuito que incorpora la entrada analógica dispuesto para conectar un sensor de luz concretamente una LDR (como por ejemplo la Norps-12 [16]), estará conectado a un pin analógico del PIC, lo cual indica que la salida

generada por este será enviada a un conversor analógico digital (ADC) para convertirla en una señal digital capaz de ser interpretada por el microcontrolador. El circuito utilizado no es más que un divisor resistivo entre una resistencia de 9,1 kΩ y la LDR conectada a dicha entrada. La tensión medida en el punto medio de este divisor (por el pin) variará según varíe la luz que incida sobre la LDR. Debido a que el valor resistivo de la LDR decrece a medida que la luz aumenta, en consecuencia la tensión en el punto medio disminuirá también a medida que la luz aumente y viceversa. Este cambio de iluminación traducido a cambio de tensión y transformado a código digital por el ADC es fácilmente entendible por el PIC. Además se ha incorporado un filtro paso bajo con una frecuencia de corte de 1 Hz aproximadamente para eliminar las oscilaciones de la señal analógica a la entrada del PIC:

$$F_c = \frac{1}{2 \cdot \pi \cdot R \cdot C} = \frac{1}{2\pi \cdot 470\text{k}\Omega \cdot 330\text{nF}} = 1{,}026\ [\text{Hz}] \qquad \textbf{(5.3)}$$

También se ha dispuesto de un potenciómetro de multivuelta para ajustar la tensión de referencia (o de fondo de escala) para el ADC. El circuito vendría a ser semejante al del sensor analógico, un divisor formado por el propio potenciómetro, entre alimentación y masa, con el pin conectado en el punto medio del divisor. Entre ese punto y masa también se ha colocado un condensador para eliminar posibles ruidos que contenga la señal de alimentación. Por último, que se ha insertado ente la alimentación (V_{CC}) y masa dos condensadores (de desacoplo) para estabilizar la alimentación que proporciona el integrado (regulador de tensión de la placa comercial).

Ya en la segunda etapa, es decir, en el sistema final, se ha creado la PCB de todo lo comentado anteriormente. Su diseño es idéntico al esquemático visto en la Fig. 5.3 para la protoboard, con la única diferencia que solo contiene el hardware que pertenece al nodo, y no aquellos dispositivos que puede tener conectados. Aun así, por razones de comodidad, se le ha incluido el sensor LDR y los jumpers que imitan la respuesta de los sensores digitales todo/nada. Estrictamente hablando, estos componentes no deberían ir incluidos en la PCB, es decir, en el hardware del nodo completo, sino que más bien deberían ir conectados al nodo a través de sus entradas correspondientes.

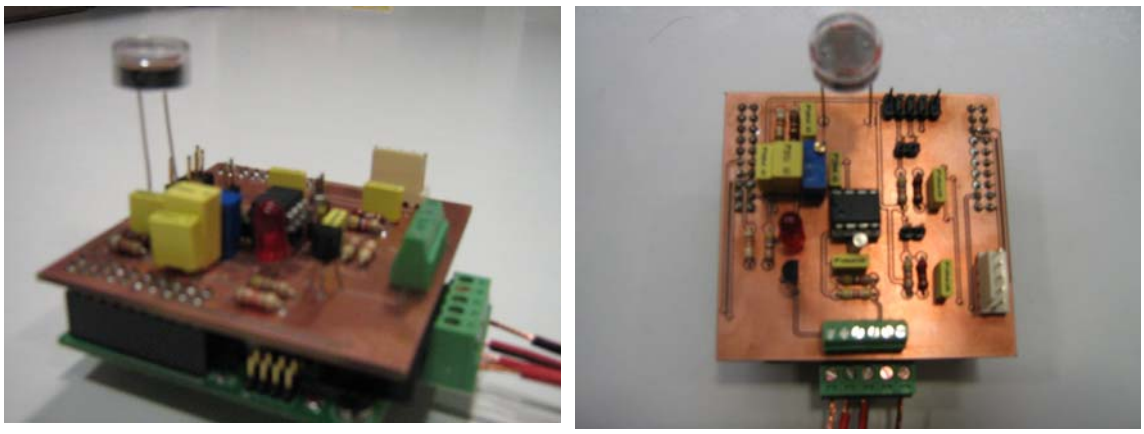

Fig. 5.5 Nodo completo: placa comercial y PCB

Otro pequeño cambio que se ha realizado en el circuito montado en la PCB respecto el esquemático de la Fig. 5.3, afecta al circuito de acondicionamiento para las entradas digitales. A cada una se le ha añadido una resistencia y un condensador en serie para proteger el conector de corrientes elevadas que se puedan introducir por error, por ejemplo, al equivocarse al conectar el dispositivo donde no toca.

5.3.3 Software

A continuación se va a describir a grandes rasgos el código en lenguaje C incluido en el módulo “Nodo”, para un nodo interior que controla una zona de paso y entrada a una vivienda y que además tiene las funcionalidades de nodo central, a partir de su diagrama de flujo:

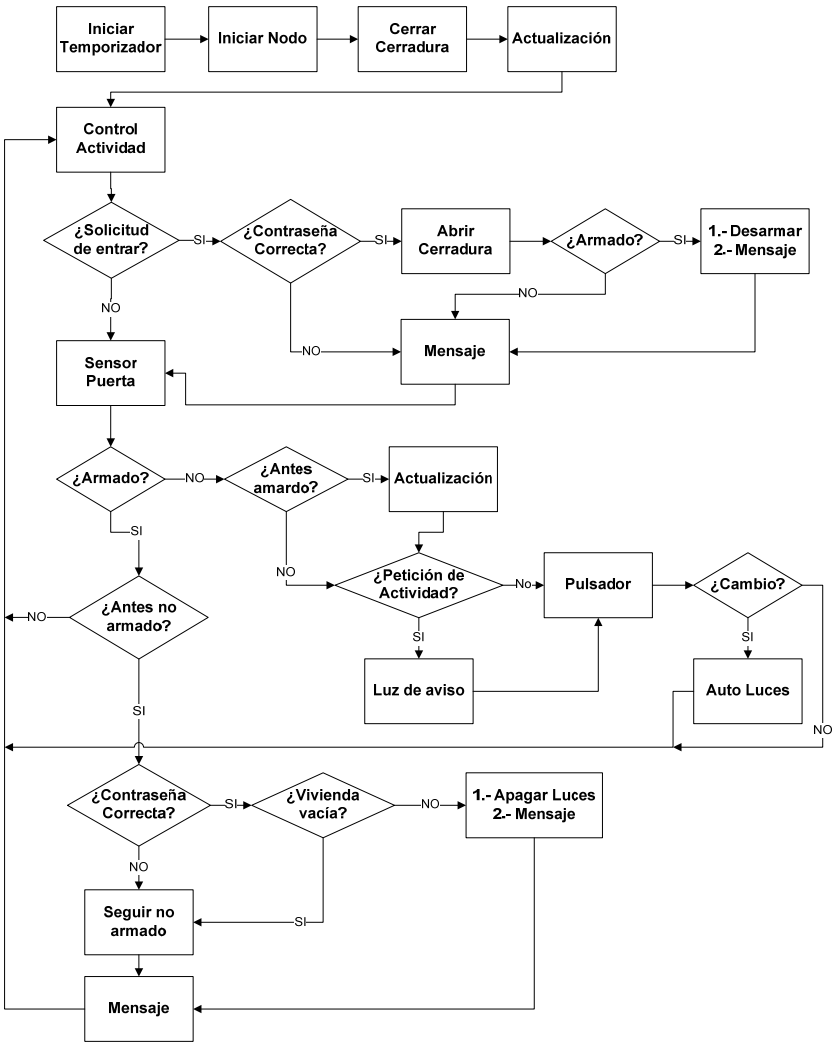

Fig. 5.6 Diagrama de flujo del Nodo Interior Zona de Paso (Central)

Inicialmente el nodo configura e inicia el temporizador. Después llama a una función del módulo “Funciones” que configura varios parámetros propios del nodo como las interrupciones, la configuración del CAN y del ADC, etc. De

seguida cierra la cerradura y actualiza el sistema (lee el estado de los sensores digitales y analógicos, y en función de éstos enciende o apaga las luces).

A partir de ahora se entra en un bucle infinito. Todas las instrucciones del código se realizan repetidamente, es decir, es un programa basado en polling. Este bucle lo primero que hace es ejecutar la función del control de actividad de la red, una característica intrínseca del nodo central. A continuación se comprueba si se ha recibido un mensaje con la petición de entrar en la vivienda. Si es así, se asegura que la contraseña que se introdujo es la correcta y abre la cerradura. Si la casa estaba armada (todos los nodos configurados en el modo establecido para cuando no hay nadie en la vivienda), la desarma indicándoselo al resto de nodos a través de un mensaje. Por último se envía un mensaje al nodo interfaz con la respuesta a la petición de entrar (si es correcto o incorrecto por error de contraseña). Después otra función propia del nodo se encarga de cerrar la cerradura cuando la puerta se cierra.

El resto de código consiste en comprobar el estado de la vivienda. Si la vivienda está armada, es decir, no hay nadie, vuelve al inicio del bucle. Si está armada pero es la primera comprobación después de recibir la petición para armarla, comprueba que la contraseña de la petición de armar sea correcta y que no haya nadie en la vivienda. En ese caso apaga todas las luces y envía un mensaje al resto de nodos para que se armen. También envía un mensaje al nodo interfaz con la respuesta de la petición de armar (si ha sido correcta o incorrecta por error de contraseña o por error de presencia).

En el caso de que la vivienda no esté armada, primero se asegura de que no venga de estarlo antes. Si es así realiza nuevamente una actualización. Seguidamente se verifica si ha llegado una petición de actividad a evento especial. Si ese fuera el caso se activa el aviso o alarma (sonoro, lumínico, etc.). El paso siguiente es constatar el pulsador. En función del tipo de pulsación se lleva a cabo las tareas oportunas (si es pulsación larga abre la cerradura y si es pulsación corta contesta a la petición de actividad desconectando el aviso). Por último, si se ha producido algún cambio en el estado de los sensores digitales o el sensor analógico detecta que el valor capturado está por debajo de cierto umbral (comprobación que se hace cada determinado tiempo, no cada instrucción) o que ha cambiado el valor de ese umbral, entonces actualiza las luces en función de los valores obtenidos.

5.4 Nodo Estándar: Zona no de Paso (Habitación)

5.4.1 Funcionalidad

El segundo nodo estándar de los dos que se han diseñado y creado es también de tipo interior, más concretamente de Zona no de Paso. Es decir, se trata de uno del resto de los nodos interiores que no controlan una zona de paso. En esta ocasión, el nodo está destinado a controlar una la dependencia de una vivienda prevista para una habitación. Sus automatismos principales son dos:

1.- Automatización y control de la iluminación. Encendido de las luces si el nivel de luz interior detectado está por debajo de cierto umbral mínimo establecido. En caso contrario permanecerá la iluminación apagada. Esta automatización será gestionada en función de si se detecta a alguien o no en la zona que controla el nodo. También existirá la posibilidad de controlar la iluminación manualmente a través de un pulsador. La conmutación entre el control manual o automático de la iluminación se realizará con otro pulsador.

2.- Automatización y control de la calefacción. Encendido de la calefacción si la temperatura interior está por debajo de cierto umbral mínimo. En caso contrario permanecerá apagado. Esta automatización será gestionada en función de los estados de la puerta y la ventana. También existirá la posibilidad de desconectar el control automático de la calefacción mediante otro pulsador.

Seguidamente se describen las salidas y entradas que tiene el nodo (tipo y cantidad), qué dispositivos se les puede conectar y cuáles son sus funciones:

Tabla 5.2 Prestaciones del nodo interior zona no de paso (habitación)

Dispositivo	Tipo de puerto	Nro. de puertos	Función
Sensor de luz	Entrada analógica	1	Controlar la luminosidad de la zona, necesaria para realizar la automatización de la iluminación.
Sensor de temperatura	Entrada analógica	1	Controlar la temperatura de la zona, necesaria para realizar la automatización de la calefacción.
Sensor digital	Entrada digital	3	Un sensor de presencia para controlar la automatización de la iluminación, y dos sensores de posición para controlar los estados de la puerta y la ventana, necesarios para realizar la automatización de la calefacción.
Alumbrado (luces)	Salida digital	3	Controlar un sistema de alumbrado, formado, por ejemplo, por tres bombillas.
Alarma	Salida digital	1	Avisador lumínico o sonoro. Se activa cuando llega una petición de evento especial (ver apartado 6.1).
Relé de 2 contactos	Salida digital	1	Control de una válvula de cierre del agua para la caldera de la calefacción.
Pulsador simple	Entrada digital	1	Para responder a una petición de actividad de evento especial (cuando se activa la alarma).
Pulsador doble	Entrada digital	2	**Pulsador 1:** para controlar manualmente las luces. La pulsación corta enciende secuencialmente las luces y la pulsación larga las apaga también secuencialmente.

Pulsador 2: para activar/desactivar los modos automáticos de la iluminación (pulsación corta) o de la temperatura (pulsación larga).

5.4.2 Hardware

En esta sección se va a explicar y detallar aquella segunda parte del hardware que pertenece a un nodo estándar, en esta ocasión, para un nodo interior zona no de paso (habitación). En las siguientes figuras se aprecia el esquemático del circuito realizado así como su montaje, todo sobre la "protoboard":

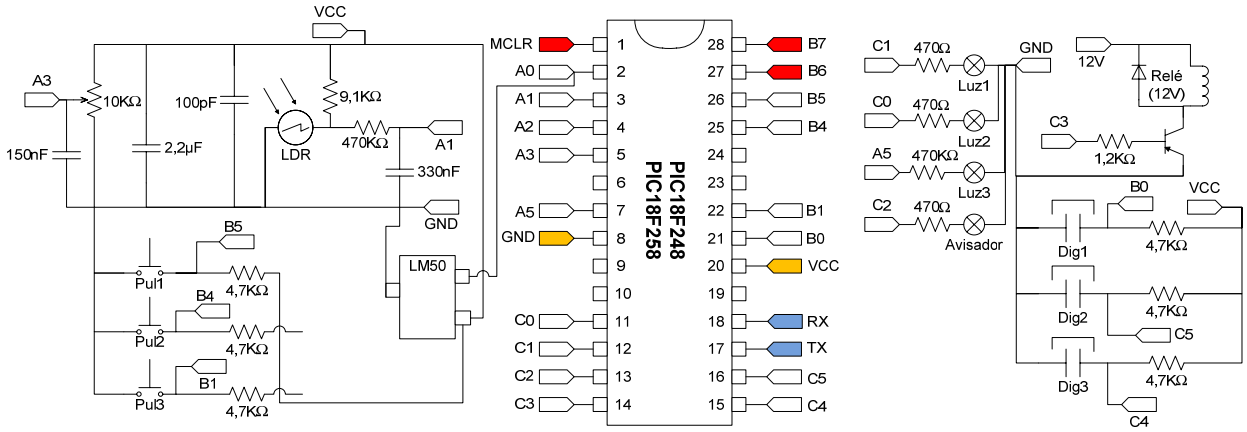

Fig. 5.7 Esquemático de una parte del hardware para el Nodo Interior Zona no de Paso (Habitación)

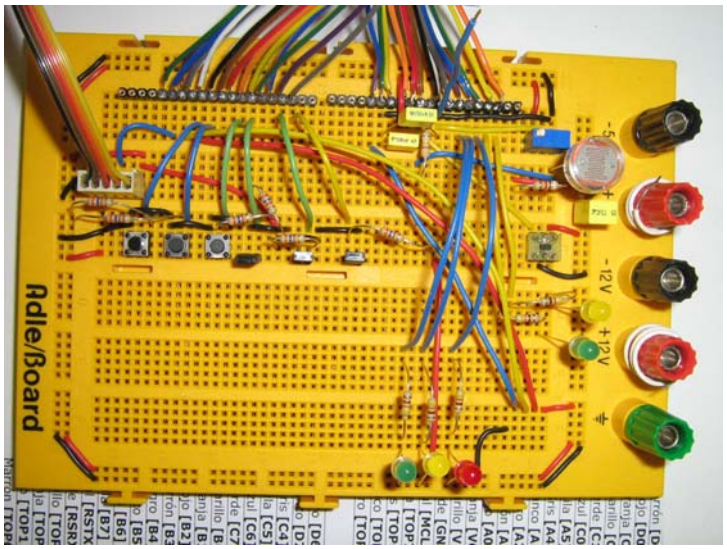

Fig. 5.8 Resultado del montaje sobre "protoboard" de una parte del hardware para el Nodo Interior Zona no de Paso (Habitación)

El circuito para esta parte del hardware del segundo nodo (que utiliza el PIC18F248) sigue siendo igual de sencillo que el del nodo anterior, y los

diseños en los que se basa cada entrada y salida son los mismos para ambos nodos. Por ejemplo, para las comunicaciones (RS232 y grabación) utiliza las mismas conexiones.

Las únicas diferencias que existen entre ambos nodos son que éste a diferencia del nodo de categoría central, no lleva un reloj en tiempo real (DS1302), pero por el contrario el nodo habitación incorpora una entrada analógica más. El circuito de acondicionamiento de la entrada analógica para un sensor de luz se implementa de la misma forma que se comentó con anterioridad. La entrada analógica extra en este nodo está prevista para conectar un sensor de temperatura integrado, como por ejemplo el LM50 [17], el cual es capaz de proporcionar para temperaturas negativas una tensión positiva. En este caso no necesitará de ningún circuito de acondicionamiento. Simplemente se alimenta el sensor, se le indica la masa y su salida se conecta a un pin analógico del PIC para que el ADC muestre la señal de tensión.

También se incorporará una salida para un relé de dos contactos, que al igual que en el nodo anterior, tendrá un montaje exactamente igual al estar previsto poder conectar un relé con las mismas prestaciones. E igualmente, habrá salidas digitales disponibles para conectar pulsadores o sensores digitales todo/nada, solo que en esta ocasión varia la cantidad: tres de cada. Para todas las salidas el circuito de acondicionamiento es el mismo que se explicó y detalló en el apartado 5.3.2.

Ya en la segunda etapa, es decir, en el sistema final, se ha creado nuevamente una PCB para este nodo con todo lo explicado en este apartado. En esta ocasión, aquellos dispositivos que incluye la PCB pero que deberían ir conectados a sus entradas correspondientes y no incluidos en el hardware de nodo, son los sensores de luz (LDR) y de temperatura (LM50), así como los jumpers que imitan la respuesta de los sensores digitales.

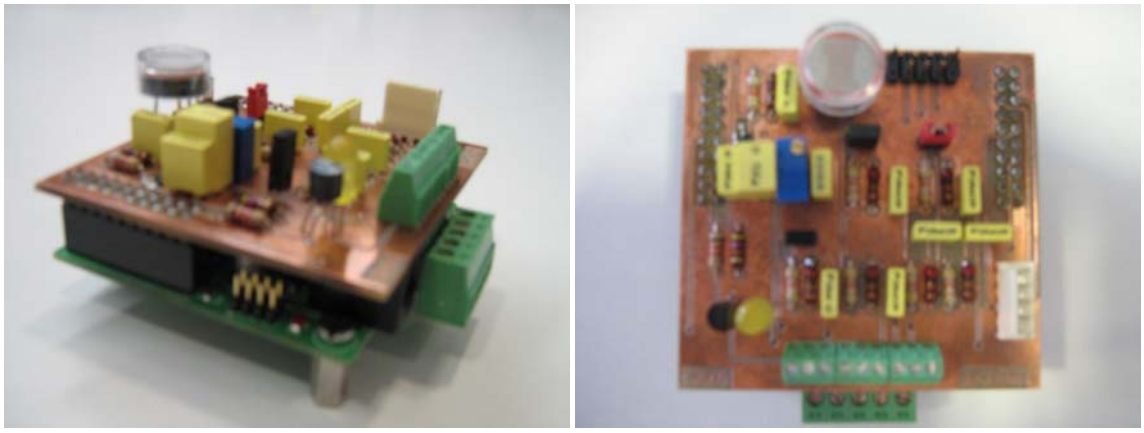

Fig. 5.9 Nodo completo: placa comercial y PCB

5.4.3 Software

En este apartado se va a describir a grandes rasgos el código en lenguaje C incluido en el módulo “Nodo” para un nodo interior, que no controla una zona de paso sino más bien una habitación, a partir de su diagrama de flujo:

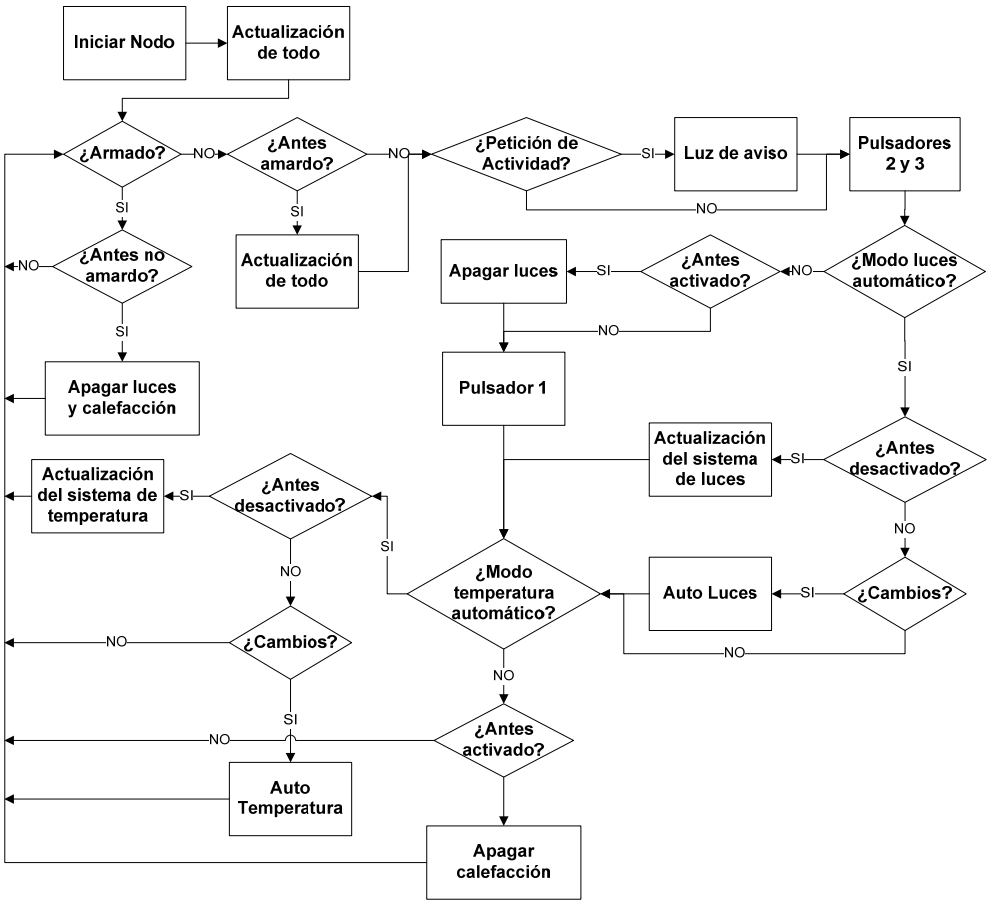

Fig. 5.10 Diagrama de flujo del Nodo Interior Zona no de Paso (Habitación)

Inicialmente el nodo llama a una función del módulo “Funciones” que configura varios parámetros propios del nodo como las interrupciones, configuración del CAN, el ADC, etc. Seguidamente actualiza todo el sistema (lee el estado de los sensores digitales y analógicos, y dependiendo de lo que haya obtenido enciende o apaga las luces y enciende o apaga la calefacción).

A partir de ahora se entra en un bucle infinito. Todas las instrucciones del código se realizan repetidamente, es decir, es un programa basado en polling. Este bucle lo primero que hace es comprobar si la casa estaba armada. Si es así, y provenía de no estarlo (es decir, que en la comprobación que hizo el bucle en la iteración anterior estuviera no armado), entonces apaga todas las luces y la calefacción. Si es el caso de que ya lleva un buen tiempo sin haber nadie en la vivienda, entonces va ejecutando continuamente el bucle sin realizar ninguna operación ya que en cada iteración del bucle vuelve al inicio.

En el caso de que la vivienda no esté armada, primero se asegura de que no venga de estarlo antes. Si es así realiza nuevamente una actualización de todo

(todo lo relacionado con las luces y la calefacción). Seguidamente se verifica si ha llegado una petición de actividad a evento especial. Si es así se activa el aviso o alarma (sonoro, lumínico, etc.). El paso siguiente es constatar los pulsadores 2 (activar/desactivar los modos automáticos de la iluminación o de la temperatura) y 3 (responder a una petición de actividad de evento especial en caso de que esté activada la alarma). En función del pulsador y del tipo de pulsación (corta o larga) se lleva a cabo las tareas oportunas. A continuación realiza dos tipos de automatismos: todo lo relacionado con el sistema de luz y todo lo relacionado con el sistema de la calefacción.

En primer lugar comprueba todo lo que tiene que ver con el sistema de las luces. Inicialmente lo que hace es mirar si está activado el modo automático de la iluminación, es decir, si la automatización de todo lo relacionado con las luces se ha de realizar. En caso de que no sea así, sino que esté activado el modo manual, si proviene de estar automático apaga todas las luces y comprueba constantemente el pulsador 1 (para controlar manualmente el encendido y apagado de las luces). Para el otro caso, es decir, que esté automático, si proviene de estar manual realiza una actualización de todo lo relacionado con la iluminación (sensor de luminosidad, sensor de presencia, luces, etc.). Sino, lo que hace es observar si se ha producido algún cambio en el estado de los sensores digitales o el sensor analógico detecta que el valor capturado está por debajo de cierto umbral (comprobación que se hace cada determinado tiempo, no cada instrucción) o ha cambiado el valor de ese umbral, entonces actualiza únicamente las luces (no el sistema relacionado con la iluminación) en función de los valores obtenidos.

En segundo lugar consulta todo lo que tiene que ver con el sistema de la calefacción. Este proceso es muy similar al que realiza el sistema de iluminación. Lo primero que hace es mirar si está activado el modo automático de la temperatura, es decir, si la automatización de todo lo relacionado con la calefacción se ha de realizar. Si no sea así no hace nada. Para el otro caso, es decir, que esté automático, si proviene de estar desactivado realiza una actualización de todo lo relacionado con el sistema de calefacción (sensor de temperatura, sensores de posición de ventanas y puertas, relé de la válvula de calefacción, etc.). Sino lo que hace es observar si se ha producido algún cambio en el estado de los sensores digitales o el sensor analógico detecta que el valor capturado está por debajo de cierto umbral (comprobación que se hace cada determinado tiempo, no cada instrucción) o ha cambiado el valor de ese umbral, entonces actualiza únicamente la válvula de la calefacción (no el sistema relacionado con la temperatura) en función de los valores obtenidos.

5.5 Nodo Interfaz

5.5.1 Funcionalidad

La programación para este nodo se ha efectuado con el software LabVIEW de National Instruments [18], una herramienta gráfica de test, control y diseño mediante la programación. Esta aplicación de alto nivel tiene el siguiente aspecto:

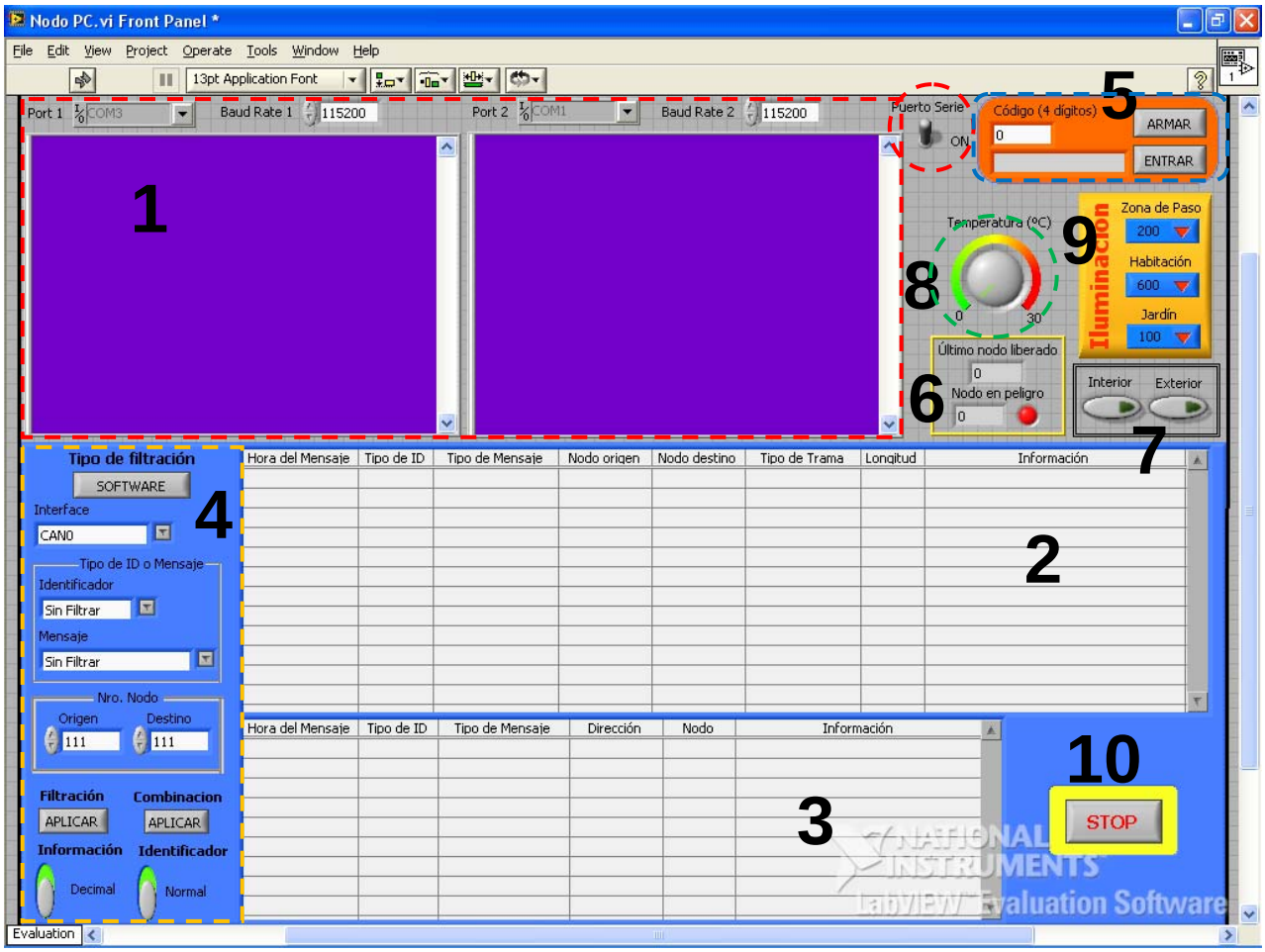

Fig. 5.11 Aplicación del Nodo Interfaz para LabVIEW

El primer bloque de la aplicación **(1)** viene a hacer las funciones que se conseguirían con el HyperTerminal, es decir, se trata de dos ventanas a través de las cuales se pueden ver todos los mensajes que circulan por dos puertos serie. Por lo tanto se pueden observar dos comunicaciones simultáneamente por puertos distintos. Se pueden seleccionar los puertos que se desean controlar y así como sus velocidades de transmisión respectivas. También existe la posibilidad de desactivarlas sino se quiere usar dicha comunicación. Se ha incluido este módulo en la aplicación para así no tener que utilizar un segundo programa con lo cual hacía más complejo tener muchas aplicaciones abiertas simultáneamente en el ordenador para comprobarlo todo.

El segundo bloque **(2)** es una tabla a partir de la cual se monitoriza todos los mensajes que circulan por el bus CAN, a excepción de los mensajes que van destinados concretamente para el nodo interfaz. En ese caso, esos mensajes aparecen en la tabla del tercer bloque **(3)**. En ambas tablas se desglosa la trama del mensaje CAN en las diferentes partes que forman el identificador (ver 6.2), así como tu tipo y la información que contiene. También se detalla la hora en la que se recibió.

En el cuarto bloque **(4)** se encuentran algunos parámetros para configurar las tablas de los bloques 3 y 4. La función principal es la de filtrado, es decir, filtrar

según varios aspectos los mensajes recibidos que posteriormente se muestran en las tablas. Existen dos tipos de filtración: una por hardware y otra por software. La filtración por hardware la lleva a cabo el programa configurando directamente los filtros y máscaras del controlador de CAN que lleva incorporado el módulo USB-CAN. Esta fue la implementación inicial que se incorporó a la aplicación. Pero este tipo de filtración introduce un inconveniente. El módulo USB-CAN utilizado tiene internamente una estructura de tipo Basic CAN, la cual, solo permite la implementación de una máscara y filtro por buffer o canal. Aunque se puede seleccionar hasta 14 canales, estos no se pueden usar simultáneamente. De esta forma no es posible filtrar las tablas por distintas partes del identificador si se tratan de mensajes de distintos tipos. Otro inconveniente, y quizás el más grave, es que la filtración por hardware impide que los mensajes que no pasan el filtrado no lleguen al código del programa. Por lo tanto si no se reciben mensajes que son enviados específicamente para el nodo interfaz, aquellas tareas que ha de realizar no se llevarán a cabo y el sistema no funcionará correctamente. Para solventar estos problemas se decidió introducir un segundo filtrado: por software. Este filtrado lo realiza la programación de la aplicación. Así todos los mensajes se reciben, son procesados y luego se filtran a la hora de mostrarlo en las tablas en función de lo configurado. En este tipo de filtrado siempre se siguen mostrando los mensajes destinados para el propio nodo interfaz, en la tabla del bloque 3, aunque el filtrado los descarte. Es decir, la filtración solo se aplica a la primera tabla. Además ahora también es posible filtrar por diferentes parámetros del identificador aunque se traten de diferentes tipos de mensajes. Otras configuraciones que se encuentra en este cuarto bloque es el formato en el que se muestran los datos en ambas tablas. Los identificadores se pueden observar por su tipo o en binario, y la información en decimal o en hexadecimal.

La simulación del sistema de la cerradura electrónica **(5)** también se realiza desde esta aplicación. Esto es simplemente un “apaño” para poder imitar su funcionamiento sin tener que implementar una en el sistema. Se entiende que en el sistema final esta cerradura sería un dispositivo más, que se encontraría en la entrada de la vivienda, el cual iría conectado al bus CAN de la instalación en caso de ser un dispositivo inteligente, o si no iría conectado directamente al nodo de esa zona. En cualquier caso no iría incorporado en el nodo interfaz. Su funcionamiento es bien sencillo. Tiene dos opciones: entrar a la vivienda, para la cual abre la cerradura y si la casa estaba armada la desarma, o bien armar la vivienda en caso de que no quede nadie cuando uno se marcha. Para ambas opciones es necesario introducir una contraseña y hay que esperar una respuesta la cual indicará si el proceso se ha realizado correctamente o si se ha producido algún error (error por contraseña o por presencia).

El sexto **(6)** bloque es meramente informativo. Indica el último nodo que fue liberado (porque se cayó de la red) y cuándo un nodo se halla en peligro como consecuencia de que no se contesta por parte del usuario. Incluye un avisador luminoso de treinta segundos que notifica cuando este último aviso se produce. Este aviso se podría substituir por cualquier otro tipo de sistema que comunicara de un peligro, como una alarma. Aún así como alternativa se ha decidió incluir tal prestación en el nodo interfaz. Para entender mejor estos dos tipos de informaciones ver el apartado 6.1.

También se incluyen dos pulsadores **(7)** para activar o desactivar los modos automáticos de iluminación de todos los nodos de la red. Un pulsador es para la iluminación de los nodos interiores y el otro para la de los nodos exteriores. Estos botones tienen preferencia sobre los de los nodos estándar, pero no son permanentes. Por ejemplo, si el nodo habitación tiene la iluminación automática desactivada a través del pulsador propio del que dispone, y se pulsa el botón del nodo interfaz para activar la iluminación automática interior, activará la automatización de todas las luces interiores de la vivienda, incluido la del nodo habitación. Pero si ahora se vuelve presionar el pulsador del nodo habitación para desactivar la iluminación automática, ésta y solo la de este nodo, se desactivará a pesar de la orden anterior que se dio desde el nodo interfaz.

La aplicación también consta de un regulador para la temperatura de la calefacción **(8)**. Cada vez que se modifica este valor y se confirma soltando el botón del ratón, se envía un mensaje con el nuevo umbral de temperatura a los nodos correspondientes. También se puede regular la luminosidad mediante el noveno bloque **(9).** En él se pueden indicar los umbrales (luxes) que han de tener en cuenta los diferentes tipos de nodos (exterior, interior de paso y interior resto) para encender o no el sistema de de luces. En cualquier momento es posible finalizar la aplicación con pulsar el botón de “Stop” **(10)**.

5.5.2 Software

No se pretende explicar en este apartado toda la programación que hay detrás de esta aplicación, ya que ésta es muy extensa y compleja. Simplemente se explicarán los conceptos utilizados para programar aquella parte clave y novedosa: el CAN. La interacción entre las diferentes capas para poder recibir o transmitir mensajes CAN con LabVIEW se detalla en el siguiente diagrama:

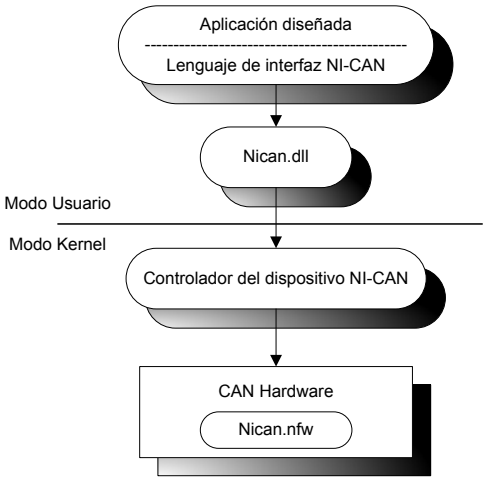

Fig. 5.12 Interacción entre las diferentes capas

El lenguaje de interfaz NI-CAN es una librería de funciones de 32 bits, nican.llb, utilizadas por las aplicaciones programadas en LabVIEW para acceder al software nican.dll. Este fichero es un archivo ejecutable que contiene las librerías de vínculos dinámicos. Estas librerías actúan de interfaz entre todas las funciones de CAN y el controlador del dispositivo NI-CAN. El controlador

facilita al usuario el acceso al dispositivo y permite a las distintas capas el acceso seguro al hardware. Nican.nfw es el software instalado en la memoria ROM del dispositivo NI-CAN. La librería de NI-CAN contiene todas las funciones necesarias para actuar sobre los controladores de CAN del dispositivo y así, poder escribir y leer en la red. Las funciones disponibles de las cuales se han dado uso para esta aplicación son las siguientes:

- **ncAction:** realiza las acciones de empezar, acabar o restaurar la comunicación de red de cualquier objeto CAN.
- **ncConfig:** antes de abrir un objeto, éste debe ser configurado. La configuración se centra en la configuración del controlador CAN.
- **ncGetAttribute:** para acceder a los datos de configuración del estado u otra información relacionada con un objeto, se ha de usar esta función.
- **ncGetTimer:** función que proporciona el tiempo de sellado (timestamp).
- **ncOpenObject:** abre el objeto especificado y devuelve una dirección que es utilizada por funciones posteriores.
- **ncRead:** proporciona varios parámetros como la longitud de los datos, datos, tiempo de sellado, identificador del mensaje y el tipo de trama.
- **ncSetAttribute:** configura el valor del atributo del objeto especificado.
- **ncWrite:** se debe configurar la longitud de los datos, los datos, el identificador del mensaje y el tipo de trama (remota o datos).

A continuación, a través del siguiente diagrama de flujo se describen los pasos a seguir para programar una aplicación (como la de este proyecto) utilizando las librerías de CAN del programa LabVIEW mencionadas con anterioridad:

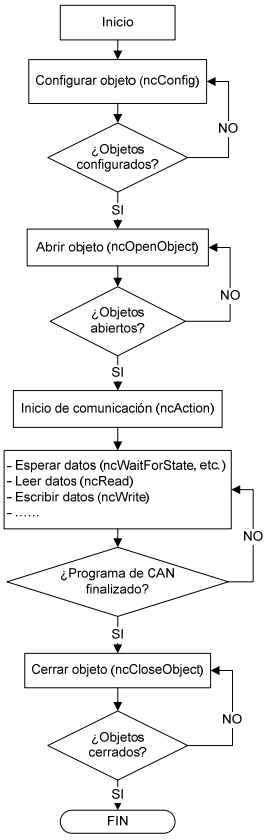

Fig. 5.13 Modelo de programación para aplicaciones CAN

CAPÍTULO 6. DEFINICIÓN DEL PROTOCOLO DE ALTO NIVEL

A continuación se explicará el nivel de aplicación, también conocido por el nombre de Protocolo de Alto Nivel -*High Level Protocol*, (HLP)- necesario para que el bus CAN realice una comunicación completa y funcione correctamente.

Existe una gran y diversa oferta de HLP's basados en el bus de comunicaciones CAN (en el apartado A.5 del anexo se ofrece información sobre los HLP's existentes). Cualquiera de ellos podría ser perfectamente aprovechable para el sistema que se propone en este proyecto. Entonces ¿por qué se ha decidido partir de cero y diseñar un HLP totalmente nuevo en vez de utilizar uno de los muchos que se nos ofrecen en el mercado? Sencillamente porque se desea un HLP que atienda las necesidades concretas del sistema diseñado. Además hay que tener en cuenta dos aspectos más. Por un lado no todos los HLP's que hay en el mercado son de libre distribución. Algunos son necesarios adquirirlos por licencia, y otros son protocolos propios y cerrados por diversos fabricantes y casas. Además, de los que son accesibles, no se proporciona ninguna librería de programación propia, y crear protocolos a partir de sus especificaciones es una tarea muy ardua, más si tenemos en cuenta que los HLP's de CAN que existen en el mercado suelen ser muy complejos ya que suelen abarcar muchos aspectos.

El CAN Kingdom, por ejemplo, fue uno de los HLP's que se barajó para implementar en la red CAN domótica. Su posible elección se sustentaba en dos razonamientos. En primer lugar porque es un HLP con mucho potencial y según se comenta desde hace tiempo se espera que sea en el futuro el protocolo por excelencia para la gran mayoría de aplicaciones basadas en el bus CAN. Aún así, esa incursión o expansión en el mercado tan esperada está por ver ya que se trata de un protocolo que ya tiene su tiempo (1991) a pesar de sus constantes revisiones, y aún con todo actualmente los protocolos predominantes son el CAN Open (en Europa) y el Device Net (en USA). En segundo lugar porque sus funcionalidades se ciñen bien a lo que requiere una red domótica, aunque quizás se llega a exceder para lo que se precisa en este proyecto. Prueba de ello es que el documento donde se define y especifica todo el protocolo tiene una extensión de 117 páginas [19]. Y eso que el CAN Kingdom es, de entre todos los HLP's de CAN, el menos complejo. Pero incluye muchas funcionalidades que para este prototipo de instalación que se propone no se llegarían a sacar partido. Así que de nuevo se llega al razonamiento del principio: se necesita un HLP que abarque las necesidades específicas del sistema a diseñar.

Seguidamente se da paso a describir todo lo relacionado con el HLP diseñado para este sistema. Los aspectos que se definirán serán los siguientes: especificación, componentes, funcionalidad y servicios que proporciona, como por ejemplo la gestión de la red; configuraciones básicas (máscara, filtros, velocidad de transmisión, etc.); y la distribución de los identificadores y de los datos de información.

6.1 Especificación, componentes y servicios

Este protocolo está diseñado para ser implementado en redes CAN basadas en el estándar ISO 11898. Por lo tanto, la capa física que impone este HLP se ha de ceñir a las características que se encuentran en la siguiente tabla:

Tabla 6.1 Características físicas para el HLP creado

CARACTERÍSTICAS	VALOR
Tasa máxima de transmisión	1 Mbps
Distancia a la máxima tasa de transmisión	40 metros
Distancia máxima de transmisión	1 Km. @ 50kbps
Impedancia característica de línea	120 Ω
Tiempo de retardo nominal de propagación	5 ns/m
Tipo de codificación de bit	NZR
Arbitración del bus	Bit-wise

Además este HLP también está pensado para trabajar en una arquitectura Full CAN, es decir, aquella implementación que está formada por un microcontrolador con el controlador de CAN incluido en el mismo encapsulado, más un transceiver externo.

El HLP podrá ser utilizado con cualquier microcontrolador de la familia PIC18FXX8. Estos PIC's contienen un módulo o controlador de CAN común para todos los de misma familia, el cual tiene las siguientes características:

Tabla 6.2 Especificaciones módulo CAN microcontroladores

MÓDULO CAN FAMILIA PIC18FXX8		
Especificación	**Valor**	**Unidad**
Tasa de transmisión de mensajes	125.000	Kbps
Longitud de identificador	11 o 29	Bits
Longitud de datos de mensaje	1 a 8	Bytes
Buffers de recepción	3	-
Buffers de transmisión	2	-
Máscaras para la recepción de mensajes	2	-
Filtros de aceptación	6	-

En cuanto al transceiver, la especificación CAN también requiere que estos cumplan con ciertas especificaciones eléctricas definidas por las ISO11898. El transceiver que se ha escogido para este HLP, el MCP2551 de Microchip [20], cumple con los requisitos citados, los cuales se describen en la siguiente tabla:

Tabla 6.3 Especificaciones CAN para los transceivers

PARÁMETRO	VALOR MÁXIMO	VALOR MÍNIMO	UNIDADES
Voltaje DC en CANH y CANL	-3	32	V
Voltaje transigente en CANH y CANL	-150	100	V
Voltaje Recesivo	2	3	V
Resistencia Interna diferencial	10	100	Ω
Voltaje dominante de salida (CANH)	2,75	4,5	V
Voltaje dominante de salida (CANL)	0,5	2,25	V

Ya se ha definido la especificación y los componentes que puede incorporar aquella red que desee integrar este HLP. Ahora solo queda por comentar los servicios y funcionalidades que incorpora.

Lo primero que hay que tener claro es el método acceso al bus que llevarán a cabo los nodos cuando se conecten por primera vez a la red. Todos los nodos de la red CAN que implemente este HLP deberán estar identificados por un número. A excepción del nodo central que por defecto siempre será el nodo número cero, el resto de nodos se les asignará un número cualesquiera comprendido entre el uno y el número máximo de nodos (112, ver apartado 6.3) que acepta a la red menos uno, es decir, 111 (ya que estamos teniendo en cuenta el cero). En realidad, los valores disponibles para asignar a un nodo estarán entre el uno y el número máximo de nodos permitidos menos dos (110), ya que el número de identificación máximo que se puede asignar, es decir, el 111 (número máximo de nodos menos uno) también está reservado para un nodo en concreto: el nodo interfaz.

Por lo tanto, lo primero que hará un nodo cuando se conecte a la red CAN será configurar la velocidad de transmisión para adaptarla a la del bus (ver apartado 6.3). Como ahora ya puede transmitir y recibir información a través del bus, el primer mensaje que transmitirá será una petición al nodo central para que le asigne un número de identificación de nodo. El nodo central, que es el encargado de asignar los números identificadores de nodo ya que él es el único que lleva constancia de cuantos nodos hay en la red y qué números de asignación están disponibles para ser asignados, responderá con un nuevo mensaje con el número de identificación que se le ha asignado al nodo que hizo la petición.

Seguidamente, el siguiente paso a realizar por parte de los nodos será el de configurar los filtros y máscaras, ya que hasta que no se conoce el número de identificación que le pertenece al nodo estos no se pueden configurar. Para más información sobre ello, revisad el apartado 6.3 de este mismo capítulo donde se dan más detalles sobre la configuración de los filtros y las máscaras. Puntualizar simplemente que por defecto, es decir, inicialmente hasta que el nodo recibe un mensaje con su número de identificación, las máscaras y filtros están configurados de forma que aceptan todo lo que se transmite por el bus. Pero a pesar que lo aceptan todo, no procesan ninguno de los datos que

reciben a menos que se trate de un mensaje que responde a su petición de asignación de número de identificación. A partir de entonces el nodo comienza a funcionar de manera normal, filtrando los mensajes para solo aceptar aquellos que vayan dirigidos a él y procesando los datos de información si la situación lo requiere.

Una vez explicado las configuraciones y rutinas que lleva a cabo el nodo al inicio así como su modo de acceso al bus, solo queda pendiente detallar los servicios de control o gestión de la red que se han diseñado.

En este HLP se ha incorporado una funcionalidad denominada “Control de Actividad” (CE). Básicamente se trata de una función que tiene como objetivo gestionar y controlar la red. Esta función solo la llevará a cabo un nodo, y como es de esperar, será el nodo central quien se encargue de ejecutarla. El proceso que lleva a cabo el CE es, simplemente, un chequeo de todos los nodos que hay en la red, a partir de la información que posee almacenada sobre ellos. Es decir, no se trata de una consulta directa nodo por nodo a través de mensajes CAN. Más bien consiste en consultar de forma interna la información que posee el nodo central almacenada sobre el estado del resto de nodos de la red. Esta rutina se realizará cada determinado tiempo. El nodo central conoce el estado del resto de nodos porque éstos le informan a través de un mensaje, denominado evento, cada vez que realizan una acción. La información que contienen los mensajes de tipo evento son la hora, minuto y segundo en la que se produjo la acción. De esta forma el nodo central siempre conocerá para cada nodo cuándo se efectuó la última acción.

A partir de este punto se le pueden implementar muchos servicios al CE. En este caso se han definido dos métodos de comprobación. Para poder diferenciar entre esas dos actividades de control, ha sido necesario diferenciar también entre los mensajes eventos:

- **Evento especial:** mensaje se genera cada vez que se produce una acción en el nodo, siempre y cuando, haya sido como consecuencia directa de la intervención de una persona. Ejemplo: cuando se activa un pulsador.
- **Evento normal:** este mensaje se genera cada vez que se produce una acción en el nodo cuando no está incluida en el marco comentado anteriormente. Es decir, aquellas acciones rutinarias que genera el propio nodo. Ejemplo: activar una salida para encender una bombilla.

Un primer chequeo consta en controlar si algún nodo se ha averiado. Si esto sucedería, el nodo caído dejaría de emitir mensajes, entre ellos los de evento normal, y por tanto el nodo central a través del CE lo detectaría pasado un cierto tiempo. El siguiente paso es dar de baja el nodo. Para ello libera su número de identificación para que se le pueda asignar a cualquier otro nodo. En ese momento, el nodo averiado ya no existe para el resto de nodos que conforman la red. Es evidente que no interesa tener un técnico de mantenimiento constantemente en el hogar, puesto que aumentaría el coste del sistema, así que este control resulta de gran ayuda: en el caso de que haya un

nodo problemático, el nodo central lo autoelimina permitiendo el correcto funcionamiento de la red en ausencia del nodo sin necesidad de una técnico.

El segundo chequeo se centra más en un control de la red desde el punto de vista del usuario. Este control se encarga nuevamente de comprobar si se ha superado un determinado tiempo desde la última vez que se registró un evento especial. En caso de que suceda así y además se detecte presencia en ese nodo, el nodo central envía un mensaje de alerta al nodo que se encuentra “en peligro” para que active un pulsador, es decir, para que dé señales de vida. Si pasado otro tiempo desde que se dio el aviso no se recibe respuesta, entonces el nodo central genera un mensaje de alerta, donde su cometido podría ser activar algún tipo de alarma, por ejemplo.

Tanto el control de eventos especiales como el de eventos normales los lleva a cabo el nodo central. Entonces, ¿quién controla al nodo central? Se autocontrola así mismo pero solo con los eventos especiales. El control de eventos normales queda totalmente descartado ya que no tiene ningún sentido que él mismo se realice un control de averías.

Todas estas funciones, servicios, controles del bus, automatismos al recibir e enviar los mensajes, etc., es decir, en sí el grueso del HLP, se ha programado en un archivo con el código fuente en C. Para se ha utilizado el PCWH Compiler de CCS para programarlo. En este archivo se incluyen todas las funciones necesarias para hacer funcionar un nodo en un bus CAN bajo las directrices de este protocolo. Por ejemplo, existe una función (que es un conjunto de varias funciones) que es la encargada de enviar los mensajes. Simplemente hay que indicarle qué tipo de información se quiere enviar y a qué dirección de nodo va destinado. Entonces cuando se invoca a esta función, ella misma genera la trama de datos y la envía por el bus. También existe la función que realiza el proceso inverso, es decir, la que recibe el mensaje. Su misión es desempaquetar la trama de datos, interpretar el mensaje (identificador) y procesarla en función del tipo información (datos) que contenga. Todo esto lo hace de forma automática para todo tipo de mensaje que reciba. En el código principal de cada nodo simplemente ejecuta esa función cuando salta alguna de las dos interrupciones, una por cada buffer de recepción, que se activan cada vez que se almacena un mensaje en el buffer después de haber sido aceptado por las máscaras y filtros.

6.2 Distribución de los identificadores y de los datos

Se ha elegido utilizar y por tanto transmitir con mensajes extendidos, lo que implica que el identificador sea de 29 bits (no de 11 bits). Por lo tanto la especificación en la que se basa este HLP es la del CAN 2.0-B. La decisión de utilizar un arbitraje extendido y no estándar se base en que, como se verá en el siguiente párrafo, en el identificador habrá constancia del número de nodo de donde proviene el mensaje así como su nodo de destino, y esto implica un número de bits considerable.

El prototipo de la instalación que se ha creado en este proyecto se basa en una red domótica muy sencilla. Al inicio de este proyecto, el número de nodos que como máximo se planteó implementar era de cinco (para identificarlos con tres bits sería suficiente) y se definió que la complejidad de parámetros a controlar y actuar no sería elevada. Esto podría permitir la opción de utilizar un identificador de 11 bits. Pero como este HLP estará pensado para ser implementado en una red domótica con nodos independientes, los cuales controlan una vivienda por zonas, esto implica que el número de nodos total o zonas a contralar pueda llegar a ser elevado o como mínimo seguro que mayor que los 5 que se contemplaban antes. Por tanto como el número se prevé que sea grande, se estipula que el máximo número de nodos permitidos por la instalación será el que restrinja el hardware de la red. Como esta cantidad es considerable (112, ver apartado 6.3), esto provoca que se tenga que aumentar el tamaño de identificador y ajustarlo al número de bits necesario. De ahí la justificación de por qué se opta por un identificador extendido para la especificación de este HLP.

Un esquemático de este campo, el campo de arbitraje, teniendo solo en cuenta las prioridades sería el siguiente:

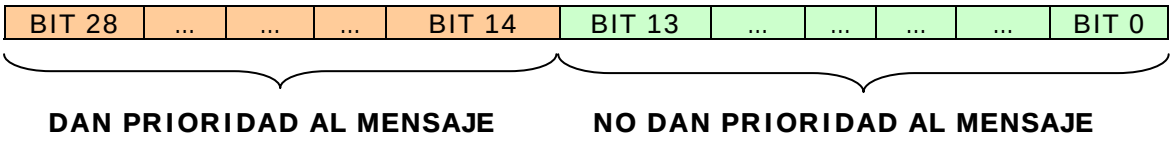

Fig. 6.1 Campo de identificación extendido de la trama de datos

Por un lado, los últimos 13, es decir, los 13 bits de menor peso servirán para indicar el nodo origen y el nodo destino de la trama. Es decir, en estos bits irían los números de identificación del nodo que envía el mensaje y del nodo que ha de recibir el mensaje en cuestión. De modo que estos 13 bits quedarían distribuidos de la siguiente manera:

- **Nodo origen:** bits del 13 al 7 del Identificador (ambos inclusive).
- **Nodo destino:** bits del 0 al 6 del Identificador (ambos inclusive).

Como se puede ver cada campo, tanto si es el de origen o el de destino, tiene un tamaño de 7 bits. Son 7 bits porque es el tamaño mínimo con el cual se puede representar en binario el número más grande posible de identificación de nodo que se puede asignar. Ese número corresponderá, como se comentó anteriormente, con el número máximo de nodos (en el apartado 6.3 de este mismo capítulo se explicará porqué ese número es 112) que acepta a la red menos uno (ya que también se tiene en cuenta el cero como número de nodo). Por lo tanto el número máximo a representar en binario sería el 1101111 (111 en decimal). Como se aprecia son necesarios 7 bits pero estos nunca deberán superar ese número, salvo en una excepción. Cuando un mensaje es de tipo broadcast, es decir, si su destino son todos los nodos, entonces los bits del 0 al 6 han de valer todos uno. Es la única ocasión en la que se podrá superar el 1101111, y solo se puede dar en los bits del identificador que indican el nodo destino. Para el nodo origen, lógicamente, no se completa esa excepción.

En cuanto a los primeros 15, es decir, los 15 bits de mayor peso, servirán para identificar los mensajes y datos que contienen los mismos. Estos 15 primeros bits serán los que realmente nos marcarán el orden de prioridad y acceso al bus CAN por parte de los mensajes (ver Fig. 6.1).

Aunque en realidad son todos los bits del identificador los que tiene en cuenta el CAN a la hora de marcar la prioridad de acceso al bus, lo que se quiere dar entender es que este HLP solo contempla los 15 primeros bits más significativos del identificador para indicar la prioridad de acceso. En el caso hipotético de que estos 15 bits sean idénticos para dos mensajes que se quieran transmitir a la vez (por lo tanto son identificadores distintos), entonces entre ambos no habrá un orden de prioridad marcada por el protocolo, pues los dos mensajes tienen el mismo nivel de relevancia con la única diferencia que los nodos origen siempre son distintos (también puede darse el caso que los nodos de destino sean diferentes, pero esto no afectaría a la prioridad). Es decir, en esta situación hipotética, la prioridad quedaría marcada en función del número de nodo de donde proviene cada mensaje. Sería una prioridad ficticia, aleatoria, pues el HLP no está determinando la prioridad ya que para dos mensajes idénticos en el tipo de información, le es indistinto quién transmita primero o no. El protocolo no establece ninguna prioridad predefinida para este tipo de situaciones. Por lo tanto es indiferente quién acceda primero al bus.

Ahora se desglosará qué información contienen esos bits que marcan prioridad (del bit 28 al 14). Los 15 bits más significativos del identificador estarán divididos en 2 subcampos:

- **Tipo de mensaje:** bits del 28 al 27 del Identificador (ambos inclusive).
- **Tipo de información:** bits del 26 al 14 del Identificador (ambos inclusive).

Se podrá tener hasta cuatro tipos de mensaje distintos, y para cada mensaje hasta 8192 posibles tipos de información (datos). Los cuatro posibles tipos de mensajes están específicamente definidos por el HLP:

- **Mensaje de orden:** este tipo de mensajes contendrán comandos y órdenes que indican realizar una acción, es decir, mensajes que se envían a otro nodo para que actúe.
- **Mensaje de petición:** mensaje que será enviado para solicitar un valor o estado de alguna variable, componente, actuador, etc. que se encuentre conectado en un nodo ajeno. También pueden solicitar peticiones de acciones a realizar que no pueden ser de orden directa.
- **Mensaje de respuesta:** este tipo de mensaje identifica a aquellos que se transmiten para responder a un mensaje de petición.
- **Mensaje de evento:** mensaje que será transmitido cuando cualquier nodo realice alguna actividad, como pueden ser lectura de datos o actuar sobre algún dispositivo, etc.

Tabla 6.4 Valores en binario de los distintos tipos de mensaje

Tipo de mensaje	ID [10:9]
Orden	00
Respuesta	01
Petición	10
Evento	11

En CAN la prioridad de mayor a menor se establece del identificador de valor más pequeño al más grande. Esto debido a que el nivel dominante es un cero lógico y el nivel recesivo corresponde a un uno lógico. Por lo tanto, según la Tabla 6.4, el tipo de mensaje con mayor prioridad será el de una orden. Normalmente en un sistema lo que prima es actuar, y eso es lo que hacen los mensajes de orden, dictar una acción al sistema. De ahí que se hayan definido como los mensajes con mayor prioridad. También es lógico que los mensajes de evento sean los de menor prioridad, pues por un lado es información extra o no relevante (utilizada por ejemplo para monitorizar el bus), y por otro lado aunque sirve para controles de gestión de la red, son menos importantes en cuanto a ejecución de tiempo que el resto de mensajes.

En cuanto a los mensajes de petición y respuesta, lo coherente a primera vista sería darle mayor prioridad a una petición que no a una respuesta. Pero en este caso se ha probado a hacerlo al revés. Los resultados de las pruebas realizadas con este orden de prioridad han resultado lo que se suponía y por tanto, lo que se esperaba. Si se la da prioridad a las peticiones, en caso de que se produzca una gran cantidad de este tipo de mensajes puede haber una saturación de peticiones sin respuesta. Si se da prioridad a las respuestas se agilizará este proceso y si no hay respuestas pendientes se dará paso a que se puedan hacer peticiones.

En resumen, todos los datos que componen el campo de arbitraje o identificación quedan estructurados de la siguiente manera:

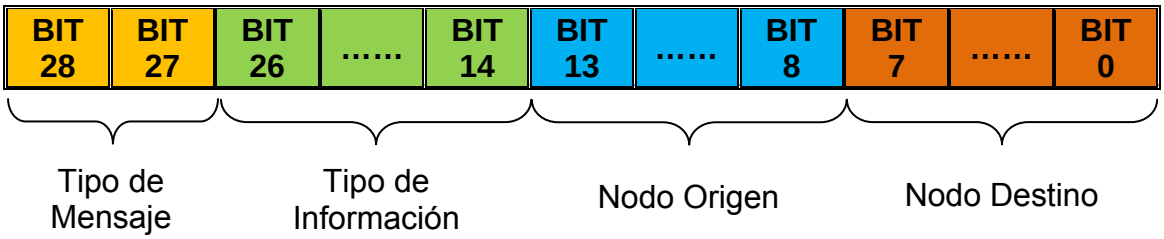

Fig. 6.2 El identificador de una trama de datos desglosado

Finalmente, en la siguiente tabla se va a mostrar todo el listado de mensajes o tramas que se han creado para este HLP. A cada mensaje se le asociará su tipo de mensaje y de información, el valor de los 15 primeros bits del identificador, lo que contiene el campo de datos, los posibles nodos de destinos y orígenes, y una breve descripción de su funcionalidad.

Tabla 6.5 Listado de todos los mensajes implementados por este HLP

Tipo de mensaje	Tipo de información	Identificador (los 15 MSB)	Datos	Nodo Origen	Nodo Destino	Función
Orden	Alarma	000000000000000	Nro. de nodo	Central	Interfaz	Se genera cuando un nodo de la red no responde a una petición de actividad a evento especial, es decir, la persona no da señales de vida.
Orden	Armar	000000000000001	Armar o desarmar	Central	Broadcast	Indica que en la vivienda no hay nadie y ha de ponerse en modo armado.
Orden	Modo interior	000000000000010	Activar o desactivar	Interfaz	Broadcast	Indica si los automatismos relacionados con el control de la luz de los nodos interiores han de estar activados o no.
Orden	Modo exterior	000000000000011	Activar o desactivar	Interfaz	Broadcast	Indica si los automatismos relacionados con el control de la luz de los nodos interiores han de estar activados o no.
Orden	Luxes interior resto	000000000000100	Luminosidad (Luxes)	Interfaz	Broadcast	Marca el umbral de luminosidad de los nodos interiores que no sean de paso.
Orden	Luxes interior paso	000000000000101	Luminosidad (Luxes)	Interfaz	Broadcast	Marca el umbral de luminosidad de los nodos interiores que son de paso.
Orden	Luxes exterior	000000000000110	Luminosidad (Luxes)	Interfaz	Broadcast	Marca el umbral de luminosidad de los nodos exteriores.

Orden	Temperatura	000000000000111	Temperatura (Grados)	Interfaz	Broadcast	Marca el umbral de temperatura de los nodos interiores que no sean de paso.
Respuesta	Poner hora	010000000000000	Fecha y hora	Interfaz	Central	Envía el día de la semana, número de día y mes, año, hora, minutos y segundos.
Respuesta	Asignar nodo	010000000000001	Nro. de nodo	Central	Cualquier	Asigna un número de nodo.
Respuesta	Actividad	010000000000010	Tipo de actividad respuesta	Cualquier	Central	Responde a una confirmación de actividad, indicando si es una respuesta de evento normal o especial.
Respuesta	Entrar	010000000000011	Resultado de la petición	Central	Interfaz	Indica si la solicitud para entrar ha sido correcta o incorrecta por de error de contraseña.
Respuesta	Armar	010000000000100	Resultado de la petición	Central	Interfaz	Indica si la solicitud para armar ha sido correcta. Si es incorrecta indica el tipo de error (por contraseña o presencia).
Respuesta	Fecha y hora	010000000000101	Fecha y hora	Central	Cualquier	Indica la fecha y hora actuales.
Petición	Poner hora	100000000000000	Código	Central	Interfaz	Solicita la fecha y hora para poder reiniciar el temporizador.
Petición	Asignar nodo	100000000000001	Código	Cualquier	Central	Solicita un número de nodo.
Petición	Actividad	100000000000010	Tipo de actividad preguntada	Central	Cualquier	Solicita confirmación de actividad porque ha detectado una anomalía de evento especial o evento normal.
Petición	Entrar	100000000000011	Contraseña	Interfaz	Central	Solicita poder entrar a la vivienda.
Petición	Armar	100000000000100	Contraseña	Interfaz	Central	Solicita armar la vivienda.
Petición	Fecha y hora	100000000000101	Código	Cualquier	Central	Solicita la fecha y hora actuales.

Evento	Pulsador	110000000000000	Hora, minuto y segundo	Cualquier	Central	Evento especial. Indica los datos de la hora en la que se produjo.
Evento	Sensor de presencia	110000000000001	Hora, minuto, segundo y estado del sensor	Cualquier	Central	Evento especial. Indica los datos de la hora en la que se produjo, y el estado del sensor.
Evento	Sensor de posición	110000000000010	Hora, minuto y segundo	Cualquier	Central	Evento especial. Indica los datos de la hora en la que se produjo. Hace referencia a aquellos sensores de posición que tienen contacto directo con la persona.
Evento	Sensor digital	110000000000011	Hora, minuto y segundo	Cualquier	Central	Evento normal. Indica los datos de la hora en la que se produjo.
Evento	Sensor analógico	110000000000100	Hora, minuto y segundo	Cualquier	Central	Evento normal. Indica los datos de la hora en la que se produjo.
Evento	Temporizador	110000000000101	Hora, minuto y segundo	Cualquier	Central	Evento normal. Indica los datos de la hora en la que se produjo.
Evento	Salida	110000000000110	Hora, minuto y segundo	Cualquier	Central	Evento normal. Indica los datos de la hora en la que se produjo.
Evento	Liberar	110000000000111	Nro. de nodo	Central	Interfaz	Se genera cuando un nodo de la red no responde a una petición de actividad a un evento normal, es decir, el nodo se ha averiado.

6.3 Configuraciones básicas

Las configuraciones básicas que se definirán sobre el HLP en este apartado serán las siguientes: velocidad de transmisión, número máximo de nodos permitidos y ajuste de filtros y máscaras de aceptación.

Velocidad de transmisión
Todos los nodos del bus CAN deben tener la misma tasa de bit nominal, es decir, han de transmitir el mismo número de bits por segundo. Por lo tanto todos los nodos deben estar configurados con la misma velocidad de transmisión. El valor de ésta depende de la frecuencia del oscilador del nodo y de sus parámetros de tiempo de bit. Como se parte de la base de que este HLP se aplicará a sistemas donde todos los nodos tendrán las mismas frecuencias para sus osciladores, por lo tanto no será necesario buscar diferentes valores para los parámetros de tiempos de bit para que coincidan las velocidades. Con una sola configuración para el tiempo de bit bastará.

Un bit está divido en 4 segmentos. Cada segmento está formado por un número entero de T_Q (el tiempo más pequeño de resolución discreto usado por CAN). Este tiempo (T_Q) depende de la frecuencia del oscilador del nodo y de una variable programable del controlador de CAN conocida como "Baud Rate Pre-escaler":

$$\mathbf{T_Q} = 2 * (\text{Baud Rate} + 1) * T_{osc} \qquad \mathbf{(6.1)}$$

Los segmentos que forman un bit son: el segmento de sincronización (Sync_Seg), el segmento de propagación (Prop_Seg) y los segmentos de fase (Phase_Seg1 y Phase_Seg2). Por lo tanto jugando con estos 4 parámetros de bit junto con el "Baud Rate" para poder configurar T_Q, se puede seleccionar el tiempo de bit deseado y de ahí extraer la velocidad ($1/T_{BIT}$) que se quiere adquirir para el bus de la red:

$$\mathbf{T_{BIT}} = T_Q * (\text{Sync\_Seg} + \text{Prop\_Seg} + \text{Phase\_Seg1} + \text{Phase\_Seg2}) \qquad \mathbf{(6.2)}$$

en donde los segmentos de bit se miden en T_Q

Por definición, el tiempo de bit tiene un mínimo de $8 \cdot T_Q$ y un máximo de $25 \cdot T_Q$. También por definición, el tiempo mínimo nominal del bit es de 1 μs, que corresponde a una tasa máxima de 1 Mbps. Por lo tanto la velocidad máxima del bus CAN según la especificación estándar en la cual se basa este HLP es de 1 Mbps (aunque algunos controladores pueden llegar a alcanzar velocidades mayores para aplicaciones especiales).

Las limitaciones de las velocidades vienen marcadas por el transceiver que se utilice y por la longitud total (extremo a extremo) de la red:

1) Según indica el fabricante del transceiver utilizado para este protocolo, MCP2551 de Microchip [20], la velocidad máxima permitida por el mismo es de 1Mbps. Así que por este aspecto el sistema no se verá restringido.

2) Por lo que hace referencia a la distancia, se ha de tener en cuenta los tiempos físicos de retardo de la red (tiempo de propagación de la señal por el bus, tiempo de retardo interno de los nodos, etc.). Este retardo se compensa con uno de los segmentos comentados anteriormente: el segmento de propagación. Debido a eso el valor de este segmento es muy importante y será uno los parámetros a tener en cuenta ya que marcará la longitud máxima para una determinada velocidad. Por lo tanto, se requiere que el frente de onda de la señal pueda propagarse entre los nodos más lejanos y luego retornar, y todo esto antes del punto de muestreo del bit.

En la siguiente tabla se hallan las recomendaciones de la CiA (CAN in Automation) en relación a la longitud del bus frente a la configuración del tiempo de bit:

Tabla 6.6 Relación velocidad respecto longitud máxima del bus

Velocidad (Kbps)	Tiempo de Bit (µs)	Longitud máxima (m)
1.000	1	30
800	1,25	50
500	2	100
250	7	250
125	8	500
50	20	1.000
20	50	2.500
10	100	5.000

Estos valores son totalmente orientativos ya que dependen de los factores comentados anteriormente.

A nivel práctico y teniendo en cuenta una pequeña aproximación de que el cableado medio de una red domótica en una vivienda no superará los 50 metros, se puede asegurar que la velocidad máxima que se puede elegir para el sistema será de 800 Kbps. Pero como en este caso la velocidad de transmisión no supone una gran restricción (en muchas ocasiones el bus estará libre ya que en domótica no se produce un uso elevado de tráfico) y para tener un poco de margen, se escoge como velocidad estándar y definitiva los 500 Kbps.

Ahora, para ajustar esta velocidad, se ha de buscar para los 4 parámetros mencionados en la ecuación 6.2 (Sync_Seg, Prop_Seg, Phase_Seg1, Phase_Seg2) una combinación de valores posibles que den como resultado la velocidad deseada.

Para calcular el valor necesario para esos parámetros, se ha utilizado un software de distribución libre, llamado "*Microchip CAN Bit Timing Calculator*". Simplemente con indicarle la frecuencia del oscilador y la velocidad que se

desea, él ya calcula el resto. Una vez introducidos los datos correspondientes ha proporcionado los siguientes resultados:

Setup Criteria	
Oscilator Frequency	20,000 MHz
Target CAN Bus Baud Rate	500,000 kbps

Selected Options	
BRP-1 (Baud Rate Prescaler)	0
Tq (Time Quanta)	100,000 ns
Number of Time Quanta	20
% Error of Target Baud Rate	0,0 %

Bit Timing Setup in Tq	
Propagation Delay	8
Phase Segment 1	8
Phase Segment 2	3
Syncronization Jump Width (SJW)	1

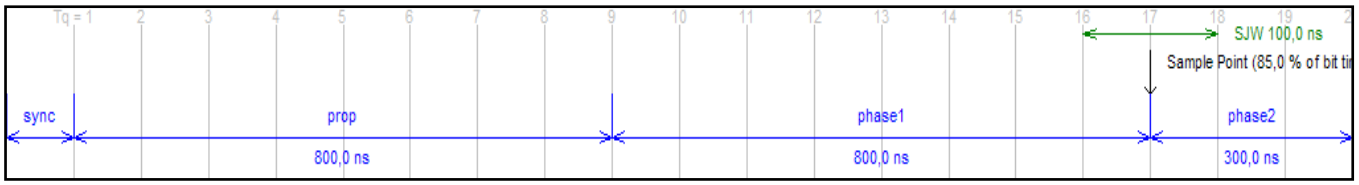

Fig. 6.3 Resultados obtenidos con el *Microchip CAN Bit Timing Calculator*

Número máximo de nodos permitidos

El número de nodos de la red no está limitado por la especificación ni por el estándar ISO. Ahora bien, uno puede equivocadamente pensar que teóricamente el número de nodos debería venir limitado por el número de identificadores posibles que permita el campo de identificación. Sin embargo, la realidad es otra. Es la capacidad de corriente disponible por el transceiver la que impone algunas restricciones, así que dependiendo del tipo del mismo, dependerá también el número de nodos posibles a conectar en la red CAN. Suele ser normal una red de entre 32 y 64 nodos, pero para este HLP la limitación que marca el fabricante del transceiver que se implementa (MCP2551 de la casa Microchip [20]) es de 112 nodos como máximo.

Ajuste de filtros y máscaras de aceptación

Como se describió en el apartado 6.1 (ver Tabla 6.2), el controlador CAN que incluye la familia de microcontroladores PIC18FXX8 dispone de 2 máscaras, 6 filtros y 2 buffers de recepción:

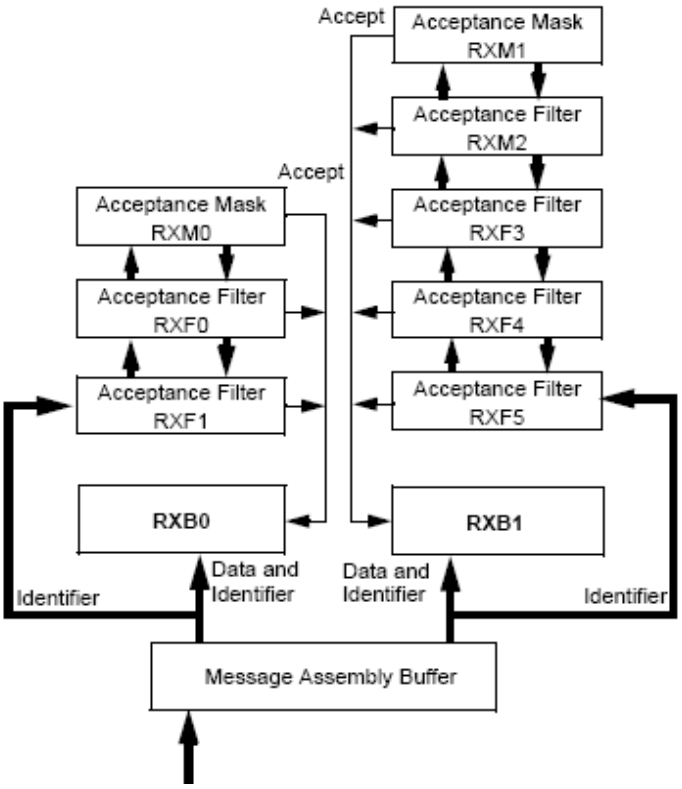

Fig. 6.4 Estructura de los filtros, máscaras y buffers del PIC18FXX8

Como se aprecia en la figura anterior, el controlador CAN que incorporan los microcontroladores de la familia PIC18FXX8 cuenta con dos buffers de recepción para evitar la pérdida de información por “overflow”, denominados RXB0 y RXB1. El primer buffer tiene asociado dos filtros y una máscara. El segundo cuenta con cuatro posibilidades de filtros e igualmente una máscara.

Para este HLP, se define que una vez que los nodos (exceptuando el nodo central pues ya conoce su número de identificación) han recibido el mensaje que les asigna un número de identificación de nodo, entonces es cuando proceden a configurar los filtros y máscaras. La máscara del buffer RXB0 valdrá 0x7F (1111111). De esta forma solo compara los 7 bits del identificador que hacen referencia al número del nodo destino. El primer filtro tendrá como valor el número del nodo para así solo recibir los mensajes que van destinados a él, y el segundo filtro valdrá 0x7F (1111111) para así aceptar todos los mensajes broadcast. Para el buffer RXB1 se configurará su máscara con el valor 0x7FFFFF (los 29 bits a 1) y los cuatro filtros se dejan como estaban por defecto, a cero. Por lo tanto, por esta máscara y filtros no se aceptará nunca nada, ya que el identificador es imposible que adopte un valor de todo los bits a cero (la máscara compara todos los bits y los filtros valen todos cero). Entonces, en el RXB1 nunca se almacenará un mensaje a excepción de que el RXB0 esté ocupado. En ese caso, si la máscara y filtros del RXB0 aceptan un mensaje pero éste está lleno, envían el mensaje al RXB1. Esta es una opción que permite configurar este controlador de CAN y para este HLP se ha activado.

CAPÍTULO 7. SISTEMA COMPLETO

7.1 Descripción del prototipo final

El prototipo para la instalación domótica completa tanto en la etapa de pruebas como en la etapa final ha quedado de la siguiente manera:

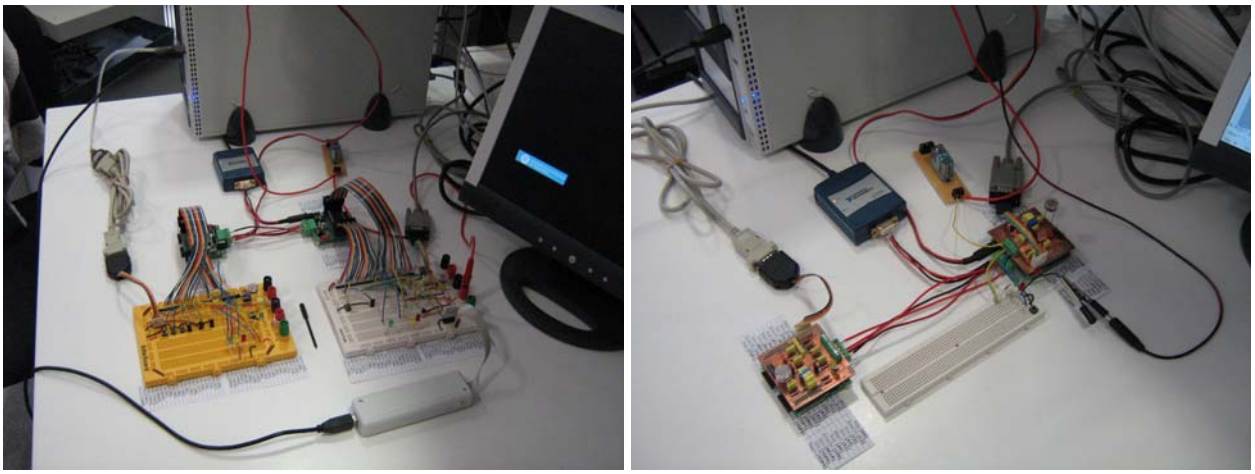

Fig. 7.1 Montaje experimental y montaje definitivo del prototipo completo

Ambos montajes están formados por los mismos componentes: dos nodos estándar y uno interfaz con las funcionalidades, hardware y software descritos en el capítulo anterior. Los tres nodos están interconectados a través del bus CAN formando una red domótica básica. Este prototipo de instalación domótica también incorpora un PC como parte del nodo interfaz con el cual se ejecuta la aplicación de alto nivel diseñada. Los puertos series de cada nodo estándar también están conectados a dicho ordenador, así como la programación realizada con el ICD. Por último, aunque no se ve en las imágenes de la Fig. 7.1, se ha necesitado de una fuente de alimentación para alimentar toda la instalación.

7.2 Verificación funcional

Como es de esperar, se han realizado las comprobaciones oportunas para verificar que realmente el diseño y el funcionamiento del prototipo de la instalación domótica creada es el esperado. Estas comprobaciones se pueden dividir en tres facetas: las realizadas en el hardware de los nodos estándar, las llevadas a cabo en el software del nodo interfaz, y las hechas a todo el sistema completo. A continuación se detallarán las características de cada una de estas tres pruebas.

Para el nodo estándar las pruebas se han ido realizando en distintas fases. En cada fase se ha probado un módulo básico o circuito de acondicionamiento para cada una de las entradas y salidas que forman el circuito del hardware

completo para los nodos estándar. Para probar el funcionamiento de cada uno de estas agrupaciones se ha programado las funciones que requieren para operar (aquellas que se encuentran en el módulo “Funciones” del software para nodos estándar). De esta forma no sólo se daba el visto bueno a la programación, sino que era una verificación doble: hardware y software. El orden de las fases llevadas a cabo ha sido el siguiente:

- Pulsador simple (única pulsación).
- Pulsador doble (pulsación larga y pulsación corta).
- Entradas para sensores digitales.
- Activar y desactivar salidas digitales normales (luces).
- Activar y desactivar salidas digitales para accionar un relé de dos contactos.
- Utilización de un reloj en tiempo real (DS1302).
- Configurar el ADC del microcontrolador, utilizarlo, etc. Para ello se introdujeron señales de tensión conocidas para hacer las pruebas pertinentes.
- Leer entradas analógicas para un sensor de luz (LDR).
- Extraer algunos valores típicos al sensor de luz con el luxómetro.
- Leer entradas analógicas para un sensor de temperatura (LM50).

Todas estas pruebas, como ya se ha comentado, son tanto de software como de hardware, ya que implica verificar el funcionamiento de los circuitos eléctricos que implementa cada módulo.

Para el nodo interfaz, también se ha hecho una batería de pruebas diferenciando las pruebas por los módulos que se explicaban en el apartado 5.5.1. Para ello, se conectaron a través del bus CAN el nodo interfaz y un nodo estándar compuesto solo de la placa comercial, sin más hardware añadido, y con un software programado para recibir los mensajes CAN y para enviar sistemáticamente cada determinado tiempo otros mensajes CAN también conocidos. De esta forma se pudieron realizar todas las combinaciones posibles tanto de recepción como de envío de mensajes para la aplicación en LabVIEW. Así que bloque a bloque se probaron sus servicios y se hicieron los retoques que fueron necesarios.

La última etapa o faceta de comprobaciones se realizaron sobre la instalación completa. De esta forma se verificó el buen funcionamiento del HLP. Para ello se simularon todas las combinaciones y situaciones posibles a las cuales se podían enfrentar cada uno de los nodos que forman la red domótica.

7.3 Valoración económica

En este apartado se estudiarán los aspectos económicos relacionas con este proyecto. Se adjunta un presupuesto sobre el coste de realización y materiales de la instalación propuesta. Para en ello, en la Tabla 7.1 se puede observar un inventario de material y componentes que forman la instalación domótica basada en un bus CAN para dos nodos concretos.

Tabla 7.1 Coste económico del proyecto

PRESUPUESTO DEL PROTOTIPO PARA INSTALACIÓN DOMÓTICA				
Componente	**Modelo**	**Unidades**	**Precio por unidad (€)**	**Precio (€)**
Placa comercial	SBC28PC	2	19,95	39,9
NODO CENTRAL				
Microcontrolador	PIC18F258	1	11,63	11,63
Sensor de luz	Norps-12	1	2,45	2,45
Sensor de presencia	LAS5201	1	30	30
Sensor de posición	MCT 302	1	31,16	31,16
Temporizador	DS1302	1	3,73	3,73
Reloj de cristal	32,768 KHz	1	0,144	0,144
Transistor NPN	BC183	1	0,048	0,048
Cerradura electrónica con teclado	CD-203	1	200	200
Otros (condensadores, sockets, resistencias, conectores, leds, etc.)	-	-	-	15
NODO HABITACIÓN				
Microcontrolador	PIC18F248	1	7,54	7,54
Sensor de luz	Norps-12	1	2,45	2,45
Sensor de temperatura	LM50	1	0,71	0,71
Sensor de presencia	LAS5201	1	30	30
Sensor de posición	MCT 302	2	31,16	62,32
Transistor NPN	BC183	1	0,048	0,048
Otros (condensadores, sockets, resistencias, conectores, leds, etc.)	-	-	-	15
NODO INTERFAZ				
Módulo CAN-USB	NI USB-8473	1	309	309
OTROS				
Cableado (bus, dispositivos, etc.)	-	-	-	100
Creación de la PCB, la soldadura y las horas del soldador	-	2	45	90
MANO DE OBRA				
Horas de trabajo	-	840	20	16.800
TOTAL				17.751,13

La suma total asciende a 17.751,13€, un precio para nada desorbitado, y eso que se han tenido en cuenta la gran mayoría de los dispositivos que no forman parte del nodo pero que sí son necesarios y no se hallan en cualquier vivienda. Además gran parte del precio se debe al trabajo realizado (30 horas semanales durante 7 meses) para hacer el diseño y elaborarlo, probarlo, etc. De la misma forma, siendo totalmente estrictos y desde el punto de vista del fabricante, habría que tener en cuenta el coste de todo el software utilizado para realizar la instalación: el compilador para los nodos estándar (CCS) y programador para el nodo interfaz (LabVIEW).

CAPÍTULO 8. CONCLUSIONES

8.1 Valoración de resultados

Como resultado de este trabajo se ha construido un prototipo de red domótica con tres nodos representativos: dos nodos estándar y un nodo interfaz. El nodo interfaz, que permite monitorizar y controlar la red, consiste en un PC, con una aplicación hecha a medida, que se integra en la red como un nodo más.

Los nodos estándar se han diseñado alrededor de un microcontrolador de 28 pines de la familia PIC18FXX8, el cual incorpora un controlador de CAN. Estos nodos ofrecen las siguientes prestaciones:

- Entradas analógicas: entre 1 y 2.
- Entradas digitales: entre 3 y 6.
- Salidas digitales: entre 3 y 4.
- Una salida para activar un relé de dos contactos.
- Reloj de tiempo real.
- Puerto serie (RS232).
- Bus Can, que permite la creación de redes de hasta 112 nodos.

La comunicación entre nodos se ha realizado mediante bus CAN. Se ha creado un protocolo de alto nivel sobre CAN que permite comunicar los distintos elementos de la red y añadir nuevos nodos de forma fácil. A la vista de los resultados, podemos decir que el bus CAN es un bus adecuado para ser aplicado en redes domóticas.

Valorando los resultados obtenidos puede afirmarse que se han alcanzado los dos objetivos planteados inicialmente que consistían en desarrollar nodos domóticos capaces de funcionar de forma independiente o en red y estudiar la viabilidad de utilizar el bus CAN en el ámbito de la domótica.

8.2 Futuras líneas de trabajo

Cabe destacar por encima de todo que este proyecto abarca un trabajo muy amplio, es decir, la temática (o temáticas, según como se mire) que engloba proporciona mucho margen de mejoras, ampliaciones, incorporaciones, etc. Por tanto, se trata de un proyecto muy escalable, lo cual, implica que las futuras líneas de trabajo son numerosas.

Desde ese punto de vista, este proyecto no se deja totalmente cerrado. Se ha realizado una base, un molde básico a partir del cual se puede continuar dándole forma. Por tanto no sería nada descabellado que a partir de este proyecto se pudiera realizar como mínimo otro en forma de continuación, ya que las ampliaciones pueden ser muchas y diversas. Algunas ideas para añadir a este trabajo podrían ser las siguientes:

1. **Ampliar la instalación domótica.** Se pueden crear más nodos e incorporarlos a la red para probar nuevas situaciones, con más tráfico de datos, etc. Esta mejora, es decir, el crear nuevos nodos estándar, no supondría de una gran complejidad ya que la base está hecha: el protocolo está creado y las funciones genéricas de los nodos también lo están. Simplemente hay que programar el módulo “Nodo” como se explicó en el apartado 5.2.
2. **Incorporar más servicios a los nodos.** Introducir más servicios como control de persianas y cortinas, modos de escenario (cine en casa), añadir más dispositivos (un dimmer para las luces), etc.
3. **Diseñar un nodo interfaz con prestaciones únicas para diseñador (técnico) o usuario.** En el caso de que fuera una interfaz para usuario, sería necesario implementar otro tipo de dispositivo como interfaz que no sea un PC, como por ejemplo, una consola táctil portátil al estilo de una TabletPC o UMPC.

8.3 Valoración personal

Personalmente he quedado muy satisfecho con el resultado final obtenido y no solo porque se hayan cumplido los objetivos planteados inicialmente. A lo largo del trabajo me he sentido muy cómodo, gracias en parte a que la temática del proyecto era muy atrayente, especialmente por el hecho de que combinara una parte de software (programación en lenguaje C para microcontroladores y programación visual en LabVIEW) con una parte de hardware. Y no solo ha contribuido esto a que el proyecto se haya hecho ameno, sino también las condiciones de trabajo que se han dado como la colaboración que he recibido.

El nivel de interés que me ha provocado el trabajado diario ha ido de menos a más, y por tanto, el tiempo dedicado o que me ha absorbido también ha crecido proporcionalmente en función de ese interés. Inicialmente había un cierto interés y expectativas, pues sino no hubiera aceptado este proyecto, pero éste ha ido poco a poco en aumento a medida que indagaba en el trabajo, hasta cotas que no me esperaba llegar. Prueba de ese nivel tan alto de implicación e interés es que me he hubiera gustado poder implementar más cosas para el prototipo de la instalación domótica creado porque realmente, este proyecto ha sido muy atrayente.

Personalmente algo que me ha sorprendido es que a pesar de que este proyecto es la culminación de unos estudios universitarios, he aprendido muchísimo, y de muchos temas. Y aunque me ha sido necesario acudir a apuntes, anotaciones o documentos que conservaba de asignaturas de la carrera, esto solo se ha producido en contadísimas ocasiones. A pesar de haberme enfrentado a conceptos técnicos totalmente nuevos para mí, con mis propios conocimientos he sido capaz de abordarlos, asimilarlos y poderlos llevar a cabo. Esto ha sido para mí una grata sorpresa pues nunca pensé que pudiera llegar a tener esa capacidad de autoaprendizaje. Supongo que la base que me ha aportado esta carrera parece que empieza a dar sus frutos.

REFERENCIAS*

[1] http://es.youtube.com/watch?v=tn4cDmPNYig&eurl=http://www.domoticaviva.com/demo.htm

[2] http://www.casadomo.com/

[3] Sistema de Gestión de Edificios, Argumentos del Sistema; *"Bus de Instalación EIB"*, Fondo Formación, EI3.

[4] CiA, "CAN" www.can-cia.org

[5] http://www.canbus.galeon.com/electronica/canbus.htm

[6] http://www.modtronix.com/product_info.php?products_id=110

[7] http://www.elettronica.ingre.unimore.it/materiale_didattico/93_PIC18Fxx8.pdf

[8] http://www.ccsinfo.com/content.php?page=overview

[9] CAN BUS Exercise Book, *"Embedded C Language for the PIC MCU, DEVELOPMENT KIT"*, CCS, Wisconsin (2006)

[10] Reese, R. B.; *"Microprocessors: From Assembly Language to C Using the PIC18Fxx2"*, Da Vinci, Canadá (2005)

[11] Palacios Municio, E., Remiro Domínguez, F., y J. López Pérez, L.; *"Microcontrolador PIC16F84. Desarrollo de proyectos"*, Rama, Paracuellos de Jarama (2005)

[12] Couëdic, M.; *"Circuitos integrados para tiristores y triacs"*, Marcombo, Barcelona (1999)

[13] Fighiera, B. y Knoerr, R.; *"30 Montajes para iniciarse en electrónica"*, Marcombo, Barcelona (1999)

[14] http://es.farnell.com/jsp/Semiconductors/Transistors/FAIRCHILD+SEMICONDUCTOR/BC183/displayProduct.jsp?sku=1017653

[15] http://es.farnell.com/jsp/Semiconductors/Clocks,+Timing+and+Frequency+Control/MAXIM-DALLAS/DS1302+/displayProduct.jsp?sku=1188041

[16] http://es.farnell.com/jsp/Optoelectronics/Light+Detectors+&+Sensors/SILONEX/NORPS-12/displayProduct.jsp?sku=327700

[17] http://www.national.com/mpf/LM/LM50.html

[18] http://www.ni.com/labview/

[19] http://www.cankingdom.org/cank.htm

[20] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010405

[21] http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1069

[22] http://www.mikroe.com/en/tools/picflashwicd/

[23] http://sine.ni.com/nips/cds/view/p/lang/es/nid/203382

* Todas las referencias relacionadas con páginas webs (links) están comprobadas a fecha de 21 de Enero del 2008.

Anexo A. BUS CAN

A.1. Orígenes y arquitectura de capas

Antiguamente todos los sistemas utilizados para intercambiar datos se basaban en la transmisión de datos convencional (se caracteriza por el hecho de que a cada señal le está asignada una conducción individual). Pero con el paso del tiempo, debido al incremento en el intercambio de datos entre los componentes electrónicos, éste ya no puede ser realizado razonablemente con interfaces convencionales.

La complejidad de los mazos de cables tan solo puede dominarse actualmente con gran esfuerzo y cada vez aumentan más las exigencias planteadas en el intercambio de datos entre las unidades de control. De ahí la aparición de la transmisión de datos en serie (CAN). Los problemas en el intercambio de datos a través de interfaces convencionales, se resolvieron mediante la aplicación de sistemas bus, como por ejemplo, el CAN.

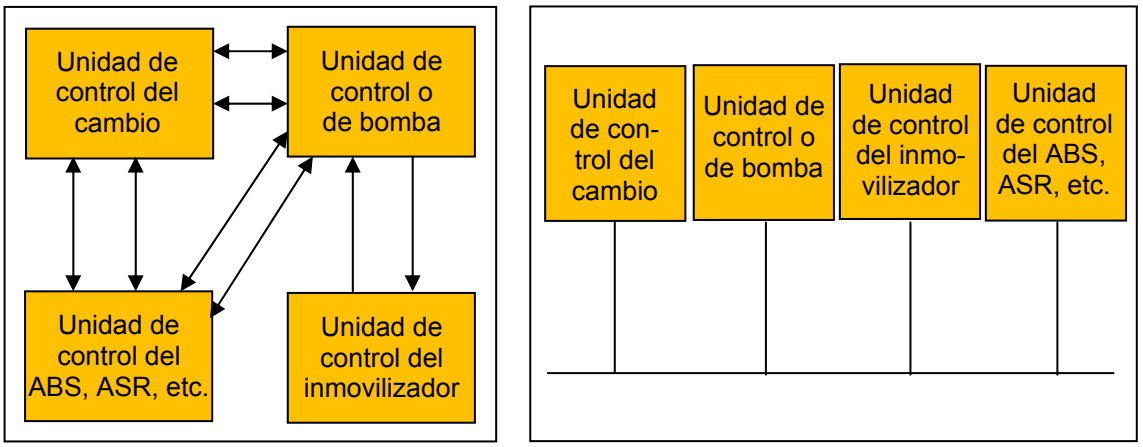

Fig. A.1 Transmisión convencional vs. Transmisión en serie

Los inicios del CAN se remontan a la década de los 80, concretamente en el año 1983, cuando Robert Bosch y su grupo de investigadores comenzaron un proyecto interno sobre el desarrollo de un protocolo para el sector del automóvil. Se pretendía establecer una comunicación capaz de unir varias centralitas electrónicas de un automóvil, con un alto nivel de fiabilidad y control de errores. Pero fue más tarde, en el año 1986, cuando presentaron oficialmente el resultado obtenido bajo el nombre de Controller Area Network (CAN).

Es interesante destacar que originalmente (1983-1986) el bus CAN fue desarrollado para aplicaciones en la industria de la automoción. Pero cuando entre 1993/94 CAN fue aceptado como un estándar mundial por la International Standardization Organization (ISO), se definió como un protocolo de bus de datos de comunicación serial para aplicaciones en tiempo real, orientado a trabajar con dispositivos inteligentes, sensores y actuadores, dentro de

diversos sistemas distribuidos. Es decir, CAN pasó de ser una idea exclusiva para el mundo del automóvil a ser adoptado también para aplicaciones industriales y de control, ya sea como protocolo en aplicaciones de bus de campo o de bus sensor/actuador. ¿Cómo fue esto posible? Como consecuencia de las características especiales que poseía, así como su robustez y excelente relación calidad/precio. Prueba de ello es que actualmente CAN ha encontrado un lugar en diversas áreas del mercado. A pesar de eso, al CAN aún le queda mucho trayecto por recorrer para extenderse por otros sectores ya que los últimos datos reflejan que un gran porcentaje de las aplicaciones CAN aún están enfocadas hacia la industria automovilística. En el apartado A.6 de este mismo anexo se puede consultar más información sobre las aplicaciones en CAN.

Toda organización de redes se basa en distintos niveles o capas. Cada uno de estos niveles ejecuta unas funciones específicas. La comunicación entre capas de igual nivel se lleva a cabo por medio de los protocolos propios de ese nivel. Se conoce con el nombre de arquitectura de red al conjunto de esas capas o protocolos.

La arquitectura del protocolo CAN, como era de esperar, también está basada en capas. Ya se comentó con anterioridad que en 1993, CAN pasó a ser un estándar de ISO, con la referencia ISO11898. Dicho estándar se basa en la Especificación de CAN 2.0 y relaciona CAN con el modelo de referencia OSI. Como muestra la Fig. A.2, el protocolo CAN es una especificación de bajo nivel ya que implementa las dos capas inferiores del modelo OSI: la capa física y la capa de enlace de datos. Es decir, solo implementa dos de las siete capas definidas por el modelo OSI. Por lo tanto, ¿qué sucede con el resto de capas? Las capas restantes, al no estar definidas por la especificación, en principio deben ser implementadas por el ingeniero de acuerdo a la aplicación que diseñe.

La capa de enlace se divide en dos, el enlace de control lógico y el control de acceso medio. El enlace de control lógico maneja las notificaciones, filtrado de mensajes, y reestablecimiento de las funciones de administración. El control de acceso medio se encarga del encapsulamiento y desancapsulamiento de datos, detección y control de errores, proceso de bit stuffing y transmisión serial. En resumen, se encarga de todo lo relacionado con el protocolo de comunicación.

La señalización física es es la parte de la capa física definida por CAN que se encarga de la codificación y descodificación de bit, el tiempo de bit y la sincronización, es decir, se encarga de las características eléctricas relacionadas con la transmisión de bits entre nodos diferentes. El medio físico, los cables y los conectores no están definidos por la especificación, así que deben ser escogidos por el ingeniero una vez conocidos los requerimientos de señalización física necesarios.

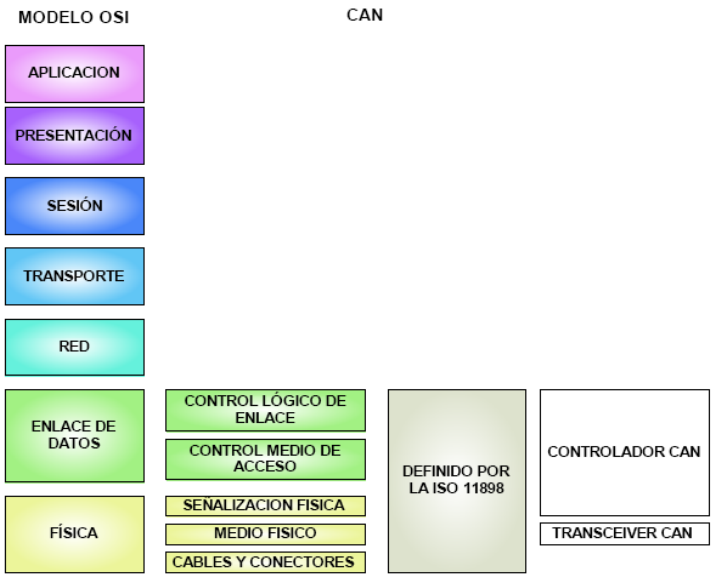

Fig. A.2 Relación entre el Modelo OSI y CAN

De todas las características eléctricas que define la capa física, es importante conocer los denominados niveles lógicos del bus. Al tratarse de un bus diferencial, éste está formado por dos señales y la diferencia que existe entre estas dos señales determinan el estado del bus. Por tanto, el CAN dispone de dos niveles lógicos. Normalmente en los sitemas digitales de dos niveles se conocen estos dos estados por nivel alto y nivel bajo, sin embargo en este caso se denominan nivel dominante y nivel recesivo:

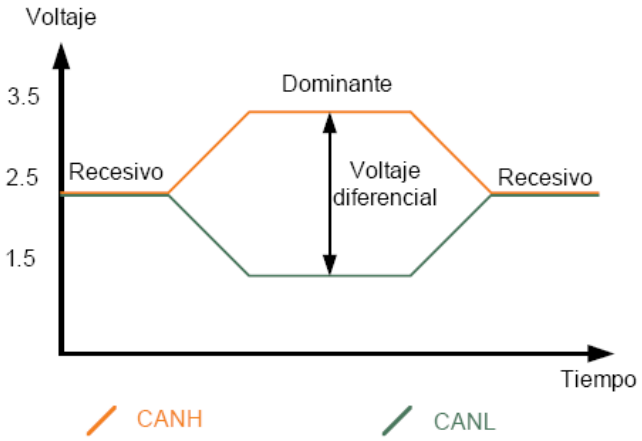

Fig. A.3 Niveles presentes en el bus CAN

Dominante. La tensión diferencial entre los pines de comunicación (CAN_H - CAN_L) ha de ser del orden de 2 V. Para conseguir esto es necesario que CAN_H tenga 3,5 V y CAN_L sea de 1,5 V (nominales). De hecho, si el voltaje de la línea CAN_H es al menos 0,9 V mayor que CAN_L, entonces ya se detectará la condición de bit dominante.

Recesivo. La tensión diferencial entre los pines de comunicación (CAN_H - CAN_L) ha de ser del orden de 0 V. Para conseguir esto es necesario que

CAN_H y CAN_L tengan 2,5 V (nominales). Aunque realmente el bus detectará una condición de recesivo si el voltaje de la línea CAN_H no es más alto que el voltaje de la línea CAN_L más 0,5 V.

A.2. El encapsulado de la trama de datos

CAN utiliza mensajes de estructura predefinida, es decir, tramas para la gestión de la comunicación. En concreto, el protocolo CAN define cuatro tipos de tramas diferentes: trama de datos, trama remota, trama de errores y trama de sobrecarga.

Tanto la trama de errores como la trama de sobrecarga son mensajes destinados al control de errores y se generan automáticamente por los nodos cuando detectan cualquiera de los muchos protocolos de error definidos por CAN. En cuanto a la trama remota, es básicamente una trama de datos pero sin datos, y se utiliza para solicitar la transmisión de una trama de datos. Su función es simplemente para tener también un control pero en este caso sobre la red. Para más información sobre estos tres tipos de tramas consultad el apartado siguiente de este mismo anexo (A.3).

Pero realmente el mensaje que es interesante conocer al detalle es la trama de datos, la trama más importante de las cuatro. En un bus CAN los nodos transmiten la información espontáneamente con tramas de datos, bien sea por un proceso cíclico o activado ante eventos en el nodo. Así que su función principal es poner información en el bus (siempre es un "Broadcast" a todos los demás nodos de la red).

Para la transmisión en el bus, se crea un marco de datos (Data Frame), cuya longitud abarca como máximo 130 bits (formato estándar) o 150 bits (formato extendido). De esta forma queda asegurado que el tiempo de espera hasta la siguiente transmisión, posiblemente muy urgente, se mantenga siempre corto. La trama de datos consta de 7 campos de bits diferentes:

Fig. A.4 Trama de datos CAN

Start of frame (SOF): El campo de inicio de la trama está compuesto de un único bit. Se trata de un bit de sintonización dominante que indica el inicio de la transmisión. El flanco descendente de este bit es utilizado por los nodos receptores para sincronizarse entre sí.

Campo de arbitraje: Se trata del campo de identificación que permite reconocer a los nodos la prioridad del mensaje. Cuanto más bajo sea el valor del identificador más alta es la prioridad, y por lo tanto determina el orden en que van a ser introducidos los mensajes en la línea. Los bits de identificador se

transmiten en orden de más significativo a menos significativo. Según sea el formato de la trama estándar (CAN 2.0-A) o extendido (CAN 2.0-B), este campo tendrá 11 ó 29 bits respectivamente. Dentro del mismo campo de arbitraje, hay un bit denominado RTR que indica si el mensaje contiene datos (0) o si se trata de una trama remota sin datos (1).

Campo de control: Este campo informa sobre las características del campo de datos. El bit IDE es el identificador del formato, es decir, si el bit IDE se transmite en nivel dominante, se trata del formato estándar y si es en nivel recesivo, es un formato extendido. Los cuatro bits que componen el campo DLC indican en binario el número de bytes de datos en el mensaje (de 0 a 8). En cuanto al bit r0, es un espacio reservado para futuras ampliaciones del bus.

Campo de datos: En este campo aparece la información del mensaje con los datos que el nodo correspondiente introduce en el bus CAN. Puede contener entre 0 y 8 bytes (de 0 a 64 bits en saltos de 8).

Cyclic Redundancy Check (CRC): Este campo tiene una longitud de 16 bits y es utilizado para averiguar si se ha recibido correctamente lo que se ha transmitido. Se produce un error de CRC cuando el resultado calculado no es el mismo que la secuencia CRC recibida. Para ello se utilizan los 15 primeros bits, mientras el último siempre es un bit recesivo que delimita el campo CRC.

Campo de reconocimiento (ACK): El campo ACK es un campo de dos bits que indica si el mensaje ha sido recibido correctamente. El nodo transmisor pone este bit como recesivo y cualquier nodo que reciba el mensaje lo pone como dominante para indicar que el mensaje ha sido recibido.

End of frame (EOF): El campo fin de trama indica el final del mensaje con una cadena de 7 bits recesivos.

Campo de intermisión (IFS): Consta de un mínimo de 3 bits recesivos que indican la separación entre dos tramas CAN seguidas.

Los campos realmente importantes a tener en cuenta serían el del arbitraje (Identificador más RTR), el del control (IDE más DLC) y el de datos. Los campos CRC, ACK e IFS se detallarán más a fondo en los siguientes apartados de este mismo anexo.

A.3. Tramas de gestión y de control de errores

Como ya se ha comentado en el apartado anterior, los datos son transmitidos por medio de mensajes llamados tramas. Por otro lado y como ya se pudo ver en el mismo apartado citado anteriormente, aparte de las tramas de datos existen otras tres tramas más, las cuales están relacionadas con la gestión y control del bus. La trama de errores y la trama de sobrecarga por un lado (sirven para manipulación de errores y son generadas por los mismos nodos cuando detectan cualquiera de los muchos protocolos de error definidos por CAN); y la trama remota por otro lado (es como una trama de datos pero sin el

campo datos). A continuación se va ver más información acerca de estas tramas.

➢ **Trama remota:** Básicamente sirve para saber qué nodo está conectado al bus. No se mandan datos ni nada por el estilo, solo el identificador y se espera la respuesta del nodo. Primero se envía el identificador y luego el bit RTR (se encuentra dentro del campo del Identificador o Arbitraje) que es recesivo. Si al mismo tiempo se manda una trama de datos, ésta gana pues el bit RTR del identificador es dominante. El formato de esta trama es el siguiente:

SOF	Arbitraje	Control	CRC	ACK	EOF

Un detalle muy importante a tener en cuenta, es que una trama remota también puede ser utilizada por un nodo para solicitar la transmisión de una trama de datos con la información asociada a un identificador dado. El nodo que disponga de la información definida por el identificador la transmitirá en una trama de datos.

Pero cuidado, porque en un bus CAN los nodos transmiten la información espontáneamente con tramas de datos, bien sea por un proceso cíclico o activado ante eventos en el nodo. Pero por el contrario, la trama remota sólo se debe utilizar para detección de presencia de nodos o para puesta al día de información en un nodo recién incorporado a la red. Estos mensajes también pueden entrar en colisión en el bus. Entonces, como sucede siempre, el del identificador de mayor prioridad sobrevivirá y los demás serán retransmitidos lo antes posible.

➢ **Trama de error:** Son generadas por cualquier nodo que detecte un error definido. Es una trama de dos campos, por un lado el Flag de error y por otro el delimitador. Éste último consiste en 8 bits recesivos consecutivos que le permite a los nodos iniciar limpiamente la transmisión. El formato de esta trama es el siguiente:

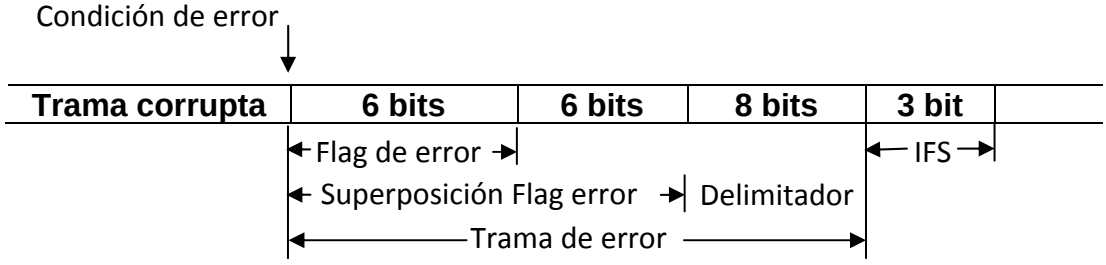

Fig. A.5 Campos que componen la trama error

La condición o indicador de error será distinto según el estado o flag de error del nodo que detecte el error. Existen dos estados o flags de error de nodo: “Activo” y “Pasivo”. El Activo consiste en seis bits dominantes consecutivos y el Pasivo consiste en seis bits recesivos consecutivos, a no ser que estén sobrescritos por otros bits dominantes de otros nodos.

Si un nodo en estado de error "Activo" detecta un error en el bus interrumpe la comunicación del mensaje en proceso generando un "Indicador o Condición de

error activo" que consiste, como ya se ha comentado antes, en una secuencia de 6 bits dominantes sucesivos y continuos. Esta secuencia rompe la regla de relleno de bits[1] y provocará que el resto de los nodos detecten error y por lo tanto transmitan también 6 bits. Por lo tanto, la longitud de esta superposición de flags oscila entre 6 y 12 bits (dependiendo del número de nodos que detectan el error en el bus). Luego dicho transmisor coloca el delimitador de error mandando los 8 bits recesivos, espera el IFS[2] y reinicia la transmisión del mensaje.

Si un nodo en estado de error "Pasivo" detecta un error, el nodo transmite un "Indicador o Condición de error pasivo" seguido, de nuevo, por el campo delimitador de error. El indicador de error de tipo pasivo consiste en 6 bits recesivos seguidos y, por tanto, la trama de error para un nodo pasivo es una secuencia de 14 bits recesivos (6 del flag más 8 del delimitador). De aquí se deduce que la transmisión de una trama de error de tipo pasivo no afectará a ningún nodo en la red, excepto cuando el error es detectado por el propio nodo que está transmitiendo. En ese caso los demás nodos detectarán una violación de las reglas de relleno y transmitirán a su vez tramas de error. En este caso nuevamente la trama tendrá la misma estructura que la del error activo.

Trama de sobrecarga: Una trama de sobrecarga tiene el mismo formato que una trama de error activo. Sin embargo, la trama de sobrecarga sólo puede generarse durante el espacio entre tramas, es decir, en el IFS. De esta forma se diferencia de una trama de error, que sólo puede ser transmitida durante la transmisión de un mensaje. La trama de sobrecarga consta de dos campos, el Indicador o Flag de Sobrecarga, y el delimitador. El indicador o flag de sobrecarga consta de 6 bits dominantes que pueden ser seguidos por los generados por otros nodos, dando lugar a un máximo de 12 bits dominantes. El delimitador es de 8 bits recesivos.

¿Cuándo se genera una trama de sobrecarga? Pueden darse tres situaciones que lo provoquen:

- Si debido a condiciones internas un nodo es incapaz de recibir un nuevo mensaje en ese instante. En este caso, la trama de sobrecarga se envía justo cuando se esperaba el primer bit del espacio de intermisión (IFS). De esta forma se consigue retrasar el inicio de transmisión del nuevo mensaje.

- Cuando un mensaje es validado por los receptores y el último bit del EOF es recibido como dominante (cuando debe ser recesivo), por tanto, este bit dominante no es considerado como error, pero la existencia de este bit ilegal puede deberse a que el receptor ha perdido la sincronización y eso requiere una reacción por parte de la red.

[1] La regla del relleno de bits consiste en que cada cinco bits de igual valor se introduce uno de valor inverso.

[2] Se conoce como IFS (Inter Frame Spacing) al espacio entre tramas. Es el espacio existente entre una trama (de cualquier tipo) y la siguiente trama de datos o remota. Esta secuencia o separación predefinida también se conoce con el nombre de intermisión o ínter-trama. La longitud mínima debe ser de tres bits recesivos. En un nodo en estado de error activo (nodo transmisor) esto es suficiente para reiniciar la transmisión, en cambio en el de error pasivo (nodo receptor o cualquier otro) tiene que esperar 8 bits más para iniciar una nueva transmisión o el bus permanecerá en reposo. De esta forma se asegura una ventaja en inicio de transmisión a los nodos en estado activo frente a los nodos en estado pasivo.

- Si durante la intermisión uno de los dos primeros bits es dominante, es decir, si la intermisión ha sido violada. En este caso, la trama de sobrecarga se enviará un bit después de recibir el bit dominante.

Cada nodo puede mandar hasta un máximo de 2 tramas de sobrecarga.

A.4. Detección y señalización de errores

Una vez vistas las tramas, se va proceder a ver como las utiliza el protocolo CAN para detectar los errores y señalizarlos. En el protocolo CAN se ha realizado un importante sistema de manejo de errores. Este sistema permite detectar errores en los mensajes que se transmiten para así en el caso que sea necesario se retransmitan los mensajes erróneos. Si un controlador del bus detecta un error, enviará un flag de error para avisar del mismo y así destruir el tráfico del bus. Los otros nodos posteriormente, detectarán un error debido a la violación de la regla del bit stuffing (relleno) en el flag de error y enviarán otros flags de error.

El protocolo CAN define cinco tipos de detección de errores, dos de estos modos son al nivel de bit y el resto al nivel de trama:

- **Error de bit:** Cuando un nodo está transmitiendo, éste monitoriza el nivel del bus y si el bit que lee en el bus no coincide con el que ha transmitido, es que se ha producido un error, así que se señala un “Error de bit”. En el siguiente tiempo de bit el nodo transmisor envía una trama de error, entonces, la trama fallida será reenviada después del espacio de intermisión.

- **Error de Stuff:** Como ya se pudo ver, este error detecta si dentro del área codificada por el método del bit stuffing, existen seis bits consecutivos del mismo nivel. El nodo que detecte esto, enviará una trama de error justo en el tiempo de bit siguiente al de detectar el sexto bit del mismo nivel. La trama fallida será retransmitida después del espacio de intermisión.

- **Error CRC:** Tanto este método para detectar errores como los dos siguientes (ACK y Forma) se realizan mediante el chequeo de la misma trama de datos a través de los campos correspondientes. El CRC es un campo de las tramas que contiene un código de redundancia cíclica el cual comprueba si hubo errores en la recepción del mensaje. Con el CRC podemos detectar errores aleatorios en hasta 5 bits o una secuencia seguida de 15 bits corruptos. Si el CRC calcula en el nodo receptor no coincide con el CRC enviado (el que contiene la trama), entonces el receptor descarta el mensaje recibido y envía una trama de error, pidiendo una retransmisión de la trama.

- **Error ACK:** Otro campo que se encuentra en las tramas. En esta ocasión se trata de dos bits que indican si el mensaje fue recibido satisfactoriamente. El nodo transmisor manda el ACK en recesivo esperando a que el nodo receptor lo sobrescriba en dominante, de lo contrario se le considera una trama corrupta y lo retransmitirá. Así, si el transmisor detecta un ACK positivo, es decir, un bit

dominante durante el campo ACK, sabrá que al menos un nodo ha recibido correctamente el mensaje.

➢ **Error de forma:** Algunas partes de los mensajes en CAN tienen formas fijas. Estas zonas son: el delimitador de CRC, el delimitador de ACK, EOF y el espacio de intermisión (IFS). Los bits de estas zonas deben ser recesivos. Si algún controlador de CAN detecta que alguno de estos bits es dominante, genera una Trama de Error porque se ha producido un "Error de forma", esta trama se enviará en el tiempo de bit siguiente al bit erróneo.

Por tanto, como se puede ver la detección de un error por parte de un nodo se hace público al resto de nodos mediante la transmisión de tramas de error. Entonces la transmisión de ese mensaje fallido es abortada y será retransmitida tan pronto como el arbitraje de la red se lo permita al nodo correspondiente.

Por otra banda, cada nodo contiene dos contadores de error: el contador de Transmisiones de Error (TEC) y el contador de Recepciones de Error (REC). Hay varias reglas que gobiernan el modo en que estos contadores deben incrementarse y/o decrementarse. En esencia, si un nodo transmite una Trama de Error su contador de Transmisiones de Error se incrementa en 8 y los nodos que reciben una Trama de Error incrementan en 1 el valor de su contador de Recepción de Error. Si un nodo transmite una trama correctamente, el TEC decrementa y si un nodo recibe una trama correctamente, su REC decrementa.

Inicialmente los nodos están en estado de Error Activo. Cuando uno de sus dos contadores llega a superar el valor 127, el nodo pasa a estar en estado de Error Pasivo.

A.5. Tipos de Protocolos de Alto Nivel (HLP's)

El protocolo CAN especifica la forma de transmitirse la información y en qué condiciones se realiza según el estándar ISO 11898 (capas de Enlace y Física). Pero eso no es suficiente para la comunicación dentro de una red, porque deja abiertos aspectos tan importantes como el comportamiento de la puesta a punto de los nodos, la asignación de identificadores, cómo estructurar y traducir el contenido de datos de la trama, mecanismos para dotar de funcionalidad específica a los nodos, etc. En consecuencia, existen diversas soluciones de software, o más conocidos como protocolos de alto nivel (HLP), que cubren las capas que faltan por definir, como las de transporte, presentación y aplicación.

Para entenderlo mejor se podría hacer un símil. Las funciones proporcionadas por CAN se asemejan a las letras latinas en la comunicación humana. Éstas son la base para escribir una lengua, pero no es suficiente para comenzar una comunicación completa y eficiente. Para especificar una lengua se necesita, entre otras cosas, una gramática para poder construir oraciones. De la misma forma, el CAN requiere definir su propia "lengua", o dicho en términos técnicos, definir su propio HLP para poder iniciar una comunicación. Hoy en día, existe

un gran abanico de protocolos de alto nivel estandarizados, la mayoría de los cuales son muy específicos. Algunos de los HLP's más populares que ofrece el mercado son: CAN Open, Smart Distributed System, Device Net, OSEK, J1939 o CAN Kingdom. A continuación se detallaran muy brevemente los rasgos principales de algunos de ellos.

CAN Open
CAN Open fue originalmente diseñado para la industria de sistemas de control, pero las redes CAN Open también son usadas para aplicaciones de campo como transporte público, equipos médicos, etc. Las especificaciones cubren el nivel de aplicación, el perfil de la comunicación, el armazón de los aparatos programables de los nodos y recomendaciones de cables y conectores. Es uno de los HLP más utilizados en las aplicaciones basadas en el bus CAN.

Device Net
El protocolo de comunicación Device Net es abierto y aceptado como un estándar de industria en todo el mundo. Fue diseñado para comunicar dispositivos de control inteligentes así como sensores de campo, estaciones de pulsadores, arrancadores, interfaces de operador simples, y variadores de control de velocidad. Es un sistema de CAN barato. Un aspecto curioso es que este protocolo también puede integrarse en CAN Kingdom.

Es rápido y llega a soportar 3 velocidades: 125Kbps, 250Kbps y 500Kbps. Además soporta 64 nodos activos. Device Net básicamente distingue entre mensajes de procesos de mayor prioridad (I/O Mensajes) y mensajes de menor prioridad (Mensajes Explícitos). Además utiliza una técnica orientada a objetos, en el que la información está estructurada en diferentes objetos.

SDS (Smart Distributed System)
Smart Distributed System, es un sistema de bus para sensores y actuadores inteligentes. Es un modo eficiente de conectar pequeños aparatos a un controlador máster. Todos los nodos SDS tienen asignados un número del rango (de 0 a 125) llamado dirección del aparato y debe ser único para todos los nodos del sistema. Este número es usado como base para seleccionar un conjunto de identificadores de CAN que pueden ser usados por este módulo.

CAN Kingdom
Este protocolo de alto nivel está diseñado principalmente para realizar sistemas de control, móviles hidráulicos, etc. Algunos denominadores comunes de estos sistemas son el alto desarrollo en tiempo real, alta demanda de seguridad. El protocolo CAN Kingdom soporta entre otras cosas, cambios dinámicos de identificadores, identificadores estándar y extendidos, un reloj global y un hardware que limita el número de módulos del sistema.

En este protocolo, el nodo por defecto en la red tiene toda la información necesaria para inicializarse en el sistema. Las especificaciones de los sistemas serán los métodos de acceso al bus, la gestión de la red, las listas de mensajes y los formatos de los datos. El diseñador de sistemas es responsable del software implementado en el nodo y él puede decidir en qué condiciones los

nodos serán aceptados por el sistema. En el fondo el diseñador de CAN Kingdom tiene la máxima libertad para crear su propio sistema.

A.6. Aplicaciones y diagnóstico

Como ya se comentó anteriormente en el apartado A.1, la mayor parte de las aplicaciones que posee el CAN están concentradas en la industria automovilística donde realiza el control del motor, de la mecánica del automóvil así como los sistemas de entretenimiento. Esto es debido en parte a que el sistema CAN fue especialmente diseñado en sus orígenes con la idea de aplicarlo en el mundo de los automóviles para la transmisión de datos entre los sistemas electrónicos de control y regulación, como por ejemplo:

- Control del cambio.
- Control electrónico del motor o de la bomba de inyección.
- Sistema antibloqueo (ABS).
- Sistema de tracción antideslizante (ASR).
- Control de estabilidad (ESP).
- Regulación del momento de arrastre del motor (MSR).
- Inmovilizador.
- Ordenador de a bordo, etc.

Relacionado con la industria automovilística, el CAN se extiende al transporte público y otras máquinas móviles como aviones, helicópteros, trenes, barcos y los controles de tráfico y sistemas de información conductor/pasajero.

Pero en general el CAN se aplica en cualquier sistema de control industrial: sistemas de control de plantas y maquinaria, redes entre máquinas, maquinaria agrícola, instrumental médico, sistemas de supervisión, etc. y en la automatización de edificios: control de ascensores, aire acondicionado, sistemas de calefacción y refrigeración, control de iluminación, etc. En resumen, en cualquier sistema que precise control en tiempo real distribuido y con escaso flujo de datos. Por lo tanto, esto augura un gran futuro para el CAN. Incluso actualmente ya se puede considerar que el CAN a alcanzado un nivel extraordinario de madurez e implantación ya que los fabricantes y procesadores digitales de señal están incorporando controladores CAN de forma bastante generalizada (se habla de cientos de millones de nodos).

Volviendo al tema de las aplicaciones CAN en vehículos motorizados, cabe decir que existen tres campos de aplicación esenciales para estos sistemas:

- Acoplamiento de unidades de control.
- Electrónica de la carrocería y de confort.
- Comunicación móvil.

De estos tres campos se va a documentar un poco sobre el primero: Acoplamiento de unidades de control.

Se acoplan entre sí sistemas electrónicos como el control del motor o de bomba de inyección, sistema antibloqueo ABS, sistema de tracción antideslizante ASR o regulación de la dinámica de marcha ESP, control electrónico de cambio, etc. Las unidades de control están aquí unidas como estaciones con igualdad de derechos, mediante una estructura de bus lineal. Esta estructura presenta la ventaja de que en caso de fallar una estación, el sistema bus continúa estando plenamente a disposición de las demás estaciones. En comparación con otras disposiciones lógicas (estructuras anulares o estructuras en estrella) se reduce así esencialmente la probabilidad de un fallo total. En el caso de estructuras anulares o en estrella, el fallo de una estación o de la unidad central, conduce a un fallo total.

Las velocidades de transmisión típicas están entre aprox. 125 Kbps y 1 Mbps (ejemplo: la unidad de control del motor y la unidad de control de bomba en la regulación electrónica diesel comunican entre sí a 500 Kbps). Las velocidades de transmisión deben ser tan altas para poder garantizar el comportamiento de tiempo real requerido. En las siguientes figuras vemos dos diferentes ejemplos:

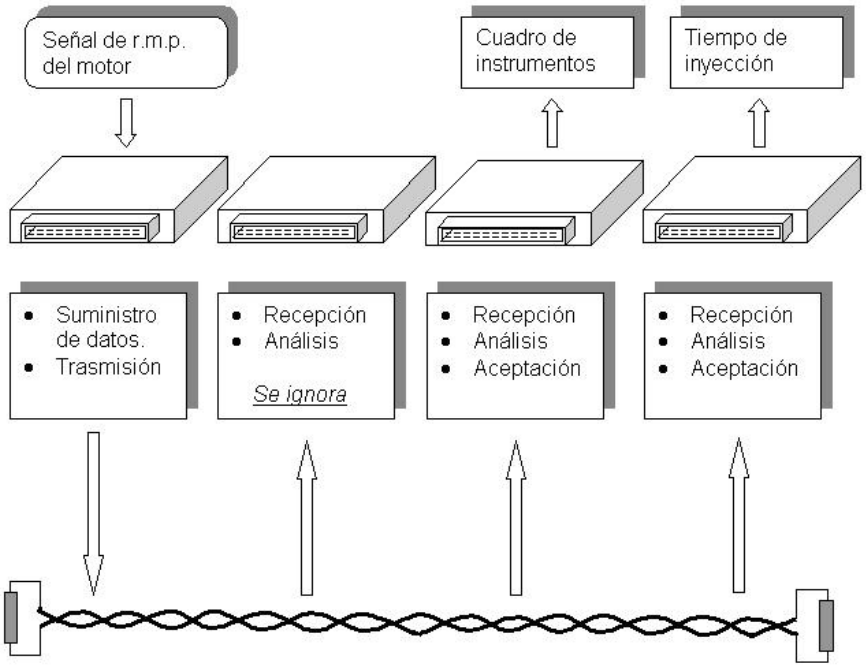

Fig. A.6 Ejemplo 1 sobre un sistema para automóvil basado en CAN

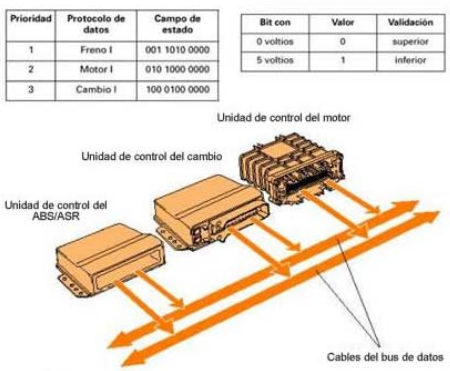

Fig. A.7 Ejemplo 2 sobre un sistema para automóvil basado en CAN

Diagnóstico

Tal como se ha explicado en este mismo anexo, el protocolo CAN dispone de una serie de mecanismos de control para el reconocimiento de anomalías. Si una estación registra una anomalía, emite entonces un "flag de error", que detiene la transmisión en curso. De esta forma se impide que otras estaciones reciban el mensaje erróneo.

Gracias a este sistema de seguridad que incorpora el CAN se consigue que las probabilidades de fallo en el proceso de comunicación sean muy bajas, pero sigue siendo posible que cables, contactos y las propias unidades de mando presenten alguna disfunción.

Por ello, para el análisis de una avería, se debe tener presente que una unidad de mando averiada abonada al CAN en ningún caso impide que el sistema trabaje con normalidad. Lógicamente no será posible llevar a cabo las funciones que implican el uso de información que proporciona la unidad averiada, pero sí todas las demás.

Una alternativa posible para localizar fallos en el CAN es emplear el programa informático CANAlyzer (Vector Informatik GmbH) con el ordenador y con la conexión adecuada. Este programa permite visualizar el tráfico de datos en el bus CAN, indica el contenido de los mensajes y realiza la estadística de mensajes, rendimiento y fallos.

Anexo B. DESCRIPCIÓN DE LAS COMUNICACIONES DEL SISTEMA

B.1. Puerto serie (RS232)

El RS232 es un protocolo de comunicación serie orientado a carácter, es decir, un protocolo donde toda la información es enviada por un solo canal bit a bit (un canal para enviar información y otro para recibirla) y donde lo que se envían son caracteres.

Este protocolo está diseñado para distancias cortas, de unos 15 metros más o menos, y para unas velocidades no muy elevadas que pueden llegar a los 20 kbps. Se puede trabajar de forma asíncrona o síncrona y con tipos de canal simplex, halfduplex y fullduplex. Como ya se ha comentado tres párrafos más arriba, en esta ocasión se hará uso como si de una comunicación simplex (unidireccional) se tratara aunque el módulo RS232 tendrá una composición halfduplex.

Una conexión RS232 es definida por un cable desde un dispositivo al otro. Hay 25 conexiones en la especificación completa pero en la mayoría de los casos se utiliza menos de la mitad. La conexión más sencilla se puede realizar con tres cables (TD, RD, y GND), que es la que se utilizará para este sistema:

- GND: Valor a 0 V (masa).
- TD: Línea de datos del transmisor al receptor.
- RD: Línea de datos del receptor al transmisor.

Uno de los parámetros importantes que se ha de configurar para poder realizar una comunicación RS232 es la velocidad de transmisión de los datos ya que este protocolo permite escoger varias velocidades. Para esta ocasión se ha elegido una velocidad de las más elevadas que permite utilizar para así evitar que la transmisión serie quede desbordada (se colapse) como consecuencia de que la velocidad de transmisión por el bus CAN es mucho más elevada: 115200 bps. Esta comunicación tiene un uso de control de errores (“debugador”) de las acciones del propio nodo, en las que están incluidas las transmisiones realizadas por el bus CAN. De ahí que haya una relación proporcionalmente indirecta entre las velocidades de ambos protocolos.

Como consecuencia de que el microcontrolador que se utilizará opera entre 0 y 5 V aproximadamente, es necesario adaptar niveles de tensión puesto que el puerto serie del ordenador necesita una tensión de ±12 V para poder funcionar. Para ello ha sido necesario incorporar un integrado, para que convierta los niveles de las líneas del RS232 a niveles TTL y viceversa. Debido a que el puerto serie no está pensado en principio como una comunicación que se mantenga en el sistema final, se ha descartado incluir el integrado en el diseño y así hacerlo más compacto. De esta forma la conexión entre el nodo (microcontrolador) y el ordenador se realiza mediante un conector (ver Fig. B.1) que ya incorpora el integrador, en este caso, el MAX3223 de la casa Maxim

[21]. Este conector incorpora por un extremo un DB-9 hembra que será el que se conectará al DB-9 macho que incorpora aquel ordenador que disponga de puerto serie. Por el otro extremo posee un conector con entrada para cinco contactos (pines) de los cuales solo se utilizaran cuatro que ya estarán incorporados en el nodo. Estos cuatro pines los forman los tres ya mencionados con anterioridad y que son necesarios para realizar la transmisión del puerto serie (GND, TD y RD) más un cuarto, VCC (5 V), que sirve para alimentar el MAX3223 que incorpora este conector.

Fig. B.1 Conector del RS232 para comunicar los nodos con un PC

El software, o más bien programa, utilizado para hacer un control de la red domótica a través del puerto serie es el "*Terminal*". Esta pequeña aplicación (es un simple ejecutable) viene a ser a un estilo al famoso "*Hyperterminal*", pero con una interfaz bastante más intuitiva, pero eso sí, menos cuidada visualmente. Además permite un sinfín de opciones configurables y accesibles desde la misma sesión, lo cual lo convierte en programa muy funcional.

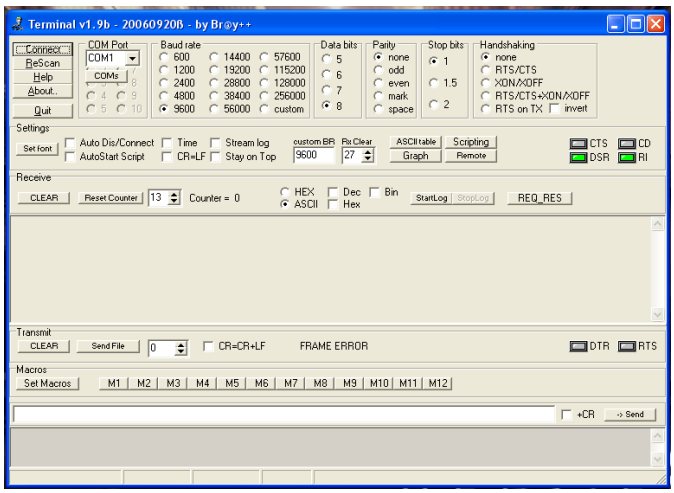

Fig. B.2 Entorno del denominado Terminal

B.2. Puerto de programación para microcontrolador

A parte del puerto serie, cada nodo también dispone de un puerto de programación. De esta forma los nodos son reprogramables ya que se les podrán cambiar las funcionalidades volviendo a grabar el software (programación en lenguaje C) nuevamente. La grabación de este software se ha llevado a cabo con un programador USB-ICD llamado *PICFLASH2* del fabricante *mikroElektronika* [22]:

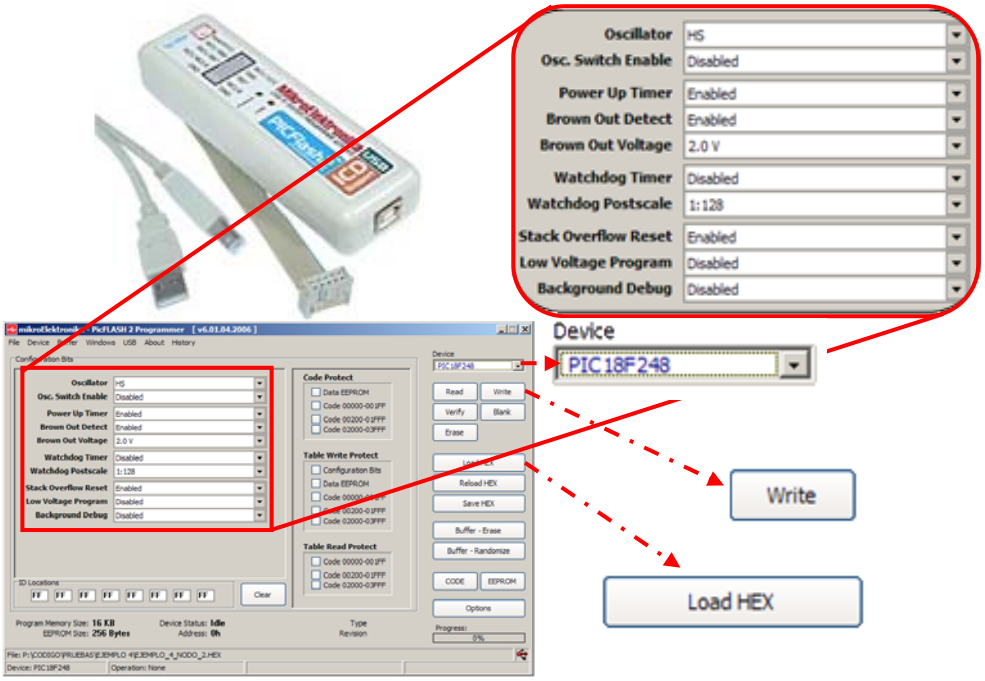

Fig. B.3 Programador PICFLASH2 y su software

La aplicación utilizada para el programador es la que se observa en la figura Fig. B.3. Simplemente hay que seleccionar el modelo del microcontrolador y configurar los parámetros de bit que solicita. Los valores que siempre se han configurado para estos parámetros están indicados en la misma figura anterior. Después se carga el archivo HEX (“Load HEX”) y se escribe (“Write”) para que realice la grabación. Dicho puerto de programación que incorpora cada nodo no es nada más que una conexión que consta de cinco pines que van conectados entre el microcontrolador y el programador de la siguiente manera:

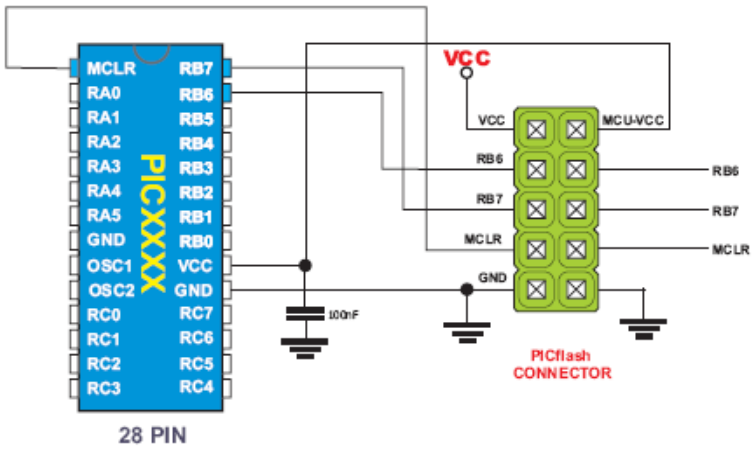

Fig. B.4 Conexionado del microcontrolador al programador: ICD

B.3. Módulo USB-CAN

Se citaba en el apartado 4.3 que las diferencias de velocidades entre el puerto serie y el CAN pueden llegar a ser considerables. La velocidad máxima que permite el RS232 es de 20 kbps y la del bus CAN puede llegar a ser de hasta 1 Mbps Por ello es lógico pensar que es totalmente inviable utilizar el puerto serie como medio de comunicación entre la red CAN y un ordenador para poder crear después una aplicación de alto nivel que procese lo que sucede en el bus y pueda también comunicarse con el mismo. Sería imposible pues se produciría un embotellamiento de datos. Por lo tanto en caso de querer comunicar el CAN con un ordenador (para, por ejemplo, diseñar una aplicación de alto nivel), es necesario un hardware adicional, un módulo o interfaz, que permita comunicar el bus CAN con el ordenador a través de otro protocolo. Es aquí donde entra en escena el *NI USB-847x CAN Interfaces* [23], una especie de conversor de CAN a USB:

Fig. B.5 Módulo USB-CAN de National Instruments

Este módulo incorpora en su interior un controlador de CAN más un transceiver. Su función vendría a ser como un nodo más pero con la

particularidad de que su arquitectura está basada en un estilo al Basic CAN. Por eso este módulo es de un único puerto, es decir, el controlador CAN únicamente posee un buffer de transmisión y recepción con su correspondiente e único filtro y máscara de aceptación. Por un extremo posee un cable USB, el cual ya permite velocidades mucho mayores incluso que el CAN mismo, y por otro lado incorpora un conector DB9 del cual se utilizarán es esta ocasión tres pines para conectarlo al bus CAN:

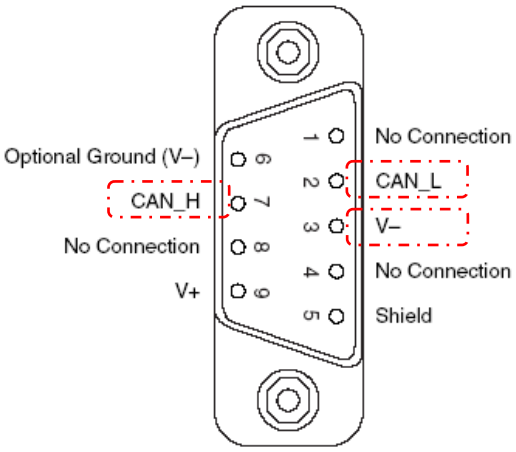

Fig. B.6 Conector DB9 del módulo CAN-USB