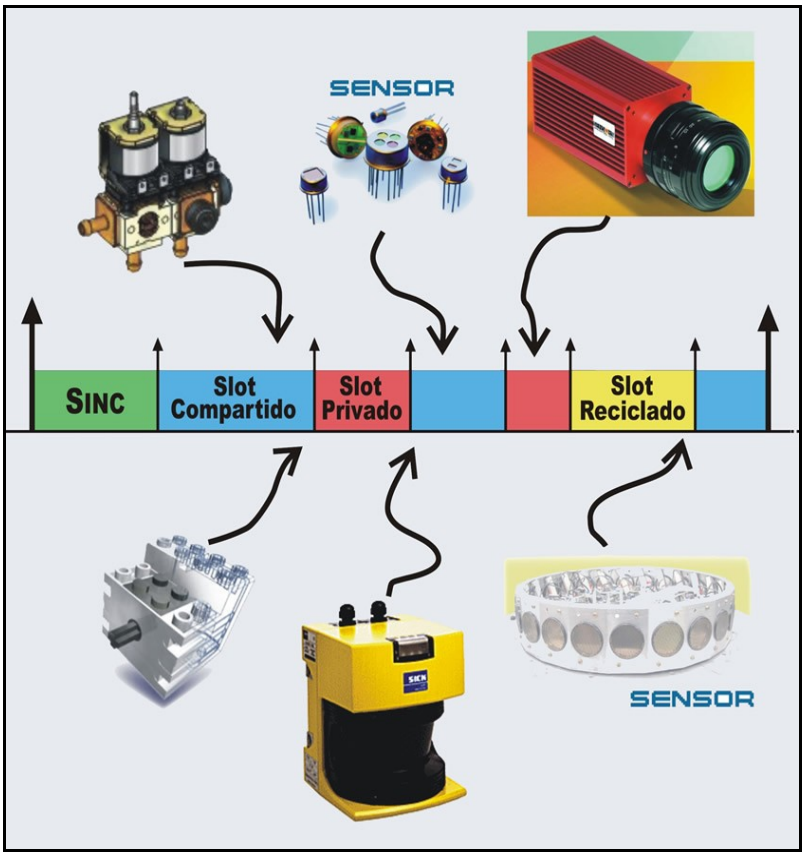
SCoCAN, UN PROTOCOLO FLEXIBLE BASADO EN CAN



- Javier O. Coronel P., M. Sc.
  Aspirante a Ph. D.
  Automática e informática Industrial.
  Universidad Politécnica de Valencia. España.

- Pascual Pérez, Ph. D.
  Profesor en la Escuela de Informática Aplicada.
  Universidad Politécnica de Valencia. España.

- Francisco Blanes, Ph. D.
  Profesor en la Escuela de Informática Aplicada.
  Universidad Politécnica de Valencia. España.

- Gines Benet, Ph.D.
  Profesor en la Escuela de Informática Aplicada.
  Universidad Politécnica de Valencia. España.

- José Simó, Ph. D.
  Profesor en la Escuela de Informática Aplicada.
  Universidad Politécnica de Valencia. España.

Grupo de Investigación de sistemas de tiempo real y robótica Industrial.
Universidad Politécnica de Valencia. España.

RESUMEN

Este artículo describe un protocolo de comunicaciones orientado a tiempos, denominado SCoCAN (Canales compartidos sobre CAN). Este protocolo se basa en un esquema de comunicación híbrido combinando tráfico *time triggered* y *event triggered* y manejado por *slots* de tiempo dedicados a cada uno. Fundamentalmente, SCoCAN es utilizado como infraestructura de comunicaciones para aplicaciones distribuidas de control con restricciones temporales.

PALABRAS CLAVES

Bus de campo, protocolo de comunicaciones CAN, sistemas distribuidos de tiempo real.

ABSTRACT

This paper describes a time-oriented communication protocol, called SCoCAN (Shared Channels on CAN). This protocol is based on a hybrid communication scheme combining time triggered (TT) and event triggered (ET) traffic with temporal isolation and they are handled by time slot dedicated to each one. Fundamentally, SCoCAN is intended as communication infrastructure for distributed control applications with timing constraints.

KEYWORDS

Field bus, CAN communication protocol, real time distributed system.

1. INTRODUCCIÓN

Los sistemas empotrados distribuidos (DES) son cada vez más utilizados en arquitecturas complejas con un alto grado de

autonomía. Es común encontrarlos en controladores de vuelo, automóviles, robots, circuitos de vigilancia, en control industrial, etc., en donde diferentes sensores, actuadores y dispositivos de control utilizan sistemas de comunicación basados en buses de campo.

Uno de los requerimientos típicos en sistemas de buses de campo es la capacidad de soportar servicios de comunicación *Time-Triggered (TT) y Event-Triggered (ET)* bajo restricciones temporales. Consecuentemente, un esquema apropiado es la combinación de ambos servicios, tratando de compartir sus respectivas ventajas. Este método es usado en el protocolo SCoCAN. Algunas discusiones sobre el uso de estos paradigmas de comunicación son encontradas en [1] y [9].

Aunque hay una gran variedad de buses de campo de tiempo real, CAN[2] (Controler Area Network) es una de las soluciones preferidas para comunicar DES en espacios reducidos. Pero el protocolo nativo de CAN no garantiza un *jitter* mínimo de comunicación ni el momento exacto de transmisión, debido a la alta variabilidad en los tiempos de respuesta de los mensajes CAN producida por condiciones de error en el canal, así como por sobre carga de tráfico en el bus [14]. Esto puede producir un incumplimiento en las restricciones temporales de algunas aplicaciones de control. Como es demostrado en [11], el *jitter* tiene un impacto negativo en la mayoría de los sistemas distribuidos de control y es la principal motivación para introducir modificaciones en el protocolo original de CAN. Esto ha generado el surgimiento de nuevas extensiones de CAN, tales como TTCAN[6], FTTCAN[1] y SCoCAN [3,4] (en nuestro caso), entre otros.

En este artículo se propone un protocolo de comunicaciones denominado "*SCoCAN - Canales Compartidos sobre CAN*", describiendo sus principales características de funcionamiento e implementación. Adicionalmente, se presenta una comparación con otros protocolos existentes y se plantea un caso de estudio para análisis de rendimiento del protocolo.

2. PROTOCOLO DE COMUNICACIONES SCoCAN

Un primer enfoque del protocolo de comunicaciones SCoCAN fue presentado en [4][5]. SCoCAN (*Shared Channels on CAN* – Canales Compartidos sobre CAN) es un protocolo de alto nivel por encima de la capa de enlace de datos de CAN, y que utiliza un esquema de comunicaciones híbrido, combinando tráfico *Time Triggered* (TT) y *Event Triggered* (ET), pero separados temporalmente, es decir, con asignación exclusiva de ancho de banda para cada tipo de tráfico.

El objetivo principal de SCoCAN es eliminar o reducir el *jitter* de comunicación y al mismo tiempo, aprovechar al máximo el ancho de banda físico del bus. Lo primero es logrado por una secuencia sucesiva de *slots* de tiempo que le proporcionan determinismo al bus de comunicaciones; y lo último es conseguido por la recuperación dinámica de ancho de banda por medio del reciclaje de *slots* inutilizados.

La nomenclatura usada en nuestro protocolo es la siguiente: en SCoCAN el tiempo de bus esta dividido en Ciclos Básicos (BC's) de duración fija. Todos los nodos están sincronizados por un mensaje especial llamado "*Mensaje de Sincronismo*" (SM), el cual es generado por un nodo master. Este SM marca el comienzo de cada BC. Dentro de cada BC, SCoCAN define 3 tipos de *slots*: *Privados*, *Compartidos* y *Reciclados* (opcional). Estas características serán explicadas en las siguientes secciones.

2.1 CICLO BÁSICO SCoCAN Y SUS SLOTS DE TIEMPO

El tiempo entre dos tramas de sincronización constituye un *Ciclo Básico* (BC) (ver figura 1). Este está formado por un conjunto sucesivo de *slots* de tiempo de diferente duración y cuya longitud dependerá de las características de los datos transmitidos. El tamaño del *BC* determinará la resolución temporal del sistema de comunicación. Por consiguiente, los períodos de transmisión del tráfico *TT* corresponderán a múltiplos enteros del *BC.*

En la red SCoCAN, todos los mensajes de comunicación utilizarán el formato de datos estándar de CAN [2] y además, una tabla de identificadores CAN deberá ser definida para cada mensaje (dependiendo del tipo de tráfico y la prioridad). Por otro lado, los nodos que integran la red, emplearán una tabla de tiempos de transmisión para definir el *BC* del protocolo, la cual es previamente definida y guardada en cada nodo durante la rutina de arranque. Adicionalmente, varios modos de operación (tablas de transmisión) pueden ser predefinidos o dinámicamente distribuidos sobre la red.

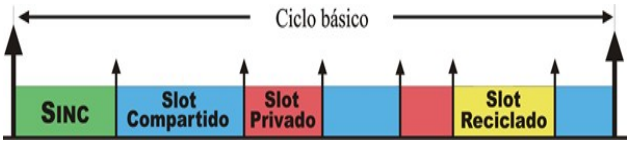


Figura 1: *Mensaje de Sincronización y slots de tiempo de un ciclo básico SCoCAN.*

2.1.1 Mensaje de Sincronización

La sincronización de los nodos en la red se hace por medio de un *Mensaje de Sincronismo (SM)*, el cual es generado por nodos maestros de sincronización primarios o secundarios. Todos los nodos de la red SCoCAN identifican el mensaje de la red por su identificador (generalmente este mensaje es el de mayor prioridad). La recepción del SM causa una nueva reconfiguración de los temporaizadores locales y restaura el tiempo de ciclo en cada nodo. Dependiendo de la implementación, la precisión del proceso de sincronización dependerá exclusivamente de la propagación física de la señal sobre el bus y del tiempo de procesamiento del mensaje en cada nodo, generándose por consiguiente un pequeño intervalo de incertidumbre (ver figura 2) que debe ser tenido en cuenta en la ejecución del protocolo.

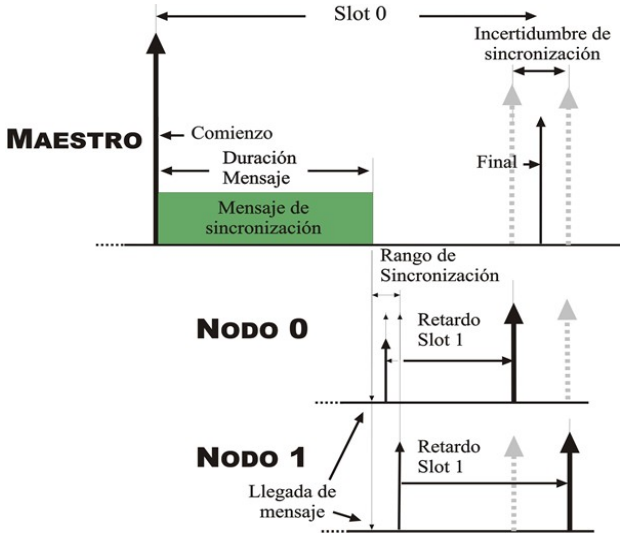


Figura 2: *Posibles retardos durante la sincronización*.

2.1.2 Slots Privados

Estos *slots* son utilizados por mensajes con restricciones de tiempo real, mensajes de sincronización ó mensajes de configuración (es decir, tráfico *time triggered*). En estos *slots*, únicamente uno de los nodos podrá transmitir datos (el nodo propietario), evitando así probables colisiones al acceder al bus. De esta manera se eliminará ó minimizará el *jitter* de comunicaciones. Para permitir una comunicación fiable, la retransmisión automática por error (característica del protocolo original CAN) es habilitada en estos *slots*.

2.1.3 Slots Compartidos

Estos *slots* son utilizados por mensajes con temporización no crítica (restricciones flexibles de tiempo real), alarmas o mensajes con grandes bloques de datos (es decir, tráfico *event triggered*). Estos *slots* no tienen propietario, por lo que los nodos utilizan el mecanismo de arbitraje del protocolo original CAN basado en prioridades y consecuentemente, hereda su eficiencia en el manejo de tráfico *event triggered*. En estos *slots* también es permitida la retransmisión automática causada por errores en la comunicación. Pero, para mantener una separación temporal estricta entre ambos tipos de tráfico (TT y ET) y proteger los *slots privados* de interferencia, se agrega un *tiempo idle* al final de cada *slot compartido* (ver figura 3). Todas las actividades de transmisión son suspendidas en este *tiempo idle*, incluyendo las retransmisiones y los nodos con peticiones de transmisión pendientes, las cuales serán conservadas hasta el próximo *slot compartido.*

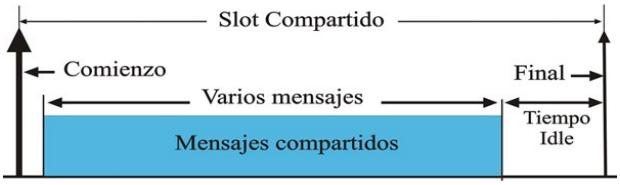


Figura 3: *Slot Compartido (Shared Slot)*

2.1.4 Slots Reciclados

Una característica importante de nuestro protocolo, es que cuando no hay datos para transmitir en los *slots privados* (para tráfico TT) estos pueden ser transformados en *slots compartidos* (para tráfico ET). Esta transformación es dinámica (durante la ejecución) proveyendo una recuperación dinámica de ancho de banda y evitando un desperdicio de recursos de bus.

Si durante un intervalo de tiempo (*tiempo de reciclaje*), después del comienzo de un s*lot privado*, se detecta inactividad sobre el bus, otro nodo con tráfico ET en cola puede comenzar a transmitir sus mensajes (ver figura 4). Dependiendo del tipo nodo, esta inactividad puede ser descubierta directamente por el muestreo del bus CAN ó por la recepción de un mensaje especial (llamado *sync1*).

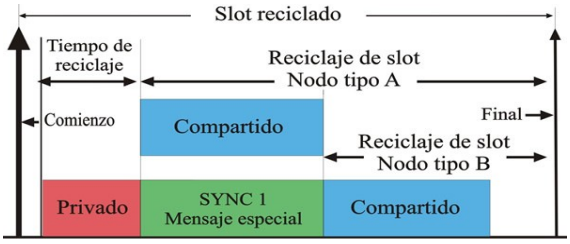


Figura 4: *Slot Privado transformado en slot reciclado*.

2.3 TIPOS DE NODOS

Nodo Maestro primario: es el encargado de sincronizar todos los nodos de la red con alta precisión. Este nodo envía mensajes SM para coordinar la máquina de estados de todos los nodos de la red y de esta forma, sincronizar sus tablas de tiempo de transmisión. Además, este nodo debe tener la capacidad de detectar inactividad vía hardware en los *slots privados* y debe indicar la condición de no ocupado enviando un mensaje particular (*sync1*).

Nodo Maestro secundario: un conjunto de nodos maestros secundarios son definidos sobre la red, los cuales monitorizan constantemente la actividad del nodo maestro primario. Sí el nodo primario falla, algún nodo secundario se encarga inmediatamente del proceso de sincronización. Para la selección del nodo secundario se puede usar un criterio similar al descrito en [10].

Nodo tipo A: nodo con tarjeta inteligente, capaz de detectar inactividad vía hardware ó por el acceso a registros de control. Esta característica le permite a estos nodos transmitir rápidamente cuando hay inactividad en los *slots privados.*

Nodo tipo B: Nodo sin capacidad de muestrear el bus. Este tipo de nodo no es capaz de reciclar los *slots privados* vacíos a menos que otro nodo especializado (nodo maestro) envíe un mensaje particular (*sync1*), que le señale la condición de inactividad.

3. OTROS PROTOCOLOS DE COMUNICACIÓN

Como se discute en [1] es conveniente soportar ambos tipos de tráfico, *TT* y *ET*. Sin embargo, existen protocolos de bus de campo que no soportan ambos tipos de tráfico (ej. TTP/C [8]), o que soportan ambos tipos de tráfico pero no tienen una separación temporal (ej. DeviceNet[5], ProfiBus[7], P-Net[7]). Y en otros casos donde la separación temporal es usada, pero el tráfico *ET* es manejado ineficientemente (ej. WorldFIP[7], Foundation Fielbus-H1[7]). Por consiguiente, únicamente los protocolos que cumplen con estas características serán nombrados.

Flexible Time Triggered CAN (FTTCAN) [1] esta extensión de CAN se basa en una planificación dinámica TDMA. FTTCAN tiene un ciclo básico dividido en dos ventanas, una asíncrona y otra síncrona. En esta ventana el tráfico es planificado dinámicamente por un nodo master central. *Time Triggered CAN (TTCAN)* [6] es otra extensión de CAN, basada en una planificación estática TDMA. TTCAN usa un mensaje de referencia para cada ciclo básico. Este último esta dividido en tres tipos de ventanas: privadas, arbitrarias y libres. El patrón completo del tráfico TTCAN esta formado por ciclos básicos consecutivos que forman una matriz.

Una de las principales ventajas de SCoCAN es su fácil implementación (no requiere un planificador maestro como en FTTCAN) y su planificación abreviada. Aunque SCoCAN tenga una planificación estática, éste proporciona cierta flexibilidad en los BC, ya que permite la transformación de *slots privados* a *slots compartidos* (*slots reciclados)*, lo que simplifica la planificación. Por otro lado, si un *slot privado* es asignado a un nodo y por alguna razón ese nodo es apagado (daños, mantenimiento, etc.), en TTCAN esto causaría un desperdicio de ancho de banda del bus, ya que la ventana privada permanecería asignada pero inutilizada. En el caso de SCoCAN, este *slot privado* puede ser reciclado para tráfico ET (*slot reciclado*), recobrando ancho de banda.

4. CASO DE ESTUDIO

El protocolo SCoCAN se ha implementado en el robot móvil autónomo YAIR, cuya arquitectura distribuida, trama de mensajes y características temporales de sensores y actuadores son definidas en [3] y [12]. Para validar el rendimiento del protocolo se han diseñado varias pruebas de campo y se han implementado herramientas de análisis y adquisición de datos, las cuales serán descritas más adelante.

4.1 CARACTERÍSTICAS BÁSICAS DE LA PRUEBA

El robot YAIR [13] es un sistema distribuido con nodos inteligentes empotrados, los cuales gestionan diferentes subsistemas, tales como sensores, actuadores y dispositivos de control. Adicionalmente, cada nodo implementa el protocolo

de comunicaciones SCoCAN. Los nodos implicados en las pruebas son: infrarrojos, motores, odometría, ultrasonidos y nodo central. En [3] y [13] se hace una descripción más detallada de cada nodo.

Teniendo en cuenta la arquitectura del robot YAIR y las características temporales de los dispositivos del sistema distribuido [3], se han definido algunos parámetros básicos para las pruebas: primero, un *ciclo básico* de 10*ms* y dividido en 20 *slots* de 500µs cada uno. Y segundo, características de transmisión de los mensajes CAN tales como identificador CAN, tamaño de datos, asignación de *slots* en el BC, tipo de *slot*, periodicidad y número de mensajes CAN, los cuales están definidos en la tabla 1. Adicionalmente, otras características son definidas: el *tiempo idle* usado es de 10µs, el *tiempo de reciclaje* es de 20 µs y la velocidad de transmisión establecida es de 1Mbps.

Tabla 1. *Características de los mensajes usados en las pruebas SCoCAN en el robot YAIR.*

Especificación / Mensaje(msj)	Tama. dato (byte)	Período de generación	Nº de msj.	Slot SCoCAN	Tipo de mensaje	Identificador
Infrarojos	8	20 ms (anillo)	4	10,19, 13,16	Privado	0x100 – 0x103
Motor-Alarma	1	Por evento	1	-	Compartido	0x010
Motor-Modo de control	1	Generado por PC	1	-	Compartido	0x300
Motor-odometría	8	10 ms	1	4	Privado	0x10A
Motor-Velocidad	4	Generado por PC	1	7	Privado	0x10B
Extra odometría	8	10 ms	1	3	Privado	0x10C
Ultra sonidos	8	200µs/ 80ms	32	-	Compartido	0x310 - 0x318

4.2 MODOS DE TRABAJO USADOS EN LAS PRUEBAS

Dos modos de trabajo del protocolo SCoCAN han sido definidos en el robot YAIR:

- SCoCANv1: únicamente un mensaje puede ser transportado en los *slots privados*. La retransmisión por error es deshabilitada para permitir una replicación física de buses.
- SCoCANv2: la retransmisión automática por error es habilitada en todos los *slots*. Además, varios mensajes compartidos (mensajes ET) pueden ser transportados al final de cada *slot privado* aún cuando un mensaje TT sea transmitido. Sin embargo, esta transmisión de mensajes ET es posible únicamente cuando no hay errores en la transmisión del mensaje TT previo.

En las dos pruebas, las demás características fundamentales del protocolo SCoCAN (tal como el reciclaje de *slots*) son implementadas.

4.3 RESULTADOS DE LAS PRUEBAS SCoCAN

El tiempo de simulación fue de 100 *ciclos básicos*. En la prueba SCoCANv1 (tabla 2) el factor de utilización (UF) máximo del bus CAN fue de 64.23% frente al 82.56% de utilización de la prueba SCoCANv2, por otro lado, el UF mínimo para la primera prueba fue de 53.83% mientras que para la segunda prueba fue del 59.03%. Adicionalmente, en las dos pruebas el 35.7% de los *slots privados* ha sido reciclado debido a inactividad. Fundamentalmente, la diferencia de resultados se debe a la capacidad de reutilización de ancho de banda al final de los *slots privados* en cada prueba.

Tabla 2: *Características principales de las pruebas*

Características	***SCoCANv1***		***SCoCANv2***			
Tiempo simulado	100 ciclos básicos		100 ciclos básicos			
Máximo UF/BC	64.23%		82.56%			
Mínimo UF/BC	53.83%		59.03%			
Promedio UF/sec	58.38%		66.85%			
Slots privados con tiempo de desviación en la TX <|x| µs	Deviation <| 6us|		Deviation <| 4us|		Deviation <| 2us|	
	100%	100%	98.1%	96.1%	71%	77.7%
Slot reciclados	250 / 700 privados		250 / 700 privados			

En la figura 5, la evolución temporal del factor de utilización es mostrada. En esta figura podemos distinguir algunos picos de transmisión, los cuales se corresponden con el comienzo de la transferencia de ficheros de datos junto con la transmisión de datos de ultrasonidos y otros mensajes TT.

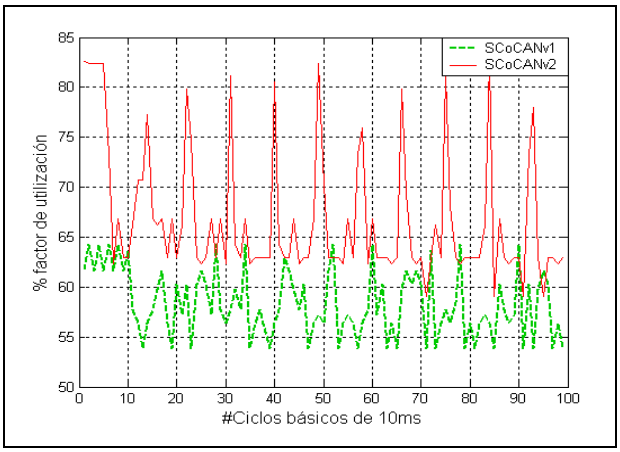


Figura 5. *Factor de utilización de CAN usando el protocolo SCoCAN en las pruebas realizadas.*

En ambas pruebas, el tiempo de desviación máximo de transmisión producido en mensajes asignados a *slots privados,* siempre fue menor de 6µs (este tiempo incluye incertidumbre de sincronización y derivas por reloj). Asimismo, en más del 70% de los *slots privados* el tiempo de desviación fue menor de 2µs.

5. CONCLUSIONES

En este artículo hemos presentado un nuevo protocolo de comunicaciones, *SCoCAN*, el cual es propuesto como infraestructura de comunicaciones para aplicaciones distribuidas de control. La característica principal de este protocolo es que soporta la combinación de tráfico TT y ET, con separación temporal entre ellos. Y adicionalmente, en los *slots privados* se ha eliminado el *jitter* de comunicación de los mensajes TT y se ha mejorado la gestión del ancho de banda de red (BW). Esto último se debe principalmente a la recuperación dinámica de *slots privados* inutilizados. El protocolo ha sido implementado y analizado sobre el robot móvil YAIR. Y según las expectativas esperadas, el análisis de los mensajes transmitidos presenta una respuesta favorable.

6. AGRADECIMIENTOS

Este trabajo se ha desarrollado con la ayuda de fondos FEDER europeos y de la Comisión Interministerial de Ciencia y Tecnología de España (CICYT). Referencia: DPI2002-04434-C04-03.

7. REFERENCIAS

[1] Almeida, L., Pedreiras, P., Fonseca, J. A. - "The FTT-CAN Protocol: Why and How", IEEE Transactions on Industrial Electronics. Volume 49, Number 6. December. 2002.

[2] CiA (CAN in Automation). “CAN Application Layer for Industrial Applications”, Documents No.: DS-201...DS-207, Version 1.1. 1996.

[3] Coronel, J.O, Blanes, F., Pérez, P., Benet,G., Simó, J.E., “Arquitectura de Control distribuida usando nodos empotrados con RT-LINUX sobre le protocolo de comunicaciones SCoCAN”, XXV Jornadas de Automática. Alicante (España). Sep. 2004

[4] Coronel, J.O, Blanes, F., Benet, G., Pérez, P., Simó, J.E., “CAN-based Distributed Control Architecture using the SCoCAN Communication Protocol”, Proc. IEEE Int'l Conf. on Emerging Technologies and Factory Automation, ETFA'2005, vol 1.

[5] DeviceNet specification— release 2.0, ODVA—Open DeviceNet Vendor Assoc., Vol. I and II, 1997

[6] Fuhrer, T., Muller, B., Dieterle, W., Hugel, R. “Time Triggerd Communication on CAN”. Cia CAN. 7th international CAN Conference 2000.

[7] General Purpose Fieldbus: Vol. 1: P-Net; Vol. 2: PROFIBUS; Vol. 3:WorldFIP, Amend.1: Foundation Fieldbus’H1, European Standard EN50170, 2000.

[8] H. Kopetz and G. Grünsteidl, “TTP—A protocol for fault-tolerant real-time systems,” IEEE Computer, vol. 27, pp., Jan. 1994.

[9] H. Kopetz, “Real-Time Systems: Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers, 1997.

[10] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. J. ACM, 32(4):841-860

[11] Pérez, P., Benet, G., Blanes, F., Simó J. E. “Communications jitter influences on Control loops using Protocols for Distributed Real-Time Systems on CAN bus”. 5th IFAC International Symposium. SICICA’03. Aveiro (Portugal). 2003

[12] Pérez, P., Posadas, J.L., Benet, G., Blanes, F., Simó, J.E., “An Intelligent Sensor Architecture for Mobile Robots” 11th International Conference on Advanced Robotics - IEEE ICAR’03. University of Coimbra (Portugal). 2003

[13] Simó, J. et al. “Behaviour Selection in the YAIR Architecture”. In proceedings of IFAC. Vilamoura, Portugal. AARTC'97. 1997.

[14] Tindell, K., Burns, A. y Wellings, A. J., “Calculating Controller Area Network (CAN) message response time”, Control Engineering Practice, Vol. 3(8). 1995.

AUTORES

Javier O. Coronel P., MsC en redes corporativas e integración de sistemas.
Aspirante a Dr. en Automática e Informática Industrial.
Universidad Politécnica de Valencia. España.

Pascual Pérez, Dr. en Informática.
Profesor de la Universidad Politécnica de Valencia. Dpto. Informática de Sistemas y Computadores. España.

Francisco Blanes, Dr. en Informática.
Profesor de la Universidad Politécnica de Valencia. Dpto. Informática de Sistemas y Computadores. España.

Gines Benet, Dr. Ingeniero Industrial.
Profesor de la Universidad Politécnica de Valencia. Dpto. Informática de Sistemas y Computadores. España.

José Simó, Dr. Ingeniero Industrial.
Profesor de la Universidad Politécnica de Valencia. Dpto. Informática de Sistemas y Computadores. España.