CAN2PCI: PLACA CON INTERFAZ AL BUS CAN Y PCI CON FINALIDAD DOCENTE

M. A. ORTIZ, F. J. QUILES, C. D. MORENO, M. A. MONTIJANO, M. BROX, A. GERSNOVIEZ
Departamento de Arquitectura de Computadores, Electrónica y Tecnología Electrónica.
Escuela Politécnica Superior. Universidad de Córdoba. España.
el1orlom@uco.es

En este trabajo presentamos una placa con interfaz al bus CAN y PCI desarrollada para la utilización en prácticas de las asignaturas relacionadas con redes de control. Se trata de una placa de altas prestaciones pensada para su utilización docente gracias a su facilidad de programación. La placa dispone de dos canales CAN independientes y permite acceso directo a los registros del controlador CAN. Al disponer de dos canales se puede tener en cada canal una red distinta y realizar prácticas de interconexión de redes CAN, o se puede tener una red CAN de dos nodos con una sola placa en cada puesto de prácticas. Se exponen también brevemente las prácticas realizadas en una de las asignaturas relacionadas con redes de control.

1. Introducción

Actualmente se tiende al diseño de sistemas de control distribuidos, y como consecuencia, las redes de control y el intercambio de información a través de ellas cobra una gran importancia. Por este motivo cada vez más se utilizan redes de control estandarizadas, que no solo definen el nivel físico sino que proporcionan un mínimo nivel de enlace que permite la transmisión de datos seguros y tolerante a fallos. Una de estas redes de control es el bus CAN que se ha convertido en el líder como bus serie para aplicaciones empotradas, especialmente en el campo de la automoción. Hemos elegido este bus para la realización de las prácticas de redes de control en la asignatura de "Arquitectura y Protocolos para Redes de Control Distribuido" que se imparte en la titulación de Ingeniero en Informática de la Universidad de Córdoba.

El bus CAN se utiliza ampliamente en el automóvil y en aplicaciones industriales, y se ha estandarizado internacionalmente como ISO 11898 para aplicaciones de alta velocidad [1] y ISO 11519-2 para aplicaciones de baja velocidad [2]. Además de esta estandarización ISO/OSI, se tienen otras descripciones CAN 2.0A y CAN 2.0B para los fabricantes de controladores CAN [3].

Una típica red CAN está formada por varios nodos conectados al bus. Todos los nodos tienen disponible o pueden leer la información que viaja por el bus. La información que circula está compuesta por mensajes. Cada mensaje tiene un identificador que, además de nombrar el mensaje, indica la prioridad de ese mensaje. Ésta se emplea en caso de que dos o más nodos intenten poner mensajes en el bus al mismo tiempo.

El identificador del mensaje debe ser único en la red. La diferencia entre la especificación CAN2.0A y CAN2.0B está en el número de bits del identificador del mensaje. A diferencia de otras redes, este identificador no tiene porqué estar relacionado con el nodo que envía o recibe la información, sino a la información en sí misma. Aunque en la práctica, algunos bits del identificador del mensaje suelen reservarse para asociarlos al nodo que envía la información o al nodo al que va destinada. El formato del mensaje se muestra en la figura 1.

Identificador	Control	Datos	CRC	ACK

Figura 1. Formato de mensaje del bus CAN.

El campo de Control especifica el tipo de trama y el número de bytes de datos que contiene el mensaje. El campo de datos contiene los datos del mensaje pudiendo ser de hasta 8 bytes. El campo CRC se utiliza para detección de errores en la transmisión. CAN define dos valores: bit dominante y bit recesivo. Por definición, el bit dominante sobrescribe al bit recesivo. En la transmisión el nodo que transmite pone el bit ACK al valor recesivo, y en caso de que cualquier nodo del bus haya recibido el mensaje sin errores, deberá poner este bit al valor dominante. Se garantiza con el campo ACK que al menos un nodo ha recibido correctamente el mensaje. Existen además mensajes especiales, que no detallaremos en este trabajo, para informar a todos los nodos del bus que se han producido determinados errores.

La arbitración en el bus se muestra en la figura 2. Cuando el bus está inactivo cualquier nodo puede comenzar la transmisión de un mensaje. Si dos o más nodos comienzan la transmisión al mismo tiempo se inicia una arbitración basada en la prioridad del mensaje, es decir de acuerdo a su identificador. Cada nodo lee del bus bit a bit y compara el valor que transmite con el valor que lee en el bus. Si un nodo pone un bit recesivo y lee uno dominante significará que el nodo ha perdido la arbitración. La arbitración en CAN es como una AND global entre nodos. Esta arbitración es no destructiva y el nodo ganador no sufre ninguna interferencia.

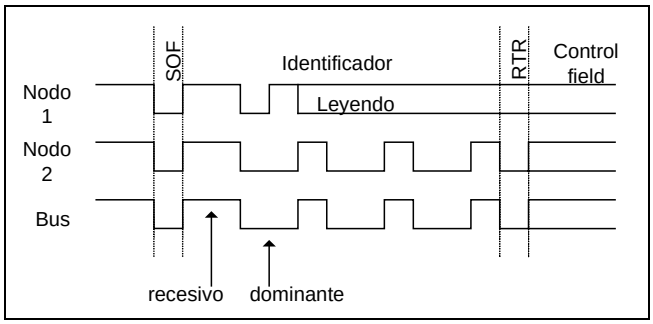

Figura 2. Arbitración en el bus CAN.

Como se puede deducir de la breve exposición de las características del bus CAN, para la realización de prácticas con el bus CAN se debe tener una red y unos nodos desde los que no solo se pueda transmitir y recibir información sino que se puedan configurar y manipular a bajo nivel. Además en estos nodos debe ejecutarse algún sistema operativo en tiempo real para conseguir unas prácticas lo más próximas a la realidad como sea posible.

En el mercado se tiene una amplia variedad de placas con interfaz al bus CAN y PCI [4], sin embargo, la dificultad que nos habíamos encontrado es que no se dispone de una información detallada del hardware de estas placas por lo que es imposible programar directamente los controladores CAN. Además los fabricantes proporcionan un conjunto de librerías para el manejo de la red CAN solamente para algunos sistemas operativos, siendo difícil encontrar estas librerías para sistemas operativos en tiempo real. Aunque desde el punto de vista industrial, la utilización de estas librerías es la manera más eficiente de trabajo, en el caso de las prácticas, interesa poder trabajar directamente con los registros de los

controladores CAN. Así el alumno puede comprender mejor como intercambian información estas redes, que es el objetivo principal de este tipo de prácticas.

Como se ha comentado el objetivo de estas prácticas es conocer el bus CAN, por lo que es necesario trabajar directamente con los controladores y realizar aplicaciones con los sistemas operativos de tiempo real que ya conoce el alumno gracias a otras asignaturas. Teniendo esto en cuenta se pensó que lo más práctico era el desarrollo de una placa con interfaz al bus CAN y PCI, de forma que se pudiera programar de una manera cómoda y sencilla desde el ordenador personal y con un sistema operativo de tiempo real para dar un mayor realismo a las prácticas.

En los siguientes capítulos describiremos la placa y el conjunto de prácticas que se realizan en la asignatura de "Arquitectura y Protocolos para Redes de Control Distribuido" que se imparte en el segundo curso de la Titulación de 2º ciclo de I. en Informática.

2. Características de la placa CAN2PCI

A la hora de decidir que características debía tener la placa había que tener en cuenta, además de las prestaciones, su finalidad docente. Por este motivo se optó por utilizar un controlador de bus CAN que fuese flexible y simple de programar. Si el espacio lo permitía debía tener dos canales para que de una manera cómoda se tuviese una red CAN de al menos dos nodos por puesto de prácticas. Teniendo en cuenta estas dos premisas optamos por utilizar el controlador SJA1000 [5] y montar dos controladores para disponer así de dos canales. Cada controlador trabajaría independientemente. El acceso a los controladores desde el ordenador sería directo, es decir, el programador accedería directamente al controlador sin utilizar ninguna memoria intermedia entre la CPU y el controlador CAN, aunque esta opción podría limitar el rendimiento del nodo CAN. Por otro lado para simplificar el acceso a los controladores decidimos mapear la placa en el espacio de entrada/salida del bus PCI.

Las características principales de la placa CAN2PCI (Fig. 3) son:

- Interfaz al bus CAN: dos canales independientes.
- Controlador de bus CAN SJA1000, con velocidad de hasta 1 MB.
- Posibilidad de utilizar alimentación del bus PCI o alimentación a través del propio bus CAN para el *transceiver* del bus CAN.
- Aislamiento por optoacopladores del bus CAN.
- Interfaz al bus PCI de 32 bits y 5 Voltios.

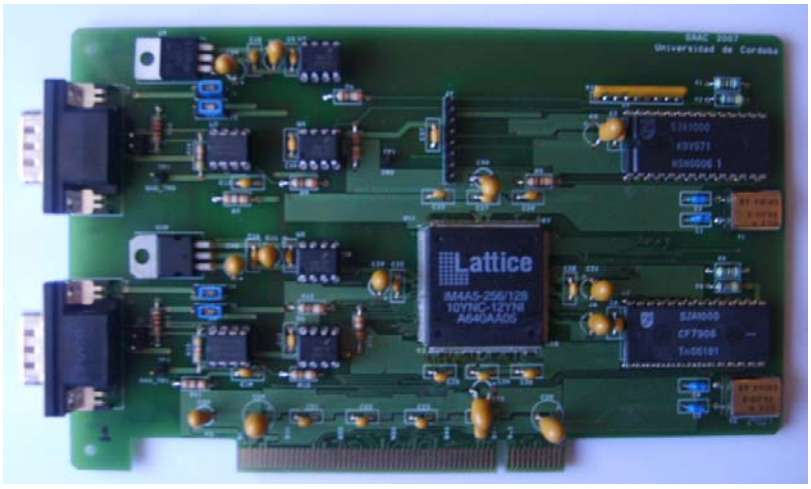

Figura 3. Fotografía de la placa CAN2PCI.

3. Descripción de la placa CAN2PCI

La figura 4 muestra el diagrama de bloques de la placa CAN2PCI. Toda la lógica de control de esta placa está integrada en el CPLD M4A5-256 de la compañía Lattice Semiconductor [6]. Dicho CPLD realiza el interfaz al bus PCI y genera el bus local de los controladores.

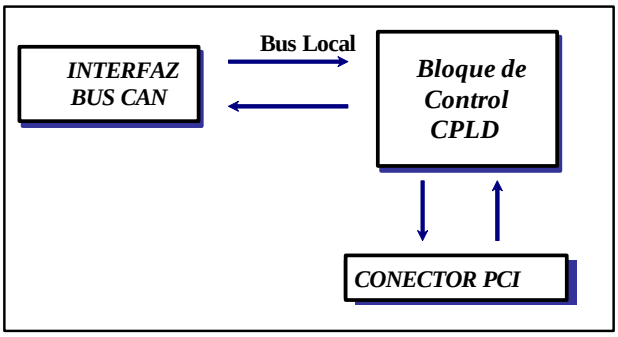

Figura 4. Diagrama de bloques de la placa CAN2PCI.

La figura 5 muestra el diagrama de bloques del interfaz de un canal al bus CAN. Este interfaz está formado por el controlador SJA1000 que se encuentra aislado del *transceiver* del bus CAN mediante optoacopladores. La alimentación del *transceiver* se puede seleccionar mediante un puente entre la alimentación del bus PCI o la alimentación proveniente del bus CAN.

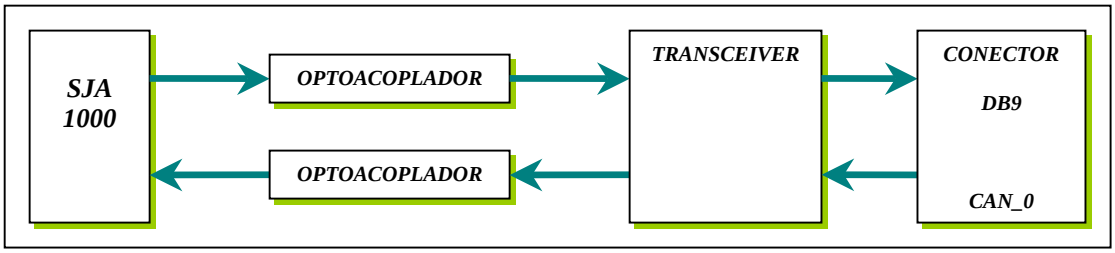

Figura 5. Interfaz al bus CAN.

4. Prácticas en redes de control utilizando la placa CAN2PCI.

La placa CAN2PCI está pensada para ser utilizada en la asignatura de "Arquitectura y Protocolos para Redes de Control Distribuido" donde se estudia el bus CAN entre otros buses de control. Se trata de una asignatura optativa de 4,5 créditos cuyo objetivo principal es estudiar los buses de control en los niveles físicos y de enlace, estudiando en profundidad la transmisión de las tramas. Como objetivo secundario se realizan aplicaciones sencillas de control. Para cumplir el objetivo principal de las prácticas, es necesario acceder a los registros del controlador SJA1000 para su configuración y a los *buffers* de datos de transmisión y recepción. Para el desarrollo de las aplicaciones se utiliza también un sistema operativo en tiempo real, que el alumno ya ha estudiado en otras asignaturas obligatorias, en concreto se utiliza el sistema operativo en tiempo real "Phar Lap ETS" [7].

Las prácticas de la asignatura "Arquitectura y Protocolos para Redes de Control Distribuido" lleva impartiéndose dos cursos y los comentarios que se realizan en este trabajo son fruto de la experiencia adquirida en este tiempo. Aunque el grado de experimentabilidad de esta asignatura permite un máximo

de 25 alumnos por grupo de prácticas, hasta el momento, el grupo con mayor número de alumnos ha sido de 14. Las prácticas se realizan en parejas y se utilizaron 7 puestos de prácticas.

Cada puesto de prácticas consta de dos ordenadores personales como muestra la figura 6. Normalmente casi todos los sistemas operativos de tiempo real y en particular el entorno de desarrollo del sistema operativo de tiempo real "Phar Lap ETS", necesitan dos equipos. Uno de ellos, denominado "target", es el equipo donde se ejecutará nuestra aplicación y el otro, denominado "host", contiene el entorno de desarrollo del sistema operativo. Ambos equipos se conectan a través de un canal de comunicación (ethernet, serie o paralelo). En el equipo "host" el usuario escribe el código, lo compila, *linka* y lo envía para su ejecución al "target", pudiendo además depurarlo. En el equipo "target" estará montada la placa CAN2PCI para la realización de las prácticas.

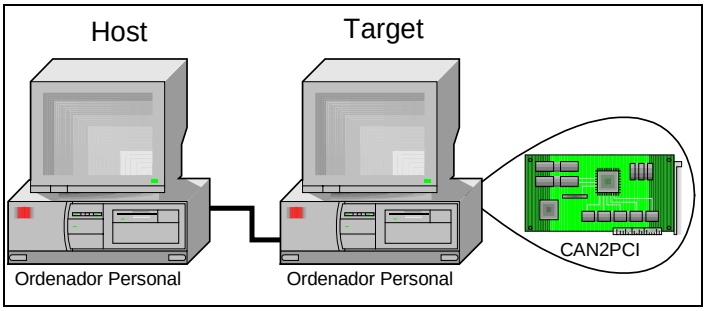

Figura 6. Puesto de prácticas.

El lenguaje de programación que se utiliza es C. Los alumnos no tienen problemas a la hora de la codificación porque utilizan C habitualmente a lo largo de la carrera y el sistema operativo "Phar Lap ETS" ya lo han utilizado en otras asignaturas. Por tanto el punto de partida es el idóneo para centrarse en los objetivos de las prácticas.

El objetivo final de las prácticas es realizar una sencilla aplicación parecida al sistema ABS de un automóvil, sin la utilización de ninguna librería o driver de manejo de la red CAN. La figura 7 muestra la estructura de capas de la aplicación.

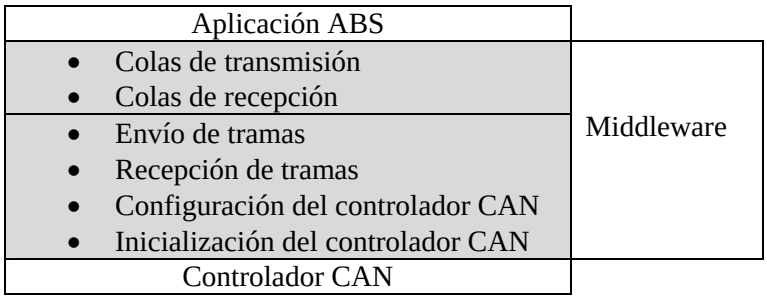

Figura 7. Estructura de capas de la aplicación objetivo de las prácticas.

Las prácticas se han dividido en cuatro bloques y se realizan en sesiones de 2 horas de duración. Se dedican un total de 20 horas a prácticas de las 45 horas totales de asignatura. A continuación realizaremos una breve descripción de las prácticas.

4.1. Transmisión de tramas CAN.

El objetivo de esta práctica es el estudio de los principales registros del controlador SJA1000 y transmisión de mensajes por la red. La meta a conseguir es que el alumno transmita mensajes con éxito por la red, por lo que previamente deberá programar la velocidad de transmisión y recepción, y los registros de máscara y aceptación del controlador. Cada puesto de prácticas está aislado del resto y se le conecta un analizador de tramas CAN. En esta práctica se le enseña al alumno a manejar el analizador de tramas CAN y se muestra también en un osciloscopio la trama a nivel físico. Habitualmente la duración de esta práctica es de 2 sesiones de 2 horas.

Estas primeras prácticas son bastante intensas, y aunque hemos comentado anteriormente que la programación en C no representa ninguna dificultad para los alumnos, sí les resulta dificultoso la tipificación de datos en C. Deben manejar los registros del controlador a nivel de bytes y bits que no son tipos de datos estándar en C. También debe adaptarse a las peculiaridades de la programación en las aplicaciones de control. Los alumnos está acostumbrados a una programación secuencial más que concurrente, y sin embargo, la naturaleza de las aplicaciones de control obliga a pensar en tareas y en muchos casos, tareas que deben realizarse con una cierta periodicidad.

4.2. Recepción de tramas CAN.

El objetivo de esta práctica es la recepción de mensajes. El alumno debe realizar las funciones que le permitan descargar del controlador los mensajes recibidos en el controlador y depositarlos en memoria. En esta práctica los nodos siguen aislados y las tramas se envían desde el analizador que se va conectando a cada puesto. Habitualmente la duración de esta práctica es de 2 horas.

4.3. Creación de un pequeño middleware.

La tercera práctica consiste en la creación de un pequeño *driver* o *middleware* que integra las funciones de transmisión y recepción de mensajes creadas en las dos prácticas anteriores. Al inicio de esta práctica se conectan todos los puestos en red y comprobamos que desde todos los puestos se puede transmitir y recibir mensajes correctamente. A continuación se crea una simple aplicación que va lanzando mensajes desde un nodo con identificadores que se incrementan a cada mensaje. En el resto de nodos se crea otra aplicación que va sacando los mensajes del controlador y llegan con el identificador esperado.

Para finalizar se crean unas colas de transmisión y recepción que se comprueban utilizando las simples aplicaciones comentadas anteriormente. De esta forma se conciencia al alumno de la necesidad de realizar pruebas a las funciones que han implementado. Habitualmente la duración de esta práctica es de 4 horas repartidas en 2 sesiones de 2 horas.

4.4. Diseño y codificación de una sencilla aplicación.

Para finalizar las prácticas se desarrolla una pequeña aplicación simulando un sistema “ABS mínimo” de un automóvil. Se crea una red CAN donde cada nodo representa una rueda de un automóvil. Cada nodo genera la velocidad de una rueda de forma simulada y la envía a la red. A su vez recoge de la red la velocidad de las distintas ruedas. Un nodo simula el freno que envía la información de su estado a la red. Para simular el bloqueo, la aplicación que se ejecuta en cada nodo genera de forma aleatoria el bloqueo de la rueda enviando velocidad cero. Para comprobar que todo funciona correctamente cada nodo muestra en pantalla del computador la información que envía y recoge de la red y si está disparando el ABS. Habitualmente la duración de esta práctica es de 5 sesiones de 2 horas.

Esta última práctica requiere un trabajo de equipo extra, ya que hay que definir los datos que contendrá las tramas que se intercambien, la periodicidad de estas tramas, la velocidad de transmisión, etc.

El alumno debe trabajar en programación multihilo, y además esta práctica tiene la complejidad añadida de ser una aplicación distribuida y, por tanto, requiere un esfuerzo de puesta a punto mayor.

5. Conclusiones

El diseño de la placa CAN2PCI con un enfoque docente nos permite realizar prácticas con el bus CAN a bajo nivel y utilizando un sistema operativo en tiempo real gracias a la posibilidad de tener un hardware "abierto". Esta posibilidad es muy importante en las asignaturas como "Arquitectura y Protocolos para Redes de Control Distribuido", donde el objetivo es el conocimiento de las redes así como de algunos algoritmos distribuidos, y no tanto el desarrollo de aplicaciones de control, que son el cometido de otras asignaturas.

Disponer de dos canales en una sola placa permite tener una red CAN por puesto de prácticas, quedando así aislados los problemas que suelen aparecer en las primeras prácticas. Queda por ultimar y explorar la realización de prácticas de interconexión de redes CAN distintas, para lo que será necesario realizar un puente entre ambas redes.

Referencias

[1] International Standard ISO 11898, *Road vehicles - Interchange of digital information - Controller area network (CAN) for high speed communication*. ISO Reference number ISO 11898, November 1993.

[2] International Standard ISO 11519-2, *Road vehicles - Low speed serial data communication - Part 2: Low speed controller area network (CAN)*. ISO Reference number ISO 11519-2, Jun 1994.

[3] Robert Bosch GmbH, *CAN specification 2.0, Part A and B*, September 1991.

[4] PCI Local Bus Specification. Revision 2.2. PCI Special Interest Group (http://www.pcisig.org/). Año 1998.

[5] NXP Semiconductors. *SJA1000 Stand-alone CAN controller*. http://www.nxp.com.

[6] Lattice Semiconductor Corporation. *The ispMACH 4A programmable logic family*. http://www.latticesemi.com.

[7] Ardence Company. *Phar Lap ETS – Real-time Operating System*. http://www.ardence.com/embedded.