Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores

Diego Encinas[1], Pablo Meilan[4], J. Alberto Bava[2, 3], R. Marcelo Naiouf[1].

[1]Instituto de Investigación en Informática (III-LIDI). Facultad de Informática. UNLP.
[2]Centro de Investigaciones Ópticas (CIOp) – CONICET – CIC.
[3]Facultad de Ingeniería. UNLP.
[4]Vehículos Espaciales de Nueva Generación S. A. (VENG S. A.)
dencinas@lidi.info.unlp.edu.ar, pfmeilan@conae.gov.ar, bava@ciop.unlp.edu.ar,
mnaiouf@lidi.info.unlp.edu.ar

Resumen. CAN (Controller Area Network) es un protocolo abierto para uso automotriz y de alta confiabilidad, apropiado para aplicaciones de tiempo real distribuidas, como una red de dispositivos de vuelo. La principal motivación del análisis y aplicación de este protocolo es poder utilizar componentes comerciales (COTS, Commercial off the Shelf) para el desarrollo de un sistema de comunicaciones en el ámbito espacial, logrando reducir costos pero cumpliendo con los requerimientos de un sistema modular distribuido de vuelo (MDAS, Modular Distributed Avionics System). En este trabajo se presenta una aplicación del protocolo de comunicación CAN a fin de ser implementado sobre un sistema aeroespacial. Se analizan sus ventajas en interconexión satelital y vehículos lanzadores de un prototipo, utilizando microcontroladores interconectados al bus mediante transceptores y controladores CAN

Palabras claves: Red CAN, Sistemas Distribuidos de Tiempo Real, Sistemas Satelitales y Vehículos Lanzadores.

Abstract. CAN (Controller Area Network) is an open and high reliable protocol for automotive uses, suitable for distributed real time applications like an avionics network. The main motivation for analysing and applying this protocol is to be able to use commercial components (COTS, Commercial off the Shelf) in developing a communication system for aerospace devices, achieving low costs but performing the requirements of a Modular Distributed Avionics System (MDAS). This paper presents an application of the CAN communication protocol to be implemented in an aerospace system. Its advantages in satellite interconnection and launch vehicles of a prototype are analysed, using microcontrollers connected to the bus with CAN transceivers and controllers.

Keywords: CAN Network, Distributed Real Time Systems, Satellite Systems and Launch Vehicles.

1 Introducción

Debido a la complejidad de los sistemas de vuelo se han desarrollado muchos protocolos de comunicaciones serie no-estandarizados para cada proyecto. Como esto implica un costo muy alto, tanto de dinero como de tiempo, surge la necesidad de utilizar protocolos estandarizados.

La ventaja del uso de componentes comerciales es que son económicos. Sin embargo, la principal desventaja es que algunos no satisfacen los requerimientos, tanto térmicos, electromagnéticos, mecánicos, entre otros, necesarios para funcionar en un sistema de vuelo.

Es preciso realizar un cuidadoso análisis de eficiencia, tolerancia a fallas y compatibilidad electromagnética del protocolo de comunicaciones y sus componentes comerciales antes de implementarlo en un sistema de vuelo.

Generalmente el protocolo de comunicaciones serie MIL-STD-1553 [4] es el utilizado en satélites y vehículos lanzadores. Aunque actualmente también está utilizándose el protocolo SpaceWire [15] (basado en el protocolo IEEE 1355 [16]) que fue diseñado especialmente para sistemas aeroespaciales. Por otro lado, para comunicaciones con requerimientos menores se utilizan los protocolos ANSI/TIA/EIA-422-B [14] y TIA/EIA-485-A [14].

Para este trabajo se ha elegido el protocolo CAN [2] el cual ha sido formalmente definido por el estándar ISO 11898 [5]. La elección se debe a su bajo costo, velocidad de transmisión media, control de errores y disponibilidad en el mercado de controladores y transceptores CAN, en circuitos integrados (stand alone). Además existe una amplia variedad de interfaces CAN-PCI y CAN-USB para utilizar en computadores de escritorio.

En principio este protocolo y su bus asociado fue desarrollado por Bosch para la industria automotriz pero actualmente es aceptado por distintas compañías de aviación e instituciones como NASA, ESA, Airbus, Boeing, Eurocopter, entre otros. Además de las aeroespaciales, otras aplicaciones posibles son: robótica, industria naval y militar (barcos, submarinos y tanques), transporte (subterráneos, semáforos), domótica (automatización de edificios) y equipamiento médico (electromedicina).

Este trabajo está organizado de la siguiente forma: en la segunda sección se describirán algunas características del protocolo de comunicaciones CAN. En la tercera sección se detalla el desarrollo de un prototipo. Finalmente en la cuarta sección se presentan las conclusiones encontradas.

2 Descripción del protocolo de comunicaciones CAN

CAN es conocido como un protocolo de comunicaciones serie de alta confiabilidad, robustez y performance; además es apropiado, mediante un desarrollo de capa de aplicación, para el control de sistemas distribuidos de tiempo real. Sus principales características son:

- Priorización de mensajes.
- Sistema Multi-maestro.
- Configuración flexible.

- Velocidad de transmisión media (hasta 1 Mbit/s).
- Señalización y detección de fallas.

Para detectar errores a nivel de mensajes el protocolo implementa tres mecanismos: chequeo de redundancia cíclica (CRC), chequeo de formato de trama y acuse de recibo (ACK).

Para detectar errores a nivel de bit el protocolo implementa dos mecanismos:

1. Monitoreo. Cada nodo transmisor monitorea el nivel del bus a fin de detectar la diferencia entre el bit enviado y el recibido.
2. Bits de Relleno (Bit stuffing). CAN utiliza la codificación NRZ (no retorno a cero) y la técnica de bit stuffing, que consiste en insertar un bit complementario cada cinco bits iguales consecutivos transmitidos.

El protocolo CAN sólo define las dos primeras capas del Modelo OSI [1] (física y enlace), aunque no especifica la interfaz al medio físico (Medio de transmisión, conectores). La norma ISO 11898 es el estándar internacional para la comunicación de alta velocidad usando el protocolo de bus CAN. Esencialmente este estándar define las capas física y de enlace de datos. La capa física se subdivide en tres subcapas, especificadas en la Tabla 1.

Tabla 1 – Arquitectura de capas del protocolo CAN [6]

<table>
<tr><td colspan="2">Especificación</td><td colspan="2">Capa OSI</td><td>Implementación</td></tr>
<tr><td colspan="2">Especificado por el diseñador del sistema</td><td colspan="2">Capa de Aplicación</td><td></td></tr>
<tr><td colspan="5"></td></tr>
<tr><td rowspan="5"></td><td rowspan="3">Especificación Protocolo CAN</td><td rowspan="2">Capa de Enlace de Datos</td><td>Enlace de control Lógico</td><td rowspan="3">Controlador CAN</td></tr>
<tr><td>Acceso Control del medio</td></tr>
<tr><td rowspan="3">Capa Física</td><td>Señalización Física</td></tr>
<tr><td colspan="2">Especificado por ISO 11898</td><td>Acceso al medio físico</td><td>Transceptor CAN</td></tr>
<tr><td colspan="2"></td><td>Interfaz dependiente del medio</td><td rowspan="2"></td></tr>
<tr><td colspan="2"></td><td colspan="2">Medio de transmisión</td></tr>
</table>

Los mensajes son gestionados por cuatro tipos de tramas:

- Trama de Datos: Es utilizada por un nodo para enviar información.
- Trama Remota: Es enviada por un nodo solicitando una Trama de Datos con el mismo identificador.
- Trama de Error: Cuando un nodo detecta un error en el bus envía esta trama al resto de los nodos (rompiendo la secuencia de Bit Stuffing).
- Trama de Sobrecarga: Fuerza a que los nodos alarguen el tiempo de transmisión entre tramas sucesivas (de Datos y Remotas)
- Espacio entre trama: Las tramas de Datos y Remotas se separan entre si por esta secuencia predefinida. Consta de tres campos: Intermisión, Bus libre y Transmisión suspendida.

La Trama de Datos posee siete campos de bits y la Trama Remota seis campos de bits (esta última con un campo menos debido a que no posee el campo de datos).

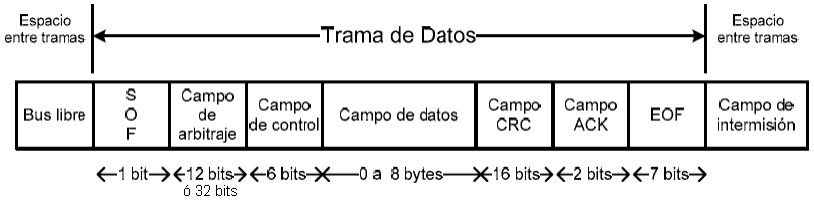

Figura 1 – Campos de la Trama de Datos

El formato de un mensaje o trama puede verse en la Figura 1. Los campos SOF (Start of Frame) y EOF (End of Frame) son delimitadores de la trama. El campo de Arbitraje define la prioridad de un mensaje cuando dos o más nodos desean acceder al bus. En el formato estándar (CAN 2.0A), el campo contiene un identificador de 11 bits
y un bit RTR (Remote Transmission Request) con el que se identifica a la trama como de datos o remota. En el formato extendido (CAN 2.0B), el campo contiene un identificador de 29 bits, un bit SRR (Substitute Remote Request), un bit IDE (Identifier Extension) y un bit RTR. El campo de Control indica el número de octetos que contendrá el campo de Datos. Los campos CRC y ACK cumplen funciones similares a las encontradas en otros protocolos.

Acceso al medio

Cuando dos o mas nodos empiezan a transmitir mensajes al mismo tiempo el conflicto de acceso al bus de comunicaciones es resuelto mediante arbitraje (bite-wise) usando un identificador. Este método de control de acceso al medio es conocido como “Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority” (CSMA/CD + AMP). El identificador es utilizado para definir la prioridad de un mensaje asegurando que el mensaje de mayor importancia sea transmitido primero. Las Tramas de Datos predominan sobre cualquier otra, previniendo el conflicto de enviar diferentes tipos de tramas con el mismo identificador.

Durante el arbitraje, cada nodo compara el nivel de bit a ser transmitido con el nivel que es monitoreado en el bus. Si los niveles son iguales el nodo puede continuar transmitiendo el próximo bit. Cuando un nodo transmite un bit con nivel recesivo y es monitoreado un nivel dominante, entonces la unidad pierde el arbitraje y debe retirarse sin enviar un bit más, pasando a un estado de “oyente” del bus. Este mecanismo es conocido como “Wired-AND” [3] en el que, los “bits dominantes” (equivalentes al nivel lógico “cero”) sobrescriben a los “bits recesivos” (equivalentes al nivel lógico “uno”). En la Figura 2 se muestra a tres nodos transmitiendo conectados al bus y aplicando el mecanismo antes mencionado.

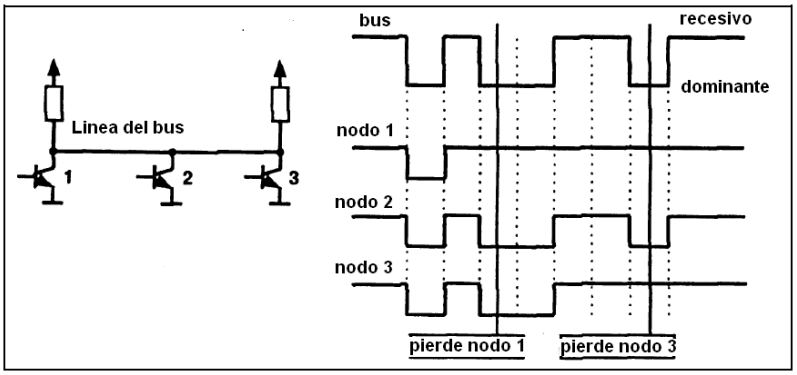

Figura 2 – Principio de arbitraje no destructivo [3]

Estados de error

Existe la distinción entre errores temporales y fallas permanentes en los nodos y desconexión automática de nodos defectuosos.

Cuando un nodo funciona en forma defectuosa debe evitarse que este envíe mensajes, ya que pueden estar corrompidos. CAN provee un mecanismo para prevenir esto. Cada nodo posee dos contadores de error: REC (Receive Error Counter) que cuenta el número de errores de transmisión en las tramas recibidas por el nodo y TEC (Transmit Error Counter) que cuenta el número de errores de transmisión en las tramas enviadas por el nodo.

Cada vez que una trama es transmitida o recibida correctamente por un nodo, el contador correspondiente implementa una cuenta regresiva. De igual forma, cuando un error de transmisión es detectado produce un incremento en el contador correspondiente. Existen reglas para el incremento y decremento de los contadores y están descriptas en la referencia [2] El nodo puede encontrarse en uno de tres estados, de acuerdo al valor de ambos contadores:

- Error Activo: el nodo puede recibir y enviar tramas normalmente. Es el estado en que se encuentra una estación al ser inicializada.
- Error Pasivo: los mensajes pueden ser transmitido y recibidos, pero después de la transmisión de un mensaje el nodo debe suspender la transmisión y esperar un tiempo definido antes de volver a transmitir. Este seria el caso de un nodo con disturbios cortos o fallas temporales. Si los contadores de error del nodo llegan a determinado valor, configurable, el nodo puede volver al estado activo.
- Bus Apagado: si el nodo posee una falla permanente, este es automáticamente desconectado del bus y ya no puede participar de la comunicación. Solo puede volver al estado activo si es reinicializado.

3. Sistema desarrollado

Con el fin de caracterizar el comportamiento dinámico en sistemas satelitales y vehículos lanzadores se requiere la utilización de instrumentos como por ejemplo: giróscopos optoelectrónicos (IFOG, Interferometer Fiber Optic Gyroscope), sistemas de posicionamiento global (GPS, Global Position System), sensores de stress mecánico, entre otros. Por lo general, los dispositivos poseen un microprocesador (por ej DSP, Digital Signal Processor) que realiza un procesamiento de las señales con el objetivo de enviarlas al bus de comunicaciones.

Al implementarse un protocolo como CAN, puede notarse un alivio del trabajo que debe realizar el procesador del nodo, ya que luego de la configuración del controlador CAN, el procesador del nodo mayormente realiza tareas relacionadas al desempeño funcional del sensor mientras que el controlador CAN se encarga de resolver el protocolo y de realizar casi todas las transacciones necesarias para la comunicación. La interacción entre el controlador y el procesador ocurre en el proceso de configuración y en la recepción y transmisión de datos. Sin embargo puede conseguirse una mayor independencia entre procesador y controlador con protocolos como MIL-STD-1553, cuyos controladores son más poderosos debido a que en forma autómata pueden controlar los datos sin intervención del procesador, pero asumiendo el costo antes mencionado.

El prototipo desarrollado está conformado por seis nodos con las siguientes características: tres unidades que se han construido y armado completamente, dos unidades están integradas en una interfaz CAN-PCI [11] y otra en una interfaz CAN-USB [12]. En la Figura 3 puede verse el conexionado de los nodos a la red y los elementos que posee cada estación.

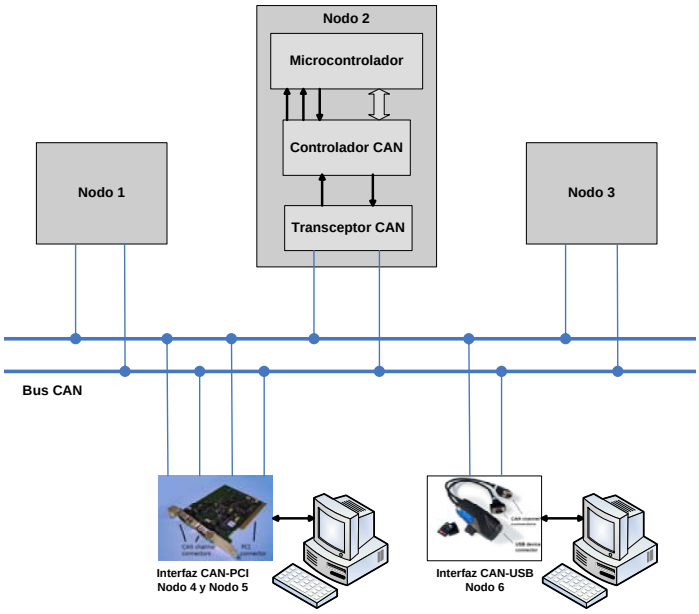

Figura 3 – Diagrama del sistema desarrollado

Las tres placas de circuito impreso (PCB, Printed Circuit Board) fueron desarrolladas por medio del software de diseño Altium Designer 6 [10] (teniendo en cuenta los siguientes integrados: microcontrolador AT89S52 [7], controlador CAN SJ1000 [8] y Transceptor CAN PCA82C250 [9]. Cada placa cuenta con un bloque ISP (In-System Programmable) que permite la conexión de una PC a la placa de desarrollo para efectuar la reprogramación del microcontrolador sin necesidad de removerlo del hardware de aplicación.

Con los microcontroladores se realizó la configuración de los nodos utilizando el entorno de desarrollo de programación KEIL IDE [13] (básicamente ANSI C). La comunicación de estos nodos se efectúa a la tasa de transferencia máxima (1 Mbits/seg) que fija la norma.

Pruebas realizadas

Las primeras pruebas consistieron en programar los nodos de tal forma que envíen mensajes y la interfaz CAN-PCI los reciba correctamente. Como esta interfaz cuenta con librerías de programación orientada a objetos se desarrolló una aplicación de alto nivel para poder analizar los mensajes recibidos. Luego con la misma interfaz se enviaron mensajes a los nodos, y para verificar la llegada de estos en forma correcta se agregaron indicadores luminosos en la placa de desarrollo (protoboard) donde estaba el nodo previamente.

Luego de verificar el funcionamiento y configuración de los nodos se diseñaron los PCB y se construyeron los nodos para formar la red definitiva.

En la Figura 4 se muestra una de las placas conectada a las interfaces CAN-PCI y CAN-USB.

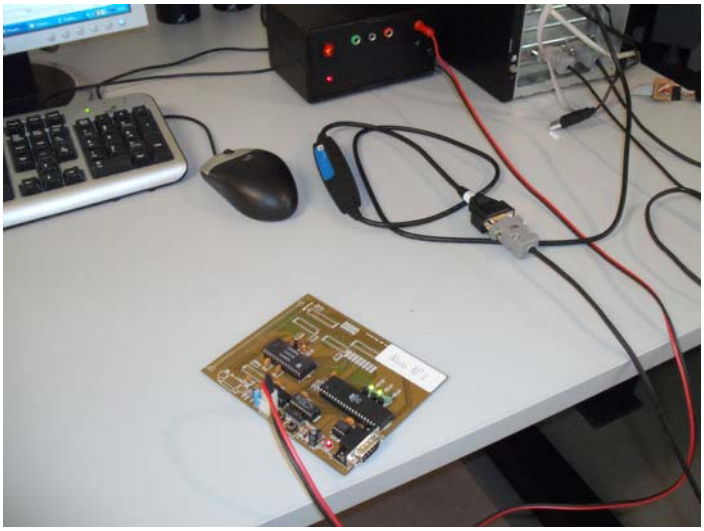

Figura 4 – Sistema desarrollado

Para monitorear las tramas se utilizó la interfaz CAN-USB ya que posee su propio software, una aplicación de alto nivel llamada CANKing (basado en el protocolo de capa de aplicación Can Kingdom [17]), que brinda un entorno de desarrollo interactivo para modificar y analizar los mensajes que circulan por la red.

Con la interfaz CAN-PCI se desarrolló una aplicación de alto nivel para monitorear los datos y realizar un escaneo de puertos. Además, se realizaron distintas pruebas funcionales como generar colisiones y errores para que determinados nodos lleguen al estado de pasivo e inactivo. También en esta interfaz comenzó a implementarse un protocolo de capa de aplicación.

4. Conclusiones

El prototipo se encuentra en la etapa de pruebas funcionales. Para configurar el controlador fue de gran utilidad elegir un procesador de nodo (microcontrolador) que tenga un puerto compatible con el funcionamiento del bus de datos y direcciones del controlador CAN. Para verificar y depurar el código procesado por el nodo, en tiempo real, es importante contar con una interfaz con la cual pueda validarse el agregado de nodos, distintos tipos de mensajes, estados de error, entre otros. Además al diseñar la placa con indicadores (banderas) luminosos se cuenta con otro elemento para verificar el procesamiento del código.

Bibliografía

1. William Stalling. Comunicaciones y Redes de Computadores. 6ª Edición. 2000.
2. Robert Bosch GmbH. CAN Specification 2.0. 1991
3. Robert Bosch GmbH. Controller Area Network – A Serial Bus System – Not Just For Vehicles.
4. Condor Engineering, Inc. MIL-STD-1553 Protocol Tutorial. 2004.
5. ISO 11898: Road Vehicles – Interchange of digital information – Controller Area Network (CAN) for high speed communication. 1993.
6. Phillips Semiconductors. PCA82C250/251 CAN Transceiver. Application Note. AN96116. 1996.
7. AT89S52. 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. 2008.
8. AN97076. SJA1000 Stand-alone CAN Controller. 1997.
9. AN96116. PCA82C250/251 CAN Transceiver. 1996.
10. Altium Designer 6 User's Guide. 2006.
11. Kvaser PCIcan Hardware Reference Manual. 2002.
12. Kvaser Leaf User's Guide. 2006.
13. Getting Started and Creating Applications with µVision2 and the C51 Microcontroller Development Tools, User's Guide, Keil Software. 2000.
14. Texas Instruments Application Report "422 and 485 Standards Overview and System Configurations". 2002.

15. Sitio sobre SpaceWire de la Agencia Espacial Europea (ESA)
http://spacewire.esa.int
16. An Overview of the IEEE-1355 Standard. Paul Walker, 1355 Association.
17. A CanKingdom Rev 3.01 by Lars-Berno Fredriksson. Kvaser. 1995.