UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"DESARROLLO DE UN SISTEMA EDUCATIVO PARA LA ENSEÑANZA DEL PROTOCOLO DE COMUNICACIONES CAN"

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA

PRESENTA

CARLOS ANTONIO CHAMÚ MORALES

DIRECTOR DE TESIS
ING. HERIBERTO ILDEFONSO HERNÁNDEZ MARTÍNEZ

HUAJUAPAN DE LEÓN, OAX.; ABRIL DE 2005

Tesis presentada el 15 de abril de 2005
ante los siguientes sinodales:

M.C. José A. Moreno Espinosa
M.C. Fermín Hugo Ramírez Leyva
M.C. Enrique Guzmán Ramírez

Director de Tesis:

C. Dr. Heriberto I. Hernández Martínez

Dedicatoria

Con mucho cariño y amor a mis padres Francisco y Cibeles, y a mis hermanos Paco y Cibe.

Carlos.

Agradecimientos

Agradezco especialmente al Profr. Heriberto I. Hernández Martínez, por compartirme sus conocimientos y experiencias, por su motivación y entusiasmo dedicados a dirigir mi trabajo, y por brindarme su confianza y su excelente amistad.

A mis padres Francisco y Cibeles, quienes me formaron y prepararon para la vida, y por todo el esfuerzo y dedicación que realizaron para brindarme una educación profesional.

A Paco y Cibeles (Ing. Beba), por su hermandad y amistad, por darme buenos consejos y ejemplos, así como por todos los momentos compartidos.

A las familias Morales Vicente y Vilchis Morales, quienes me abrieron las puertas de su hogar en los tiempos de estudiante y apoyaron incondicionalmente.

A mis primos Diana, Rubén, Edgard, Víctor Hugo y Viridiana por todo el apoyo moral que me brindaron durante esta etapa de mi vida.

A los profesores José Antonio Moreno Espinosa, Enrique Guzmán Ramírez y Fermín Hugo Ramírez Leyva, cuyas sugerencias y observaciones contribuyeron a mejorar el documento de tesis.

A mis amigos: Pablo Acevedo Cuéllar, por permitirse adoptar como un hermano; Teté Palacios Díaz, por su amistad y apoyo incondicional en todo momento; Josué N. García Matías (Ing. Boti) por su valiosa ayuda y por pasarme la estafeta; e Isaac García López, por ilustrarme con sus ideas y contribuciones.

A la Universidad Tecnológica de la Mixteca, por permitirme desarrollar la presente investigación dentro de sus instalaciones.

Carlos.

Índice

Lista de tablas

Lista de figuras

Resumen

El diseño de un sistema empotrado (*embedded system*) implica que tanto el software (SW) como el hardware (HW) se diseñen en paralelo mediante una metodología de desarrollo.

El Sistema Educativo para la Enseñanza del Protocolo de comunicaciones CAN (SeeCAN) tiene como principal objetivo ser una herramienta de apoyo didáctico en la enseñanza de dicho protocolo. El SeeCAN se diseñó mediante una metodología de desarrollo de sistemas empotrados.

Como resultado de este trabajo, se obtuvo un sistema de gran trascendencia para la enseñanza del protocolo CAN. Cabe mencionar que los sistemas de entrenamiento (*starter kit*) comerciales no están enfocados a la enseñanza general del protocolo y presentan costos elevados.

Se define cada una de las fases de la metodología del desarrollo de sistemas empotrados, y con base a ésta se describe el desarrollo del SeeCAN.

Abstract

The design of an embedded system is usually carried out following a development methodology in which the software (SW) and the hardware (HW) are designed in parallel. As a training system of the CAN protocol, the SeeCAN becomes a powerful educational tool that is directly included in an embedded system-type with a specific development methodology.

This work presents a flexible and configurable training system for the CAN protocol with emphasis on the teaching of industrial protocols. Similar systems are available in the market with distinct applications but with the disadvantage of their high cost.

Hence, this thesis has as a main objective to development every phase of the design in the embedded system and it also describes the CAN protocol in detail. It concludes that the applied SeeCAN can be simpler and more efficient in educational applications than those protocols accessible in the market.

Introducción

Los buses de campo (FBs, *Fieldbus*) son sistemas de transferencia de información en serie destinados a aplicaciones de tiempo real distribuidas, sistemas de automatización, sistemas de supervisión y control en el ámbito de celdas de producción [13]. Los FBs interconectan dispositivos electrónicos, tales como sensores, actuadores y unidades de control electrónico (ECU, *Electronic Control Unit*), que gobiernan desde complicados procesos industriales hasta simples procesos en el hogar [26, 27, 31].

El surgimiento de los FBs fue motivado por una serie de necesidades, entre ellas [28, 44]:

- Reducir el cableado en las instalaciones.
- Lograr una mayor integración de los datos respecto a la planta en los sistemas de información de la empresa.
- Dotar de inteligencia a los sensores y actuadores.
- Desarrollar sistemas de control distribuido.
- Diseñar equipos de control con conexión normalizada, con la finalidad de establecer compatibilidad entre productos, equipos y dispositivos industriales de diferentes fabricantes.
- Obtener mayor seguridad en la transferencia de información.

Los sistemas distribuidos surgieron al incrementar la funcionalidad y complejidad de las unidades de control, lo cual motivó el desarrollo de nuevos sistemas de comunicaciones para interfazar dichas unidades. Además, se observó que la ubicación de los sensores y actuadores está determinada por la disposición física del proceso principal, y por lo tanto se requieren canales apropiados para una correcta comunicación [41].

La aparición del microprocesador, y su continua evolución en relación precio/prestaciones, ha conducido a una nueva generación de dispositivos inteligentes (*smart devices*) de control industrial. En la actualidad se cuenta con sensores y actuadores que tienen integrada su propia unidad de proceso, basada generalmente en un microcontrolador (MCU, *Microcontroller*) [45].

Dada la dispersión existente en una planta industrial respecto a sus unidades de proceso, se requiere inevitablemente un enlace entre ellas, lo cual se consigue mediante una red de comunicaciones. Una red orientada al enlace de dispositivos de control industrial se conoce como FB, sin embargo, estas redes no son de uso exclusivo de las plantas industriales, ya que también se implementan en entornos de menor amplitud como son los entornos automotrices.

Dichas tendencias implicaron un aumento en el número de canales de comunicación, que condujo a una implementación de varios canales en un sólo medio físico de comunicación, el cual se puede compartir mediante una multiplexación por división en el tiempo (TDM, *Time-Division Multiplexing*). Debido a los problemas resultantes de la utilización de un sólo canal físico, se debe defi-

nir cada cuándo un nodo puede utilizar el medio físico, conocido como bus. Tal definición se especifica en el protocolo de comunicaciones; se ha especificado un gran número de protocolos de comunicaciones, los cuales se distinguen entre sí por sus características particulares.

Los protocolos de redes aplicados en automóviles deben satisfacer requerimientos únicos que no se presentan en otros protocolos de redes, dichos requerimientos incluyen detección de errores de alto nivel, tiempos de latencia bajos y flexibilidad en la configuración [13].

CAN (*Controller Area Network*) es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH, basado en una topología de bus para la transmisión de mensajes en ambientes distribuidos, además ofrece una solución a la gestión de la comunicación entre múltiples unidades centrales de proceso [27].

CAN ha sido ampliamente aceptado en aplicaciones de redes automotrices debido a su bajo costo, alto rendimiento y disponibilidad de diversas implementaciones del protocolo en circuito integrado [29, 32]. El protocolo CAN proporciona los siguientes beneficios:

- 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.
- El CPU anfitrión (*host*) delega la carga de comunicaciones a un periférico inteligente, por lo tanto el CPU anfitrión tiene más tiempo para ejecutar sus propias tareas.
- Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto.

Por otro lado, un sistema empotrado (*embedded system*) es un sistema basado generalmente en un MCU, que realiza una tarea específica mediante la correcta configuración/programación de sus recursos hardware (HW)/software (SW). El diseño de un sistema empotrado implica que tanto el HW como el SW se diseñen en paralelo mediante una metodología de desarrollo.

El presente trabajo de investigación propone el estudio del protocolo de comunicaciones CAN, cuya aplicación inicial fue la conexión de ECUs en los automóviles y que en la actualidad se extiende a la automatización de procesos, control de maquinaria y aplicaciones ferroviarias y marítimas.

Como antecedentes, se retoman los trabajos realizados en [10, 16, 17, 40] y la experiencia del director de tesis sobre el protocolo CAN [29, 30, 31, 32, 33, 34]. En los trabajos citados no se han considerado aspectos relevantes para el desarrollo de sistemas robustos, por ello este trabajo considera la metodología de desarrollo de sistemas empotrados propuesta en [4, 5].

Con base en lo anterior se propone el desarrollo de un Sistema Educativo para la Enseñanza del Protocolo de Comunicaciones CAN (SeeCAN) como herramienta propietaria de la Universidad Tecnológica de la Mixteca (UTM) para apoyo a las materias de redes de computadoras que se imparten en las carreras de Ingeniería en Electrónica, Ingeniería en Computación e Ingeniería Industrial, en los niveles de licenciatura y posgrado.

Planteamiento y objetivos de la tesis

El objetivo principal de este trabajo de tesis es profundizar en el conocimiento del protocolo de comunicaciones CAN mediante el diseño y elaboración de nodos de red con fines académicos.

Este trabajo forma parte de la línea de investigación del Cuerpo Académico de Redes de Instrumentación del Instituto de Electrónica y Computación (IEC) de la UTM, destinada al estudio y especificación de los protocolos de comunicaciones industriales.

Una vez planteado el objetivo principal y el ámbito donde se centra el trabajo de investigación, se proponen los siguientes objetivos secundarios:

- Elaborar un estado del arte del protocolo de comunicaciones CAN.

- Estudiar el protocolo de comunicaciones CAN con base a su arquitectura de protocolos establecida en su especificación [6] y estándares ISO [22, 23].
- Poner en funcionamiento el sistema de entrenamiento CAN de la firma esd GmbH [11, 12].
- Diseñar y construir nodos de red CAN que implementen, a nivel hardware, las características más importantes del protocolo considerando aspectos académicos.
- Realizar una red CAN con base a los sistemas desarrollados.

Estructura de la tesis

A continuación se detalla la estructura del documento de tesis (Figura I.1):

El capítulo 1 presenta el estado del arte del protocolo de comunicaciones CAN, en el que se incluye una reseña histórica del protocolo, la clasificación de las aplicaciones automotrices, las características del protocolo y sus principales aplicaciones.

El capítulo 2 describe la arquitectura del protocolo de comunicaciones CAN de acuerdo al modelo de referencia OSI.

El capítulo 3 presenta una clasificación de dispositivos CAN en cuanto a las principales características del protocolo.

El capítulo 4 describe la metodología de desarrollo de sistemas empotrados empleada, y expone las fases de desarrollo del SeeCAN.

El capítulo 5 presenta los resultados obtenidos.

El capítulo 6 plantea las conclusiones obtenidas y los trabajos futuros de investigación.

Por último se presentan las referencias bibliográficas utilizadas en el desarrollo de esta tesis y los anexos, referentes al manual de usuario y al diagrama esquemático del SeeCAN.

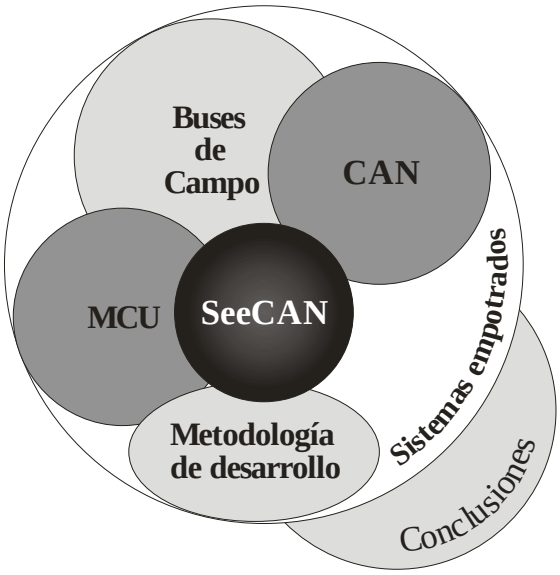


Figura I.1. Estructura del trabajo de tesis.

1. Estado del arte del protocolo de comunicaciones CAN

A inicios de la década de los ochentas, un grupo de ingenieros de la compañía alemana Robert Bosch GmbH [URL 2] estudiaron la posibilidad de aplicar sistemas de bus seriales dentro de los automóviles con la finalidad de satisfacer las demandas de la Sociedad de Ingenieros Automotrices (SAE, *Society of Automotive Engineers*). Como resultado se obtuvo la especificación del protocolo de comunicaciones CAN, del cual a continuación se presenta una reseña histórica y sus principales aplicaciones.

1.1. Reseña histórica del protocolo de comunicaciones CAN

En 1983, la compañía Robert Bosch GmbH inició un proyecto interno para desarrollar una red de interconexión entre los diversos dispositivos electrónicos instalados en los automóviles (*in-vehicle network*), con la finalidad de agregar mayor funcionalidad y reducir el cableado del sistema. Dicho proyecto fue dirigido por Siegfried Dais, Martin Litschel y el Dr. Uwe Kiencke[1]; como consultor intervino el Dr. Wolfhard Lawrenz[2], y como asistente académico el Dr. Horst Wettstein (Figura 1.1). En la investigación de los mecanismos de detección de errores intervinieron Wolfgang Borst, Wolfgang Botzenhard, Otto Kart, Helmut Schelling y Jan Unruh.

Durante la especificación del sistema de bus serial se agregó la empresa fabricante de automóviles Mercedes-Benz [URL 17] y el fabricante de semiconductores Intel Corp. [URL 10].

En febrero de 1986, Robert Bosch GmbH presentó oficialmente el sistema de bus serial CAN[3] en el congreso de la SAE celebrado en la ciudad de Detroit, E.U.A.[4]

Figura 1.1. Dr. Uwe Kiencke, Siegfried Dais y Dr. Wolfhard Lawrenz, creadores de CAN.

[1] En la actualidad, considerados los padres del protocolo CAN.
[2] El Dr. Wolfhard Lawrenz propuso el nombre de CAN (Controller Area Network).
3 Su forma original en el idioma inglés es "*Automotive Serial Controller Area Network*".
4 Estados Unidos de América.

Un año después, la firma Intel realizó la primera implementación física del protocolo en el controlador CAN (*CAN Controller*) 82526[5] y tiempo después la firma Philips [URL 19] introdujo el controlador CAN 82C200, ambos son la base del desarrollo de los controladores actuales 82527 de Intel y SJA1000 de Philips respectivamente, los cuales tienen diferencias significativas en cuanto al manejo de errores y filtrado de admisión. El controlador 82526/82527 implementa el principio *FullCAN* (apartado 3.3.2), cuya arquitectura requiere menor carga de procesamiento del MCU al que se encuentra conectado, a diferencia del principio *BasicCAN* implementado por Philips (apartado 3.3.1); además el principio *FullCAN* limita el número de mensajes que puede recibir, y el principio *BasicCAN* requiere menor superficie de silicio para su producción.

A pesar de que el bus CAN fue desarrollado para aplicaciones dentro de los automóviles, sus primeras aplicaciones, principalmente en Europa, se llevaron a cabo en sectores de mercado ajenos a la industria automotriz, por ejemplo:

- *Finlandia*: la empresa Kone implementó el bus CAN en el diseño y fabricación de elevadores [URL 12].
- *Suecia*: la empresa Kvaser [URL 13] sugirió el uso de CAN a los fabricantes de máquinas textiles Lindauer Dornier GmbH (Alemania) y Sulzer Corporation (Suiza) [URL 21], lo cual dio origen al grupo de usuarios textiles (*CAN Textile User's Group*) dirigido por Lars Berno Fredriksson[6].
- *Holanda*: la división de Sistemas Médicos de la firma Philips implementó una red interna basada en CAN para la comunicación de sus máquinas de rayos X, dando origen a la especificación de mensajes de Philips (PMS, *Philips Message Specification*)[7].
- *Alemania*: en la Universidad de Ciencias Aplicadas de Weingarten, el Dr. Konrad Etschberger, aplicando las mismas ideas de la PMS, desarrolló el protocolo conocido como STZP (*Steinbeis Transfer Center for Process Automation*).

A partir de las necesidades que representaban los diferentes sectores de aplicación, empezaron a surgir varios protocolos que cubrían la capa de aplicación. La mayoría de los pioneros del bus CAN utilizaban un enfoque monolítico, es decir, que las funciones de comunicación, administración de la red y código de aplicación, se realizaban en un mismo módulo de software y no consideraron las desventajas que surgían al desarrollar una solución propia. Lo anterior originó la necesidad de establecer un grupo encargado de reglamentar el desarrollo de aplicaciones del bus CAN.

La década de los noventas fue muy significativa para el protocolo de comunicaciones CAN, ya que en este periodo ocurrieron diversos acontecimientos importantes que fomentaron su desarrollo, difusión y aplicación. En 1991, Robert Bosch GmbH publicó la especificación CAN en su versión 2.0 (*CAN Specification 2.0*) [6], posteriormente dicha especificación fue sometida a su estandarización internacional, proceso que se vio afectado por disputas políticas con el protocolo francés VAN (*Vehicle Area Network*). En el mismo año, la firma Kvaser dio a conocer el protocolo de capa de aplicación CANKingdom [URL 4, URL 13].

En marzo de 1992, se estableció oficialmente la organización CiA (*CAN in Automation*) la cual es una organización no lucrativa formada por un grupo de fabricantes y usuarios internacionales que proporcionan información técnica, de productos y comercialización para fomentar la utilización del bus CAN [URL 3].

---

[5] Considerado el primer prototipo en circuito integrado del protocolo CAN.

[6] Los principios de comunicación, desarrollados por este grupo a inicios de 1989, dieron origen al protocolo de capa de aplicación CANKingdom en 1990 [URL 4].

[7] Considerada la primera capa de aplicación para redes CAN.

A pocas semanas de su fundación, CiA publicó un artículo técnico en donde recomendaba la utilización de transceptores (*transceivers*) en la capa física del protocolo CAN y posteriormente, basándose en las especificaciones PMS y STZP, publicó el protocolo de capa de aplicación CAN (CAL, *CAN Application Layer*). Otra de las tareas de CiA fue organizar el intercambio de información entre los expertos y usuarios de CAN.

Por su parte, en 1992 Mercedes-Benz implementó el protocolo CAN en sus vehículos Clase S. El sistema estaba compuesto por dos redes CAN, una red de alta velocidad, en la que se comunicaban las ECUs del motor, de la caja de cambios y el tablero de instrumentos; y una red de baja velocidad, para el control del aire acondicionado y de los dispositivos electrónicos internos[8]. La implementación realizada por Mercedes-Benz propició que otros fabricantes de automóviles comenzaran a utilizar redes CAN en sus modelos de lujo, por ejemplo BMW, Jaguar, Volvo, Saab y VW, más tarde se agregaron a la lista Fiat e inclusive Renault y PSA[9].

En noviembre de 1993, el protocolo CAN logró su estandarización internacional bajo la norma ISO 11898 [23], en la que se define una capa física para velocidades de transferencia de datos de hasta 1 Mbps. En 1994, CiA organizó la primera conferencia internacional de CAN (iCC, *internacional CAN Conference*); en el mismo año, la firma Allen-Bradley [URL 1] publicó un protocolo de capa de aplicación CAN conocido como DeviceNet, que actualmente se utiliza en los E.U.A.

En 1995, CiA lanzó al mercado CANopen, que al igual que DeviceNet es un protocolo que cubre la capa de aplicación para CAN pero con mayor aceptación y demanda por parte de los fabricantes europeos [14]. En el mismo año se publicó una mejora al estándar ISO 11898 incluyendo las recomendaciones de la especificación CAN 2.OB de Robert Bosch GmbH.

En marzo de 1995, la SAE emprendió una investigación para valorar las necesidades comunes que representa la implementación de CAN en los automóviles. A principios de 2000, se dio a conocer el protocolo accionado por tiempo basado en CAN (TTCAN, *Time Trigger CAN*) bajo el estándar ISO 11898-4 [23]. La Figura 1.2 muestra la evolución del protocolo CAN.

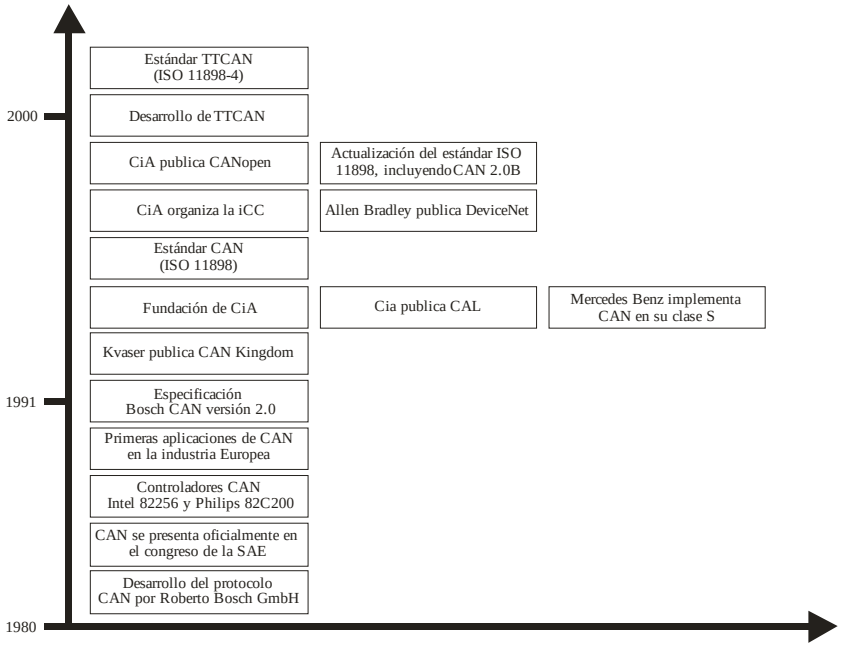


Figura 1.2. Evolución del protocolo CAN.

[8] En el idioma inglés se utiliza el término *body electronics* para referirse a los dispositivos electrónicos internos tales como elevalunas, seguros, control y ajuste de asientos, aire acondicionado, espejos, luces interiores y exteriores, etc.

[9] Renault y PSA fueron las firmas que desarrollaron el protocolo VAN.

1.2. Clasificación de las aplicaciones automotrices

La SAE estableció la siguiente clasificación formal de acuerdo a las áreas de aplicación dentro del automovil:

- *Clase A*: se refiere a la comunicación entre el campo eléctrico y electrónico, conectando nodos no inteligentes como interruptores, luces delanteras, traseras, de freno, de emergencia, posición del asiento, posición de los espejos, elevalunas eléctricos, seguros de puertas, etc. La información que se transmite es poca y requiere bajas velocidades de actualización, la velocidad de transferencia de datos es baja, típicamente menor a los 10 Kbps. La técnica de instalación de cables es sencilla y el costo por conexión de nodos es bajo.
- *Clase B*: a diferencia de las aplicaciones Clase A, en éstas se distribuye una mayor cantidad de información, por ejemplo la transferencia de datos para actualizar el tablero de instrumentos y el control del aire acondicionado. La velocidad de transferencia de datos requerida está en el rango de los 40 Kbps.
- *Clase C*: comprende la transmisión de información crítica en tiempo real (*real-time*) con un ciclo de reloj en el rango de 1 a 10 ms, asimismo se requieren tiempos de latencia de mensajes menores a 1 ms. Los paquetes de datos a comunicar son mayores a 1 octeto de longitud. Dentro de esta categoría están las aplicaciones de comunicación entre los diferentes sistemas de control de motor, transmisión, estabilidad, frenos y dirección. La velocidad de transferencia de datos requerida comprende el rango de 250 Kbps a 1 Mbps.
- *Clase D*: en esta categoría se comunican grandes bloques de datos con una longitud de cientos a miles de octetos. Algunas aplicaciones son: conexión del sistema de radio, teléfono, navegación GPS, reproductor de discos compactos, consola de interfaz para controladores genéricos que tienen la finalidad de descargar programas, parámetros, etc. La velocidad de transferencia de datos requerida está en el rango de 1 a 10 Mbps.

1.3. Principales características del protocolo CAN

Actualmente los conceptos de productor/consumidor (*producer*/*consumer*) y cliente/servidor (*client*/*server*) son los modelos de comunicaciones de datos más utilizados en la automatización industrial.

El protocolo de comunicaciones CAN se basa en el modelo productor/consumidor, el cual es un concepto, o paradigma de comunicaciones de datos, que describe una relación entre un productor y uno o más consumidores. Este modelo describe un servicio de comunicación en el que un proceso proporciona datos, por difusión de mensajes, a otros procesos sin que éstos los soliciten. El productor proporciona servicios a otros procesos mediante un indicador de servicio; los consumidores (procesos) pueden aceptar o ignorar dichos servicios; y debido a que el productor no conoce a los consumidores, no se requiere una confirmación del servicio.

Los protocolos de capas de aplicación (CAL, CANopen y DeviceNet) dan soporte al modelo cliente/servidor.

Existen dos tipos de protocolos de comunicaciones de datos:

- *Protocolos orientados a nodos:* el intercambio de información entre nodos se basa en el direccionamiento de los mismos, generalmente las tramas de datos que se transmiten contienen las dirección destino y algunas veces la dirección fuente.
- *Protocolos orientados a mensajes:* la información a intercambiar se descompone en mensajes, a los cuales se les asigna un identificador y se encapsulan en tramas para su transmisión. Cada mensaje tiene un identificador único, con el cual los nodos deciden aceptar o no dicho mensaje.

CAN es un protocolo orientado a mensajes, y dentro de sus principales características se encuentran [6, 13, 27]:

- Prioridad de mensajes.
- Garantía de tiempos de latencia.
- Flexibilidad en la configuración.
- Recepción por multidifusión (*multicast*) con sincronización de tiempos.
- Sistema robusto en cuanto a consistencia de datos.
- Sistema multimaestro.
- Detección y señalización de errores.
- Retransmisión automática de tramas erróneas tan pronto como el bus esté libre.
- Distinción entre errores temporales y fallas permanentes de los nodos de la red, y desconexión autónoma de nodos defectuosos.

1.4. Aplicaciones de CAN

Actualmente, el sistema de bus serie CAN se utiliza en diversas áreas de aplicación industriales debido, principalmente, a su bajo costo, alto rendimiento, proliferación rápida guiada por la industria automotriz y respaldo de un gran número de fabricantes de dispositivos CAN.

Considerando las características anteriores y la excelente relación costo/desempeño, CAN ha sido adoptado en la mayoría de campos relacionados con la automatización de control industrial como el protocolo de comunicaciones para aplicaciones de FBs y de sensor/actuador. Las principales compañías de control industrial ofrecen, al menos, una interfaz CAN para sus sistemas en una gran variedad de aplicaciones [URL 6].

A continuación se mencionan las principales aplicaciones CAN respaldadas por CiA dentro de los sectores automotriz, automatización industrial, control de maquinaria, y marítimos y ferroviarios.

1.4.1. Sector automotriz

CAN fue desarrollado, inicialmente, para aplicaciones en los automóviles y por lo tanto la plataforma del protocolo es resultado de las necesidades existentes en el área automotriz. A nivel global, los fabricantes Europeos implementan al menos una red CAN en sus automóviles para el control de motor; los fabricantes americanos utilizan CAN en aplicaciones de control de energía del motor; y los fabricantes del Lejano Oriente han iniciado el desarrollo de redes basadas en CAN para aplicarlas en sus automóviles.

La Organización Internacional para la Estandarización (ISO, *International Organization for Standarization*) define dos tipos de redes CAN: una red de alta velocidad, bajo el estándar ISO 11898-2 [23], para controlar el motor e interconectar las ECUs; y una red de baja velocidad, bajo el estándar ISO 11519-2 [22], dedicada a la comunicación de los dispositivos electrónicos internos de un automóvil como son control de puertas, techo corredizo, luces y asientos.

En algunos automóviles Europeos se está implementando una interfaz de diagnóstico CAN, basada en el estándar ISO 15765 [15, 24], la cual define las capas física, de transporte y aplicación, así como el uso de los servicios Keywords 2000 [URL 23]. Otra aplicación en los automóviles basada en CAN es la interconexión de dispositivos multimedia y de entretenimiento, para ello la SAE ha iniciado la especificación del protocolo IDB-C [URL 8], consistente en una red basada en CAN que utiliza un formato de trama extendida.

Las grandes compañías de la industria automotriz, como DaimlerChrysler, Ford, General Motors, etc., se encuentran vinculadas al desarrollo de redes CAN mediante el proyecto *x-by-wire* (inci-

so 1.4.1.4), que pretende implantar un nuevo paradigma del automóvil consistente en sustituir los sistemas de control mecánicos e hidráulicos del automóvil por sistemas electrónicos distribuidos [28]. Dichas compañías utilizan CAN en redes de alta velocidad y para aplicaciones de baja velocidad utilizan el protocolo J1850 [36]. La firma Toyota ha implementado CAN y próximamente lo harán los demás fabricantes japoneses y coreanos.

La difusión de CAN a nivel mundial ha sido rápida, al grado que cada año se instalan millones de nodos CAN. Diariamente se producen cerca de 100,000 unidades ABS (*Anti-lock Break System*) y por lo menos la mitad de ellas cuenta con interfaz CAN. El Programa Electrónico de Estabilidad (ESP, *Electronic Stability Program*), desarrollado por Bosch GmbH, es una extensión del sistema ABS y del ASR (*Anti Slip Regulator*) que se conecta a una red CAN. La firma Mercedes-Benz utiliza unidades ESP, fabricadas por Bosch e implementadas desde 1995 en los automóviles Clase S, en toda su línea de automóviles desde la Clase A hasta la Clase S. Otro ejemplo es el Peugeot 607, el cual cuenta con un ESP que se comunica con las ECUs de la caja automática de velocidades y el sistema de gestión del motor a través de una red CAN [2].

En 1999, DaimlerChrysler en colaboración con la firma Siemens Automotive [URL 20], introdujeron una nueva tecnología de acceso y encendido en automóviles mediante la implementación del sistema Keyless Go con interfaz CAN, inicialmente pensado para la Clase S de Mercedes-Benz [2].

En la actualidad, los fabricantes de ECUs integran la opción de comunicación CAN para satisfacer las necesidades de conectividad dentro de los automóviles. La creciente demanda de ECUs en las futuras generaciones de automóviles conduce a nuevos desafíos técnicos para el establecimiento de una red como son el número total de nodos en la red y el rendimiento de procesamiento de la información.

En 1999, la firma Hella, fabricante de luces para autos, reportó la utilización de 15 millones de dispositivos CAN en sus sistemas [URL 9].

Los cambios que enfrentan los automóviles en aspectos de suministro de energía enfrentan un cambio de conmutación pasiva de carga hacia una administración activa de energía y carga, por ello se está llevando a cabo la transición del suministro de energía principal de 12 V hacia suministros de energía de 14 o 42 V. La firma Yazaki introdujo los módulos inteligentes para el control de carga SmartLCC (*Load Control Center*), los cuales se conectan mediante una red CAN [URL 24].

Las nuevas tecnologías aplicadas a los automóviles incluyen sistemas de asistencia para estacionamiento y de respaldo, dichos sistemas necesitan conectarse a la red CAN para obtener la información producida por otros sistemas (ECUs, sensores de radar, posición, etc.) Los sistemas bajo investigación y desarrollo demandan la existencia de redes CAN para supervisión, por ejemplo:

- Sistema de alerta al conductor (*Driver Alertness Monitoring*).
- Colección electrónica de peaje (*Electronic Toll Collection*).
- Datos flotantes del vehículo (*Floating Vehicle Data*).
- Advertencia de colisión frontal (*Front Collision Warning*).
- PDA (*Personal Digital Assistant*).

A continuación se exponen los principales sistemas dentro del automóvil que implementan el protocolo de comunicaciones CAN.

1.4.1.1. Frenos inteligentes

Los sistemas de precisión y control inteligente están facilitando el desarrollo de futuros sistemas de manejo asistidos electromecánicamente.

Figura 1.3. Módulos SBS.

DaimlerChrysler trabaja en el sistema de frenado Sensortronic (SBS, *Sensortronic Brake System*) conectado mediante CAN (Figura 1.3). SBS combina las lecturas obtenidas por los sensores del ABS y del ESP de manera que el MCU del sistema calcula, dependiendo de la situación, la presión óptima a aplicar a los frenos. Los frenos hidráulicos controlados electrónicamente necesitan conocer los siguientes parámetros [URL 5]:

- Velocidad de las ruedas.
- Ángulo de dirección de las ruedas y ángulo del volante de dirección.
- Aceleración transversal.
- Movimiento circular.

De esta manera se mejora notablemente el rendimiento de los sistemas ABS, ESP y ASC (*Automatic Slip Control*).

1.4.1.2. Sistemas electrónicos inteligentes

De forma paralela a las redes de interconexión en los automóviles, la industria automotriz ha introducido sistemas electrónicos inteligentes (*smart electronics*), los cuales reciben y envían información del exterior del vehículo para retroalimentar al conductor y a las ECUs del automóvil.

Ejemplo de dichos sistemas es la unidad de control de crucero adaptable (ACC, *Adaptive Cruise Control*) que requiere de una comunicación confiable entre otras ECUs. La unidad ACC deslinda al conductor de ajustes de velocidad repetitivos y monótonos a través de un control automático de aceleración, desaceleración y frenado en condiciones de tráfico variable. Varios distribuidores automotrices han anunciado sistemas ACC, por ejemplo Bosch y Siemens, dichos sistemas deben comunicarse con varios dispositivos, radares, bolsas de aire, cinturones de seguridad, ECUs, etc., y dado que la comunicación requiere un gran ancho de banda, se utiliza CAN para dar soporte confiable a bajos costos. Actualmente el modelo XKR de Jaguar cuenta con un sistema estándar ACC.

Para la conexión de sistemas de clasificación de peso (WCS, *Weight Classification System*) con otras unidades de control dentro del vehículo, es adecuado el uso de redes basadas en CAN. El sistema WCS fue desarrollado por la firma Siemens Automotive. El módulo de control WCS recoge la información de los sensores medidores de tensión en los cinturones de seguridad para procesar los datos de entrada, y con ello definir y clasificar al ocupante como infante, niño pequeño, adolescente, adulto pequeño o adulto grande.

1.4.1.3. Control de telemática

En el campo de control de telemática existen numerosos distribuidores de dispositivos de entretenimiento que implementan interfaces CAN, por ejemplo:

- *Bosch/Blaupunkt*: fue una de las primeras compañías en utilizar CAN como una red de control para entretenimiento.
- *Bosch*: desarrolló el perfil de aplicación MCNet (*Mobile Communication Network*), el cual está basado en CAN y se distribuye mediante la firma Vector Informatik, sin embargo no es aceptada por los fabricantes de automóviles debido a que los buses de audio y video no están estandarizados.
- *Ford Motor Co.*: compañía integrada por Ford, Jaguar, Lincoln, Mazda, Mercury Volvo, está colaborando con el AMIC (*Automotive Multimedia Interface Consortium*) para el desarrollo del bus inteligente de datos (IDB, *Intelligent Data Bus*). Adicionalmente a la red de audio y video, el estándar IDB necesita de una red CAN para la comunicación de órdenes y realimentación. Las redes IDB conectan sistemas de navegación, sistemas de entretenimiento en los asientos traseros y sistemas de entretenimiento en general.

Actualmente, se encuentra en proceso de normalización un sistema de bus para audio y video, sin embargo existen soluciones como el sistema de red MOST (*Media Oriented System Transport*). La firma Yazaki suministra una pasarela (*gateway*) MOST/CAN que transmite información de audio en una red CAN hacia dispositivos de entretenimiento como radios, amplificadores y cambiadores de CD.

En el 2000, la firma Philips introdujo al mercado su sistema Nexperia CIP (*Car Infotainment Platform*) basado en un procesador MIPS, el cual proporciona la funcionalidad DSP (*Digital Signal Processing*) requerida para el procesamiento de audio y video, además está equipado con diversos módulos como memoria de video, decodificador y multiplexor de video MPEG, acelerador de 3D y navegación GPS. Nexperia CIP cuenta con interfaz CAN para la comunicación entre dichos dispositivos.

1.4.1.4. Tecnología x-by-wire

Un sistema *x-by-wire* es un sistema electrónico tolerante a fallos que se relaciona con la seguridad en vehículos sin respaldo mecánico alguno. El objetivo de un sistema *x-by-wire* es asistir al conductor en diversas situaciones y hacer más segura la conducción, lo cual contribuye a incrementar la seguridad total del vehículo deslindando al conductor de algunas tareas rutinarias. Su principal ventaja es su bajo costo de producción. Un sistema *x-by-wire* es también llamado un sistema seco (*dry system*) ya que se requieren menos fluidos hidráulicos, lo cual conduce a un sistema más simple y fácil de mantener [28].

El concepto *x-by-wire* surgió como consecuencia de los sistemas de redes dentro de los automóviles, ya que éstos han permitido su desarrollo e implementación. CAN es el protocolo de comunicaciones pionero en los sistemas *x-by-wire*, y éstos a su vez están basados en dicho protocolo.

Los fabricantes de automóviles se han preocupado por la evolución de los mismos e intentan ofrecer vehículos más seguros, confortables, baratos y ecológicos, lo cual se refleja en la continua investigación y desarrollo de prototipos que cuentan con sistemas *x-by-wire*. A continuación se describen dos prototipos que establecen una nueva filosofía en el diseño, producción y conducción de automóviles.

1.4.1.4.1. BMW Z22

El fabricante de automóviles BMW ha puesto gran interés en el desarrollo y aplicación de nuevas tecnologías que contribuyan al mejoramiento de sus automóviles.

Los ingenieros de BMW desarrollan un prototipo denominado Z22, considerado el automóvil mecatrónico más ligero y avanzado (Figura 1.4). El diseño exterior del Z22 es bastante conservador en contraste con su tecnología de desarrollo. Las tecnologías *steer by wire* y *brake by wire* (Figura 1.5) han sustituido por completo a la dirección mecánica y al freno hidráulico, con lo cual se abrió un nuevo capítulo en la llamada tecnología *by wire*, ya que estos sistemas que anteriormente se accionaban mecánicamente, ahora se regulan de forma electrónica, proporcionando ventajas considerables en cuanto a comodidad, rapidez y calidad de respuesta.

Al suprimir componentes mecánicos, como la columna de dirección y las partes del sistema de pedales, se han incrementado tanto la seguridad como la comodidad.

En forma general, se establece una comunicación de alta velocidad que permite la tolerancia a fallos (*fault tolerance*) entre diferentes dispositivos electrónicos como sensores de desplazamiento, ángulo, velocidad, unidades de control, etc. que gobiernan los sistemas de frenado y dirección.

1.4.1.4.2. Autonomy de GM

La compañía General Motors (GM) desarrolla el Autonomy (Figura 1.6), el cual es un vehículo que prácticamente no cuenta con ningún sistema mecánico. Es un automóvil que combina un sistema de propulsión a base de celdas de hidrógeno con tecnología *x-by-wire*, lo cual permite que los sistemas de frenado, dirección y tracción sean controlados electrónicamente.

Figura 1.4. Detalles de diseño del automóvil BMW Z22.

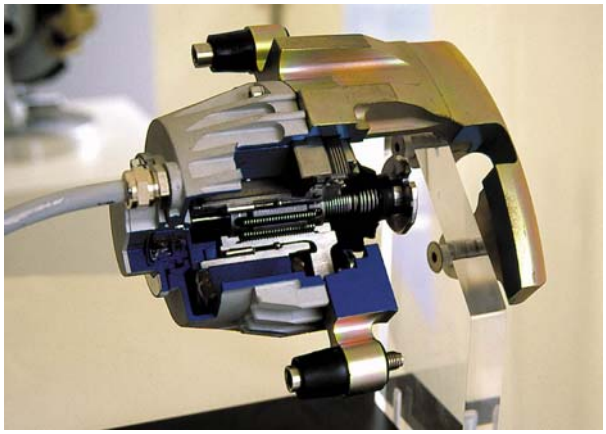


Figura 1.5. Sistema *brake-by-wire* del BMW Z22.

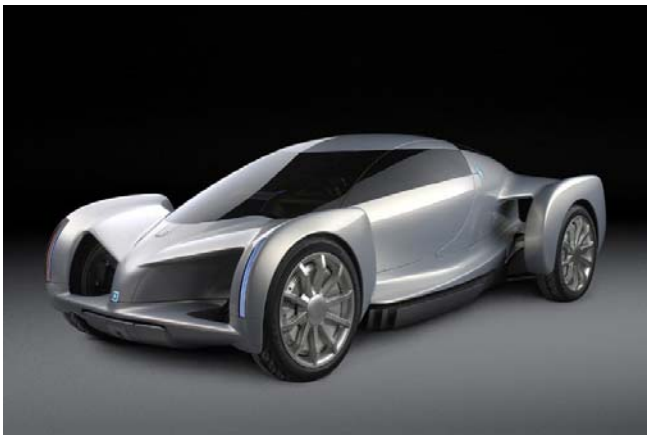



Figura 1.6. Detalles de diseño del automóvil Autonomy de GM.

El Autonomy es un proyecto más ambicioso que el Z22 y los sistemas en los que se respalda la comunicación entre los diferentes dispositivos electrónicos dentro del vehículo están basados en un sistema de red CAN.

1.4.2. Automatización de industrias

En el área de la automatización industrial, en los E.U.A. se está utilizando el protocolo de comunicaciones basado en CAN conocido como DeviceNet para construir fábricas inteligentes (*digital smart factory*). El protocolo DeviceNet es el más utilizado en esta área geográfica gracias a la difusión que le otorgó la firma Rockwell Automation (actualmente Allen-Bradley).

DeviceNet se utiliza generalmente en la automatización de procesos industriales y a continuación se describen algunos ejemplos de su aplicación.

1.4.2.1. Domótica

Una de las primeras aplicaciones de CAN se realizó en 1990 en un sistema de control de elevadores (subcapítulo 1.1), a partir de entonces muchas compañías han empleado sistemas basados en CAN en el diseño de edificios inteligentes (*smart buildings*).

Se han instalado redes CAN en sistemas empotrados de calefacción y enfriamiento de edificios, por lo que miles de aparatos de refrigeración, aire acondicionado, calefacción, etc. cuentan con nodos CAN.

En la Tabla 1.1 se muestran los lugares donde se han instalado redes CAN y las funciones que desempeñan dichas aplicaciones.

Tabla 1.1. Aplicaciones CAN en domótica.

Lugar de aplicación	Funciones
Supermercados	Oscurecimiento de ventanas, control integrado del cuarto, control y acceso de puertas, aire acondicionado, iluminación interior y exterior, sistema de alarma
Universidades, escuelas, salones deportivos	
Estaciones de tren	
Edificios de oficina, fábricas, bancos y aseguradoras	
Otro tipo de edificios	Sistemas rociadores y aspersores, sistema de control de camerinos, equipo de estudio (incluyendo control de iluminación, audio y video)

1.4.2.2. Línea de producción

A continuación se presentan tres ejemplos de soluciones realizadas mediante redes CAN en la línea de producción.

1.4.2.2.1. Empresa cementera

Lafarge Group es una empresa dedicada al suministro de materiales para construcción como son cemento, concreto y agregados [URL 14]. Dicha empresa utiliza DeviceNet en su planta de Alpena, Michigan (E.U.A.) para el control de motores en su sistema de control total, con lo cual ha reducido el tiempo de instalación y ha mejorado la operación de sus hornos.

Las características principales de la planta cementera son:

- Cuenta con cinco hornos que operan las 24 horas y, a excepción del periodo anual de mantenimiento en el que se tienen que apagar, opera todos los días del año.
- Está equipada con aproximadamente 500 motores que realizan varias funciones de los hornos, dichos motores son de diferente potencia y van desde 0.5 a 450 h.p.
- Tiene una producción anual de 2.5 millones de toneladas de cemento.

El fallo de un motor puede traer consecuencias críticas en cuanto a la producción y rentabilidad de la planta, lo cual se ha calculado en 9,500 toneladas de cemento a la semana.

La Tabla 1.2 lista los beneficios y logros obtenidos gracias a la implementación de DeviceNet en dicha planta.

Tabla 1.2. Beneficios y logros obtenidos a partir de la implementación de DeviceNet.

Beneficios	Logros
Menores tiempos de instalación	
Mejora en las operaciones de los hornos	Control distribuido en la planta
Rápidez en la obtención de diagnósticos más fiables	Ahorro en costes de cableado y de instalación
Mejor planeación en cuanto al mantenimiento preventivo	Confiabilidad y alta integración de la planta
Mayor inteligencia en el área de control de máquinas	

1.4.2.2.2. Empresa cervecera

El Grupo Modelo, fabricante de la cerveza Corona, es una de las industrias cerveceras más grandes del mundo y en su planta de Zacatecas (México) cuenta con un sistema de control. Tiene una capacidad de producción de 45 millones de hectolitros y para competir a escalas mundiales decidió instalar un equipo de embotellado de alta velocidad para producir 132,000 botellas por hora [URL 7]. Lo anterior atendiendo a las siguientes necesidades:

- El sistema de embotellamiento necesitaba instalar una línea de alta velocidad preservando la producción en la línea existente.
- Adicionalmente, el equipo se extendería sobre un área extensa, surgiendo la necesidad de distribuir inteligencia a través de líneas de producción con una longitud de hasta 100 m.
- Se necesitaba integrar y comunicar diferentes tipos de equipos, motores, PLCs (*Programmable Logic Controllers*), módulos de E/S, terminales de operador con interfaz, etc. sin incrementar el cableado existente.

La firma Sasib Beverage propuso e instaló el sistema Rockwell Automation en la planta mencionada, dicho sistema es una solución integrada basada en DeviceNet. Una de las características de DeviceNet es permitir la comunicación de dispositivos de distintos fabricantes como si todos fueran inteligentes; el suministro de energía y la comunicación de datos se realiza a través de un sólo cable.

Dicha solución resultó de gran interés para el Grupo Modelo y en la actualidad sus líneas de producción cuentan con FBs que conectan cientos de motores y módulos de E/S. Los principales logros obtenidos en la utilización de DeviceNet son la migración a un sistema distribuido y la considerable reducción del cableado.

1.4.2.2.3. Empresa embotelladora

La empresa Rhode Island Beverage, dedicada al envasado de bebidas tales como té, limonada, jugos, etc., fue la precursora del uso de DeviceNet en la línea de producción industrial. Antes de la adopción de DeviceNet como solución, su planta tenía instalado un sistema de embotellado de dos líneas basado en dispositivos de E/S con las siguientes condiciones [URL 3]:

- Contaba con 100 empleados en su instalación de Warwick, Rhode Island.
- Trabajo de 20 horas diarias durante seis días de la semana.
- Necesidad de demostrar a sus clientes que podían satisfacer sus necesidades de calidad, velocidad y volumen.
- Dos líneas de producción idénticas en su operación general.

La Tabla 1.3 muestra las ventajas de utilizar una red DeviceNet en una de las líneas de producción.

Tabla 1.3. Características y ventajas del uso de DeviceNet.

Características de las líneas de embotellado		Logros obtenidos
Antes	**Después**	
Utiliza una plataforma física de E/S Sólo proporciona información sobre el estado de los interruptores de E/S	Uso de DeviceNet Los costos de instalación son los mismos, pero incrementa el ciclo de vida del sistema Información de alto grado que notifica la necesidad de mantenimiento requerido por sensores	Mejoras en la línea de embotellado Reducción de los costos de instalación Mejoras en el desempeño total del sistema Mejora la respuesta a cambios en el mercado Mejoras en el sistema de mantenimiento y de control Se mejoran las condiciones higiénicas de los transportadores de botellas Permite realizar correcciones rápidas en caso de fallos o desajustes en los dispositivos

La experiencia obtenida al actualizar su línea de embotellamiento demuestra los beneficios que puede proporcionar la tecnología evolutiva de los FBs y de las redes de dispositivos en el ciclo de trabajo de un sistema de control automatizado. La Figura 1.7 ilustra la instalación e implementación de DeviceNet respectivamente.

1.4.2.3. Control de calidad en las empresas

La firma Compaq Computers, fabricante de computadoras, cuenta con celdas de producción dedicadas a la fabricación y pruebas de sus unidades centrales de proceso (CPU, *Central Processing Unit*). En la actualidad ha instalado una red DeviceNet con productos de la firma Rockwell Automation que le ha permitido disminuir tiempos y costos de producción [URL 3].

Inicialmente, las celdas en el área de fabricación de CPUs estaban constituidas por estaciones de fabricación y prueba; posteriormente, se expandieron mediante la instalación de nuevas celdas de distintos fabricantes, lo cual requería la conexión de 120 terminales durante dos semanas de proceso.

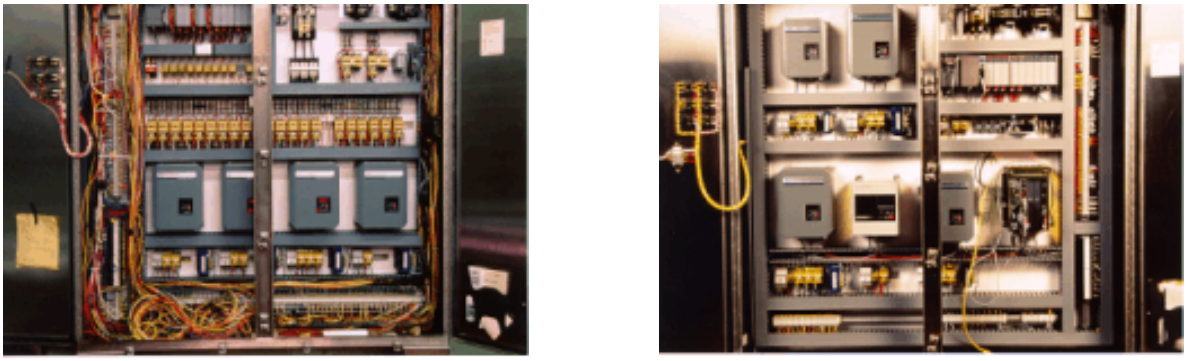


Figura 1.7. Instalación del control de la planta antes y después de implementar DeviceNet.

Con la instalación de DeviceNet se mejoró el rendimiento del sistema en un 50% utilizando únicamente 16 enlaces a las celdas.

La red DeviceNet comunica los siguientes equipos en una celda de fabricación y prueba:

- 28 fotoceldas reflectivas de la serie 9000 de la firma Rockwell Automation.
- Un controlador SLC5/03 de la firma Rockwell Automation.
- Bloques de E/S para la comunicación entre celdas y el control de dispositivos neumáticos.
- Relevadores electrónicos modelo SMP3 para protección de sobrecarga de corriente a los motores de las transportadoras.

Actualmente el 80% de las celdas se conectan a la red DeviceNet y los principales logros obtenidos son:

- Se ha simplificado el mantenimiento del sistema.
- Se pueden sustituir los dispositivos dañados sin afectar el rendimiento del sistema.
- El tiempo para detección de fallos se ha reducido en una sexta parte.
- Disminución de costos de instalación.
- Capacidad añadida de acceso de datos para almacenar eventos, supervisar y temporizar el tiempo de inoperabilidad.

1.4.2.4. Otras aplicaciones

Otras aplicaciones, principalmente de DeviceNet, son las siguientes [URL 3]:

- Sistema de automatización en almacenes.
- Producción de vajillas.
- Sistemas de calefacción, ventilación y aire acondicionado (HVAC, *Heating, Ventilating and Air Conditioned*) en autos, camiones y equipo militar, desarrollado por la firma Delphi Harrison Thermal Systems.
- Sistemas de empaquetamiento de comida (FMS, *Food Machinery Sales*).
- Sistemas dispensadores para la industria automotriz, utilizados por DaimlerChrysler, Ford y GM, y distribuidos por Ingersoll-Rand Johnstone Dispensing Systems.
- Línea de producción de esponjas.
- Producción de cortinas y puertas de acero para cocheras.
- Depósitos de granos en un almacén.
- Elaboración de pan.
- Fabricación de mallas y medias.

1.4.3. Control de maquinaria

En el área de control de maquinaria, se utilizan redes CAN para control interno. Los fabricantes de máquinas textiles fueron unos de los primeros usuarios de CAN (subcapítulo 1.1), con base en los resultados obtenidos se ha ampliado el uso a máquinas de impresión, de moldeado por inyección, de procesamiento de madera, de venta y juegos.

En dichas aplicaciones se utiliza CAN como una red empotrada (*embedded network*) que conecta controladores programables, dispositivos de E/S y controladores de movimiento.

Las redes basadas en CAN proporcionan capacidad de respuesta en tiempo real y la flexibilidad necesaria para optimizar la comunicación interna de las máquinas.

Inicialmente, los usuarios de CAN desarrollaron soluciones propias, pero en la actualidad la gran mayoría han emigrado a CANopen. Por ejemplo, la industria alemana de impresión ha decidido utilizar CANopen como plataforma de integración para subsistemas de terceros y para la comunicación interna de sus máquinas. Otras aplicaciones basadas en CANopen son las siguientes:

- Máquinas para moldear vidrios, cerámica, plástico, etc. por soplado y por inyección.
- Máquinas para fabricación de envolturas para barras de chocolate.
- Sistemas para fabricación y generación de pruebas para semiconductores (CANtrol).
- Máquinas de diseño para fibras de vidrio.
- Máquinas manipuladoras de obleas de silicio (CAN I/O).
- Máquinas industriales de costura.

1.4.4. Aplicaciones marítimas y ferroviarias

A continuación se mencionan los aspectos más importantes de la integración de redes basadas en CAN en aplicaciones marítimas y ferroviarias.

1.4.4.1. Redes empotradas en subsistemas marítimos

En el caso de las aplicaciones marítimas como son botes deportivos, barcos y embarcaciones, se utilizan redes CAN como redes empotradas en subsistemas y como redes integradas que conectan subsistemas que requieren soluciones tolerantes a fallos.

En la exhibición SMM (*Ship Machinery and Marine Technology*) llevada a cabo en el 2000 en la ciudad de Hamburgo (Alemania), diversas compañías presentaron sistemas automatizados para embarcaciones basados en CAN, subsistemas con interfaz CAN y dispositivos marítimos sencillos con conectividad CAN. Las aplicaciones específicas se detallan a continuación.

1.4.4.1.1. Establecimiento de redes a bordo

Particularmente, la industria europea de equipo y sistemas marítimos utiliza redes CAN en lugar de las redes basadas en RS-485 como Modbus.

Existen varios protocolos de la capa de aplicación CAN que se utilizan en aplicaciones marítimas, por ejemplo, algunos distribuidores de equipo naval electrónico están utilizando CANopen, y la firma Mercury, fabricante de botes deportivos, utiliza CANKingdom.

En los E.U.A. se utiliza el perfil de aplicación NEMA 2000, basado en J1939, en sistemas de navegación. Por otro lado, varios distribuidores de sistemas marítimos han desarrollado soluciones CAN propietarias.

1.4.4.1.2. Sistema tolerante a fallos CANopen para automatización marítima

Los sistemas tolerantes a fallos han adoptado la comunicación basada en CANopen, entre las aplicaciones más importantes se encuentran: alarmas de barcos, supervisión y sistemas de control en transportadores de productos, barcos contenedores, embarcaciones de pasajeros, transbordadores y buques de carga. Dichas aplicaciones requieren que el sistema sea tolerante a fallos, lo cual implica que debe proporcionar redundancia tanto en la configuración general del sistema como en el sistema de comunicaciones.

La firma IXXAT Automation ha desarrollado un sistema de comunicación basado en CANopen que cumple los requisitos de un sistema tolerante a fallos para la empresa Noruega Kongsberg Norcontrol, distribuidor de sistemas de automatización para barcos [URL 11]. El sistema resultante se basa en CANopen y está destinado a satisfacer las necesidades de las aplicaciones marítimas, como proyecto fue desarrollado el programa CANopen SIG "*Maritime Electronic*" de CiA [URL 3].

El sistema de automatización de barcos está integrado por diferentes subsistemas. La idea básica de la arquitectura del sistema es la división en varios segmentos de proceso y en subsistemas con funciones tales como el manejo de energía, bombas y válvulas o sistemas auxiliares que integran un segmento de proceso (sistema distribuido).

Basándose en la capacidad de procesamiento de las unidades distribuidas, los segmentos de proceso son independientes del resto incluso cuando se aíslan del sistema. El intercambio de información dentro de cada segmento de proceso está basado en una comunicación local (*local process bus*). Un bus de proceso global conecta los diferentes segmentos de procesos con las estaciones remotas de operación, los subsistemas se conectan al bus global mediante controladores de proceso segmentado (PSC, *Process Segment Controllers*), los cuales proporcionan funciones de puente entre los subsistemas y el bus global. Ambos tipos de sistemas de comunicación se implementan como una red redundante basada en CAN.

1.4.4.1.3. Sistema de automatización en barcos pesqueros

El campo de la pesca comercial requiere de tareas especiales, gran cantidad tiempo y equipo complejo. Las embarcaciones para este tipo de pesca presentaban problemas en cuanto a la automatización de pesca, conservación y traslado.

La firma FHE diseñó un sistema basado en DeviceNet para el control de equipos de supervisión del motor y control de temperatura en la embarcación Ocean Breeze. Con lo cual se han logrado identificar problemas potenciales antes de que éstos puedan ocasionar daños.

La arquitectura DeviceNet modificó la estructura tradicional del cableado en los barcos pesqueros, suministrando robustez y flexibilidad, por lo que no es extraño que en futuros diseños se establezca dicha solución como un estándar.

1.4.4.2. Sistemas de control en ferrocarriles y sistemas de carga

En la feria internacional Innotrans 2002, realizada en Berlín (Alemania), se propuso la estandarización de una red para componentes electrónicos y de redes empotradas en vagones de tren. Anterior a este hecho, el foco de atención recaía en los buses de comunicación WTB (*Wide Train Bus*) y MVB (*Multiple Vehicle Bus*), sin embargo en la actualidad CANopen es el candidato idóneo para su estandarización en redes empotradas.

Además, en el mismo campo de aplicación, CANopen se utiliza en las siguientes áreas:

- Locomotoras Diesel.
- Trenes ligeros.
- Tranvías.
- Información para los pasajeros.

2. Protocolo de comunicaciones CAN

CAN es un protocolo de comunicaciones serie que soporta control distribuido en tiempo real con un alto nivel de seguridad y multiplexación [6].

El establecimiento de una red CAN para interconectar los dispositivos electrónicos internos de un vehículo tiene la finalidad de sustituir o eliminar el cableado [44]. Las ECUs, sensores, sistemas antideslizantes, etc. se conectan mediante una red CAN a velocidades de transferencia de datos de hasta 1 Mbps.

La arquitectura de protocolos CAN, de acuerdo al modelo de referencia OSI (*Open Systems Interconnection*) [21], incluye tres capas: física, enlace de datos y aplicación, además establece una capa especial para la gestión y control del nodo llamada capa de supervisor (Figura 2.1).

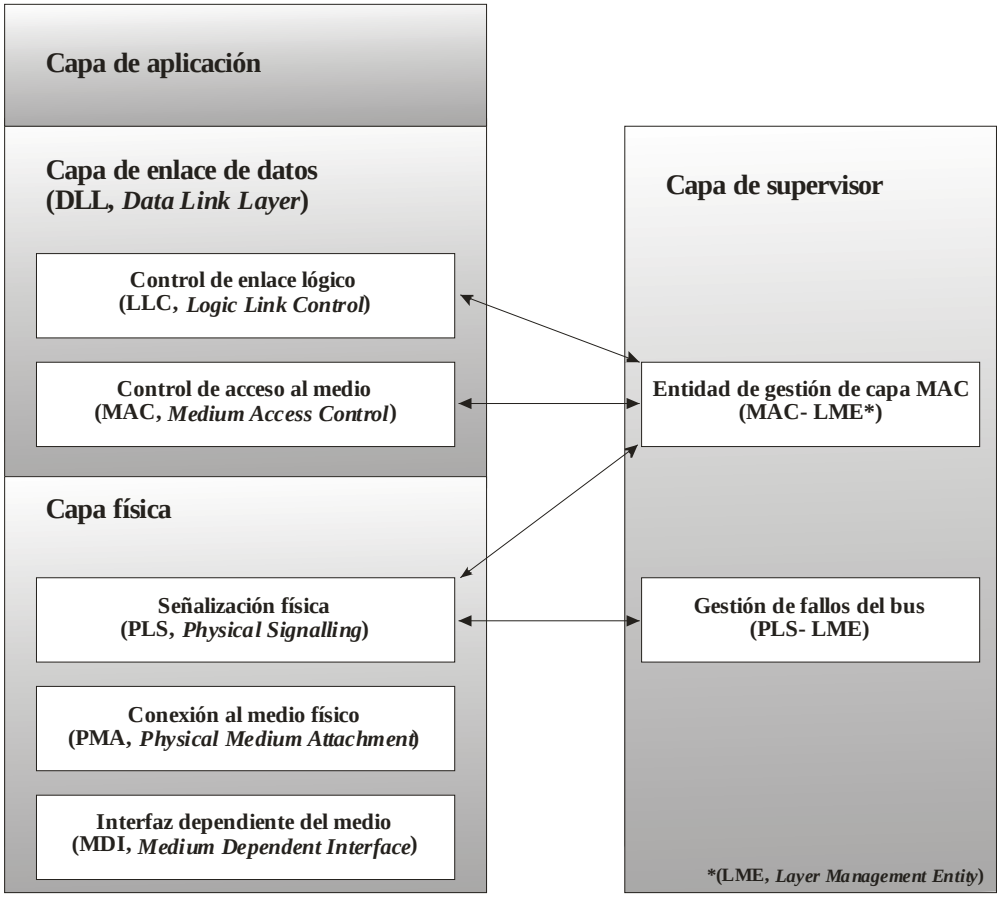


Figura 2.1. Arquitectura de protocolos CAN.

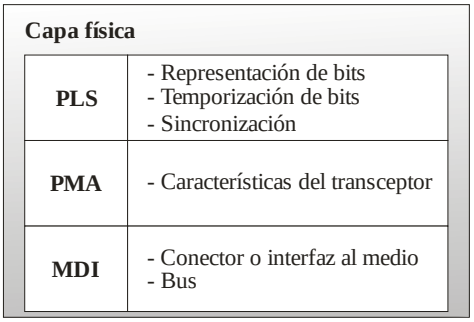


Figura 2.2. Arquitectura de la capa física del protocolo CAN.

2.1. Capa física

La capa física de un sistema de comunicaciones define los aspectos del medio físico para la transmisión de datos entre los nodos de una red, los más importantes hacen referencia a los niveles de señal, representación, sincronización y tiempos en los que los bits se transfieren al bus [27]. El diseño de una red CAN varía de acuerdo a las necesidades de desempeño y para ello se deben considerar los requisitos de la capa física.

La especificación del protocolo CAN [6] no define una capa física, sin embargo, los estándares ISO 11898 [23] e ISO 11519 [22] establecen las características que deben cumplir las aplicaciones para la transferencia en alta y baja velocidad respectivamente. Cabe mencionar que las características definidas para la capa física deben implementarse en todos los nodos que se encuentren conectados dentro de una red CAN. La capa física se divide en tres secciones o subcapas (Figura 2.2), las cuales se describen a continuación.

2.1.1. Subcapa de señalización física

La subcapa de señalización física (PLS, *Physical Signalling*) define las funciones relacionadas con la representación, tiempo y sincronización de los bits, y está implementada en los controladores del protocolo CAN.

2.1.1.1. Representación de bits

Una trama CAN está codificada de acuerdo con el método NRZ (*Non Return to Zero*), el cual establece que durante todo el tiempo de bit se genera un nivel de señal que puede ser dominante (d) o recesivo (r). Una ventaja de dicho método, en comparación con la codificación Manchester, es que produce una frecuencia menor de operación, ya que la codificación Manchester requiere de flancos en la mitad del tiempo de bit.

Sin embargo, en el caso de transmitir una gran cantidad de bits con la misma polaridad, la codificación NRZ no proporciona flancos que puedan utilizarse en la sincronización y por ello se implementa el procedimiento de inserción de bit (*bit-stuffing*), el cual asegura que en la transmisión de una trama sólo puede haber un máximo de cinco bits consecutivos con la misma polaridad como se muestra en la Figura 2.3.

En una trama CAN, de datos o remota (inciso 2.2.2.1), la codificación NRZ junto con el procedimiento de inserción de bit se aplican a los campos de inicio de trama (SOF, *Start of Frame*), arbitraje, control, datos[10] y código de redundancia cíclica (CRC, *Cyclic Redundant Check*). Los campos restantes, delimitador CRC, aceptación (ACK, *Acknowledgement*) y fin de trama (EOF, *End of Frame*), tienen un formato fijo y son transmitidos sin emplear el procedimiento de inserción de bit. Lo anterior también se aplica a las tramas de error o de sobrecarga [13].

[10] El campo de datos no se incluye en una trama remota.

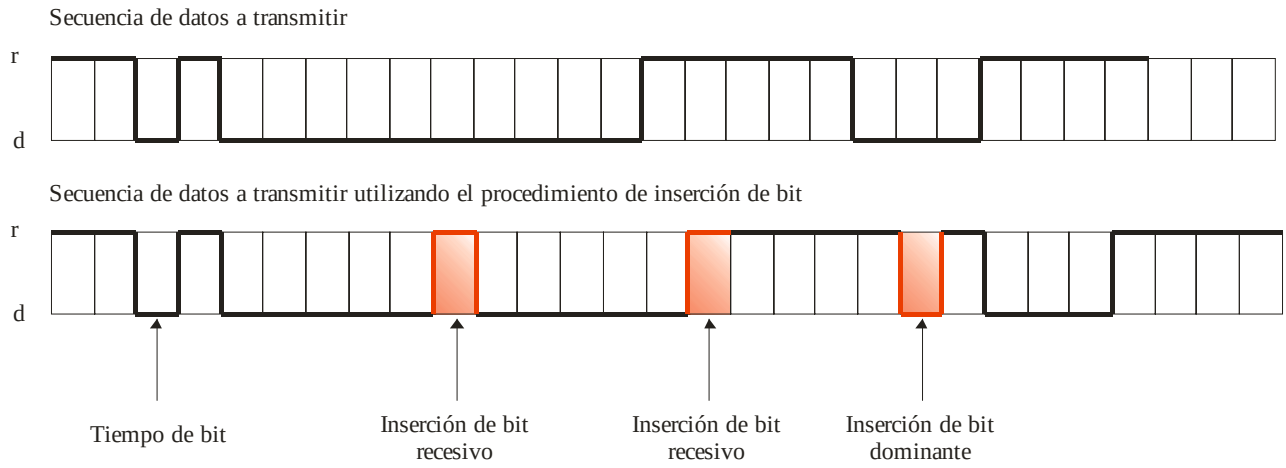


Figura 2.3. Ejemplo del procedimiento de inserción de bit.

2.1.1.2. Temporización de bits

Una característica importante del protocolo CAN es la flexibilidad para determinar los parámetros de velocidad de transferencia, punto de muestreo de bit y número de muestras realizadas en un periodo de bit. Por lo tanto, el diseño de una red CAN debe considerar los siguientes conceptos [6, 22, 23]:

- *Tiempo de bit (t_b)*: se define como el tiempo de duración de un bit.
- *Velocidad de transferencia nominal (f_b)*: es el número de bits por segundo que un transmisor ideal emite sin resincronización.
- *Tiempo de bit nominal*: se obtiene mediante la Ecuación 2.1 y se divide en cuatro segmentos de tiempo no traslapados (Figura 2.4). La longitud de los segmentos de tiempo en un intervalo de bit está definida por múltiplos enteros de la unidad básica de tiempo (t_q, *time quantum*) derivada del periodo del oscilador t_{CLK} (Figura 2.5). El parámetro t_q es la unidad de tiempo discreta más pequeña utilizada por un nodo CAN.

$$t_b = \frac{1}{f_b} \qquad \text{Ecuación 2.1}$$

Los cuatro segmentos que forman un tiempo de bit nominal son:

- *Segmento de sincronización (Sync_Seg)*: se utiliza para sincronizar los diferentes nodos en el bus mediante un flanco dentro del mismo segmento.
- *Segmento de tiempo de propagación (Prop_Seg)*: se utiliza para compensar los tiempos de retardos físicos originados por la propagación de la señal en el bus y por los retardos internos de los nodos.

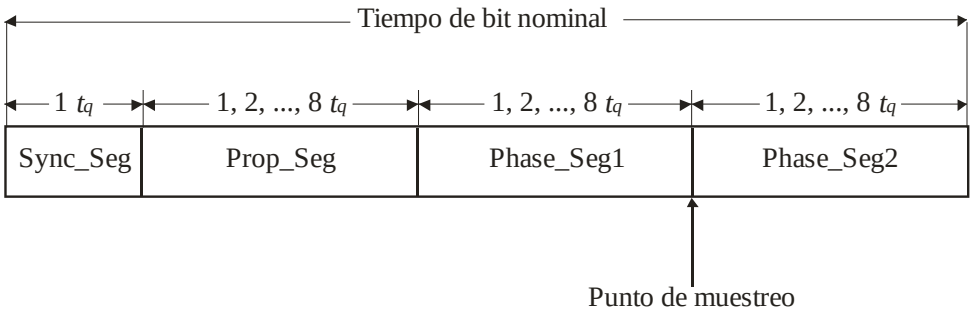


Figura 2.4. Segmentos del tiempo de bit.

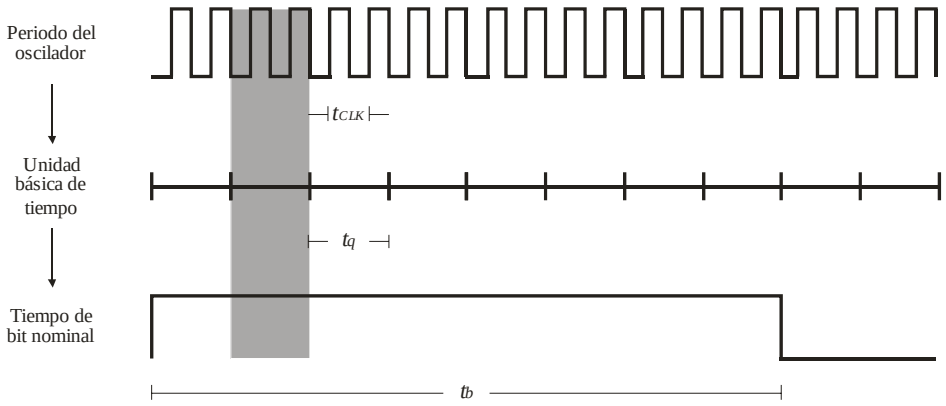


Figura 2.5. Principio de derivación del periodo de bit.

- *Segmento de memoria temporal de fase 1 (Phase_Seg1)*: segmento de longitud ajustable que se utiliza para compensar variaciones de tiempo entre nodos, puede incrementarse durante la resincronización.
- *Segmento de memoria temporal de fase 2 (Phase_Seg2)*: segmento de longitud ajustable que se utiliza para compensar variaciones de tiempo entre nodos y puede reducirse durante la resincronización.

El número total de unidades básicas en un tiempo de bit debe ser programable al menos de 8 a 25 t_q.

- *Punto de muestreo (sample point)*: instante de tiempo en el que se lee e interpreta el nivel del bus y se proporciona el valor del bit respectivo.
- *Tiempo de procesamiento de la información*: es el periodo de tiempo que comienza con el punto de muestreo y se utiliza para calcular el nivel de bit subsecuente.

2.1.1.3. Mecanismos de sincronización de bits

El protocolo CAN utiliza transferencia de datos sincrónica, lo cual incrementa la capacidad de transmisión pero requiere un método de sincronización de bits debido a que sólo está disponible un bit de inicio al comienzo de la trama. Lo anterior no es suficiente para mantener sincronizado el muestreo de bit del receptor con el del transmisor, y para lograrlo se requiere realizar una resincronización continua del receptor [13].

En una red CAN cada nodo se sincroniza con su propio oscilador (inciso 2.1.1.5), debido a ello pueden ocurrir desplazamientos de fase en los diferentes nodos y por lo tanto los controladores CAN proporcionan un mecanismo de sincronización para compensar dichos desplazamientos mientras reciben una trama CAN.

Antes de analizar las dos formas de sincronización descritas por el protocolo CAN, es necesario definir [6]:

- *Error de fase del flanco (phase error of an edge)*: el error de fase del flanco (*e*) está dado por la posición del flanco respecto al segmento de sincronización y se mide en unidades t_q. El signo del error de fase se define de la siguiente manera:
  - *e = 0*, sí el flanco se detecta dentro del segmento *Sync_seg*.
  - *e > 0*, sí el flanco se detecta antes del punto de muestreo.
  - *e < 0*, sí el flanco se detecta después del punto de muestreo del bit previo.
- *RJW (Resychronization Jump Width)*: es el valor programado de unidades t_q que se suma a la longitud del *Phase_Seg1*, o se reduce de la longitud del *Phase_Seg2*.

Se definen dos tipos de sincronización [6]:

- *Sincronización al comienzo de la trama (Hard synchronization)*: después de una sincronización, se reinicia el tiempo de bit al finalizar el segmento de sincronización, sin tener cuidado de un error de fase, con ello la sincronización obliga al flanco a caer dentro del segmento de sincronización del bit reiniciado.
- *Resincronización dentro de la trama (Soft synchronization)*: cuando la magnitud de un error de fase del flanco que generó la resincronización es menor o igual al valor programado del RJW, el efecto de la resincronización es el mismo que el de una sincronización; y cuando la magnitud del error de fase es mayor que el valor de RJW:
  - y sí el error de fase es positivo: el segmento de fase 1 se prolonga por una cantidad igual al valor de RJW;
  - y sí el error de fase es negativo: el segmento de fase 2 se reduce por una cantidad igual al valor de RJW.

Ambas formas de sincronización siguen las siguientes reglas [6]:

1. Dentro de un tiempo de bit sólo se permite una sincronización.
2. Para efectos de sincronización se utiliza un flanco sólo si el valor detectado en el punto de muestreo anterior difiere del valor del bus inmediatamente después del flanco.
3. Se realiza una sincronización siempre que haya un flanco recesivo a dominante, y cuando el bus esté desocupado (*bus idle*).
4. Todos los flancos recesivos a dominantes, opcionalmente los flancos dominantes a recesivos en el caso de bajas velocidades de transferencia, que satisfagan las reglas 1 y 2 se utilizan para una resincronización, a excepción de que un nodo que se encuentre transmitiendo un bit dominante no puede realizar resincronización como resultado de una flanco recesivo a dominante con un error de fase positivo, si sólo son utilizados para la resincronización flancos recesivos a dominantes.

2.1.1.4. Impacto del tiempo de bit y de la amplitud de la señal en la longitud del bus

La longitud máxima del bus CAN depende básicamente de dos aspectos:

- *Aspectos de corriente alterna*: que corresponden al tiempo de bit, tolerancia del oscilador, retardo de propagación y capacitancia de la red.
- *Aspectos de corriente directa*: correspondientes a las consecuencias de la amplitud de la señal del bus, impedancia característica de los cables del bus y la impedancia de entrada de los nodos.

Como se menciona en el inciso 2.1.3.1, la capa física se encarga de representar los estados dominante y recesivo. Durante el proceso de arbitraje para acceder al medio, el nodo compara los valores de bit que transmite con los que recibe para decidir si continua con la transmisión, para ello los nodos deben asegurar que la transmisión de un bit dominante se realice dentro del periodo de bit correspondiente y por consiguiente puedan detectar una condición de pérdida del arbitraje y detengan su transmisión.

El retardo de propagación entre los nodos es resultado del retardo de línea del bus específica (menor a la velocidad de la luz) y del retardo de los componentes electrónicos [27]. Por esta razón, los requerimientos de tiempo de bit implican que la velocidad de transferencia disminuya a medida que la longitud del bus y/o la tolerancia del oscilador incrementan.

Existen dos criterios para optimizar la relación entre la velocidad de transferencia y la longitud del bus:

- El primero se enfoca en obtener una longitud de bus máxima dada una velocidad de transferencia y para ello es necesario que la ubicación del punto de muestreo esté lo más cercano posible del final del periodo de bit, y la tolerancia del oscilador debe minimizarse utilizando un oscilador de cristal (inciso 2.1.1.5).
- El segundo hace hincapié en el uso de osciladores menos precisos y por lo tanto, el tiempo de bit debe ajustarse a la tolerancia máxima del oscilador, sin embargo, la optimización de la tolerancia del oscilador implica una reducción considerable de la longitud de bus a una velocidad de transferencia dada.

La relación entre la máxima velocidad de transferencia y la máxima longitud de bus, pueden calcularse mediante la Ecuación 2.2.

$$f_B * long\_bus < 50000m * Kbps \qquad \text{Ecuación 2.2}$$

Utilizando transceptores de alta velocidad se pueden lograr distintas longitudes de bus (Tabla 2.1 y Figura 2.6). A velocidades de transferencia de datos menores a 1 Mbps, la longitud del bus puede incrementarse significativamente. El estándar ISO 11898 establece una longitud de bus máxima de 1 Km., pero permite el uso de puentes (bridge) y/o repetidores (repeaters) para incrementar la distancia entre los nodos.

Tabla 2.1. Velocidad de transferencia de datos y longitud de bus CAN.

Velocidad de transferencia [Kbps]	Longitud del bus [m]	Tiempo de bit nominal [μs]
1000	40	1
500	100	2
250	250	4
125	500	8
62.5	1000	20
20	2500	50

2.1.1.5. Tolerancia de oscilación

Como se mencionó en los incisos anteriores, cada nodo CAN deriva su señal de tiempo de bit de su propio oscilador. En sistemas reales, la frecuencia del oscilador f_{CLK} se desvía de su valor nominal debido al desplazamiento de tolerancia inicial, al envejecimiento de los componentes electrónicos y a las variaciones de la temperatura ambiental.

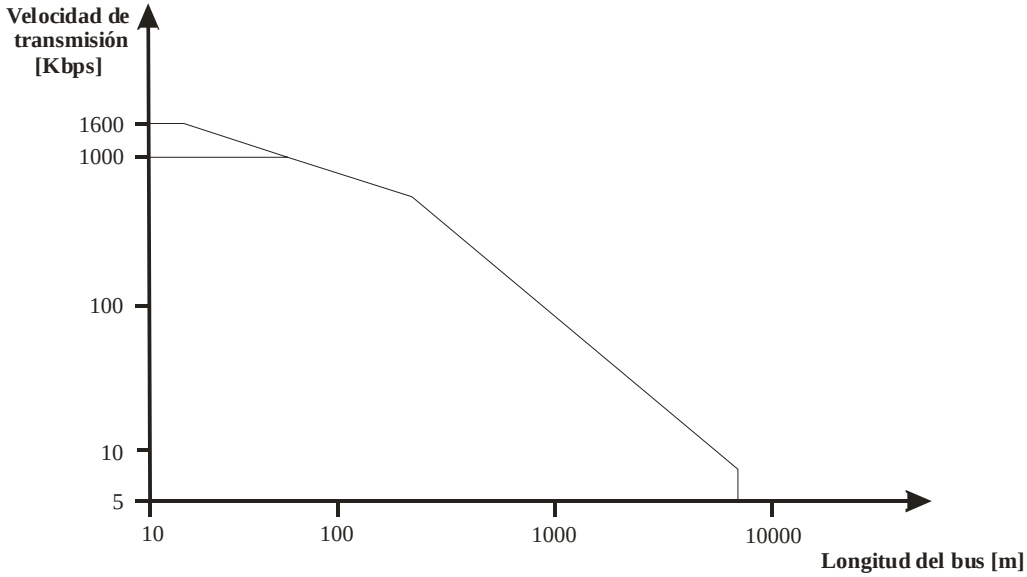


Figura 2.6. Relación entre velocidad de transferencia y longitud del bus.

La suma de desviaciones resulta en una tolerancia total del oscilador (Δf), la cual es una tolerancia relativa que representa la desviación de la frecuencia del oscilador normalizada a la frecuencia nominal (Ecuación 2.3).

$$\Delta f = \left| \frac{f_{CLK.\max/\min} - f_{CLK.nom}}{f_{CLK.nom}} \right| \qquad \text{Ecuación 2.3}$$

Así como la señal de reloj de un sistema CAN se deriva de la frecuencia del oscilador, Δf representa la tolerancia relativa de la señal de reloj del sistema; los valores mínimo y máximo para la duración del periodo del reloj del sistema se aproximan como definen las Ecuaciones 2.4 y 2.5.

$$t_{SCL\min} = \frac{t_{SCL.nom}}{1+\Delta f} \approx t_{SCL.nom}(1-\Delta f) \qquad \text{Ecuación 2.4}$$

$$t_{SCL\max} = \frac{t_{SCL.nom}}{1-\Delta f} \approx t_{SCL.nom}(1+\Delta f) \qquad \text{Ecuación 2.5}$$

Las aproximaciones que se establecen en las Ecuaciones 2.4 y 2.5 son válidas asumiendo un valor de $\Delta f << 1$. Las referencias de reloj utilizadas en sistemas reales, tales como osciladores de cristal ($\Delta f < 0.1\%$), frecuencias derivadas de PLL (*Phase Locked Loop*) ($\Delta f < 0.5\%$) y resonadores cerámicos ($\Delta f < 1.2\%$), satisfacen tal consideración [25].

La especificación CAN, establece una tolerancia máxima para el oscilador de 1.58%, y recomienda el uso de un resonador de cerámica en aplicaciones que necesiten una velocidad de transferencia menor a 125 Kbps [6].

Cuando se requiere de toda la gama de velocidades que establece el protocolo CAN, es necesario utilizar un oscilador de cuarzo. En una red CAN, la exactitud de oscilación requerida por los demás nodos está determinada por el circuito integrado que demande mayor precisión en su oscilador.

Los resonadores de cerámica se utilizan únicamente cuando todos los nodos de la red cumplen con la especificación del protocolo CAN posterior a la versión 1.2. En controladores de protocolo que cumplan con la versión 2.0 A/B y anteriores, se recomienda utilizar osciladores de cristal siempre y cuando pertenezcan a la misma red.

2.1.2. Subcapa de unión al medio físico

La conexión entre el controlador CAN y el medio físico se define en la subcapa de unión al medio físico (PMA, *Physical Medium Attachment*). Esta subcapa describe las características funcionales y eléctricas que debe cumplir el transceptor para hacer posible la transmisión/recepción entre un nodo y la red, además algunas de sus implementaciones en circuitos integrados o discretos proporcionan los medios para detectar fallos en el bus.

La interfaz eléctrica con el bus consiste principalmente de un transmisor y un receptor. La Figura 2.7 ilustra el diagrama a bloques básico de un transceptor en un nodo CAN, el controlador CAN provee funcionalidad básica de transceptor, que consiste de un comparador en la recepción y un amplificador en la salida.

Además de la adaptación de señales entre el controlador CAN y el bus, el transceptor tiene que cumplir con las siguientes funciones:

- Suministrar la capacidad de rendimiento al controlador CAN.
- Proteger al controlador CAN contra sobrecargas.
- Reducir la radiación electromagnética.

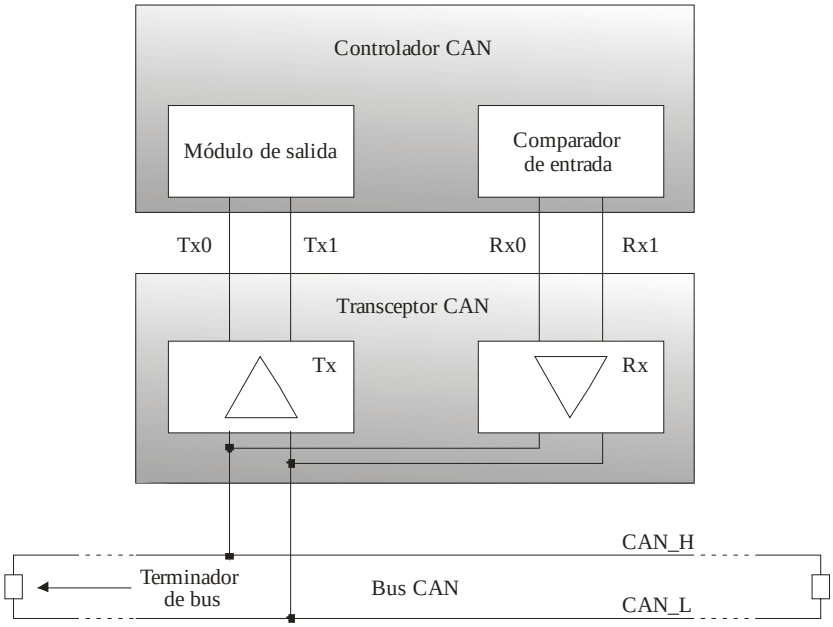


Figura 2.7. Arquitectura de un nodo CAN con transceptor.

Como receptor realiza las siguientes funciones:

- Suministrar un nivel de señal recesivo.
- Proteger el comparador de entrada del controlador CAN contra sobrevoltajes de las líneas del bus.
- Suministrar suficiente sensibilidad para un mejor reconocimiento de la señal de entrada.
- Detectar errores en el bus tales como: ruptura de la línea, corto-circuito y conmutación a operación asimétrica de una sola línea.

Una función opcional del transceptor es proporcionar aislamiento galvánico, al nodo CAN de las líneas de bus, mediante el empleo de optoacopladores.

2.1.3. Subcapa de interfaz dependiente del medio

La interfaz dependiente del medio (MDI, *Medium Dependent Interfaz*) define los aspectos eléctricos y mecánicos para la conexión entre el medio físico y el nodo.

Las especificaciones mecánicas de la capa física definen las características de la interfaz respecto al medio de transmisión y al tipo de conector.

2.1.3.1. Medio físico

Un requisito indispensable para realizar el método de arbitraje en el bus, es la capacidad de representar los niveles de señal recesivo y dominante. Dicho principio se realiza con medios eléctricos u ópticos, los cuales se describen a continuación.

2.1.3.1.1. Medio de transmisión eléctrico

La implementación de redes CAN generalmente utiliza como medio físico un bus de dos hilos con retorno común controlados diferencialmente (Figura 2.8). En aplicaciones específicas se utiliza un bus de un sólo hilo, por ejemplo en dispositivos electrónicos internos de un vehículo. En la actualidad se desarrollan soluciones en las que se transmiten las señales CAN junto con la fuente de alimentación en el mismo bus:

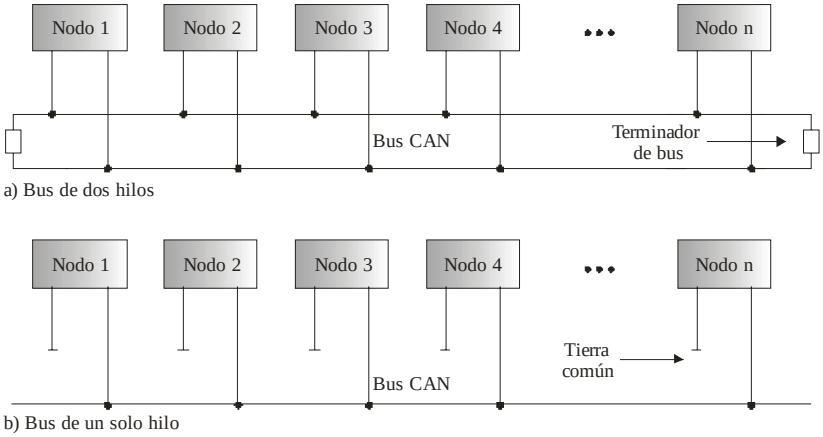


Figura 2.8. Medio de transmisión eléctrico.

- *Bus de dos hilos (two wire bus)*: permite una transmisión diferencial y es resistente a los errores de modo común (Figura 2.8a). Las líneas del bus deben contar con un terminador de bus en cada extremo, el cual consiste en resistores de 120 Ω para evitar la reflexión de la señal. Se garantiza la transmisión de una señal a pesar de sus niveles bajos, además, la interferencia de radio inducida de forma electromagnética se puede compensar por cables del tipo par trenzado. Si se implementa un manejo adecuado de los errores en el bus, es posible que la comunicación continúe en condiciones de inmunidad de ruido reducida aún si una línea se rompe o se encuentra en corto circuito.
- *Bus de un hilo (single wire bus)*: esta solución da por hecho que está disponible una tierra común para todos los nodos (Figura 2.8b) y sólo se implementa en aplicaciones automotrices para la interconexión de dispositivos electrónicos internos. El bus de un hilo está expuesto a la interferencia inducida eléctricamente cuando no está blindado, por lo tanto es necesario realizar un gran desplazamiento del nivel de la señal para mejorar la relación señal/ruido, lo cual afecta el grado de emisiones electromagnéticas y requiere la reducción de las caídas de la señal y por lo tanto de la máxima velocidad de transferencia de datos.
- *Transmisión de la señal y fuente de alimentación integradas en la misma línea*: muchos sistemas de FBs, como AS-I y PROFIBUS-PA[11], suministran la fuente de alimentación junto a la transmisión de datos sobre la misma línea del bus. El suministro de voltaje sobre el bus es conveniente en sistemas CAN, sin embargo no se recomienda la transmisión simultánea de alimentación y datos debido al arbitraje no destructivo bit a bit que utiliza dicho protocolo. Actualmente este tipo de transmisión se encuentra en fase de investigación y una posible solución es utilizar modulación OOK (*On-Off Keying*) para la transmisión de datos. Las investigaciones muestran que la transmisión simultanea de datos y alimentación implica la utilización de circuitos de alto costo, pérdida de inmunidad a la interferencia y reducción en la longitud del bus, por lo que esta solución no es competente frente a las ya existentes en el mercado.

2.1.3.1.2. Medio de transmisión óptico

Los avances relacionados con los medios de transmisión ópticos han obtenido gran importancia en el desarrollo de redes CAN, especialmente en el área de la fibra óptica. El comportamiento eléctricamente neutro de un medio óptico es ideal para aplicaciones en ambientes potencialmente explosivos y entornos electromagnéticos perturbados.

[11] AS-I (*Actuator-Sensor Interface*); PROFIBUS-PA (*PROFIBUS Profile for Process Automation*).

Las líneas de transmisión y recepción deben proporcionarse de forma separada, debido al acoplamiento directo en el medio óptico, además cada línea de recepción debe acoplarse externamente con cada línea de transmisión con el propósito de asegurar el monitoreo de bits que requiere el protocolo CAN, esta función puede implementarse por un acoplador de tipo estrella. El uso de acopladores pasivos tipo estrella es posible con una cantidad pequeña de nodos, lo cual limita la expansión de este tipo de redes.

2.1.3.2. Topología de una red CAN

Si no se consideran las medidas apropiadas, las señales eléctricas transmitidas en las líneas del bus CAN se reflejan al final de la línea eléctrica principal y en las líneas de extensión, por ello es necesario que las reflexiones sobrepuestas de la señal estén debidamente atenuadas cuando se muestrea el nivel de bit para interpretar los niveles de bus recibidos. Las reflexiones de la señal se pueden evitar al colocar resistores de terminación con una impedancia equivalente a la de la línea en ambos extremos del bus, y al evitar líneas de extensión de grandes longitudes. De esta forma, mediante una topología de bus se puede lograr el producto más alto de velocidad de transferencia y longitud de línea (Figura 2.8).

En aplicaciones industriales es común no conectar un nodo al bus mediante líneas de extensión muy cortas, sin embargo, con la configuración apropiada de los parámetros de los tiempos de bit, el punto de muestro de bit puede colocarse al final del tiempo de bit, y con ello se minimizan los efectos de reflexión de la señal.

En muchas aplicaciones es inevitable utilizar topologías extendidas, por ejemplo para conectar herramientas de diagnóstico o servicio. Para superar las limitaciones de la topología de bus CAN se emplean repetidores, puentes y pasarelas con la finalidad de adaptar la topología de red de acuerdo con las necesidades geográficas de cada aplicación específica.

2.1.4. Estándares de capa física

Como se mencionó en el subcapítulo 1.4, las redes CAN tienen diversas aplicaciones, sin embargo la especificación CAN [6] no define las características del transceptor, y deja abierta la posibilidad de implementar el medio de transmisión y los niveles de señales que más convengan al diseñador de la red. Por ello se han definido diferentes estándares, los cuales especifican: niveles de señales, topología de la red, máximo número de nodos, requisitos para el bus y conectores.

El estándar ISO 11898-2 destinado a la comunicación de alta velocidad en vehículos, es la norma más importante y CiA lo recomienda para aplicaciones industriales [7]. Las capas de aplicación CANopen [8] y DeviceNet especifican suplementos al estándar ISO 11898-2.

2.1.4.1. Estándar ISO 11898-2

El estándar ISO 11898-2 establece los fundamentos de una serie de estándares y es la norma de capa física más importante en aplicaciones industriales y automotrices. Dicha norma describe las funciones de la subcapa PMA, y algunas características de la subcapa MDI. Sus principales características son:

- Velocidades de transferencia de datos de hasta 1 Mbps.
- Longitud máxima del bus de 40 m@1 Mbps.
- El número total de nodos está limitado por la carga eléctrica en el bus.
- Bus diferencial de dos hilos.
- Seguridad contra corto circuito en el rango de voltaje comprendido entre -3 a +16 V.
- Impedancia característica de la línea de 120 Ω.

- Rango de voltaje en modo común desde -2 V (CAN_L) a +7 V (CAN_H).

El estándar ISO 11898-2 define un bus de dos hilos terminado en ambos extremos con la impedancia de línea específica del medio físico y con las siguientes características eléctricas:

- Longitud máxima de línea de extensión (*stub*): 30 cm.
- Resistencia de línea nominal por metro: 70 mΩ/m.
- Retardo de propagación nominal: 5 ns/m.

El bus de dos hilos especificado en el estándar ISO 11898, contiene dos señales, CAN_H y CAN_L, en donde todo nodo CAN debe proporcionar el voltaje diferencial de salida en el bus (Ecuación 2.6):

$$V_{diff} = V_{CAN\_H} - V_{CAN\_L} \qquad \text{Ecuación 2.6}$$

- Bit recesivo: - 500 mV a 500 mV, sin carga.
- Bit dominante: +1.5 V a 3.0 V, con una carga de 60 Ω.

Los nodos deben detectar los dos estados posibles del bus (Tabla 2.2 y Figura 2.9).

Tabla 2.2. Niveles nominales de voltaje en el bus.

Señal CAN		**Niveles nominales de voltaje [V]**		
		CAN_H	**CAN_L**	V_{diff}
Estado de bus	Dominante	3.5	1.5	0.9 – 2.0
	Recesivo	2.5	2.5	0 – 0.5

CAN utiliza la función *Wired-AND*, la cual consiste en detectar que un nodo transmita un bit dominante para que el bus pase al estado dominante; sólo en el caso que todos los nodos de la red transmitan bits recesivos, el bus deberá pasar al estado recesivo (Tabla 2.3 y Figura 2.10). Es decir, que un bit recesivo representa un estado de bus que puede ser sobrescrito por un bit dominante.

Tabla 2.3. Lógica del bus CAN.

Nodo 1	**Nodo 2**	**Nodo 3**	**Bus**
d	d	d	d
d	d	r	d
d	r	d	d
d	r	r	d
r	d	d	d
r	d	r	d
r	r	d	d
r	r	r	r

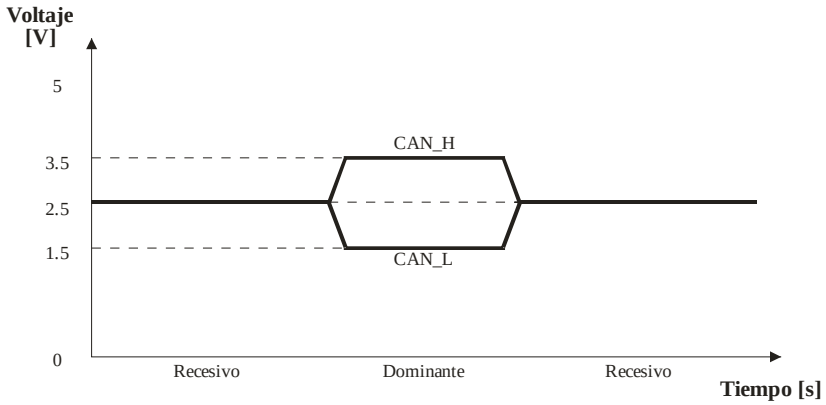


Figura 2.9. Niveles nominales de las señales CAN recomendados por el estándar ISO 11898.

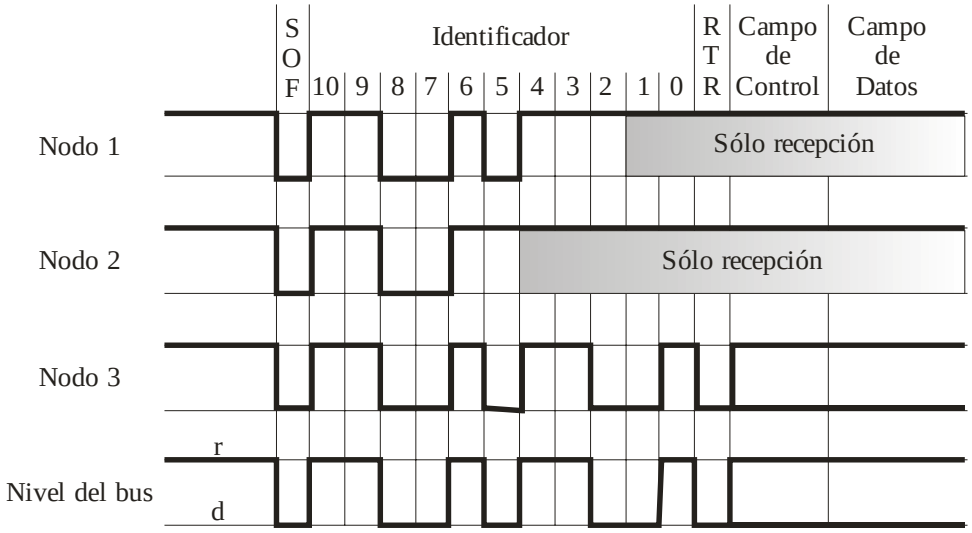


Figura 2.10. Lógica del bus CAN.

Según el estándar ISO 11898-2, el acceso al medio se realiza mediante un transceptor de alta velocidad, el cual además realiza otras tareas como obtener las medidas de protección requeridas en vehículos de motor. Un transceptor debe cumplir con las siguientes características:

- Compatible con la norma ISO 11898-2.
- Soportar velocidades de transferencia de datos de hasta 1 Mbps.
- Proteger contra sobrecarga térmica.
- Proteger contra corto circuito, tanto de tierra como de fuente de voltaje.
- Consumir poca corriente en modo de espera (*standby*).
- Protección contra perturbaciones de la línea del bus.

Para que un nodo cumpla con el estándar ISO 11898-2 debe contar con un MCU y un controlador CAN, los cuales se conectan al bus mediante un transceptor a través de dos líneas, una línea de salida de datos (Tx) y otra línea de entrada de datos (Rx) (Figura 2.7).

Las líneas Tx y Rx manejan niveles TTL (*Transistor-Transistor Logic*) y son unidireccionales; las líneas CAN_H y CAN_L se utilizan para enlazar el transceptor al bus. La Figura 2.11 muestra los niveles nominales de las señales CAN que se presentan en un transceptor CAN de alta velocidad.

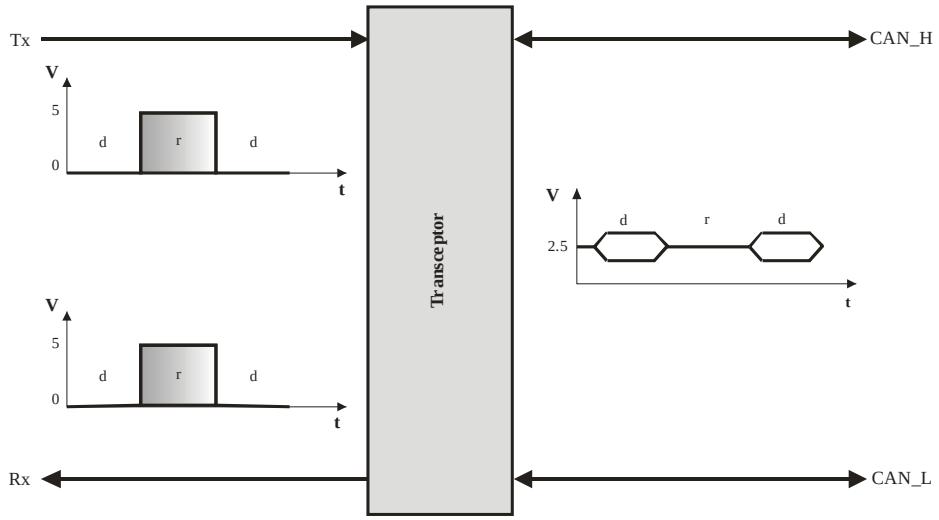


Figura 2.11. Niveles nominales de las señales presentes en un transceptor CAN.

2.1.4.2. Recomendación CiA DS-102

Respecto a la capa física, la recomendación DS-102 brinda soporte a la realización de redes CANopen para la comunicación entre diversos tipos de módulos electrónicos en aplicaciones industriales [7]. Dicha recomendación está basada en los estándares ISO 11898-1 y 11898-2, y define lo siguiente:

- Velocidades de transferencia estandarizadas desde 10 Kbps a 1 Mbps, con recomendaciones para la configuración de los parámetros de tiempos de bit (Tablas 2.4 y 2.5).
- Recomendaciones para la línea de bus.
- Características mecánicas y eléctricas de los conectores, así como la asignación de sus terminales (Figura 2.12).

Tabla 2.4. Velocidades de transferencia y parámetros de tiempos de bit recomendados en DS-102.

Velocidad de transferencia [Kbps]	Longitud de línea máxima [m]	Tiempo de bit nominal [µs]	Número de t_q	Duración de t_q [ns]	Punto de muestreo [t_q]
1000	25	1	8	125	6
800	50	1.25	10	125	8
500	100	2	16	125	14
250	250	4	16	250	14
125	500	8	16	500	14
50	1000	20	16	1250	14
20	2500	50	16	3125	14
10	5000	100	16	6250	14

Tabla 2.5. Parámetros de tiempos de bit recomendados por DS-102.

Frecuencia del oscilador:	16 Mhz +/- 0.1%
Método de muestreo:	Muestreo sencillo
Modo de sincronización:	Sólo flancos recesivos a dominantes
Duración de *SJW:**	1 t_q
Duración de *Phase_Seg2*:	2 t_q

* Parámetro equivalente al RJW definido en el inciso 2.1.1.3.

De acuerdo a la recomendación DS-102, se implementa un par trenzado blindado o no blindado terminado en ambos extremos con una impedancia equivalente a la de la línea y tierra común, asimismo especifica el uso de repetidores para incrementar el número de nodos en el bus.

Es posible el aislamiento galvánico de transceptores, sin convertidores DC/DC propios, mediante una línea de fuente de alimentación común.

Existen dos conceptos básicos que indican cómo implementar la línea de bus, línea de bus interconectada y línea de bus dividida, los cuales se describen a continuación.

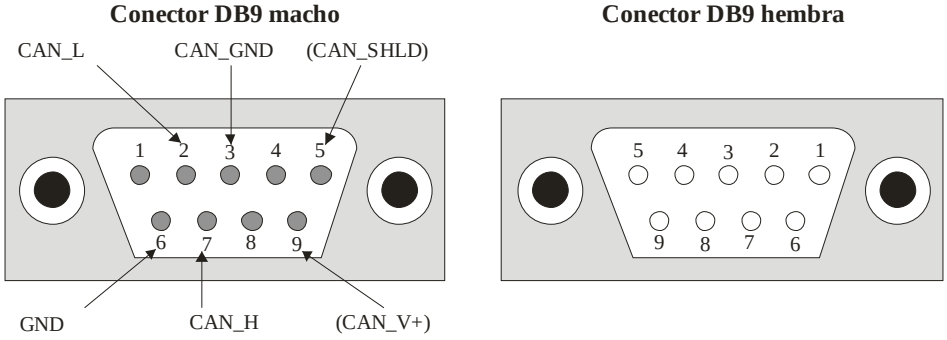


Figura 2.12. Asignación de terminales en el conector CAN de acuerdo a la recomendación DS-102.

2.1.4.2.1. Línea de bus interconectada

La línea de bus consiste de un número de secciones interconectadas mediante dos opciones (Tabla 2.6):

- *Opción A*: se utiliza un conector tipo T[12] para interconectar las secciones de línea de bus y el bus con el nodo.
- *Opción B*: los módulos electrónicos están equipados con dos conectores para el bus, interconectando las secciones de línea de bus.

Tabla 2.6. Línea de bus interconectada de acuerdo a la recomendación DS-102.

Opción	A	B
Nodo	1 conector macho	1 conector macho y 1 conector hembra
Conector tipo T	1 conector macho y 2 conectores hembras	No requerido
Sección de línea de bus	1 conector macho y 1 conector hembra	1 conector macho y 1 conector hembra

2.1.4.2.2. Línea de bus no dividida

La línea de bus consiste de un sólo cable sin dispositivos interconectados:

- *Nodo*: un conector macho.
- *Línea de bus*: un conector hembra por nodo, además un conector hembra y un conector macho para terminación.

2.1.4.3. Recomendaciones de capa física por los estándares HLP

La meta más importante que se intenta al estandarizar los protocolos de capas altas (HLP, *High Layer Protocol*) y los perfiles de aplicación, tales como CANopen y DeviceNet, es la interoperabilidad e intercambiabilidad de dispositivos, lo cual asegura la compatibilidad total de las propiedades físicas de los dispositivos y de la red. Por lo tanto, además de las características básicas de la capa física definida en el estándar ISO 11898, se define: topología de red, tipo de línea, tecnología de conexión, fuente de alimentación, velocidad de transferencia de datos y transceptor.

2.1.4.3.1. CANopen

CANopen permite la implementación de redes, hasta con 127 nodos, cumpliendo con el estándar ISO 11898-2 (inciso 2.1.4.1) y la recomendación DS-102 (inciso 2.1.4.2). El aislamiento galvánico de nodos es opcional y sólo se recomienda para buses con longitudes mayores a 200 m.

Todos los dispositivos CANopen deben dar soporte a la velocidad de transferencia de 20 Kbps y a las longitudes de línea recomendadas en DS-102.

CANopen permite utilizar una fuente de alimentación común y recomienda el uso de los siguientes conectores [8]:

- Conector DB9 (recomendado en DS-102, Figura 2.12).
- Conector tipo mini (*mini style*) de cinco terminales (Tabla 2.7 y Figura 2.13).
- Conector multipolo (*multipole*) (Tabla 2.8 y Figura 2.14).
- Conector tipo abierto (*open style*) (Tabla 2.9 y Figura 2.15).
- Conector tipo micro (*micro style*) (Tabla 2.10 y Figura 2.16).
- Conector RJ10/RJ45 (Tabla 2.11 y Figura 2.17).
- Conector DIN redondo (*round connector)* de 7 u 8 terminales (Tabla 2.12 y Figura 2.18).

[12] Un conector tipo T es un dispositivo pasivo que enlaza tres diferentes conectores.

Tabla 2.7. Asignación de terminales del conector tipo mini para CANopen.

Pin	Señal	Descripción
1	CAN_SHLD	Blindaje (opcional)
2	CAN_V+	Fuente de alimentación externa (opcional)
3	CAN_GND	Señal de tierra (GND)
4	CAN_H	Señal de bus CAN (dominante alto)
5	CAN_L	Señal de bus CAN (dominante bajo)

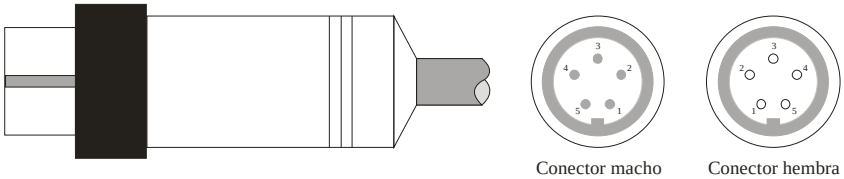


Figura 2.13. Conector tipo mini.

Tabla 2.8. Asignación de terminales del conector multipolo para CANopen.

Pin	Señal	Descripción
1	-	Futuras aplicaciones
2	(GND)	Tierra (opcional)
3	CAN_L	Señal de bus CAN (dominante bajo)
4	CAN_H	Señal de bus CAN (dominante alto)
5	CAN_GND	Señal de tierra (GND)
6	-	Futuras aplicaciones
7	-	Futuras aplicaciones
8	CAN_V+	Fuente de alimentación externa (opcional)
9	-	Futuras aplicaciones
10	-	Futuras aplicaciones

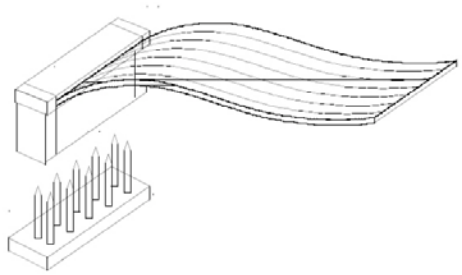


Figura 2.14. Conector multipolo.

Tabla 2.9. Asignación de terminales del conector tipo abierto para CANopen.

Pin	Señal	Descripción
1	CAN_GND	Señal de tierra (GND)
2	CAN_L	Señal de bus CAN (dominante bajo)
3	CAN_SHLD	Blindaje (opcional)
4	CAN_H	Señal de bus CAN (dominante alto)
5	CAN_V+	Fuente de alimentación externa (opcional)

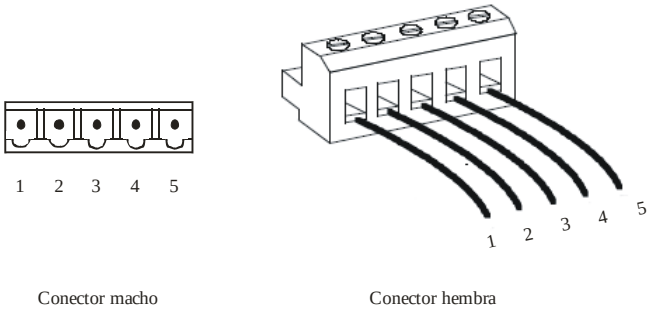


Figura 2.15. Conector tipo abierto.

Tabla 2.10. Asignación de terminales del conector tipo micro para CANopen.

Pin	Señal	Descripción
1	(CAN_SHLD)	Blindaje (opcional)
2	(CAN_V+)	Fuente de alimentación externa (opcional)
3	CAN_GND	Señal de tierra (GND)
4	CAN_H	Señal de bus CAN (dominante alto)
5	CAN_L	Señal de bus CAN (dominante bajo)

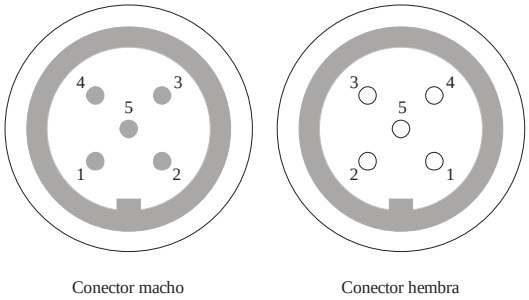


Figura 2.16. Conector tipo micro.

Tabla 2.11. Asignación de terminales de los conectores RJ10 y RJ45 para CANopen.

Pin	Conector RJ10		Conector RJ45	
	Señal	Descripción	Señal	Descripción
1	(CAN_V+)	Fuente de alimentación externa (opc)	CAN_H	Señal de bus CAN (dominante alto)
2	CAN_H	Señal de bus CAN (dominante alto)	CAN_L	Señal de bus CAN (dominante bajo)
3	CAN_L	Señal de bus CAN (dominante bajo)	CAN_GND	Señal de tierra (GND)
4	CAN_GND	Señal de tierra (GND)	-	Futuras aplicaciones
5			-	Futuras aplicaciones
6			(CAN_SHLD)	Blindaje (opcional)
7			(GND)	Tierra (opcional)
8			(CAN_V+)	Fuente de alimentación externa (opc)

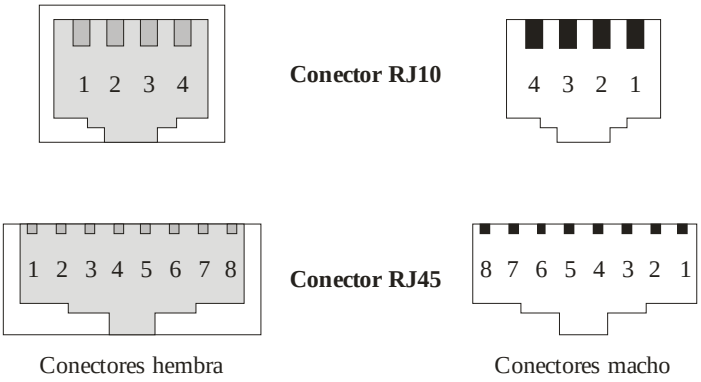


Figura 2.17. Conectores RJ10 y RJ45.

Tabla 2.12. Asignación de terminales para los conectores DIN redondo para CANopen.

Pin	Conector DIN redondo 7 terminales		Conector DIN redondo 8 terminales	
	Señal	Descripción	Señal	Descripción
1	(CAN_V+)	Fuente de alimentación externa (opc.)	CAN_V+	Fuente de alimentación externa
2	CAN_GND	Señal de tierra (GND)	GND	0 V
3	CAN_H	Señal de bus CAN (dominante alto)	CAN_H	Señal de bus CAN (dominante alto)
4	CAN_L	Señal de bus CAN (dominante bajo)	CAN_L	Señal de bus CAN (dominante bajo)
5	DIL*-1	Interruptor 1 conectado a CAN_V+	CAN_GND	Señal de tierra (GND)
6	DIL-2	Interruptor 2 conectado a CAN_V+	-	Futuras aplicaciones
7	DIL-3	Interruptor 3 conectado a CAN_V+	-	Futuras aplicaciones
8			-	Futuras aplicaciones

*DIL (*Dual In Line*)

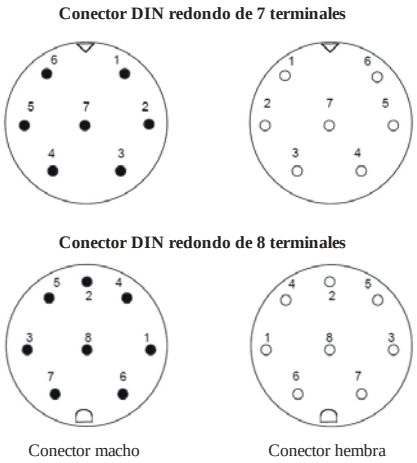


Figura 2.18. Conector tipo DIN redondo de 7 y de 8 terminales.

2.1.4.3.2. DeviceNet

De la misma forma que CANopen, la especificación de capa física que implementa DeviceNet es una extensión del estándar ISO 11898. DeviceNet permite redes de hasta 64 nodos con topología de bus y sus líneas de bus están terminadas en ambos extremos con una impedancia R_t=121 Ω +/- 1%.

DeviceNet define dos pares de cables, uno para la transmisión de la señal y el otro para la fuente de alimentación. Se permiten tres tipos de cable variantes en diámetro (grueso, delgado y plano), los cables grueso y plano se utilizan en redes de longitud mayor a 100 m., y el cable delgado se utiliza en la conexión del nodo al bus (*stub*) y en redes de longitudes menores a 100 m (Tabla 2.13). Es obligatorio el empleo del código de colores para hilos individuales.

Tabla 2.13. Extensión de red y longitud de línea de extensión para DeviceNet.

Velocidad de transferencia [Kbps]	Longitud de la línea principal [m]			Longitud máxima de la línea de extensión [m]	
	Cable grueso	Cable delgado	Cable plano	Acumulada	Individual
125	500	100	420	156	6
250	250	100	200	78	6
500	100	100	100	39	6

De acuerdo a la especificación de DeviceNet, el protocolo brinda soporte a tres diferentes velocidades de transferencia de datos: 125, 250 y 500 Kbps. Las conexiones del bus y los dispositivos electrónicos pueden alimentarse de la fuente de voltaje común de 24 V.

DeviceNet define tres tipos de conectores:

- Tipo micro (Tabla 2.13Tabla 2.14 y Figura 2.16).
- Tipo mini (Tabla 2.15 y Figura 2.13).
- Tipo abierto (Tabla 2.16 y Figura 2.15).

Tabla 2.14. Asignación de terminales del conector tipo micro para DeviceNet.

Pin	Señal	Descripción	Color
1	Drain	Blindaje	-
2	V+	Fuente de alimentación externa	Rojo
3	V-	Tierra	Negro
4	CAN_H	Señal de bus CAN (dominante alto)	Blanco
5	CAN_L	Señal de bus CAN (dominante bajo)	Azul

Tabla 2.15. Asignación de terminales del conector tipo mini para DeviceNet.

Pin	Señal	Descripción	Color
1	Drain	Blindaje	-
2	V+	Fuente de alimentación externa	Rojo
3	V-	Tierra	Negro
4	CAN_H	Señal de bus CAN (dominante alto)	Blanco
5	CAN_L	Señal de bus CAN (dominante bajo)	Azul

Tabla 2.16. Asignación de terminales del conector tipo abierto para DeviceNet.

Pin	Señal	Descripción	Color
1	V-	Tierra	Negro
2	CAN_L	Señal de bus CAN (dominante bajo)	Azul
3	Drain	Blindaje	-
4	CAN_H	Señal de bus CAN (dominante alto)	Blanco
5	V+	Fuente de alimentación externa	Rojo

La especificación de DeviceNet presenta ciertas recomendaciones en cuanto a la protección de las conexiones. Opcionalmente la conexión al bus puede aislarse galvánicamente y la corriente necesaria para alimentar a los transceptores puede tomarse de la fuente de alimentación común.

2.1.4.4. Estándar ISO 11898-3

El estándar ISO 11898-3 define una capa física CAN de baja velocidad tolerante a fallos, destinada a aplicaciones con escasos requerimientos en cuanto a velocidad de transferencia y longitud de bus [22, 23]. Fue desarrollado por las firmas Philips y Bosch con la finalidad de reemplazar al estándar ISO 11519.

Este estándar está dirigido principalmente a la realización de redes de unidades electrónicas que aumentan la comodidad en un automóvil (*comfort electronics*).

Si se considera una red eléctricamente corta, no se considera el problema de reflexión de la señal y puede utilizarse una línea de bus abierta. Lo anterior significa que pueden utilizarse controladores de baja potencia, y con ello realizarse redes de bajo consumo. Otra característica es la topología de red, la cual no se restringe a una estructura lineal con longitudes de extensión cortas para la conexión entre los nodos y el bus. Asimismo existe la posibilidad de transmitir datos asimétricamente sobre un sólo hilo del bus en caso de ocurrir una falla eléctrica en sus líneas.

Los parámetros más importantes de este estándar son los siguientes:

- Velocidades de transferencia de datos de hasta 125 Kbps.
- Redes de hasta 32 nodos.
- Transmisión de señal simétrica mediante par trenzado.
- Prueba de corto circuito en un rango de voltaje de -6 V a +16 V.
- Corriente de salida del transmisor mayor a 1 mA.
- Rango de voltaje en modo común entre -2 V a +7 V.
- Fuente de alimentación de 5 V.

En la Figura 2.19 se muestran los niveles nominales de las señales CAN en especificados en el estándar ISO 11898-3. Los niveles de la señal en el bus son diferentes para el nivel recesivo y para el nivel dominante, y de esta forma dichos niveles del bus pueden detectarse al compararse con un voltaje de referencia de 2.5 V.

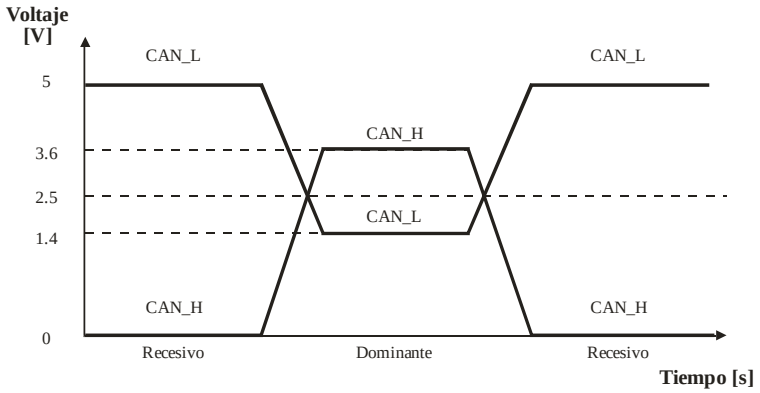


Figura 2.19. Niveles nominales de voltaje en el bus de acuerdo al estándar ISO 11898-3.

El estándar ISO 11898-3 considera la detección y el manejo de las siguientes condiciones de error en el bus:

- Interrupción de la señal CAN_H.
- Interrupción de la señal CAN_L.
- Corto circuito de la señal CAN_H con V_{cc}.
- Corto circuito de la señal CAN_L con GND.
- Corto circuito de la señal CAN_H con GND
- Corto circuito de la señal CAN_L con V_{cc}.
- Corto circuito de la señal CAN_H con la señal CAN_L.

Las distintas condiciones de error pueden detectarse al comparar ambas líneas del bus CAN, y detectando el nivel dominante a través de un tiempo fuera (*timeout*) al deshabilitar la línea defectuosa y conmutar a operación asimétrica; la comunicación puede continuar sobre la línea sobrante, sin embargo, se pierde la inmunidad a la interferencia eléctrica.

Una solución de interfaz de bus que cumpla con el estándar ISO11898-3 puede realizarse mediante un transceptor modelo TJA 1054 de la firma Philips (Figura 2.20).

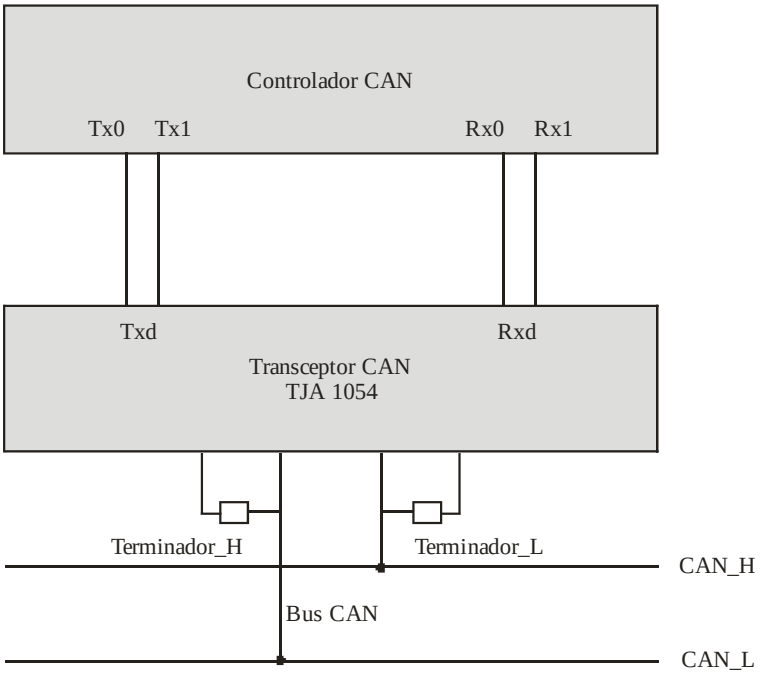


Figura 2.20. Interfaz para bus CAN utilizando un transceptor TJA 1054 de la firma Philips.

Dicho transceptor da soporte total al manejo de errores, incluyendo la detección de errores en el bus y la conmutación automática al modo de transmisión asimétrico. Cuando la condición de error deja de existir, automáticamente se regresa al modo de transmisión simétrico. Además el transceptor contiene un filtro de interferencia integrado en el módulo de recepción y un modo de ahorro de corriente, el cual habilita la conexión de componentes adicionales que pueden agregarse cuando el bus está activo.

2.1.4.5. Estándar SAE J2411

El estándar de un sólo hilo SAE J2411 (*single-wire CAN*)[13] se aplica en redes CAN con requisitos de velocidad de transferencia y longitud de bus menores a los establecidos por el estándar ISO 11898-3. Los parámetros básicos de este estándar son los siguientes:

- Bus de comunicación de un sólo hilo.
- Velocidad de transferencia nominal de $33^1/_3$ Kbps en modo normal.
- Alta velocidad de transferencia de datos para propósitos de diagnóstico de $83^1/_3$ Kbps.
- Hasta 32 nodos por red.
- Modo de temporizador selectivo.

La principal aplicación del estándar SAE J2411 está enfocada a la realización de redes de ECUs que aumentan la comodidad en un automóvil (subcapítulo 1.2).

El medio físico se define como un cable de un sólo hilo no blindado. Debido a la baja velocidad de transferencia, la topología del bus no está limitada a una estructura lineal con longitudes cortas de la línea de conexión del nodo con el bus. La Figura 2.21 muestra los niveles nominales de la señal CAN en el bus, en modo normal.

El estándar SAE J2411 puede realizarse utilizando un transceptor modelo TLE 6255 de la firma Infineon Semiconductors.

2.1.4.6. Estándar ISO 11992

El estándar ISO 11992 es una alternativa de red CAN para baja velocidad basado en conexiones punto a punto, utilizada en el control de camiones de carga (*truck/trailer*) y vehículos remolcadores o grúas (*towing vehicles*). Los parámetros básicos de dicho estándar son:

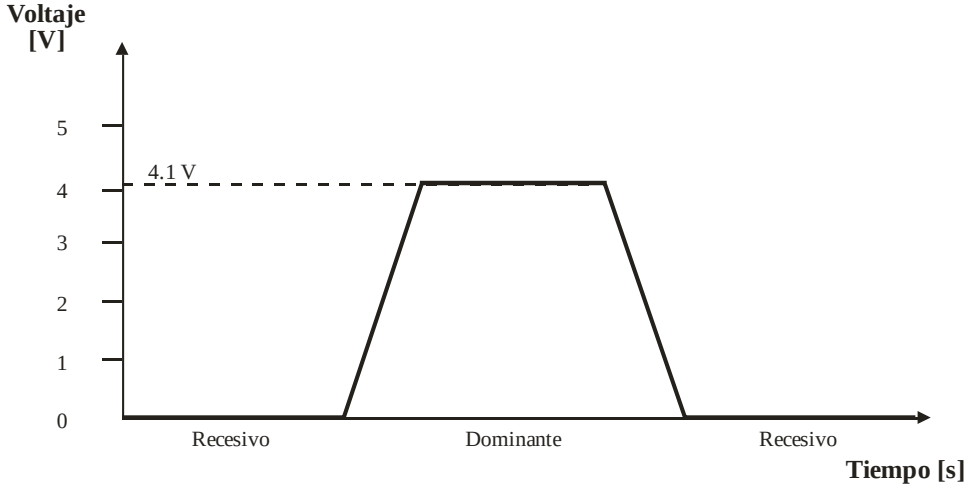


Figura 2.21. Niveles nominales de la señal CAN en el bus de acuerdo al estándar SAE J2411.

[13] Desarrollado originalmente por la firma GM, como una alternativa al bus UART de un sólo hilo (*single-Wire UART bus*) de acuerdo a la norma SAE J1850.

- Conexiones punto a punto para vehículos con un remolque.
- Conexión lineal (*daisy-chain*) para vehículos con dos remolques.
- Velocidad de transferencia nominal de 125 Kbps.
- Longitud de línea máxima de 40 m.
- Transmisión simétrica sobre un par de hilos con línea de tierra y fuente de alimentación.
- Manejo de errores de bus.
- Voltaje de alimentación V_{cc} = 12 V o 24 V.

El medio físico está definido como un cable par trenzado y no se especifican resistores de terminación en el bus debido a la corta longitud del mismo y su baja velocidad de transferencia.

El estándar ISO 11992 especifica la capacitancia máxima por unidad de longitud, la impedancia de la línea, las capacitancias de contacto máximas e impedancias de conexión para los conectores, las condiciones extremas del ambiente para el cable, tipo de conectores y los ciclos de conexión para intercambio de los remolques.

La Figura 2.22 muestra los niveles nominales de las señales CAN del bus de acuerdo al estándar ISO 11992. Los niveles nominales en el bus dependen de suministro de voltaje V_{cc} de los vehículos. Los niveles nominales de las señales para 12 V y 24 V se muestran en la Tabla 2.17.

Tabla 2.17. Niveles nominales de las señales CAN en el bus recomendados por el estándar ISO 11992.

Nivel	Señal	Voltaje del vehículo	
		12 V	24 V
Dominante	CAN_H	6.0 a 10.6 V	10.6 a 21.3 V
	CAN_L	3.0 a 5.3 V	5.3 a 10.6 V
Recesivo	CAN_H	3.0 a 5.3 V	5.3 a 10.6 V
	CAN_L	6.0 a 10.6 V	10.6 a 21.3 V

El estándar ISO 11992 también especifica el manejo de errores del bus al activar el cambio a transmisión de señal asimétrica cuando se detecta un error de bus físico. La comunicación se puede mantener por operación asimétrica en caso de ruptura, corto circuito con V_{cc}, con tierra o entre las líneas. Durante la emisión de línea sencilla se verifica el modo de transmisión de señal, para asegurar que en funcionamiento normal, o después de corregir un error de bus, el modo de transmisión sea simétrico.

La Tabla 2.18 lista los principales aspectos de los estándares analizados anteriormente.

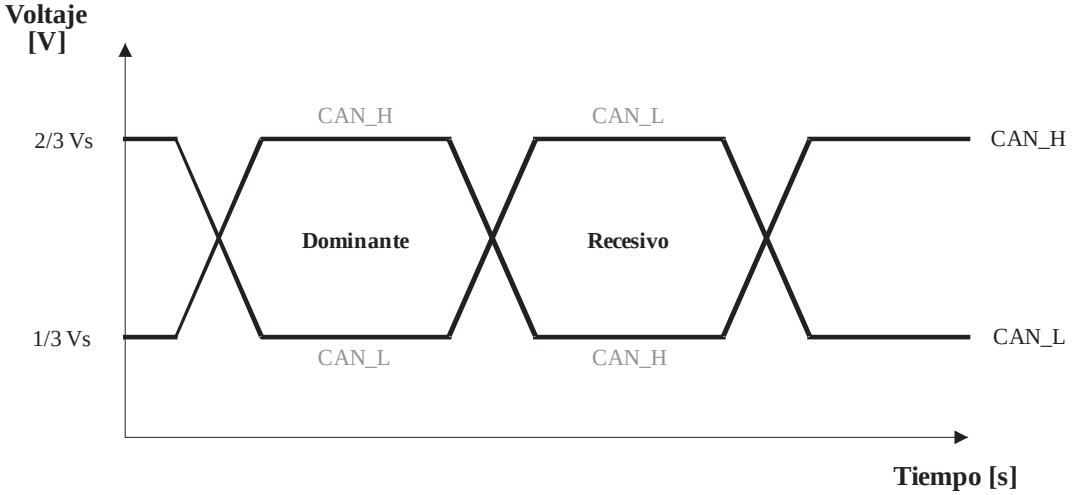


Figura 2.22. Niveles nominales de las señales CAN en el bus de acuerdo con el estándar ISO 11992.

Tabla 2.18. Comparación de los diferentes estándares para implementar la capa física del bus CAN.

Estándar	Velocidad de transferencia vs Longitud de bus	Tipo de transmisión	Número máximo de nodos	Conectores	Tipo de bus	Impedancia de las terminaciones	Transceptor comercial	Características destacadas
ISO 11898-2	Máx. 1 Mbps @ 40 m	Diferencial	Limitado por la carga eléctrica máxima del bus	–	2 hilos (par trenzado)	120 Ω	Philips PCA 82C250	Es el más implementado y es la base para los demás estándares
CiA DS-102	10 Kbps @ 5,000 m – 1Mbps @ 25 m	Diferencial	–	DB9	2 hilos (par trenzado)	Igual a la de la línea	–	–
CANopen	10 Kbps @ 5,000 m – 1Mbps @ 25 m	Diferencial	127	DB9 (DS-102) Tipo mini Tipo abierto Multipolo Tipo micro RJ10/RJ45 DIN redondo	2 hilos (par trenzado)	120 Ω	–	Suministro de voltaje opcional en 2 hilos independientes sobre el bus
DeviceNet	125 Kbps @ 500 m, 250 Kbps @ 250 m, 500 Kbps @ 100 m	Diferencial	64	Tipo mini Tipo abierto Tipo micro	2 hilos p/señal CAN, y 2 hilos p/V_{cc} y Tierra	121 Ω +/-1%	–	Codificación individual de cables con colores
ISO 11898-3	Máx. 125 Kbps	Asimétrica y simétrica	32	–	2 hilos	Sin terminaciones	Philips TJA 1054	Detección y manejo de condiciones de error en el bus
SAE J2411	$33^1/_3$ Kbps	–	32	–	1 hilo no blindado	–	Infineon TLE 6255	Modo de operación a $83^1/_3$ Kbps para propósitos de diagnóstico
ISO 11992	125 Kbps @ 40 m	Simétrica con cambio a asimétrica	–	–	2 hilos p/señal CAN, y 2 hilos p/V_{cc} y Tierra	Sin terminaciones	Atmel B10011S	Manejo de errores en el bus

2.2. Capa de enlace de datos

En este apartado se describen los fundamentos y principios básicos de la capa de enlace de datos (DLL, *Data Link Layer*), la cual se divide en dos subcapas: control de enlace lógico (LLC, *Logic Link Control*) y control de acceso al medio (MAC, *Medium Access Control*).

2.2.1. Control de enlace lógico

La subcapa LLC describe la parte alta de la capa DLL y define las tareas independientes del método de acceso al medio, asimismo proporciona dos tipos de servicios de transmisión sin conexión al usuario LLC (*LLC user*):

- *Servicio de transmisión de datos sin reconocimiento*: proporciona, al usuario LLC, los medios para intercambiar unidades de datos de servicio de enlace (LSDU, *Link Service Data Units*) sin establecer una conexión de enlace de datos. La transmisión de datos puede ser punto a punto, multidifusión o difusión.
- *Servicio de petición de datos remota sin reconocimiento*: proporciona, al usuario LLC, los medios para solicitar que un nodo remoto transmita sus LDSUs sin establecer una conexión de enlace de datos.

De acuerdo con los tipos de servicios, se definen dos formatos de tramas, de datos LLC y remota LLC. Ambos formatos definen identificadores de 11 bits (estándar) y de 29 bits (extendida), los cuales se describen en el inciso 2.2.2.1.

2.2.1.1. Funciones de la subcapa LLC

La subcapa LLC realiza las siguientes funciones:

- *Filtrar mensajes (frame acceptance filtering)*: el identificador de una trama no indica la dirección destino pero define el contenido del mensaje, y mediante esta función todo receptor activo en la red determina si el mensaje es relevante o no para sus propósitos.
- *Notificar sobrecarga (overload notification)*: si las condiciones internas de un receptor requieren un retraso en la transmisión de la siguiente trama de datos o remota, la subcapa LLC transmite una trama de sobrecarga. Solamente se pueden generar dos tramas de sobrecarga como máximo.
- *Proceso de recuperación (recovery management)*: la subcapa LLC proporciona la capacidad de retransmisión automática de tramas cuando una trama pierde el arbitraje o presenta errores durante su transmisión, dicho servicio se confirma al usuario hasta que la transmisión se completa con éxito.

2.2.2. Control de acceso al medio

Una red CAN brinda soporte para procesamiento en tiempo real a todos los sistemas que la integran. El intercambio de mensajes que demanda dicho procesamiento requiere de un sistema de transmisión a frecuencias altas y retrasos mínimos [13]. En redes multimaestro, la técnica de acceso al medio es muy importante ya que todo nodo activo tiene los derechos para controlar la red y acaparar sus recursos.

Para acceder al medio, los nodos CAN utilizan el mecanismo de arbitraje que se describe a continuación (Figura 2.10):

- Cuando un nodo necesita enviar información a través de una red CAN, la capa de aplicación realiza una petición, de forma asíncrona, para transmitir una trama. Puede ocurrir que varios nodos inicien el mismo procedimiento simultáneamente. CAN resuelve lo anterior al asignar prioridades mediante el identificador de cada mensaje, donde dicha asignación

se realiza durante el diseño del sistema en forma de números binarios y no puede modificarse dinámicamente. El identificador con el menor número binario es el que tiene mayor prioridad.

- El método de acceso al medio utilizado es el de acceso múltiple por detección de portadora, con detección de colisiones y arbitraje por prioridad del mensaje (CSMA/CD+AMP, *Carrier Sense Multiple Access with Collision Detection and Arbitration Message Priority*). De acuerdo con este método, los nodos en la red que necesitan transmitir información deben esperar a que el bus esté libre (detección de portadora); cuando se cumple esta condición, dichos nodos transmiten un bit de inicio (acceso múltiple). Cada nodo lee el bus bit a bit durante la transmisión de la trama y comparan el valor transmitido con el valor recibido; mientras los valores sean idénticos, el nodo continúa con la transmisión; si se detecta una diferencia en los valores de los bits, se lleva a cabo el mecanismo de arbitraje.
- De acuerdo a la Figura 2.10, durante la transmisión del bit 5, el nodo 1 transmite un bit dominante y el nodo 2 transmite un bit recesivo; el nivel dominante sobrescribe al recesivo; ambos nodos comparan los valores transmitidos con los que reciben, el nodo 2 detecta una diferencia (detección de colisión), ha perdido el arbitraje e inmediatamente deja de transmitir para conmutar a modo de recepción; mientras tanto el nodo 1 continua transmitiendo (AMP) hasta que se presenta la misma situación en la transmisión del bit 2. Al final, el nodo 3 es quién gana el arbitraje y transmite su trama. Lo anterior se conoce como arbitraje no destructivo bit a bit, es decir que a pesar de que varios nodos inicien la transmisión de sus tramas al mismo tiempo, el ganador del arbitraje, el mensaje con la mayor prioridad, continua con la transmisión de su trama sin necesidad de retransmitirla desde el principio.
- Si lo anterior se detecta en un campo distinto al bit de inicio, campo de arbitraje y ranura ACK, se activa el proceso de control de errores (inciso 2.2.2.4) por parte del administrador del nodo, quien lo considera un error de bit [27].
- Si se inicia simultáneamente la transmisión de una trama de datos (párrafo 2.2.2.1.1) y una trama remota (párrafo 2.2.2.1.2), solicitada por un receptor, el proceso de arbitraje no puede resolverse únicamente con el identificador de la trama, sino que tiene que utilizarse el bit RTR.

El principio de arbitraje utilizado en CAN limita la extensión máxima de la red a una velocidad de transferencia de datos específica [13].

2.2.2.1. Transmisión de mensajes

CAN establece dos formatos de tramas de datos (*data frame*) que difieren en la longitud del campo del identificador, las tramas estándares (*standard frame*) con un identificador de 11 bits definidas en la especificación CAN 2.0A, y las tramas extendidas (*extended frame*) con un identificador de 29 bits definidas en la especificación CAN 2.0B.

Para la transmisión y control de mensajes CAN, se definen cuatro tipos de tramas: de datos, remota (*remote frame*), de error (*error frame*) y de sobrecarga (*overload frame*).

Las tramas remotas también se establecen en ambos formatos, estándar y extendido, y tanto las tramas de datos como las remotas se separan de tramas precedentes mediante espacios entre tramas (*interframe space*).

2.2.2.1.1. Trama de datos

Una trama de datos se compone de siete campos (Figura 2.23):

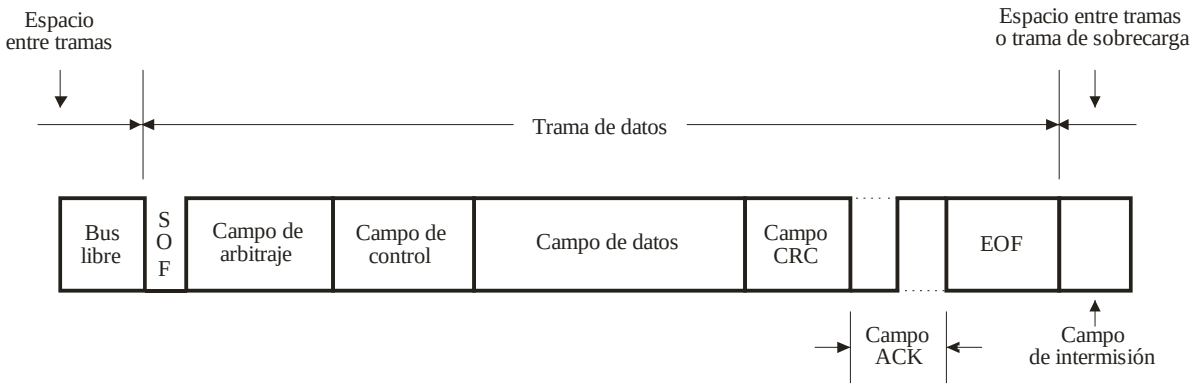


Figura 2.23. Formato de trama de datos CAN.

- *Inicio de trama (SOF, Start of Frame)*: indica el inicio de una trama de datos o una trama remota, y consiste en un bit dominante que sincroniza a todos los nodos activos en la red. Es el mismo para la trama estándar y extendida.
- *Campo de arbitraje (Arbitration field)*: difiere según el formato de trama (Figura 2.24):
  - En el formato estándar está constituido por un identificador de 11 bits y el bit de petición de transmisión remota (RTR, *Remote Transmission Request*). El bit menos significativo del identificador se transmite al último y los 7 bits más significativos no pueden ser todos recesivos.
  - En el formato extendido está formado por un identificador de 29 bits, el bit de petición remota substituta (SRR, *Substitute Remote Request*), el bit de extensión del identificador (IDE, *Identifier Extension*) y el bit RTR. El identificador se divide en dos secciones, la primera de 11 bits denominada base (*Base ID*) que corresponde al identificador del formato estándar, y la segunda sección de 18 bits conocida como extendida (*Extended ID*). Los bits de ambas secciones del identificador se transmiten en orden de mayor a menor prioridad.

  El bit RTR debe ser dominante para ambos formatos de trama de datos y el bit SRR es un bit recesivo, por lo tanto las posibles colisiones entre ambos tipos de formatos de trama que tengan el mismo valor en el campo *Base ID* se resuelven de manera que el formato de trama estándar predomina sobre el formato de trama extendida. En dicha resolución también se involucra al bit IDE, el cual pertenece al campo de arbitraje en el caso de un formato extendido y se encuentra en el campo de control para el caso de un formato de trama estándar. La transmisión del bit IDE es dominante para el formato estándar y recesivo para el extendido.

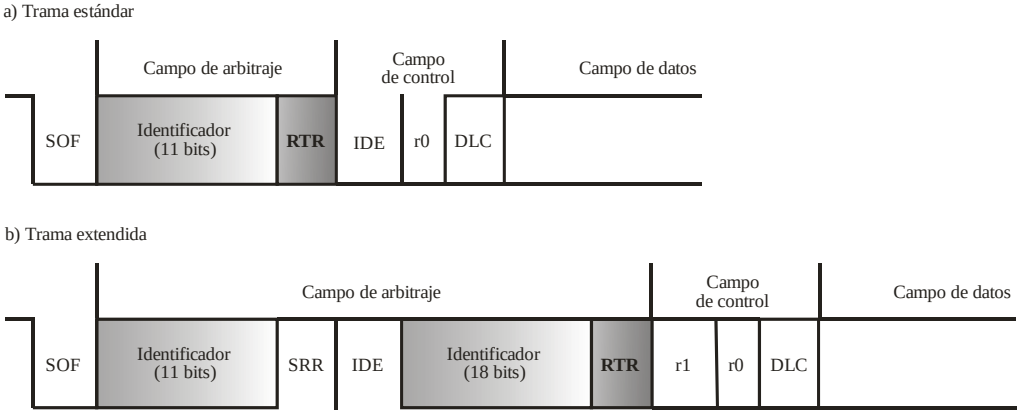


Figura 2.24. Formatos de tramas de datos CAN, estándar y extendida.

- *Campo de control (Control field)*: está compuesto de seis bits, IDE/r1, r0 y cuatro bits que forman el código de longitud de datos (DLC, *Data Length Code*). El primer bit que se transmite es IDE, el cual distingue entre los dos tipos de tramas; en seguida r0, en nivel dominante y está reservado para futuras aplicaciones del protocolo CAN; finalmente se transmite el DLC para indicar el número de octetos contenidos en el campo de datos.
- *Campo de datos (Data field)*: contiene el mensaje a transmitir dentro de una trama CAN y puede tener una longitud de 0 a 8 octetos. Se transfiere primero el bit con mayor prioridad.
- *Campo CRC (CRC field)*: está constituido por una secuencia de 15 bits de verificación y un bit delimitador CRC (*CRC delimiter*), este último se transmite en un nivel recesivo. Mediante este campo, el receptor verifica si la secuencia de bits recibidos fue alterada. Se utiliza el polinomio generador $X^{15}+X^{14}+X^{10}+X^{8}+X^{7}+X^{4}+X^{3}+1$.
- *Campo de aceptación (ACK field)*: está constituido por dos bits, ranura ACK (*ACK slot*) y delimitador ACK (*ACK delimiter*), este último siempre se transmite en un nivel recesivo. Todo nodo activo en la red CAN que recibe una trama válida, sobrescribe la ranura ACK con un nivel dominante, y con ello el transmisor verifica que su mensaje se envió correctamente; si por el contrario ningún nodo sobrescribe dicha ranura, el transmisor considera un error de transmisión.
- *Fin de trama (EOF, End of Frame)*: tanto la trama de datos como la trama remota están delimitadas por una secuencia de 7 bits recesivos que indican el fin de trama CAN. Cuando EOF está activo se realiza una violación al procedimiento de inserción de bit (inciso 2.1.1.1), por ello dicho procedimiento no se aplica a este campo.

2.2.2.1.2. Trama remota

Un nodo CAN, en modo receptor, puede iniciar la transmisión de su información mediante el envío de una trama remota, la cual se compone de seis campos tanto en formato estándar o extendido (Figura 2.25). Los campos de una trama remota son los mismos que los de una trama de datos, a excepción que la trama remota no contiene el campo de datos y el bit RTR es recesivo. El valor del DLC debe coincidir con el de la trama de datos correspondiente.

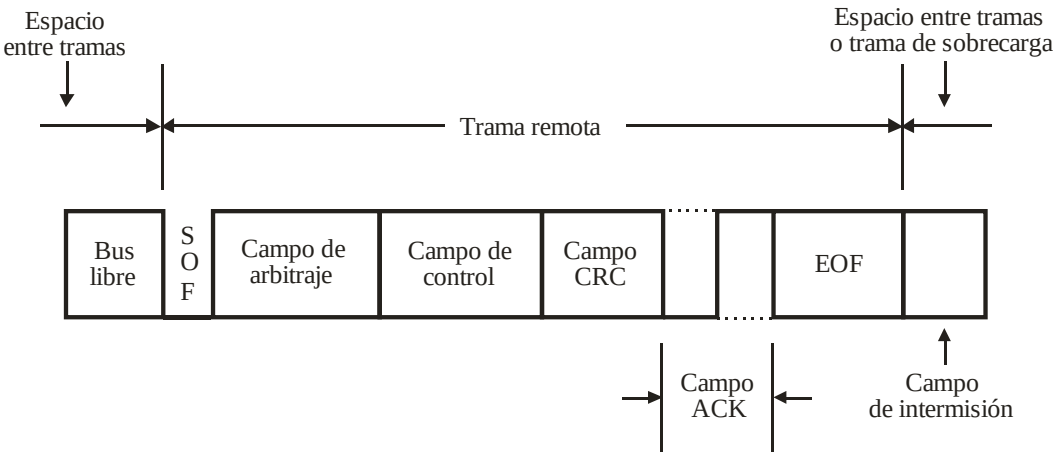


Figura 2.25. Formato de trama remota CAN.

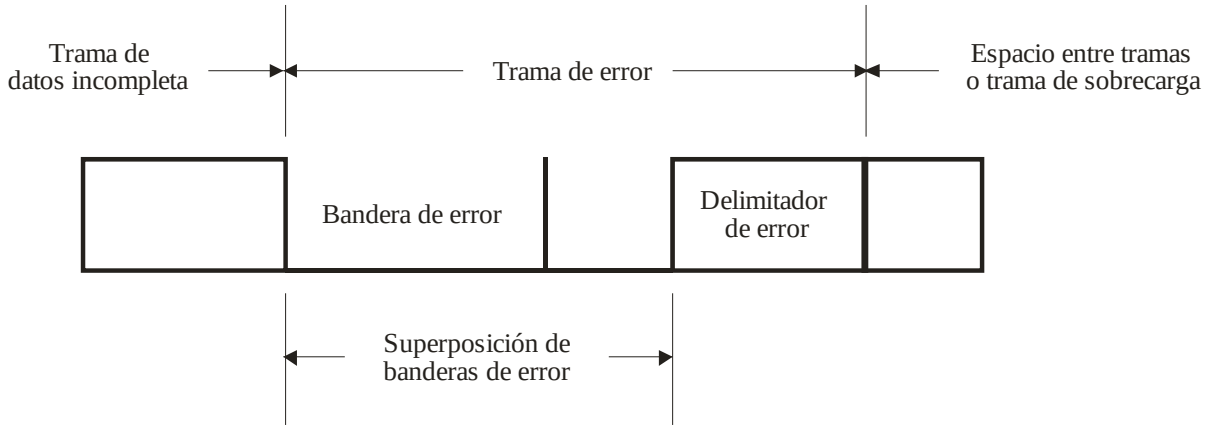


Figura 2.26. Formato de trama de error CAN.

2.2.2.1.3. Trama de error

La trama de error señaliza la detección de cualquier error durante la transmisión o recepción de una trama de datos o remota, la cual viola el procedimiento de inserción de bit y ocasiona que el transmisor reenvíe la trama. Asimismo la detección de un error, durante la transmisión o recepción de una trama de sobrecarga o error, genera la transmisión de una nueva trama de error.

La trama de error está formada por dos campos (Figura 2.26):

- *Bandera de error (Error flag)*: existen dos formas de representar una bandera de error:
  - *Bandera de error activo (Active error flag)*: consiste en seis bits dominantes consecutivos.
  - *Bandera de error pasivo (Passive error flag)*: está formada por seis bits recesivos consecutivos, a menos que sean sobrescritos por bits dominantes de otros nodos.
- *Delimitador de error (Error delimiter)*: una trama de error termina con una secuencia de ocho bits recesivos. Posterior a la transmisión de una bandera de error, el nodo transmite bits recesivos y verifica el nivel del bus hasta que reconozca un bit recesivo, entonces comienza la transmisión de otros siete bits recesivos. Con este mecanismo, el nodo puede determinar si fue el primero en transmitir una bandera de error y con ello detectar una condición de error.

2.2.2.1.4. Trama de sobrecarga

La trama de sobrecarga se utiliza para que un receptor solicite un retraso en la transmisión de la trama siguiente, ya sea de datos o remota, o para señalizar condiciones de error relacionadas con el campo de intermisión. El protocolo CAN permite la generación de dos tramas de sobrecarga como máximo para retrasar la transmisión de la siguiente trama.

Las tramas de sobrecarga se transmiten después de detectar las siguientes condiciones de error:

- Detección de un bit dominante durante los primeros dos bits del campo de intermisión. La detección de un bit dominante en el tercer bit del campo de intermisión se interpreta como un SOF.
- Cuando un receptor detecta un bit dominante en el último bit del campo EOF; o cuando un nodo, receptor o transmisor, detecta un bit dominante en el último bit del delimitador de una trama de error o de sobrecarga.

Una trama de sobrecarga se considera una forma especial de trama de error y consta de los siguientes campos (Figura 2.27):

- *Bandera de sobrecarga (Overload flag)*: se constituye por seis bits dominantes. La forma completa corresponde a la bandera de error activa.
- *Delimitador de sobrecarga (Overload delimiter)*: está formado por ocho bits recesivos.

2.2.2.1.5. Espacio entre tramas

Las tramas de datos y remotas están separadas de tramas precedentes por un espacio entre tramas, a diferencia de las tramas de error y de sobrecarga que se transmiten en forma sucesiva, es decir sin un espacio entre tramas.

El espacio entre tramas está formado por tres campos:

- *Intermisión (Intermission)*: consiste en tres bits recesivos. Durante su transmisión la única acción que puede realizarse es señalar una condición de sobrecarga, y no se permite que ningún nodo inicie la transmisión de una trama de datos o remota.
- *Bus libre (Bus idle)*: este periodo es de longitud arbitraria y tiene un nivel recesivo hasta que algún nodo inicie la transmisión de una nueva trama.
- *Suspender transmisión (Suspend transmission)*: adicionalmente, el espacio entre tramas contiene un tiempo de inhibición de transmisión de ocho bits para nodos que se encuentren en estado de error pasivo.

2.2.2.2. Codificación de tramas

Los campos inicio de trama, arbitraje, control, datos y secuencia CRC, están codificados de acuerdo al procedimiento de inserción de bit que se describió en el inciso 2.1.1.1. Los campos restantes, delimitador CRC, ACK y EOF, tienen un formato fijo y no siguen el procedimiento de inserción de bit, de igual forma, las tramas de error y de sobrecarga tienen un formato fijo y no se codifican por dicho procedimiento [23].

2.2.2.3. Validación de tramas

El instante de tiempo en el que se valida una trama difiere según:

- *Transmisor*: la trama es válida si no existen errores hasta el final del campo EOF. Si existe un error, en la trama se activa el proceso de recuperación.
- *Receptor*: la trama es válida si no existen errores hasta el siguiente bit después del campo EOF.

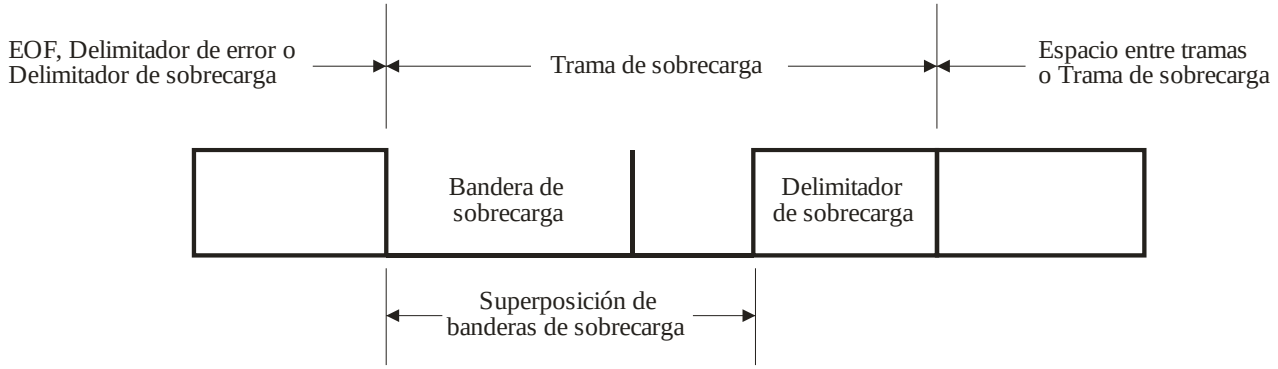


Figura 2.27. Formato de trama de sobrecarga CAN.

2.2.2.4. Detección y manejo de errores

Un controlador CAN cuenta con la capacidad de detectar y manejar los errores que surjan en una red. Todo error detectado por un nodo, se notifica inmediatamente al resto de los nodos.

Un nodo puede tener una alteración local permanente, lo que provoca el envío continuo de tramas de error. Para prevenir dicho comportamiento, el protocolo CAN describe un algoritmo, basado en la actividad del bus, que obliga a los nodos con errores permanentes a desconectarse de la red (*bus off*), y que los demás nodos no sean perturbados por los nodos defectuosos [27].

2.2.2.4.1. Mecanismos de detección de errores

Para cumplir con las demandas relativas a la seguridad en la transmisión de datos, el protocolo CAN define los siguientes mecanismos para detección de errores:

- *Monitoreo de bits*: todo nodo verifica que el nivel del bus transmitido sea el mismo nivel en el bus, y cuando dichos valores difieren se detecta un error de bit (*bit error*). El monitoreo de bits representa un mecanismo de seguridad global para la detección de todos los errores efectivos.
- *Verificación del procedimiento de inserción de bit*: hace referencia al hecho de detectar un error de inserción de bit cuando ocurren seis niveles consecutivos de bits con el mismo valor en un campo de trama codificado por el procedimiento de inserción de bit (*stuff error*).
- *Verificación de redundancia cíclica de 15 bits*: se presenta un error de CRC (*CRC error*) cuando la secuencia CRC recibida no se corresponde con la secuencia CRC calculada.
- *Verificación de trama*: detecta un error cuando un campo de bit de formato fijo contiene uno o más bits no válidos (*form error*).
- *Verificación de aceptación*: un transmisor detecta un error de aceptación (*ACK error*) cuando la ranura ACK (*ACK slot*) no cambia a estado dominante (párrafo 2.2.2.1.1).

2.2.2.4.2. Manejo de errores

Cuando un nodo detecta algún tipo de error, de bit, de inserción de bit, de forma o de aceptación, inicia la transmisión de una bandera de error (*error flag*) en el siguiente bit. Cuando se detecta un error de CRC, se inicia la transmisión de una trama de error después del delimitador ACK, a excepción de que previamente se haya transmitido otra trama de error.

El manejo de errores se lleva a cabo de acuerdo con el diagrama de flujo de la Figura 2.28.

2.2.2.4.3. Capacidad de detección de errores

Los protocolos de comunicaciones de bus seriales emplean diferentes mecanismos de detección de error, los cuales proporcionan al receptor la capacidad de detectar si la trama recibida es errónea o no. Por otro lado, la capacidad de detectar errores de transmisión depende de los mecanismos de error y del protocolo utilizado.

Para valorar la integridad de los datos en un sistema de transmisión se deben considerar las interferencias electromagnéticas y la capacidad para detectar errores de transmisión. En un sistema de transmisión de datos existe una medida estadística que representa la integridad de datos transferidos conocida como probabilidad de error residual, la cual especifica la probabilidad de que existan tramas erróneas sin detectar y está dada por la Ecuación 2.7.

$$\textit{Probabilidad de error residual} < (4.7 \times 10^{-11})(\textit{velocidad de error de trama}) \qquad \text{Ecuación 2.7}$$

La cual define el número de errores de transmisión sin detectar durante el tiempo de operación de una red CAN, que puede ser estimado por la velocidad de error de trama, el porcentaje de carga del bus, la velocidad de transferencia de datos y la longitud de la trama.

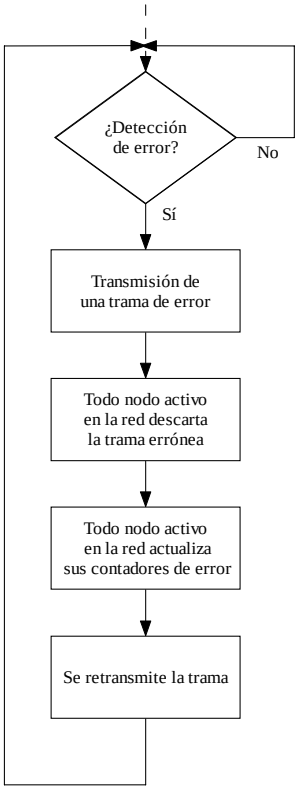


Figura 2.28. Diagrama de flujo para el manejo de errores.

La mayoría de errores de transmisión son debido a interferencias electromagnéticas. La susceptibilidad de error de un sistema de transmisión de datos se puede caracterizar mediante parámetros estadísticos tales como la velocidad de error de trama (*frame error rate*) y la probabilidad de error de bit (*bit error probability*). La velocidad de error (*error rate*) está dada por la relación entre el número de tramas erróneas y el número total de tramas transmitidas durante un periodo de tiempo, con lo cual se describe la probabilidad de que exista una trama errónea, por otro lado, la probabilidad de error de bit especifica la probabilidad de que exista un bit erróneo en una trama transmitida.

Los errores de transmisión no ocurren en todos los nodos de una red, sino que aparecen en nodos individuales con diversos patrones de error. El protocolo CAN tiene la capacidad de señalizar los errores detectados mediante la transmisión de una bandera de error, sin embargo existe la posibilidad de no detectar errores locales cuando se cumplen las siguientes condiciones en la distribución del error de bit:

- El nodo que transmite funciona adecuadamente, y puede detectar un error al monitorear el bit y señalizarlo a todo el sistema.
- Cuando todos los nodos receptores que reciben una trama errónea detectan un patrón de error no definido (indetectable). Debido a que estos patrones son muy raros, todos los nodos que recibieron la trama errónea deben mostrar un patrón de error idéntico. Esta condición es cada vez más improbable en un sistema distribuido con un gran número de nodos.

La DLL del protocolo de comunicaciones CAN garantiza un alto grado de detección de errores. Todos los errores se detectan mediante el proceso de monitoreo de bit realizado por el transmisor de la trama, mediante dicho proceso el transmisor puede detectar errores ocasionados a nivel global por interferencia electromagnética, la cual aparece en todos los nodos, y además tiene la capacidad de detectar errores locales.

Los errores que sólo ocurren localmente, en receptores individuales, se detectan mediante la verificación de la secuencia CRC de 15 bits, realizada por el mismo receptor.

2.3. Capa de supervisor

La sustitución del cableado convencional por un sistema de bus serie presenta el problema de que un nodo defectuoso puede bloquear el funcionamiento del sistema completo. Como se ha descrito con anterioridad, cada nodo activo transmite una bandera de error cuando detecta algún tipo de error (principio de señalización de errores) y puede ocasionar que un nodo defectuoso pueda acaparar el medio físico, para eliminar este riesgo el protocolo CAN define un mecanismo autónomo para detectar y desconectar un nodo defectuoso del bus, dicho mecanismo se conoce como aislamiento de fallos (*fault confinement*)[14].

2.3.1. Aislamiento de fallos

El objetivo del aislamiento de fallos es preservar la disponibilidad del sistema de transmisión de datos, incluso en el caso de que existan nodos defectuosos, para ello se definen estrategias que incrementan la seguridad en los siguientes aspectos:

- Distinguir entre errores temporales (*short disturbances*) y fallas permanentes (*permanent failures*).
- Localizar y desconectar (*switched off*) nodos defectuosos (*defective nodes*).

Por ello, cada nodo tiene un contador de errores de transmisión (TEC, *Transmit Error Counter*) y un contador de errores de recepción (REC, *Receive Error Counter*) para registrar el número de errores respectivamente. Si las tramas se transmiten y reciben correctamente, dichos contadores decrementan sus valores, y en caso de ocurrir errores, los contadores se incrementan en proporciones mayores a los que se decrementan.

Los contadores incrementan o decrementan sus valores dependiendo de la relación existente entre las tramas enviadas y recibidas correctamente y aquellas con errores. Los valores en los contadores indican la frecuencia relativa de perturbaciones previas.

El comportamiento de los nodos se modifica, en relación a los errores, dependiendo de los valores del contador correspondiente. El incremento o decremento de los contadores se modifica de acuerdo a las reglas establecidas en [6, 23].

Dependiendo del valor de sus contadores de error, un nodo puede estar en uno de los siguientes estados (Figura 2.29):

- *Error activo (Active error)*: un nodo toma parte en la comunicación de bus y cuando detecta un error envía una bandera de error activo; destruye la trama que transmitía, viola el procedimiento de inserción de bit y previene con ello a los demás nodos de la presencia de una trama errónea.
- *Error pasivo (Passive error)*: un nodo no puede enviar una bandera de error activo pero aún toma parte en la comunicación del bus; cuando detecta un error, sólo puede enviar una bandera de error pasivo, la cual no interfiere en la comunicación del bus.
- *Desconectado (Bus off)*: en este estado, un nodo no tiene ninguna influencia en el bus y por ello no puede transmitir tramas, aceptación ACK, tramas de error o de sobrecarga.

[14] El estándar ISO 11898 [23] y la espeficación CAN 2.0 [6] denominan a este mecanismo como aislamiento de fallos, mientras que en la literatura se denomina limitación de errores (*error limitation*) [11]. En este trabajo de tesis se emplea el término aislamiento de fallos.

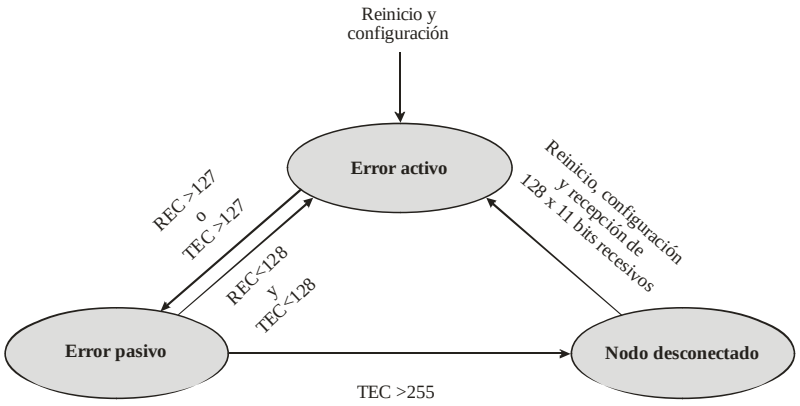


Figura 2.29. Diagrama de estados de error de un nodo CAN.

2.4. Capa de aplicación

CAN se utiliza en aplicaciones en las que no está determinada una estructura de capas entre el nivel de enlace proporcionado por el controlador de protocolo y la aplicación en el nodo. Actualmente, con la implementación de sistemas distribuidos basados en CAN han surgido nuevos requerimientos que no han sido considerados en el estándar ISO 11898-2, siendo los más importantes:

- Disponibilidad de servicios de transmisión para bloques de datos mayores a 8 octetos.
- Soporte al modelo cliente/servidor.
- Funciones dedicadas a la gestión de red (*network management*).
- Métodos para asignar identificadores de mensaje y configuración de parámetros específicos del nodo, de forma transparente al usuario.
- Interoperabilidad e intercambio de dispositivos de diferentes fabricantes.
- Estandarizar la funcionalidad y definir nuevos perfiles para dispositivos.

La necesidad de estandarizar las capas de aplicación ha surgido sobre todo en el sector de los FBs industriales, donde la comunicación entre dispositivos de diferentes fabricantes es una característica fundamental.

Respecto al protocolo CAN, existen diferentes estándares que definen su capa de aplicación; algunos son muy específicos y están relacionados con sus campos de aplicación. Entre las capas de aplicación más utilizadas cabe mencionar las siguientes:

- *CAL*: define un amplio conjunto de funciones para la comunicación y gestión de una red CAN (apartado 2.4.1).
- *CANopen*: protocolo de ámbito Europeo, respaldado por CiA. Utiliza parte de CAL y le agrega nuevos perfiles de protocolo y dispositivos (apartado 2.4.2).
- *DeviceNet*: desarrollado por la firma Allen-Bradley y de mayor utilización en E.U.A. Es un enlace de bajo costo que conecta dispositivos industriales a una red CAN (apartado 0).
- *SDS (Smart Distributed System)*: sistema de bus desarrollado por la firma Honeywell para la interconexión de sensores y actuadores inteligentes.
- *OSEK (Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug)*: en inglés "*Open Systems and the Corresponding Interfaces for Automotive Electronics*", desarrollado por la firma Siemens como resultado de la cooperación entre fabricantes de automóviles Alemanes para desarrollar una arquitectura abierta para unidades de control distribuido [URL 18].

- *CANKingdom*: desarrollado por la firma Kvaser y compatible con DeviceNet, protocolo reconocido a nivel mundial en el control de maquinaria [URL 4].

2.4.1. CAL

La organización CiA estandarizó el protocolo de comunicaciones CAL con la finalidad de facilitar las implementaciones de sistemas abiertos de redes CAN en aplicaciones de control industrial [9, 27]. CAL se considera la única implementación independiente dedicada exclusivamente a la capa de aplicación en redes CAN.

La funcionalidad de CAL consiste en cuatro elementos de servicio:

- *Especificación de mensajes basada en CAN (CMS, CAN Based Message Specification*): proporciona los medios para la descripción e implementación de una comunicación orientada a objetos. Contiene distintos tipos de datos estructurados y diferentes objetos de comunicación con características de transmisión. Mediante reglas de codificación, especifica un formato de datos común para todos los mensajes de la red.
- *Gestión de la red (NMT, Network Management)*: asegura un inicio ordenado de toda la red y contiene las medidas de precaución necesarias para la supervisión e intercambio/inserción de nodos en tiempo de ejecución.
- *Distribuidor de identificadores (DBT, Identifier Distributor)*: se encarga de la asignación dinámica de identificadores y trabaja en conjunto con el servicio NMT.
- *Gestión de capa (LMT, Layer Management)*: se encarga de la configuración y parametrización (*parametrization*) de CAL a través de la red.

Los elementos anteriores se pueden configurar para diseñar sistemas con diferentes capacidades y requerimientos. En la Figura 2.30 se muestra la arquitectura de un sistema basado en CAL.

2.4.2. CANopen

CANopen es un estándar de comunicaciones y dispositivos basado en CAL [14, 27]. Inicialmente se desarrolló como una red empotrada de configuración flexible y con apoyo de CiA logró su estandarización para utilizarse en sistemas distribuidos de automatización industrial. En Europa se considera el estándar *de facto* para la implementación de sistemas industriales basados en CAN, y sus campos de aplicación son: equipo médico, vehículos para terreno especiales (*off road vehicle*), electrónica marítima, transporte público, domótica, etc.

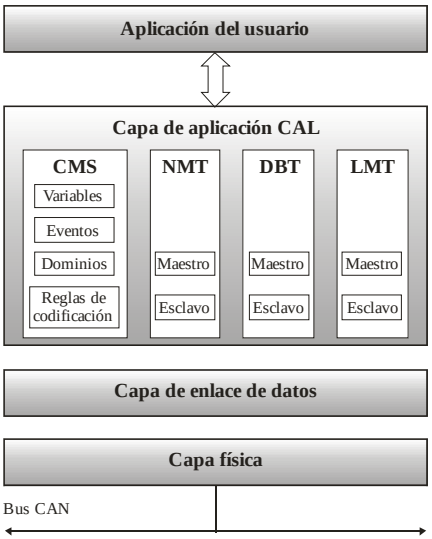


Figura 2.30. Arquitectura general de un sistema basado en CAL.

CANopen está integrado por un perfil de comunicaciones (*communication profile*) que especifica y define los mecanismos de comunicación. Los perfiles de dispositivos (*device profile*) describen a los dispositivos utilizados en la tecnología de automatización industrial tales como módulos de E/S analógicos y digitales (*digital and analog I/O module*), controladores (*drives*), unidades de interfaz hombre máquina (MMI-unit, *Man Machine Interface unit*), codificadores (*encoders*), etc. El perfil de dispositivo define la funcionalidad de un dispositivo en particular.

La principal característica de CANopen es la descripción funcional de parámetros, y datos de un dispositivo, la cual se encuentra en un diccionario de objetos (OD, *Object Dictionary*). La funcionalidad y características de un dispositivo CANopen se definen en una hoja de datos electrónica (EDS, *Electronic Data Sheet*) estandarizada en formato ASCII.

De forma similar a otros FBs, CANopen define dos mecanismos básicos de transmisión:

- El intercambio crítico en tiempo real de datos de proceso mediante objetos de proceso (PDO, *Process Data Object*).
- El acceso en tiempo crítico a las entradas del diccionario de objetos mediante objetos de servicio (SDO, *Service Data Object)*.

La Tabla 2.19 muestra las ventajas y características, mientras que la Figura 2.31 ilustra la arquitectura general de un sistema CANopen.

Tabla 2.19. Ventajas y características de CANopen.

Ventajas	Características
Permite la interoperabilidad entre diferentes dispositivos	Configuración automática de la red
Capacidad de respuesta en tiempo real	Fácil acceso a todos los parámetros del dispositivo
Modular, cubre dispositivos desde simples a complejos	Sincronización de dispositivos
Amigable con el usuario, disponibilidad de herramientas	Transmisión de datos controlada por eventos y cíclica
Protocolo abierto e independiente del vendedor, estandarizado en EN50325-4	Lectura sincrónica de configuración de entradas, salidas o parámetros

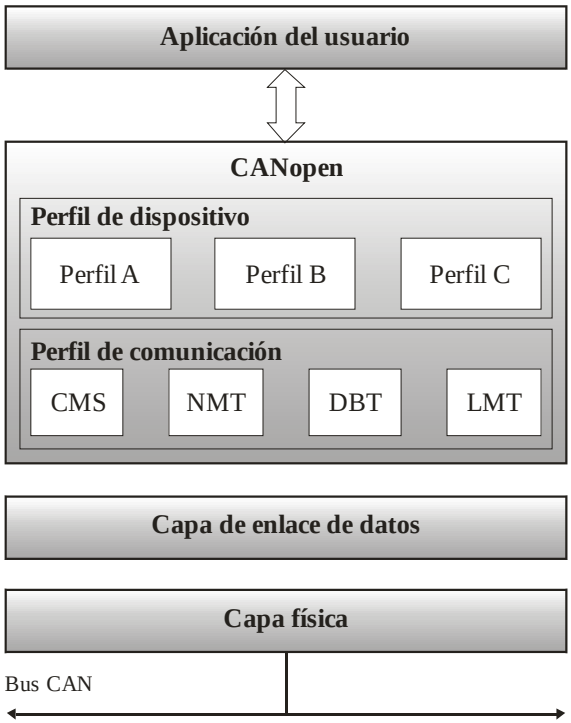


Figura 2.31. Arquitectura general de un sistema basado en CANopen.

2.4.3. DeviceNet

Desarrollado por la firma Rockwell Automation[15] y publicado como un estándar de FB abierto. Actualmente desempeña un papel importante en la industria de los E.U.A. y Asia, y en Europa se extiende cada vez más.

DeviceNet es un protocolo basado en CAN, simple, de bajo costo, eficiente, etc. que fue diseñado para el nivel más bajo de los FBs, por ejemplo sensores y actuadores, y unidades de control asociadas.

DeviceNet es una de las tres redes abiertas (DeviceNet, ControlNet, y Ethernet/IP) que utilizan el protocolo de control e información (CIP, *Control and Information Protocol*). El CIP es un protocolo de comunicaciones común y sus interfaces hardware/software permiten la conexión de dispositivos industriales a Internet mediante dos tipos de mensajes:

- *Control*: se utiliza para control de las E/S en tiempo real o mensajes implícitos.
- *Información*: está dedicado al intercambio de mensajes o mensajes explícitos.

Ambos tipos de mensajes están destinados el área de control industrial y proporcionan las siguientes características:

- Servicios de control común.
- Servicios de comunicación común.
- Capacidades de encaminamiento común.
- Base de conocimiento común.

La especificación de DeviceNet define parte de la capa física y el medio de transmisión (párrafo 2.1.4.3.2). La Figura 2.32 muestra la arquitectura de protocolos de DeviceNet.

2.4.4. SDS

SDS es un sistema de bus basado en CAN para la conexión de sensores y actuadores inteligentes. SDS implementa un protocolo de capa de aplicación diseñado para cumplir los requerimientos establecidos en la automatización industrial (*industrial automation*), como son: velocidad de transferencia de datos, confiabilidad, flexibilidad, etc. Algunas de sus principales características son la utilización de métodos de detección y corrección de errores, y confiabilidad en el reconocimiento de mensajes.

El protocolo de capa de aplicación SDS proporciona un conjunto de mensajes que abarca desde mensajes de cambio de estado controlados por eventos, hasta operaciones complejas transportando valores binarios, analógicos y alfanuméricos.

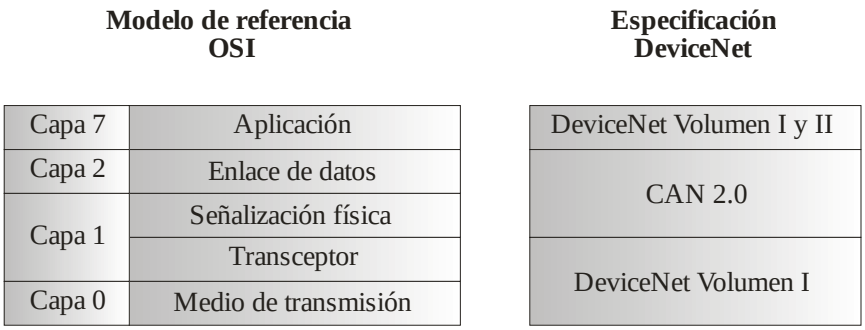


Figura 2.32. Arquitectura de protocolos DeviceNet.

[15] La firma Rockwell Automation escribió la especificación original DeviceNet [37]. En Europa, DeviceNet forma parte de la norma EN-50325-2 [42] y se incluye en el estándar IEC 62026-3 [1].

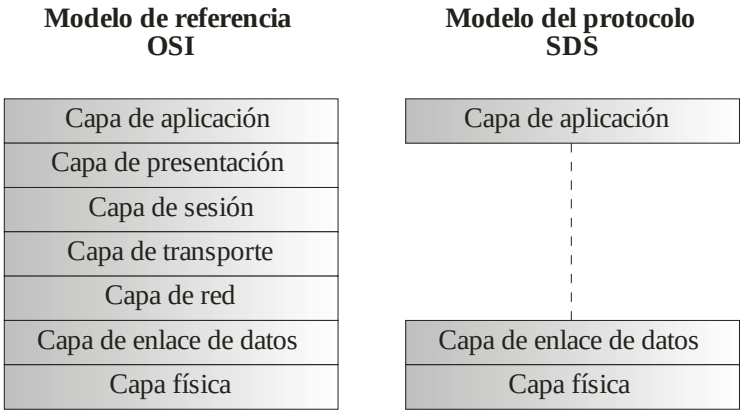


Figura 2.33. Arquitectura de protocolos SDS.

La arquitectura del protocolo de capa de aplicación SDS se basa en el modelo OSI, utiliza los servicios que proporciona la capa de enlace de datos CAN y debe implementar una capa física compatible con CAN (Figura 2.33).

2.4.5. OSEK/VDX

OSEK/VDX es un proyecto común de la industria del automóvil alemana, su objetivo es obtener una arquitectura abierta estandarizada para las unidades de control distribuidas (*distributed control units*) en vehículos [35]. OSEK/VDX introduce una arquitectura abierta que comprende tres áreas (Figura 2.34):

- *Comunicación (OSEK COM)*: proporciona servicios para el intercambio de datos entre tareas y/o rutinas de servicio de interrupción (ISR), comunicación interna de una ECU (*internal communication*), y externa entre diferentes ECUs (*external communication*); el acceso a los servicios de comunicación OSEK se realiza mediante interfaces de programación de la aplicación específica (API, *Application Program Interface*).
- *Gestión de red (Network Management)*: define un conjunto de servicios para determinar y supervisar la configuración del nodo. Debe adaptarse a los requerimientos específicos del sistema de bus utilizado (métodos globales) o a los recursos de cada nodo (métodos locales).
- *Sistema operativo (OS, Operating System)*: las aplicaciones automotrices se caracterizan por tener requerimientos de tiempo real rigurosos. El OS de OSEK proporciona la funcionalidad necesaria para dar soporte a sistemas de control manejados por eventos. Los servicios especificados por el OS constituyen una base para hacer posible la integración de módulos software realizados por distintos fabricantes.

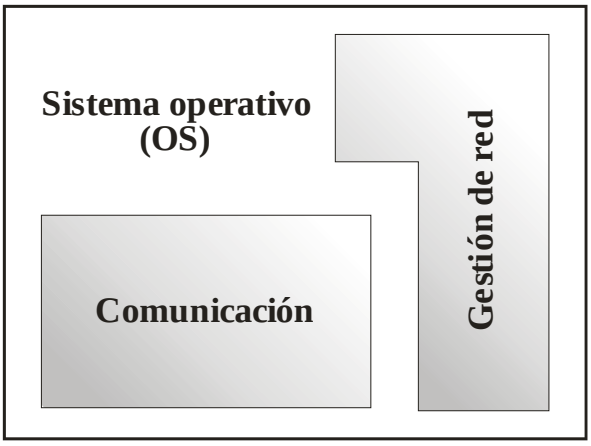


Figura 2.34. Arquitectura OSEK/VDX.

2.4.6. CAN Kingdom

La organización CiA respalda a cuatro HLPs: CANopen, DeviceNet, CAN Kingdom y SDS. CAN Kingdom fue desarrollado específicamente para aplicaciones de control de maquinaria que requieren un desempeño en tiempo real, los demás protocolos están enfocados a la automatización industrial.

CAN Kingdom es un conjunto de primitivas del protocolo basado en CAN y es una herramienta que el diseñador de sistemas puede utilizar para diseñar y optimizar HLPs. Propone una filosofía para el desarrollo de máquinas basada en comprensibilidad, seguridad, simplicidad y efectividad.

El desarrollo de un sistema CAN Kingdom sigue el principio de que los módulos deben servir a la red (MSN, *The Modules are to Serves the Network*) y como consecuencia todo nodo en la red tiene la información necesaria para inicializar el sistema.

CAN Kingdom describe un sistema como si fuera un país (*Country*), un reino (*Kingdom*), con su respectiva capital y ciudades (*Capital and Cities*). El rey (*King*) gobierna al reino desde la capital, y cada ciudad tiene un Alcalde (*Mayor*) responsable del gobierno local. El único medio para comunicarse dentro de la ciudad es el correo (*Mail*). La red CAN se describe como el sistema postal real (*The Royal Postal System*), cada ciudad tiene una oficina de correos (*Post Office*) y un director de correos (*Postmaster*) el cual simboliza a un controlador CAN (Figura 2.35).

Cada ciudad produce algo y puede importar o exportar información por correo. El alcalde de la ciudad organiza cualquier información de importación o exportación dentro de listas, dichas listas forman parte de la documentación del módulo. El diseñador de sistemas elige los módulos específicos que se utilizarán en su máquina, y para ello debe conocer completamente las listas. Asimismo el diseñador crea un protocolo optimizado para su máquina, al asignar identificadores a las variables que estén en dicha lista.

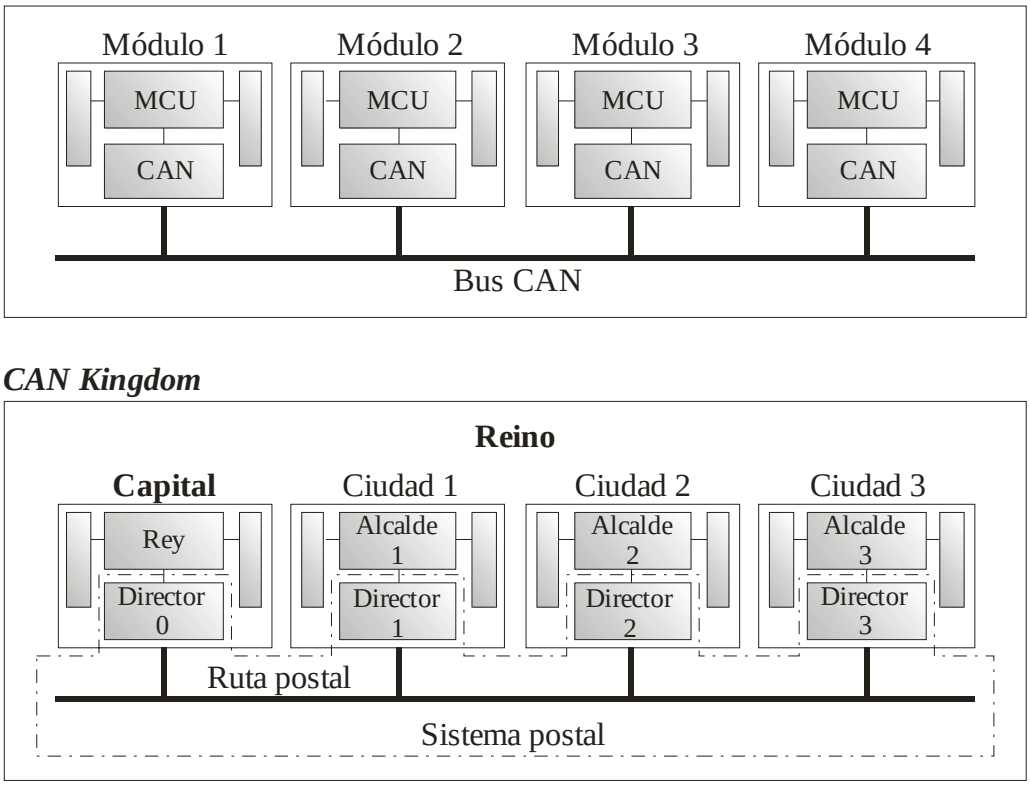


Figura 2.35. Representación de una red CAN con CAN Kingdom.

3. Clasificación de dispositivos CAN

Debido a la aceptación que ha logrado el protocolo de comunicaciones CAN, apoyado por fabricantes de semiconductores, organizaciones de estandarización y la SAE, actualmente existe una gran variedad de dispositivos que implementan dicho protocolo para diversas aplicaciones (subcapítulo 1.4). Por ello, los sistemas y dispositivos CAN se clasifican con base a distintos criterios, entre ellos: clasificación ISO, versión del protocolo, estructura de memoria intermedia (*mailbox structure*) y grado de integración (Figura 3.1).

Un controlador CAN debe realizar las siguientes tareas correspondientes a las capas física y de enlace de datos:

- Arbitraje del bus (apartado 2.2.2).
- Transmisión/recepción de tramas (inciso 2.2.2.1).
- Cálculo y verificación del CRC (párrafo 2.2.2.1.1).
- Detección y señalización de errores (inciso 2.2.2.4).
- Construcción de tramas CAN (subcapítulo 2.2).
- Inserción y eliminación de bits de relleno (inciso 2.1.1.1).
- Generación y verificación del bit de aceptación (párrafo 2.2.2.1.1).
- Sincronización del flujo de bits recibido (inciso 2.1.1.3).

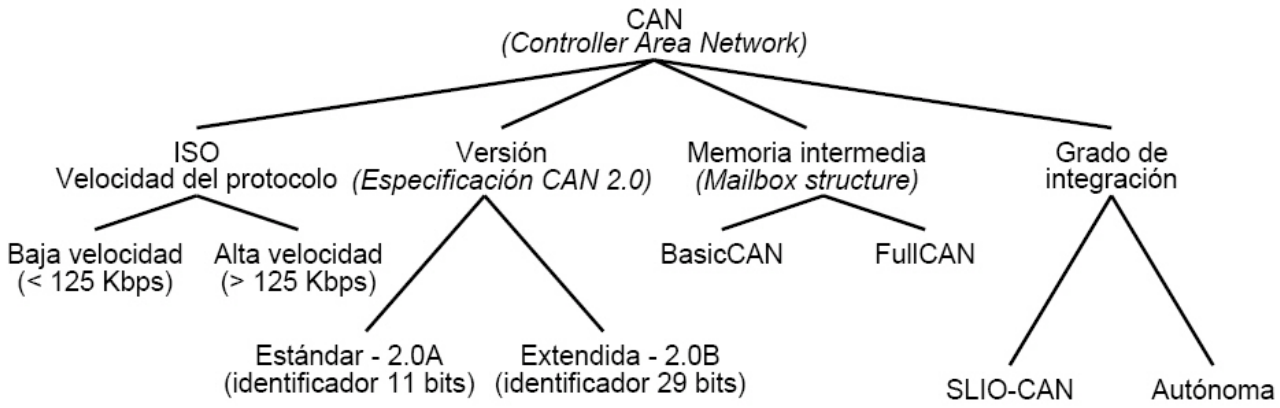


Figura 3.1. Clasificación de dispositivos CAN.

3.1. Clasificación ISO

La clasificación propuesta por la ISO se basa en las aplicaciones dentro del automóvil definido por la SAE (subcapítulo 1.2), y define dos tipos:

- Comunicación de baja velocidad: < 125 Kbps.
- Comunicación de alta velocidad: > 125 Kbps.

3.2. Versión del protocolo

El protocolo CAN define que el acceso al medio debe resolverse mediante la prioridad del identificador del mensaje, de forma tal que el mensaje con mayor prioridad continúa su transmisión mientras que los mensajes de menor prioridad detienen y posponen su proceso de transmisión. Dependiendo de la versión de la especificación CAN se puede tener un identificador de 11 bits (versión 2.0A) o de 29 bits (versión 2.0B) (párrafo 2.2.2.1.1).

3.3. Manejo de mensajes

El manejo de mensajes hace referencia al proceso de recibir tramas del bus, filtrar su identificador y, con base a ello, almacenarlos en la memoria intermedia (*buffers*) del controlador CAN o descartarlos. Existen dos principios de implementación de la memoria intermedia, *BasicCAN* y *FullCAN*, y actualmente existe una gran cantidad de combinaciones entre ambos (Figura 3.2) [26].

3.3.1. Principio BasicCAN

El principio *BasicCAN* asocia a los mensajes recibidos una memoria como filtro, el cual consiste en un selector de identificador de mensaje (*Selector ID-Rx*) y su respectiva máscara (*Máscara ID-Rx*). Ambos parámetros son definidos por el usuario al configurar la interfaz *BasicCAN*. El identificador de cada trama recibida se compara con el *Selector ID-Rx* y se descarta cuando no coincide con la *Máscara ID*; si pasa el filtrado anterior, el mensaje se almacena en la memoria intermedia (*ID-Rx/Reg. Datos*) para ser procesado por el controlador CAN. Debido a que pueden existir varios mensajes que cumplan con el filtrado descrito anteriormente, la memoria intermedia proporciona capacidad de almacenamiento de más de un mensaje.

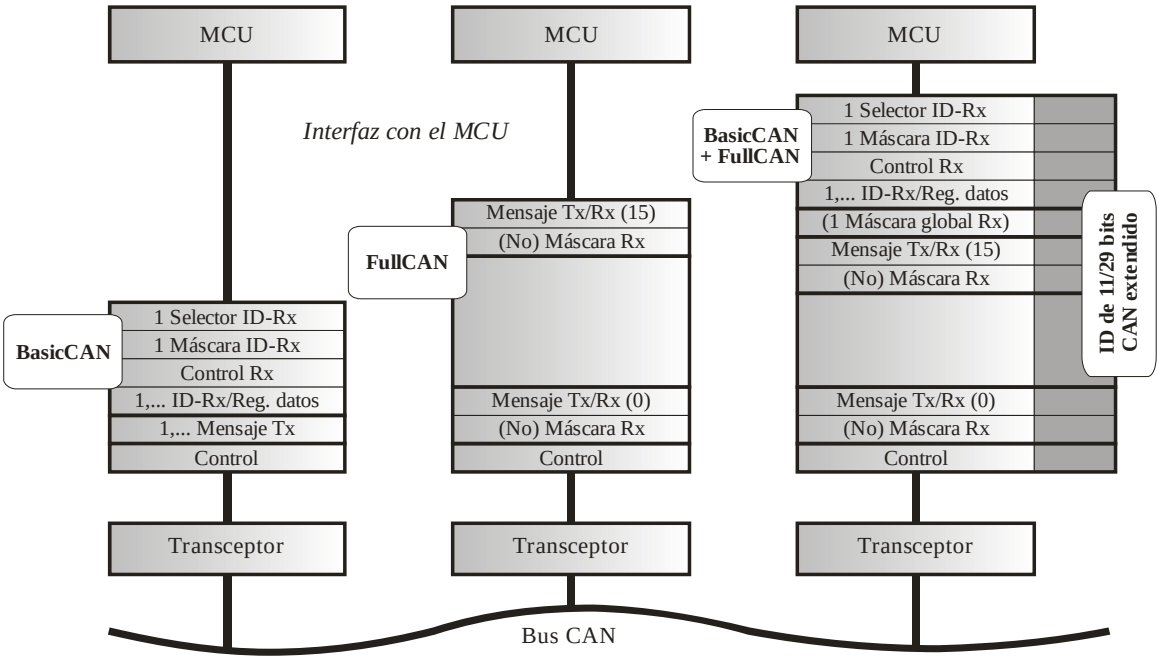


Figura 3.2. Estructuras de memoria intermedia.

Una vez aceptado un mensaje, el registro de control (*Control*) genera una interrupción, solicitando atención al controlador CAN. Asimismo, si ocurre un desbordamiento (*overflow*) en el registro de recepción, debido a que el controlador no pueda atender las solicitudes de los mensajes entrantes, se produce una señal de advertencia (*warning signal*).

Cuando el controlador CAN necesita transmitir un mensaje, almacena el identificador y datos correspondientes en el registro de transmisión (*Mensaje Tx*), y posteriormente activa el proceso de transmisión (*Control*).

3.3.2. Principio FullCAN

El principio *FullCAN* se puede entender como una estructura *BasicCAN* especializada donde la máscara (*Máscara ID-Rx*) es toltalmente transparente y por lo tanto no se requiere tal registro. Como consecuencia, sólo un mensaje puede validarse en cada selector de identificador de mensaje (*Mensaje Tx/Rx*) y por ello se requieren varias memorias intermedias para almacenar los mensajes, dichas memorias pueden configurarse como de recepción o de transmisión, con lo cual se incrementa la flexibilidad del sistema.

Cada memoria intermedia es capaz de almacenar el mensaje completo que se recibe o se desea transmitir, el cual consta de un identificador único, datos, definición de recepción/transmisión y demás información de control como puede ser: longitud de los datos, interrupción de recepción de nuevo mensaje, petición de transmisión, etc.

Si una memoria intermedia, de recepción, contiene un mensaje y recibe un nuevo mensaje con el mismo identificador, ésta se sobrescribe, de modo que siempre contiene la información más reciente.

Con este mecanismo, el controlador general del nodo no está obligado a leer la memoria intermedia cada que llega un nuevo mensaje , sino que ocasionalmente puede leer el contenido de una determinada memoria. Lo anterior tiene ventajas, especialmente cuando se reciben varios mensajes con diferentes identificadores dentro de intervalos cortos de tiempo, y el controlador CAN distribuye y almacena los mensajes en sus memorias correspondientes. Dicho proceso se lleva a cabo sin la intervención del MCU como en el caso del principio *BasicCAN*.

Actualmente, los dispositivos que implementan el principio *FullCAN* proporcionan hasta 16 memorias intermedias.

3.3.3. Combinación de principios BasicCAN y FullCAN

Algunos dispostivos CAN implementan una combinación proporcionando al menos una memoria intermedia *BasicCAN* (*Selector ID-Rx*, *Máscara ID-Rx*, *Control Rx* y *ID-Rx/Reg. Datos*) y múltiples memorias intermedias *FullCAN* (*Mensaje Tx/Rx*), estas últimas pueden configurarse como de recepción o transmisión, y tienen asociada una máscara global (*Máscara global Rx*). Además proporcionan la opción para elegir entre tramas estándares y extendidas (párrafo 2.2.2.1.1).

Ejemplo de lo anterior es el principio *PeliCAN* implementado por la firma Philips, el cual es una mejora al principio *BasicCAN* para dar soporte a la versión 2.0B de la especificación CAN. Además agrega un mecanismo de detección y manejo de errores (incisos 2.2.2.4, 2.2.2.4.1 y 2.2.2.4.2).

3.4. Grado de integración

Actualmente existen diversas formas para diseñar un sistema de comunicaciones CAN y los dispositivos, de acuerdo al grado de integración en su realización física, se clasifican en (Figura 3.3):

- *Controladores CAN independientes (stand alone)*: implementan todas las funciones necesarias para el control de las capas física y de enlace de datos del protocolo, y brindan so-

porte a funciones avanzadas mediante un controlador anfitrión (*host controller*). Este tipo de controladores CAN se encargan de transmitir o recibir las tramas, para ello requieren de un MCU que se encargue de administrar los procesos de transmisión y recepción respectivamente, así como también de interfazar con los dispositivos de E/S (*MCU + E/S*) [43].

- *MCUs con controlador CAN integrado (MCU + E/S + CAN)*: existe una gran cantidad de MCUs de propósito general que integran el protocolo CAN como medio de comunicación al exterior, en su mayoría implementan la solución *FullCAN* y con ello liberan al procesador de realizar la tarea de comunicación. Esta solución proporciona: mejor interfaz entre las memorias intermedias del controlador CAN y el MCU, dispositivos de menor tamaño y bajo costo, aunque son menos flexibles en cuanto a configuración de usuario [43].
- *ASICs*[16] *con controlador CAN integrado*: son controladores CAN flexibles que pueden modificarse de acuerdo a los requerimientos de una aplicación específica, además pueden integrar la funcionalidad de un transceptor de bus (*MCU + E/S + CAN + Transceptor*). En consecuencia, los fabricantes de semiconductores ofrecen controladores CAN junto a periféricos como son convertidores analógico a digital, puertos de E/S, comparadores, controladores de interrupción, etc.
- *Dispositivos de E/S con controlador CAN integrado*: se diseñaron para implementar redes de sensores y actuadores de bajo costo. Estos son controladores de E/S (SLIO, *Serial Linked I/O*) que integran un controlador CAN, los cuales son configurados y controlados por nodos inteligentes o maestros (*host node/master node*) a través del bus CAN. El nodo maestro tiene acceso directo a los puertos de E/S de un controlador de nodo SLIO a través de un protocolo fijo, por lo tanto, no es necesario un software local en el nodo SLIO (*Lógica + E/S + CAN + SLIO*). Una red CAN SLIO está limitada en características tales como número de nodos, velocidad de transferencia, protocolo específico, etc., por ello dichos controladores son cada vez menos utilizados [13]. Actualmente, están disponibles MCUs avanzados con menor costo, que integran uno o más controladores CAN, los cuales están supliendo a los controladores CAN SLIO.

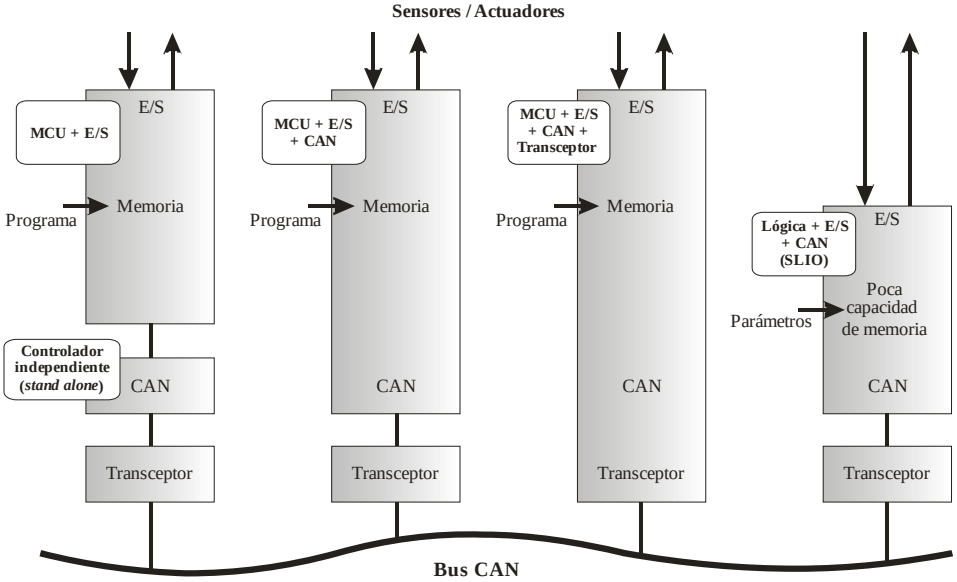


Figura 3.3. Clasificación de nodos CAN según el grado de integración.

[16] Un circuito integrado de aplicación específica (ASIC, *Application Specific Integrated Circuit*) integra la funcionalidad de varios controladores de periféricos discretos y logra una disposición compacta.

3.5. Controlador CAN SJA1000

El SJA1000 es un controlador independiente del protocolo de comunicaciones CAN utilizado en ambientes automotrices e industriales. En la Figura 3.4 se muestra el diagrama a bloques del SJA1000, el cual se describe a continuación:

- *Lógica para gestión de la interfaz (IML, Interface Management Logic)*: interpreta las órdenes que provienen del MCU, controla el direccionamiento de los registros CAN y proporciona información al MCU sobre interrupciones y estado actual del controlador.
- *Memoria temporal de transmisión (TXB, Transmit Buffer)*: es una interfaz entre el MCU y el procesador de flujo de bits (BSP, *Bit Stream Processor*); puede almacenar un mensaje completo para que se transmita sobre la red CAN y cuenta con 13 octetos de memoria temporal para transmisión.
- *Memoria temporal de recepción (RXB, Receive Buffer + RXFIFO)*: es una interfaz entre el filtro de admisión y el MCU, se encarga de almacenar los mensajes que se reciben y aceptan de la línea del bus CAN. La RXB representa una ventana de 13 octetos en la cola de recepción (RXFIFO), con una capacidad total de 64 octetos. El MCU sólo tiene acceso a la RXB, y con ayuda de la RXFIFO, el MCU puede procesar un mensaje mientras recibe otro en forma simultánea.
- *Filtro de admisión (ACF, Acceptance Filter)*: compara el identificador del mensaje recibido con el contenido en el registro de filtro de admisión, y decide si el mensaje es aceptado o no. En el caso de ser aceptado, se almacena el mensaje en la RXFIFO.
- *Procesador de flujo de bits (BSP, Bit Stream Processor)*: controla el flujo de datos entre la RXB, la RXFIFO y el bus CAN, realiza la detección y manejo de errores, e implementa los mecanismos de arbitraje y la técnica de inserción de bit.
- *Lógica de temporización de bit (BTL, Bit Timing Logic)*: supervisa la línea de bus CAN y controla la temporización de bit, implementando los mecanismos de sincronización descritos en el apartado 2.1.1.
- *Lógica de gestión de errores (EML, Error Management Logic)*: realiza el aislamiento de errores de los módulos de la DLL; recibe las señales de error del BSP, e informa al BSP y a la IML sobre estadísticas de errores.

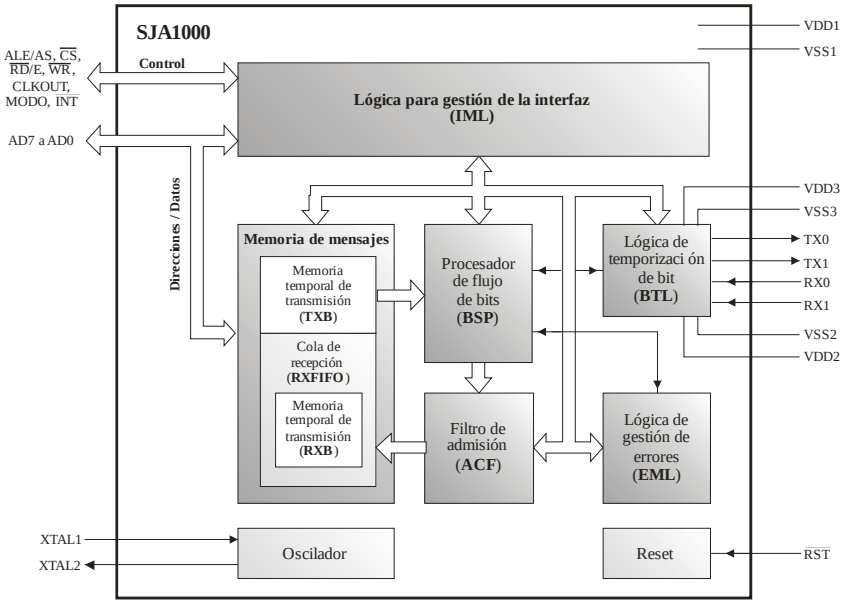


Figura 3.4. Diagrama a bloques del controlador CAN SJA1000.

3.5.1. Modos de operación del controlador CAN SJA1000

El controlador CAN SJA1000 es el sucesor del controlador PCA82C200 (*BasicCAN*) de la firma Philips Semiconductors, ya que para lograr la compatibilidad por software implementa dos modos de operación[17] (Tabla 3.1):

- *Modo BasicCAN*: compatible con PCA82C200.
- *Modo PeliCAN*: soporta la especificación CAN 2.0B.

Tabla 3.1. Principales características del controlador CAN SJA1000.

BasicCAN	***PeliCAN***
Compatible con el controlador PCA82C200.	Contadores de error con acceso de lectura y escritura.
Modo de operación por defecto.	Limite de advertencia de error programable.
Memoria intermedia de recepción extendida (FIFO de 64 octetos).	Registro de código del último error.
Compatible con la especificación del protocolo CAN 2.0B.	Interrupción de error independiente para cada tipo de error en el bus CAN.
Soporta identificadores de 29 bits en modo pasivo.	Soporta identificadores de 11 y 29 bits.
Velocidad de transferencia máxima de 1 Mbps.	Interrupción de pérdida de arbitraje con posición de bit detallada.
Frecuencia de reloj máxima de 24 MHz.	Filtro de admisión configurable.

El modo de operación se selecciona configurando el bit de modo CAN (*CAN mode*), localizado en el registro divisor de reloj (CDR, *Clock Divider Register*), el cual por defecto toma el modo *BasicCAN*.

El controlador CAN SJA1000 fue diseñado para cumplir con la especificación CAN 2.0B, para ello proporciona una mayor tolerancia de oscilador y procesamiento de mensajes de trama extendida. En modo *BasicCAN* se pueden transmitir y recibir tramas con identificador de 11 bits únicamente, y si se detectan tramas con identificador de 29 bits, éstas se aceptan y se activa el reconocimiento siempre que la trama sea válida (modo pasivo), pero no se genera la interrupción de recepción.

A continuación se describe la disposición de las direcciones de memoria en ambos modos, *BasicCAN* y *PeliCAN*.

3.5.1.1. Disposición de las direcciones de memoria en modo BasicCAN

El MCU considera al controlador SJA1000 como un dispositivo de E/S direccionado en memoria externa. El SJA1000 implementa sus registros internos en memoria tipo RAM (*Random Access Memory*) para garantizar un funcionamiento autónomo respecto al MCU. Dichos registros se agrupan en los siguientes segmentos:

- *Segmento de control*: realiza el intercambio de las señales de estado (SR, *Status Register*), control (CR, *Control Register*) y órdenes (CMR, *Command Register*) entre el MCU y el SJA1000. El MCU controla la comunicación CAN a través de este segmento, el cual se configura durante la inicialización para seleccionar los parámetros de comunicación (BTR0, BTR1, OCR y CDR). La Tabla 3.2 muestra el mapa de memoria de este segmento.

  Los registros del segmento de control son los siguientes:

  - *Registro de control (CR, Control Register)*: se utiliza para modificar el comportamiento del SJA1000.

[17] A nivel teórico, *BasicCAN*, *FullCAN* y *PeliCAN* se conocen como principios, y a nivel de implementación se conocen como modos de operación.

- *Registro de órdenes (CMR, Command Register)*: cada uno de sus bits se encarga de realizar una acción especifica. Es necesario al menos un ciclo de reloj de separación para procesar correctamente dos órdenes.
- *Registro de estado (SR, Status Register)*: indica el estado actual del SJA1000.
- *Registro de interrupciones (IR, Interrupt Register)*: contiene información acerca de las interrupciones generadas en el SJA1000, y mediante su terminal 16 (INT) indica al MCU la activación de una interrupción; el MCU lee el contenido del IR, una vez realizada esta operación, se restablecen los bits del IR y se libera la terminal INT.
- *Registro de código de admisión (ACR, Acceptance Code Register)*: cuando un mensaje cumple con el filtro de admisión y la RXB cuenta con suficiente espacio, se almacena el identificador y el campo de datos de dicho mensaje en la RXFIFO. Para informar al MCU de la recepción de un nuevo mensaje, existen dos técnicas:
  - *Técnica de sondeo (polling)*: el MCU monitorea constantemente el bit de estado de recepción (RBS, *Receive Buffer Status*) del SR, el cual indica la existencia de uno o más mensajes disponibles en la RXFIFO.
  - *Técnica de interrupción*: el MCU se libera de la tarea de monitorear la llegada de nuevos mensajes al SJA1000, ya que éste le informa al MCU de la llegada de nuevos mensajes al activar la interrupción de recepción (RI, *Receive Interrupt*) del IR, para ello se debe configurar el bit de interrupción de recepción (RIE, *Receive Interrupt Enable*) del CR.
- *Registro de máscara de admisión (AMR, Acceptance Mask Register)*: determina el código de admisión mediante la configuración de bits relevantes (*AM.X=0*) o cuyo valor no importa (*AM.X=1*) para el filtro de admisión.
- *Registro de temporización de bus 0 (BTR0, Bus Timing Register 0)*: su contenido define los valores del multiplicador de velocidad de transferencia (BRP, *Baud Rate Prescaler*) y de la duración del salto de sincronización (SJW, *Sinchronization Jump Width*).
- *Registro de temporización de bus 1 (BTR1, Bus Timing Register 1)*: su contenido define la longitud del periodo de bit, la ubicación del punto de muestreo y el número de muestras que se toman en cada punto de muestreo.
- *Registro de control de salida (OCR, Output Control Register)*: permite la activación de diferentes configuraciones del controlador de salida (*output driver*) mediante software.
- *Registro divisor de reloj (CDR)*: configura la frecuencia de la señal CLKOUT, la cual se utiliza de forma opcional para proporcionar una señal de reloj al MCU, asimismo el CDR permite desactivar la terminal CLKOUT. Adicionalmente se realiza la selección del modo de operación *BasicCAN* o *PeliCAN*, la conexión del puente del comparador de recepción (*comparator bypass*) y se habilita la terminal TX1 para generar una interrupción de recepción.

- *Segmento de memoria temporal de transmisión de mensajes*: consta de dos campos, descriptor y datos, y se encarga de almacenar el mensaje a transmitir. El acceso de lectura y escritura sólo se lleva a cabo en modo de funcionamiento, el cual se describe en este mismo inciso. La Tabla 3.3 muestra el mapa de memoria correspondiente.
  - *Registro de identificador 1 (ID.10 a ID.3)*: define los ocho bits más significativos del identificador.
  - *Registro de identificador 2 (ID.2 a ID.0, RTR, DLC)*: define los tres últimos bits del identificador, asimismo se configura el bit RTR para seleccionar el tipo de trama que

se va a transmitir (datos o remota), y se indica la longitud, en octetos, de la trama a transmitir (DLC.3 a DLC.0).

- *Registros de datos 1 a 8 (TX data)*: la información que se transmite se almacena previamente en estos registros de ocho bits (trama de datos), el primer bit que se transmite es el más significativo; la longitud de datos a transmitir debe coincidir con el código que se define en el DLC.

- *Segmento de memoria temporal de recepción de mensajes*: es la parte accesible de la RXFIFO, y su capacidad para almacenar mensajes depende de la longitud de éstos. Si en determinado momento, no cuenta con capacidad suficiente para almacenar un nuevo mensaje, el SJA1000 genera una interrupción para indicar una condición de datos sobrantes (*data overrun condition*), la cual se indica al MCU mediante el SR; en caso de contar con suficiente capacidad, almacena el mensaje recibido hasta que el MCU lo procesa y posteriormente libera el espacio de memoria para uso futuro. El mapa de memoria de este segmento es similar al del segmento anterior (Tabla 3.3).
  - *Registro de identificador 1 (ID.10 a ID.3)*: en este octeto se reciben los ocho bits más significativos del identificador.
  - *Registro de identificador 2 (ID.2 a ID.0, RTR, DLC)*: en este octeto se reciben los tres últimos bits del identificador, asimismo se informa del tipo de trama que se recibe (datos o remota), y se indica la longitud de la trama en octetos (DLC.3 a DLC.0).
  - *Registros de datos 1 a 8 (RX data)*: almacenan los datos de una trama que se recibe correctamente.

Tabla 3.2. Mapa de memoria del segmento de control en modo *BasicCAN*.

Dirección CAN del registro	Modo de funcionamiento		Modo de reinicio	
	Lectura	**Escritura**	**Lectura**	**Escritura**
0h	Control	Control	Control	Control
1h	FFh	Orden	FFh	Orden
2h	Estado	-	Estado	-
3h	Interrupción	-	Interrupción	-
4h	FFh	-	Código de admisión	Código de admisión
5h	FFh	-	Máscara de admisión	Máscara de admisión
6h	FFh	-	Temporización de bus 0	Temporización de bus 0
7h	FFh	-	Temporización de bus 1	Temporización de bus 1
8h	FFh	-	Control de salida	Control de salida
9h	Prueba	Prueba	Prueba	Prueba

Existen dos modos para tener acceso a los registros internos del SJA1000:

- *Modo de reinicio (Reset Mode)*: se configuran los parámetros de comunicación del SJA1000.
- *Modo de funcionamiento (Operating Mode)*: se realiza la transmisión y recepción de tramas, así como la señalización y manejo de errores.

Tabla 3.3. Mapa de memoria temporal de transmisión y recepción.

Dirección CAN		Campo	Nombre de registro	Bits							
TX	**RX**			**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0Ah	14h	Descriptor	Identificador 1	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
0Bh	15h		Identificador 2	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
0Ch	16h	Datos	Dato TX/RX 1	Bits 7 al 0 del dato TX/RX 1							
0Dh	17h		Dato TX/RX 2	Bits 7 al 0 del dato TX/RX 2							
0Eh	18h		Dato TX/RX 3	Bits 7 al 0 del dato TX/RX 3							
0Fh	19h		Dato TX/RX 4	Bits 7 al 0 del dato TX/RX 4							
10h	1Ah		Dato TX/RX 5	Bits 7 al 0 del dato TX/RX 5							
11h	1Bh		Dato TX/RX 6	Bits 7 al 0 del dato TX/RX 6							
12h	1Ch		Dato TX/RX 7	Bits 7 al 0 del dato TX/RX 7							
13h	1Dh		Dato TX/RX 8	Bits 7 al 0 del dato TX/RX 8							

3.5.1.2. Disposición de direcciones en modo PeliCAN

Al igual que en *BasicCAN*, en modo *PeliCAN* el SJA1000 aparece como un dispositivo de E/S direccionado en memoria externa por el MCU. En modo *PeliCAN*, los registros BTR0, BTR1, OCR y CDR, tienen la misma dirección CAN y funcionalidad que en el modo *BasicCAN*.

Los registros internos del SJA1000 en modo *PeliCAN* se agrupan en segmentos similares a los del modo *BasicCAN*:

- *Segmento de control*: realiza el intercambio de las señales de estado (SR, *Status Register*), de modo (MOD, *Mode Register*) y órdenes (CMR, *Command Register*) entre el MCU y el SJA1000. El MCU controla la comunicación CAN a través de este segmento, el cual se configura durante la inicialización para seleccionar los parámetros de comunicación (BTR0, BTR1, OCR y CDR). La Tabla 3.4 muestra el mapa de memoria de este segmento.

  Los registros del segmento de control son los siguientes:

  - *Registro de modo (MOD, Mode Register)*: su función es la misma que el registro CR en modo *BasicCAN*.
  - *CMR*: su función es la misma que el registro CMR en modo *BasicCAN*.
  - *SR*: su función es la misma que el registro SR en modo *BasicCAN*.
  - *IR*: su función es la misma que el registro IR en modo *BasicCAN*.
  - *Registro de habilitación de interrupciones (IER, Interrupt Enable Register)*: permite habilitar los diferentes tipos de fuentes de interrupción e indicarlos al MCU.
  - *Registro de captura de pérdida de arbitraje (ALC, Arbitration Lost Captured Register)*: contiene información referente a la posición de bit en la que se perdió el arbitraje.
  - *Registro de captura del código de error (ECC, Error Code Capture Register)*: contiene información acerca del tipo y localización de errores en el bus.
  - *Registro de límite de advertencia de error (EWLR, Error Warning Limit Register)*: define el límite de advertencia de error.
  - *Registro contador de errores de recepción (RXERR, RX Error Counter Register)*: contiene el valor actual del contador de errores de recepción (REC).
  - *Registro contador de errores de transmisión (TXERR, TX Error Counter Register)*: contiene el valor actual del contador de errores de transmisión (TEC).

Tabla 3.4. Mapa de memoria en modo *PeliCAN*.

Dirección CAN	Modo de funcionamiento		Modo de reinicio	
	Lectura	**Escritura**	**Lectura**	**Escritura**
0h	MOD	MOD	MOD	MOD
1h	00h	CMR	00h	CMR
2h	SR	-	SR	-
3h	IR	-	IR	-
4h	IER	IER	IER	IER
5h	Reservado	-	Reservado	
6h	BTR0	-	BTR0	BTR0
7h	BTR1	-	BTR1	BTR1
8h	OCR		OCR	OCR
9h	Prueba	Prueba	Prueba	Prueba
Ah	Reservado	-	Reservado	
Bh	ALC		ALC	
Ch	ECC		ECC	
Dh	EWLR		EWLR	EWLR
Eh	RXERR		RXERR	RXERR
Fh	TXERR		TXERR	TXERR

- *Segmento de memoria temporal de transmisión*: se tiene que distinguir entre la configuración del formato de trama estándar (SFF, *Standard Frame Format*) y el formato de trama extendida (EFF, *Extended Frame Format*). La memoria temporal de transmisión permite enviar mensajes con una longitud máxima de 8 octetos.

  La memoria temporal de transmisión está dividida en un campo descriptor y un campo de datos, el primer octeto del campo descriptor es el octeto de información de trama (*frame information*). En el octeto de información de trama se describe el formato, SFF o EFF; trama de datos o remota; y la longitud de los datos. En el caso de SFF se asignan los dos octetos siguientes para el identificador, y en el caso de EFF se asignan los cuatro octetos siguientes para el identificador. El campo de datos se compone de 8 octetos y la memoria temporal de transmisión tiene una longitud de 13 octetos.

- *Segmento de memoria temporal de recepción*: es similar a la disposición de la memoria temporal de transmisión (Tabla 3.5) y es la parte accesible de la RXFIFO.

- *Segmento de filtro de admisión*: con la ayuda del filtro de admisión, el SJA1000 acepta o impide el almacenamiento de los mensajes recibidos en la RXFIFO. La Tabla 3.6 muestra el mapa de memoria de este segmento.

  El filtro de admisión se define mediante los registros de código de admisión (ACRn, *Acceptance Code Registers*) y los registro de máscara de admisión (AMRn, *Acceptance Mask Registers*). El patrón de bits del mensaje que se recibe se define dentro de los registros de código de admisión. Los registros de máscara de admisión correspondientes permiten definir ciertas posiciones de bits que son relevantes o no importan para el filtro de admisión.

Tabla 3.5. Mapa de memoria temporal de transmisión y recepción en modo *PeliCAN*.

Dirección CAN	Modo de funcionamiento			
	Lectura		Escritura	
10h	RX SFF	RX EFF	TX SFF	TX EFF
11h	RX ID1	RX ID1	TX ID1	TX ID1
12h	RX ID2	RX ID2	TX ID2	TX ID2
13h	RX Dato 1	RX ID3	TX Dato 1	TX ID3
14h	RX Dato 2	RX ID4	TX Dato 2	TX ID4
15h	RX Dato 3	RX Dato 1	TX Dato 3	TX Dato 1
16h	RX Dato 4	RX Dato 2	TX Dato 4	TX Dato 2
17h	RX Dato 5	RX Dato 3	TX Dato 5	TX Dato 3
18h	RX Dato 6	RX Dato 4	TX Dato 6	TX Dato 4
19h	RX Dato 7	RX Dato 5	TX Dato 7	TX Dato 5
1Ah	RX Dato 8	RX Dato 6	TX Dato 8	TX Dato 6
1Bh	-	RX Dato 7	-	TX Dato 7
1Ch	-	RX Dato 8	-	TX Dato 8

Tabla 3.6. Disposición de memoria del filtro de admisión en modo *PeliCAN*.

Dirección CAN	Modo de reinicio
	Lectura / Escritura
10h	ACR1
11h	ACR2
12h	ACR3
13h	ACR4
14h	AMR1
15h	AMR2
16h	AMR3
17h	AMR4

Se pueden seleccionar dos modos de filtrado mediante el bit de modo del filtro de admisión (AFM, *Acceptance Filter Mode*) del registro MOD:

- *Modo de filtrado sencillo (MOD.3=1)*: se define un filtro de 4 octetos de longitud, y la correspondencia de bits de filtrado con los bits de mensaje depende del formato de trama que se recibe:
  - *Trama estándar*: se aplica el filtrado de admisión al campo de identificador, incluyendo además al bit RTR y los dos primeros octetos del campo de datos.
  - *Trama extendida*: se aplica el filtrado de admisión al campo de identificador incluyendo al bit RTR.
- *Modo de filtrado doble (MOD.3=0)*: se definen dos filtros cortos. Al recibir un mensaje, éste se compara con ambos filtros para determinar si se almacena en la RXB; el

mensaje es válido si al menos uno de los filtros lo acepta. La correspondencia de bits de filtrado con los bits de mensaje depende del formato de trama que se recibe:

- *Trama estándar*: los dos filtros definidos funcionan diferente, el primer filtro de admisión se aplica al campo de identificador, incluyendo además al bit RTR y al primer octeto del campo de datos; el segundo filtro sólo compara al campo de identificador y al bit RTR.
- *Trama extendida*: los dos filtros definidos funcionan igual, ambos filtros sólo comparan los primeros dos octetos del identificador.

4. Desarrollo del SeeCAN

En este capítulo se presenta el desarrollo del SeeCAN mediante una metodología de desarrollo de sistemas empotrados.

4.1. Metodología de desarrollo de un sistema empotrado

El ciclo de vida del desarrollo de sistemas empotrados plantea el diseño en paralelo del software (*SW*) y del hardware (*HW*) correspondiente, en donde se incluye una cantidad considerable de iteración y optimización en siete fases [5].

Las siete fases son las siguientes: especificación del producto, división HW y SW, iteración e implementación, diseño detallado HW y SW, integración de componentes HW y SW, prueba y liberación del producto, y mantenimiento y actualización (Figura 4.1). En los siguientes apartados se describe cada una de estas fases.

4.1.1. Especificación del producto

El proceso de desarrollar un sistema empotrado inicia con una meta, la especificación del producto, la cual describe *lo qué será* y *lo que hará* el producto final [4]. Para la mayoría de ingenieros, el desarrollo de un producto se traduce en considerar, en lo posible, todas sus especificaciones para asegurar un producto robusto [5].

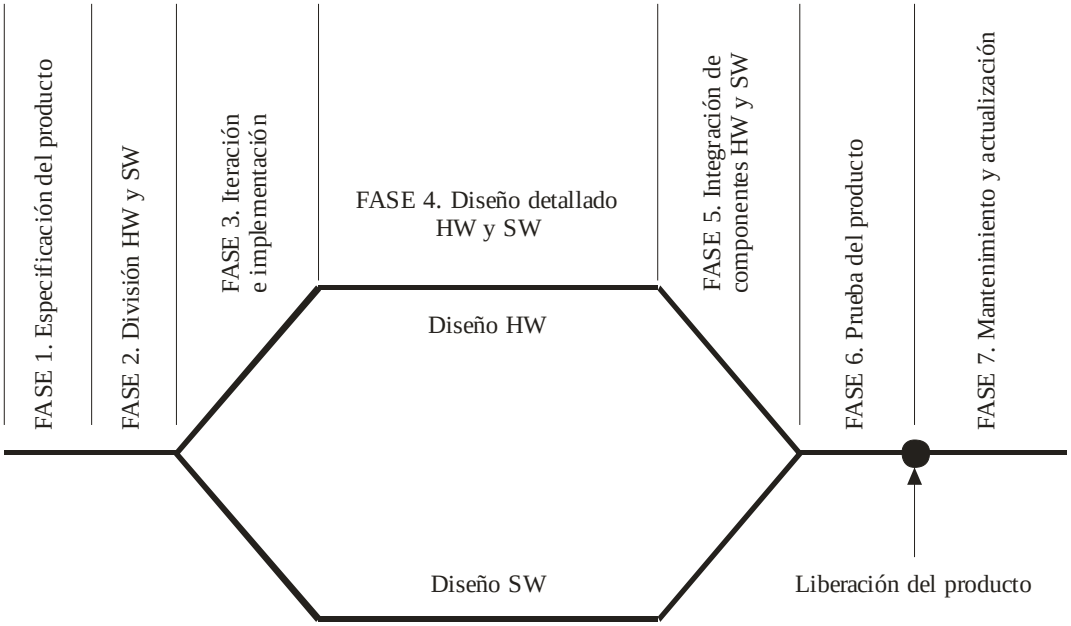


Figura 4.1. Diagrama del ciclo de vida del desarrollo de sistemas empotrados.

La especificación del producto define las interacciones, la interfaz de usuario y las condiciones de error del sistema, además incluye:

- La funcionalidad del sistema.
- Las entradas y salidas con el mundo real.
- Las interfaces externas con otros sistemas.

El equipo de trabajo encargado del diseño del producto debe contar con la descripción general del producto, lo cual es un factor clave en el desarrollo de productos exitosos.

El resultado de esta fase es la descripción del producto en forma de abstracción con base en las necesidades del usuario final, por ello debe considerarse lo siguiente:

- Entender las necesidades del cliente.
- Determinar las mejores características y costos.
- Manejar detalles respecto a investigación.

Cabe señalar que en esta fase se eligen las herramientas de desarrollo HW y SW, de acuerdo a los requisitos establecidos en la especificación del producto, con la finalidad de asegurar opciones viables en todo el ciclo del desarrollo del producto y minimizar el riesgo de no cumplir con los objetivos iniciales [5].

4.1.2. División hardware y software

Generalmente el desarrollo de un sistema empotrado incluye componentes HW y SW, por lo que se deben dividir las tareas del problema que corresponden a cada caso, esta elección se conoce como la división HW y SW (Figura 4.2).

Si se define un algoritmo como los pasos requeridos para implementar un diseño, entonces se puede considerar una combinación de componentes HW y SW, en donde dicho algoritmo se puede implementar totalmente en SW, totalmente en HW, o como una combinación de ambos.

Los requerimientos HW son más rigurosos que los de SW, debido a que es más complicado y costoso corregir un defecto HW que un error de SW.

La división HW y SW es un problema de optimización, la división del algoritmo depende del MCU seleccionado en el diseño y de cómo se implementa el diseño a nivel HW.

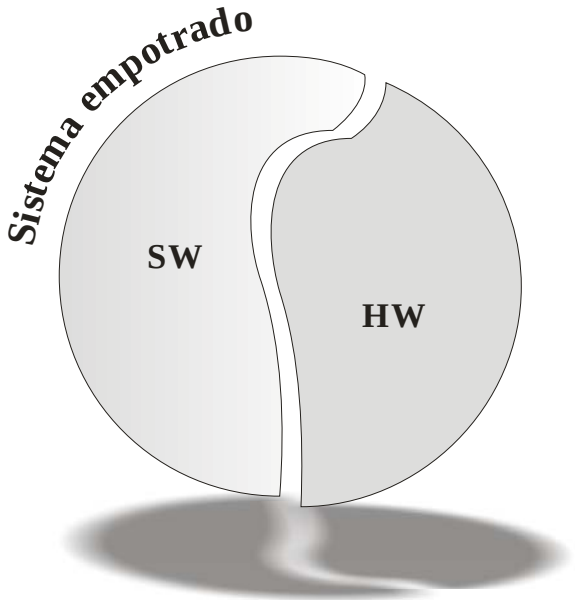


Figura 4.2. División HW y SW de un sistema empotrado.

4.1.2.1. El proceso de selección

La optimización de cualquier proyecto depende directamente de la elección del MCU, siendo ésta una tarea compleja que debe considerar lo siguiente:

- *¿Está disponible en una implementación apropiada?*: en muchas ocasiones la tecnología de encapsulado e integración del MCU limitan significativamente la elección de la arquitectura y del conjunto de instrucciones (*instruction set computer*).
- *¿Brinda un funcionamiento suficiente?*: el MCU debe realizar tareas de procesamiento complejas en tiempo real, para ello la optimización de sus recursos es un aspecto de suma importancia.
- *¿Existe soporte de un sistema operativo?*: actualmente, con los MCUs de 32 bits, resulta conveniente elegir un sistema operativo en tiempo real (RTOS, *Real Time Operating System*).
- *¿Existen herramientas de soporte apropiadas?*: el desarrollo de un proyecto debe considerar las herramientas apropiadas, se debe contar por lo menos con un compilador y un depurador; en algunas situaciones especiales se requiere de otras herramientas tales como emuladores (ICE, *In Circuit Emulator*), simuladores, etc.

No obstante que se recomienda cumplir con dichas pruebas, en muchos casos la elección del MCU queda a criterio de los desarrolladores [5].

4.1.3. Iteración e implementación

Esta fase representa un área confusa entre la implementación y la división HW y SW, aquí el diseño es fluido y los bloques pueden dividirse en componentes HW y SW.

Los diseñadores del HW pueden utilizar herramientas como son los simuladores de arquitecturas para modelar el desempeño del MCU y la memoria del sistema; los diseñadores de SW desarrollan código ejecutable. Actualmente los equipos de desarrollo HW y SW trabajan conjuntamente para mantener activo el proceso de iteración.

4.1.4. Diseño detallado HW y SW

El objetivo principal de esta fase es obtener un diseño detallado del sistema con base en los requerimientos iniciales. Se debe considerar la interfaz de usuario y la funcionalidad del sistema.

Para un diseño adecuado se consideran los siguientes aspectos:

- Entornos de desarrollo y técnicas especiales de SW.
- Técnicas especiales de programación.
- Diseño digital y arquitectura de MCUs.

4.1.4.1. Diseño HW

En la fase de diseño HW se realizan las tareas específicas para el desarrollo del HW. La interfaz HW se define en la especificación del sistema, la cual debe soportar cualquier funcionalidad que el sistema requiera.

4.1.4.2. Diseño SW

En la fase de diseño SW se elabora un documento de requerimientos, el cual incluye:

- *Una declaración de requerimientos*: consta de requerimientos, especificaciones de ingeniería, definiciones de HW, etc.

- *El protocolo de comunicación con otro SW*: describe los mecanismos de acceso a otros dispositivos como memorias temporales, órdenes o respuestas de otros dispositivos, etc.
- *Una descripción de la implementación del sistema*: realizada mediante diagramas de flujo, pseudocódigo, u otros métodos.

4.1.5. Integración de componentes HW y SW

En esta fase se debe contar con herramientas y métodos especiales para el manejo de la complejidad. La clave en el diseño de sistemas empotrados es combinar el primer prototipo HW, el SW de aplicación, el código del controlador y el SW del sistema operativo.

Los métodos generales de depuración que se aplican en las computadoras personales (PC, *Personal Computer*) o estaciones de trabajo (*workstation*), son muy similares a las que se utilizan en sistemas empotrados, muchos de ellos son imposibles de depurar hasta que se encuentren operando a su máxima velocidad. En general, existen tres requisitos para depurar un sistema empotrado en tiempo real:

- Control de ejecución.
- Sustitución de memoria.
- Análisis en tiempo real.

4.1.6. Prueba y liberación del producto

Las pruebas y requisitos de seguridad de un sistema empotrado son más estrictas que la mayoría de las aplicaciones de escritorio. Esta fase tiene un significado especial ya que incluye aspectos de seguridad del sistema. Las pruebas consisten en determinar que el sistema final funcione correctamente.

4.1.7. Mantenimiento y actualización de productos existentes

La mayoría de diseñadores de sistemas empotrados más que diseñar nuevos productos, mantienen y actualizan productos existentes. Gran parte de estos ingenieros no son miembros del equipo original de diseño, por lo que tienen que confiar en su experiencia, habilidades, documentación existente y el producto en cuestión, para entender el diseño original y con ello proporcionarle mantenimiento y, si es necesario, actualizarlo.

Esta fase requiere de herramientas adecuadas para la reingeniería. El equipo de soporte tiene acceso a herramientas sofisticadas que le permiten observar la ejecución del código en tiempo real.

4.2. Desarrollo del SeeCAN

A continuación se describe el diseño del SeeCAN con base en la metodología de desarrollo descrita.

4.2.1. Especificación del SeeCAN

El sistema educativo para la enseñanza del protocolo CAN (*SeeCAN*) tiene como principal objetivo ser una herramienta de apoyo didáctico en la enseñanza y aprendizaje del protocolo de comunicaciones CAN.

El SeeCAN es un sistema que implementa el protocolo CAN, compuesto por nodos y el medio físico de transmisión (bus CAN). El bus CAN es un cable tipo par trenzado, con terminadores de 120 Ω, que permite la transmisión diferencial mediante tres líneas (CAN_H, CAN_L y GND) y utiliza conectores tipo DB9.

Cada nodo cuenta con un administrador general, un controlador CAN, y una interfaz con el medio físico. Asimismo cuenta con un decodificador de direcciones para el MCU (GAL22V10 de la firma Lattice) y una interfaz de usuario, la cual consiste de un visualizador de cristal líquido (LCD, *Liquid Crystal Display*), un módulo de señalización de errores (MSE) y dos módulos de entradas y salidas (E/S) digitales (DIP1 y M8DES) (Figura 4.3).

4.2.1.1. Comunicación CAN

La implementación del protocolo CAN se realiza mediante el controlador SJA1000 [18, 39] y el transceptor PCA82C250 [38], ambos de la firma Philips, con lo que se obtienen las siguientes características:

- Soporte a la especificación CAN 2.0 A/B en modo *PeliCAN*.
- Filtros configurables de admisión de mensajes.
- Manejo y señalización de errores.
- Velocidades de transferencia de datos de hasta 1 Mbps.
- Medio de transmisión diferencial.
- Señalización por pérdida de arbitraje.

Cada nodo permite dos modos de operación:

- *Modo de reinicio*: en este modo se configuran los registros que contienen los parámetros de comunicación, los filtros de mensajes y el tipo de transmisión (manual o automática).
- *Modo de funcionamiento*: en este modo el nodo puede transmitir y recibir tramas CAN, y el usuario puede definir 7 bits del identificador, el tipo de trama y un octeto del campo de datos.

4.2.1.2. Administrador del nodo

El administrador general del nodo es el MCU ATmega8515 de la firma Atmel [3], cuyas principales tareas son:

- Configurar y gestionar el SJA1000 y el LCD.
- Controlar las E/S del DIP1 y del M8DES.
- Mostrar la información del MSE.
- Controlar las interrupciones.

Para comunicarse con los diversos controladores y periféricos, el administrador del nodo cuenta con direccionamiento de localidades de memoria externa, sin embargo no tiene capacidad suficiente para direccionar varios periféricos, por lo que se utiliza un dispositivo lógico programable (PLD, *Programmable Logic Device*) para incrementar el número de periféricos a controlar.

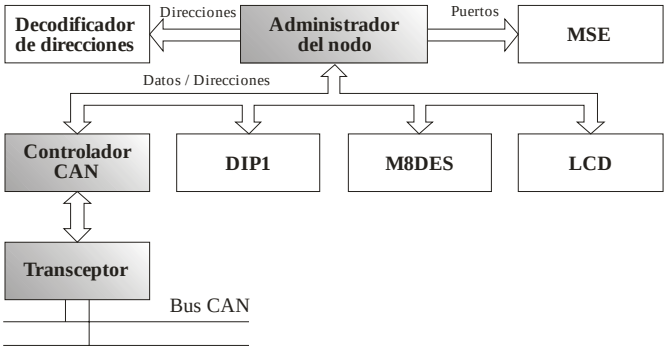


Figura 4.3. Diagrama a bloques del SeeCAN.

4.2.1.3. Interfaz de usuario

La interfaz de usuario se compone de los siguientes elementos:

- Módulo de 8 entradas digitales (DIP1).
- Módulo de 8 E/S digitales (M8DES).
- Visualizador de cristal líquido (LCD).
- Módulo de señalización de errores (MSE) del nodo.
- Botón para transmitir en forma manual (TxM).
- Botón para reinicializar el nodo (RST_NOD).
- Botón para reinicializar el controlador CAN (RST_SJA).

La Figura 4.4 muestra las opciones de configuración del SeeCAN, las cuales se describen a continuación.

En modo de reinicio, el DIP1 se utiliza para configurar los siguientes parámetros:

- *Velocidad de transferencia de datos*: 20 Kbps, 100 Kbps, 125 Kbps, 250 Kbps, 500 Kbps y 1 Mbps (interruptores 0 a 2).
- *Filtro de admisión*: aceptar todos los mensajes o definir 3 rangos (interruptores 3 y 4).
- *Transmisión de tramas CAN*: en forma automática o manual (interruptor 5).

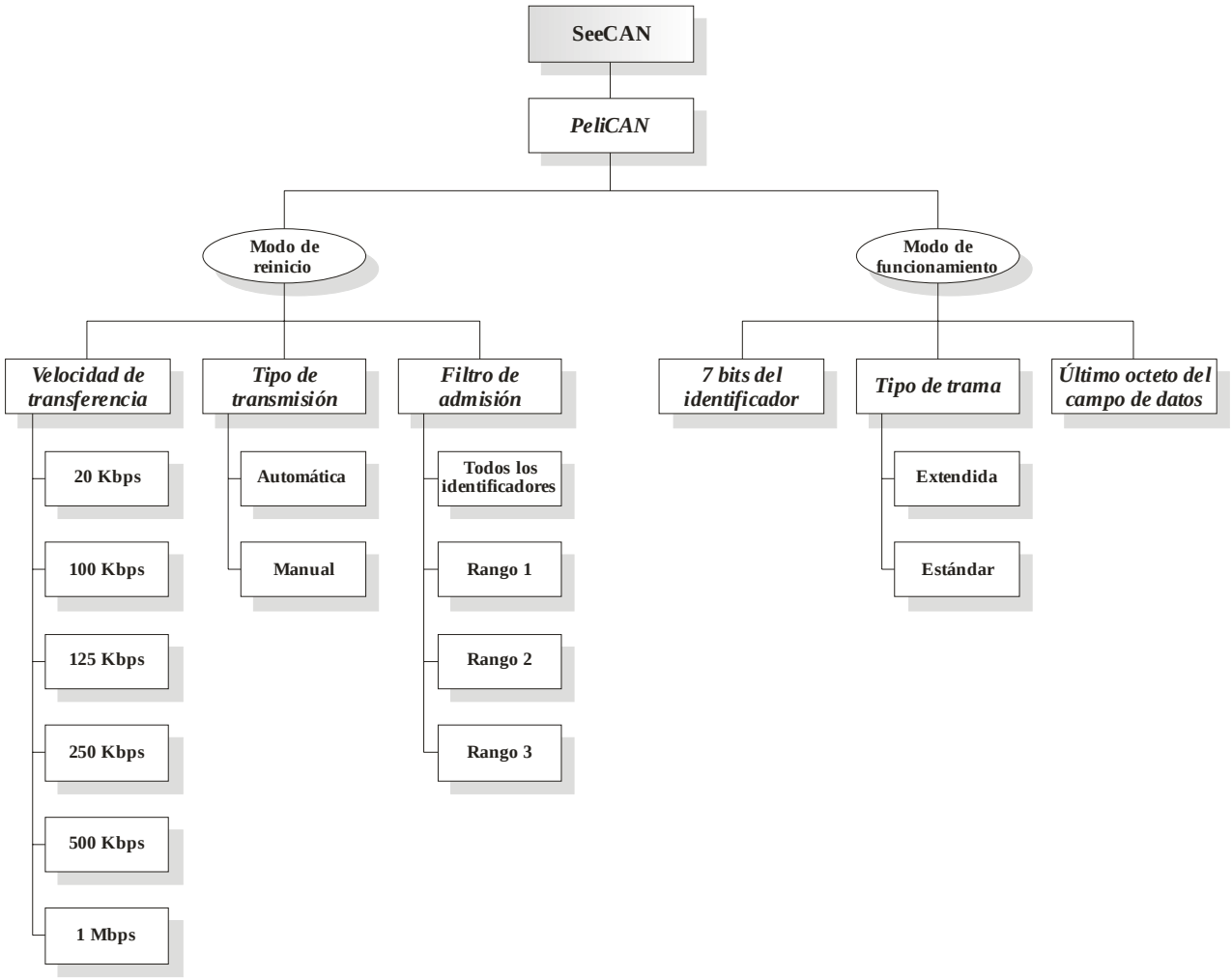


Figura 4.4. Diagrama de opciones para configurar el SeeCAN.

En modo de funcionamiento, el DIP1 se utiliza para seleccionar:

- El tipo de identificador de la trama a transmitir, 11 o 29 bits (interruptor 7).
- 7 bits del identificador de la trama a transmitir (interruptores 0 a 6).

El módulo M8DES se utiliza sólo en modo de funcionamiento para activar o desactivar los ocho bits del último octeto del campo de datos en una trama CAN. Consta de 8 interruptores configurados como entradas digitales (información a transmitir), y 8 leds que muestran las salidas digitales (información que se recibe).

El LCD se encarga de mostrar la información respecto al estado del nodo, mensajes recibidos, indicación de inicio del nodo, errores en el bus y pérdida de arbitraje.

El MSE indica el estado de error en el que se encuentra el nodo, consta de 3 leds: verde, para indicar estado de error activo; amarillo, para indicar estado de error pasivo; y rojo, para indicar estado de nodo desconectado (apartado 2.3.1).

Cada nodo se alimenta con una fuente de 5 VCD @ 500mA.

4.2.2. División del diseño del SeeCAN en sus componentes HW y SW

Con base en las especificaciones del SeeCAN, su diseño se dividió de acuerdo a la Tabla 4.1:

Tabla 4.1. División del diseño del SeeCAN (*Inicialización de SJA1000, LCD).

HW	SW
Comunicación CAN	Configuración/Control del nodo*
Unión al medio físico	Lectura del DIP1
Control de LCD	Escritura de mensajes en LCD
Control de MSE	Información del MSE
Decodificador de direcciones	Control del M8DES
Reinicio del nodo	Control de interrupciones
Envío de tramas en forma manual	Envío de tramas en forma automática

El HW de los nodos SeeCAN se compone de los siguientes dispositivos:

- MCU ATmega8515L de la firma Atmel.
- Controlador independiente CAN SJA1000.
- Transceptor de bus CAN PCA82C250.
- Visualizador de cristal líquido LCD.
- Dispositivo lógico programable GAL22V10.
- Memoria intermedia 74HC245 de las firmas Motorola y National.
- Optoacopladores 6N137 de la firma Agilent.
- Convertidores CD/CD DCP010505BP de la firma Texas Instruments.

4.2.2.1. Selección HW y SW

Como administrador del nodo se seleccionó al MCU ATmega8515 por las siguientes razones:

- Direccionamiento de memoria externa compatible con la familia 8051 de Intel.
- Compatibilidad en disposición física con el MCU 8051 de Intel.
- Arquitectura Harvard.
- Tarjeta de desarrollo STK500 de la firma Atmel, compatible con el MCU ATmega8515 y el AVR Studio.
- Velocidad de operación de hasta 8 MHz.

Como controlador CAN se seleccionó el SJA1000 por las siguientes características:

- Interfaz directa con la familia de MCUs Intel 8051 y compatibilidad con la interfaz del MCU ATmega8515.
- Frecuencia de reloj máxima de 24 MHz.
- Soporte a la especificación CAN 2.0A/B.
- Permite el uso de una gran variedad de transceptores discretos o integrados.

Como interfaz física entre el SeeCAN y el bus, se seleccionó al transceptor PCA82C250 debido a que cumple con la norma ISO 11898.

Como herramientas de desarrollo para el diseño HW y SW, simulación, implementación, pruebas y actualización del SeeCAN se utilizaron:

- *SW de aplicación AVR Studio de la firma Atmel*: se utilizó en su versión 4.10, en un entorno de sistema operativo Windows 2000/XP, para la escritura y depuración de aplicaciones AVR. Incluye un entorno de desarrollo integrado (IDE, *Integrated Development Enviroment*), el cual contiene editor, ensamblador, depurador y simulador, además de brindar soporte a emuladores AVR y tarjetas de desarrollo como la STK500. El AVR Studio es una aplicación de uso público distribuido gratuitamente (*freeware*).
- *SW para programación OPAL de la firma National*: SW de programación de dispositivos lógicos (GAL y PAL) que permite desarrollar y simular sistemas digitales, se utilizó la versión 1.02.
- *Tarjeta de desarrollo STK500*: sistema de entrenamiento y herramienta de programación y desarrollo para los MCUs AVR de la firma Atmel. Permite la implementación de código en forma rápida debido a sus características para el desarrollo de prototipos y prueba de nuevos diseños.

4.2.3. Iteración y desarrollo del SeeCAN

Las tareas que se realizaron en el entorno AVR Studio son las siguientes (Figura 4.5):

- Escribir y depurar programas de prueba.
- Simular la lectura y escritura a direcciones de memoria externa, así como la configuración de los registros del MCU.
- Verificar la ejecución de las subrutinas, la activación de interrupciones y el tiempo de ejecución de las mismas.
- Descargar el programa al MCU mediante el sistema STK500.
- Programar los bits internos del MCU para seleccionar las opciones de funcionamiento.

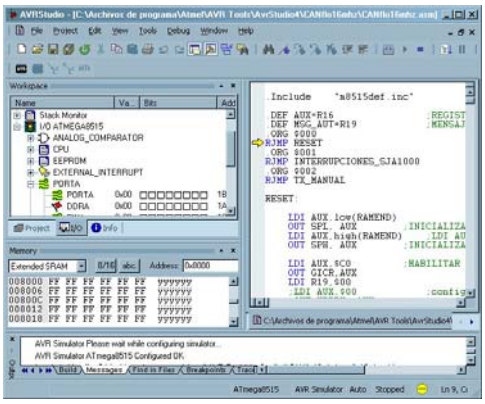


Figura 4.5. Entorno de desarrollo AVR Studio.

La simulación del PLD se llevó a cabo utilizando el software OPAL, mediante las siguientes actividades:

- Escribir el programa para la decodificación de direcciones de memoria externa.
- Verificar la decodificación de direcciones de memoria externa.

Las tareas iterativas HW y SW, de corrección o de ajuste, son las siguientes:

- Asignar direcciones de memoria externa en el PLD.
- Asignar funciones a los puertos del MCU.
- Configurar los registros del MCU.
- Agregar y/o modificar las conexiones HW necesarias.

4.2.4. Diseño paralelo HW y SW del SeeCAN

4.2.4.1. Diseño HW del SeeCAN

Una vez seleccionado los componentes HW del SeeCAN, se definen los siguientes aspectos:

- Mapa de direcciones de memoria externa para cada dispositivo (Tabla 4.2).
- Asignación de funciones a los puertos del MCU (Tabla 4.3).
- Estrategias de oscilador y señal de reloj del MCU y del controlador CAN (Tabla 4.4).

4.2.4.1.1. Mapa de memoria de los registros del MCU (Interfaz MCU/Controlador CAN

El SJA1000 proporciona un bus multiplexado de datos y direcciones, así como las señales de control para la conexión con el MCU. La interfaz MCU/Controlador CAN se realiza configurando al SJA1000 en modo Intel mediante la conexión de su terminal 11 a V_{cc}.

- Tabla 4.5).

Tabla 4.2. Mapa de direcciones de memoria externa.

Direcciones	Dispositivo o periférico
FB00h / FC00h	LCD (Orden/Dato)
FD00h – FDFFh	SJA1000
FE00h	DIP1
FF00h	Interruptores del M8DES
8000h – FAFFh	Futuros periféricos

Tabla 4.3. Función para cada puerto del MCU.

Puerto	Función
A	Bus de datos y direcciones (AD0 - AD7)
B	M8DES
C	Bus de direcciones (A8 – A15)
D	Bits 4 y 5 del MSE
E	Bit 2 del MSE

Tabla 4.4. Estrategias de oscilador y señal de reloj.

Dispositivo	Frecuencia del oscilador externo		Observaciones
	Máxima	Utilizada	
SJA1000	24 MHz	16/24 MHz	Deshabilitar la terminal CLKOUT para mejorar la relación señal/ruido en el nodo.
Atmega8515	8 MHz	8 MHz	Deshabilitar el oscilador interno programando los bits respectivos en el MCU.

4.2.4.1.2. Interfaz MCU/Controlador CAN

El SJA1000 proporciona un bus multiplexado de datos y direcciones, así como las señales de control para la conexión con el MCU. La interfaz MCU/Controlador CAN se realiza configurando al SJA1000 en modo Intel mediante la conexión de su terminal 11 a V_{cc}.

Tabla 4.5. Mapa de memoria de los registros del MCU.

Registro	Nombre	Dirección
SPH	Apuntador de pila (alto)	3Eh
SPL	Apuntador de pila (bajo)	3Dh
GICR	Control de interrupciones general	3Bh
MCUCR	Control del MCU	35h
PORTB	Datos del puerto B	18h
DDRB	Dirección de datos del puerto B	17h
PORTD	Datos del puerto D	12h
DDRD	Dirección de datos del puerto D	11h
PORTE	Datos del puerto E	07h
DDRE	Dirección de datos del puerto E	06h

4.2.4.1.3. Control de la comunicación CAN

El programa del MCU se encarga de la funcionalidad, respecto a configuración y tareas del controlador CAN, para cumplir con los requisitos de un sistema de comunicaciones CAN.

El intercambio de datos entre el MCU y el controlador CAN se realiza a través de un conjunto de registros (segmento de control) y una RAM (memoria temporal de mensajes) (inciso 3.5.1.2). En la Tabla 4.6 se agrupan los registros del controlador CAN (modo *PeliCAN*) que se utilizan en el diseño del SW del SeeCAN.

Tabla 4.6. Mapa de memoria de los registros del controlador CAN.

Tipo de registro	Nombre de registro	Dirección
Para seleccionar los modos de operación	Modo (MOD)	00h
	Divisor de Reloj (CDR)	1Fh
Para configurar la comunicación CAN	Código de admisión (ACR)	10h-13h
	Máscara de admisión (AMR)	14h-17h
	Temporización de bus 0 (BTR0)	06h
	Temporización de bus 1(BTR1)	07h
	Control de salida (OCR)	08h
Básicos para la comunicación CAN	Órdenes (CMR)	01h
	Estado (SR)	02h
	Interrupción (IR)	03h
	Activar interrupción (IER)	04h
Para analizar y detectar errores	Captura de pérdida de arbitraje (ALC)	11h
	Captura de código de error (ECC)	12h
	Límite de advertencia de error (EWL)	13h
	Contador de errores de RX (RXERR)	14h
	Contador de errores de TX (TXERR)	15h
Memoria temporal de mensajes	Memoria temporal de TX (TXBUF)	10h-1Ch
	Memoria temporal de RX (RXBUF)	10h-1Ch

4.2.4.1.4. Interfaz de capa física CAN

Al utilizar el PCA82C250 como transceptor se debe considerar lo siguiente:

- Activar la función de enlace del comparador (*bypass comparator*) del SJA1000.
- Aislar eléctricamente el nodo mediante la utilización de optoacopladores.

4.2.4.1.5. Información necesaria para el Ingeniero de SW

El Ingeniero de SW debe conocer el mapa de direcciones de memoria externa (Tabla 4.2), las funciones asignadas a cada puerto del MCU (Tabla 4.3), y el vector de interrupciones del MCU y del SJA1000.

4.2.4.2. Diseño de SW del SeeCAN

El diseño SW del SeeCAN se basa en las especificaciones iniciales del sistema, considerando los siguientes factores:

- Interfaz del MCU con el controlador CAN.
- Interfaz del MCU con otros periféricos.
- Mapas de memoria del SeeCAN.
- Asignación de puertos del MCU.
- Estrategia de oscilador.
- Interfaz de capa física.
- Interfaz de usuario.

El SW del SeeCAN se basa en un programa principal, el cual se compone de varias subrutinas y rutinas de servicio a interrupción (ISR, *Interrupt Service Routine*). La Figura 4.6 muestra el diagrama de flujo del programa principal del SeeCAN y la Figura 4.7 muestra el diagrama de flujo de las ISR.

El código de configuración del MCU se basó en una programación estructurada considerando la fase de mantenimiento y actualización.

Las subrutinas son: inicializar administrador del nodo, inicializar controlador del protocolo CAN, inicializar LCD, configurar E/S, y procedimiento local.

El SeeCAN utiliza las interrupciones del MCU, INT0 e INT1, para proporcionar los siguientes servicios:

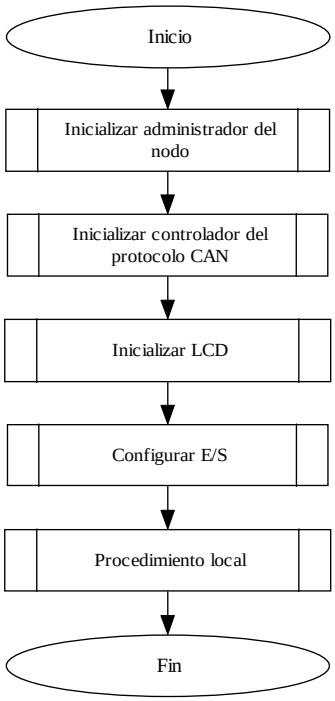


Figura 4.6. Diagrama de flujo del programa principal del SeeCAN.

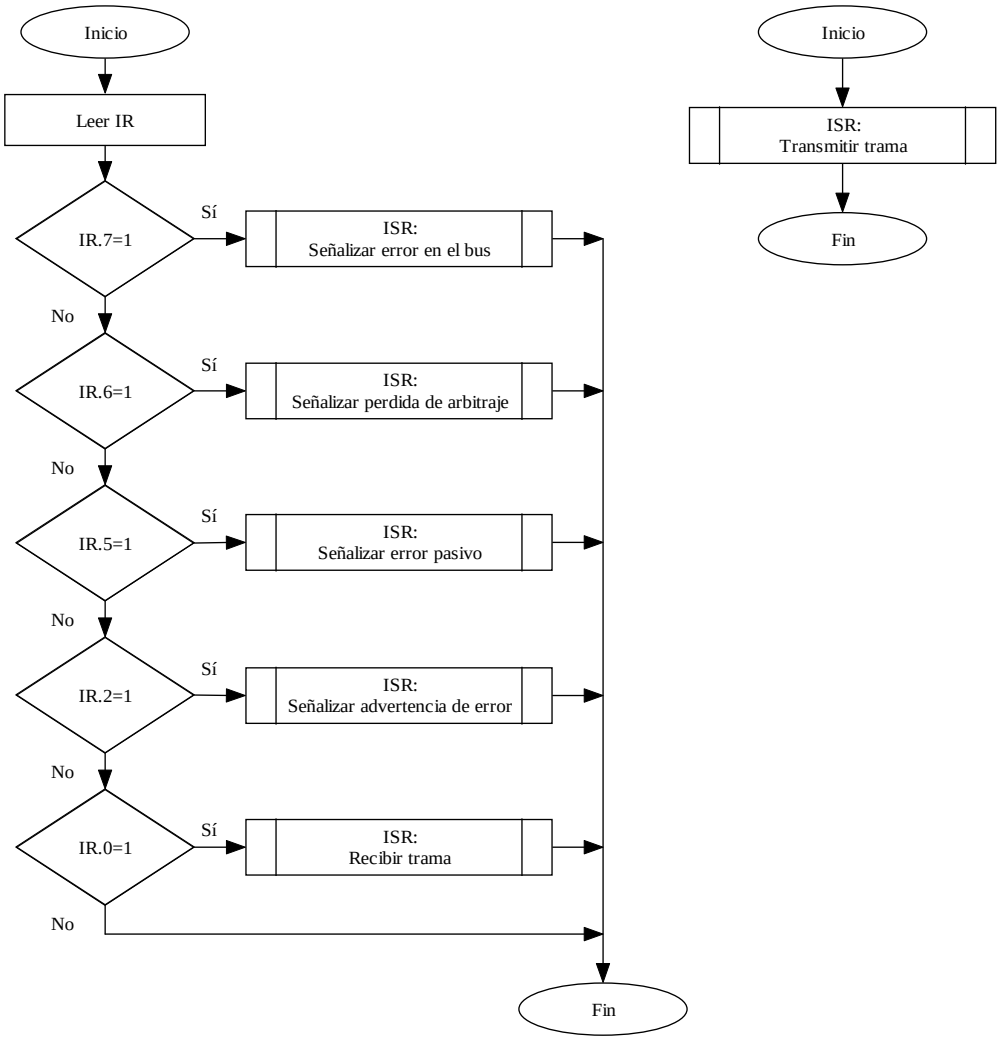


Figura 4.7. Diagrama de flujo de las ISR, INT0 e INT1, del SeeCAN.

- *Control de interrupciones requeridas por el controlador CAN (INT0)*: incluye las funciones de recepción de tramas CAN, señalización de error de bus y de error pasivo, advertencia de errores y pérdida de arbitraje. Para el manejo de las diversas interrupciones, el MCU debe identificar el tipo de interrupción mediante el registro IR.
- *Control manual para la transmisión de tramas CAN (INT1)*: realiza la función de enviar una trama CAN a cualquier otro nodo en la red, configurando previamente el identificador del mensaje y parte de los datos. Dicha interrupción se activa por el usuario.

En los siguientes párrafos se describe el SW del SeeCAN.

4.2.4.2.1. Inicializar administrador del nodo

El administrador del nodo es el primer componente HW del SeeCAN que se inicializa y configura, por lo tanto la primera subrutina del programa principal se encarga de ello mediante la configuración de los registros del MCU. La subrutina se lleva a cabo cada vez que el SeeCAN inicia su funcionamiento.

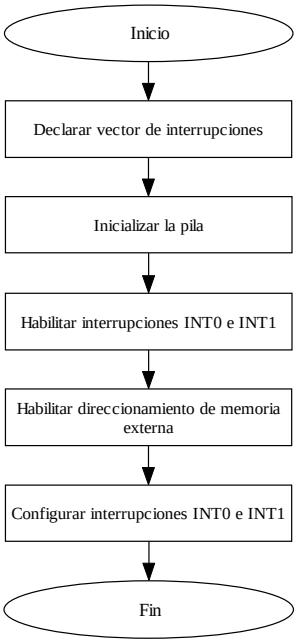


Figura 4.8. Diagrama de flujo de la subrutina de inicialización del administrador del nodo.

De acuerdo con la Figura 4.8, la subrutina realiza lo siguiente:

- *Declarar el vector de interrupciones del MCU*: a cada interrupción se le asigna su correspondiente número de vector, así como sus respectivas etiquetas de identificación.
- *Inicializar la pila del MCU*: se configuran los registros SPL y SPH para asignar la dirección de memoria que deben tener los apuntadores de la pila del MCU.
- *Habilitar las interrupciones INT0 e INT1*: se configura el registro GICR para habilitar las interrupciones INT0 e INT1.
- *Habilitar el direccionamiento de localidades de memoria externa*: se configuran los bits del MCUCR para habilitar el direccionamiento de las localidades de memoria externa.
- *Configurar las interrupciones INT0 e INT1*: se configuran los bits del MCUCR para seleccionar la activación por flanco de bajada de las interrupciones INT0 e INT1.

4.2.4.2.2. Inicializar controlador del protocolo CAN

La subrutina de inicialización del controlador CAN se ejecuta después de que el administrador del nodo se encuentra configurado correctamente.

El controlador CAN se configura después que se alimenta con 5V o después de un reinicio del sistema. La Figura 4.9 muestra el diagrama de flujo de la subrutina de inicialización del controlador CAN. El procedimiento es el siguiente:

- *Habilitar el modo de reinicio del controlador CAN*: el administrador del nodo realiza la petición de reinicio mediante la configuración del bit cero del registro MOD.
- *Verificar el ingreso al modo de reinicio del controlador CAN*: mediante una comparación se verifica que el registro MOD tenga el valor correspondiente al modo de reinicio.
- *Configurar el registro CDR*: el administrador del nodo escribe el valor correspondiente en el CDR para seleccionar el modo *PeliCAN*, tipo de configuración del comparador de entrada, activación de la señal CLKOUT, y función de la terminal TX1.
- *Deshabilitar fuentes de interrupción CAN en el administrador del nodo*: el administrador del nodo configura el valor correspondiente para deshabilitar las interrupciones en el IER.

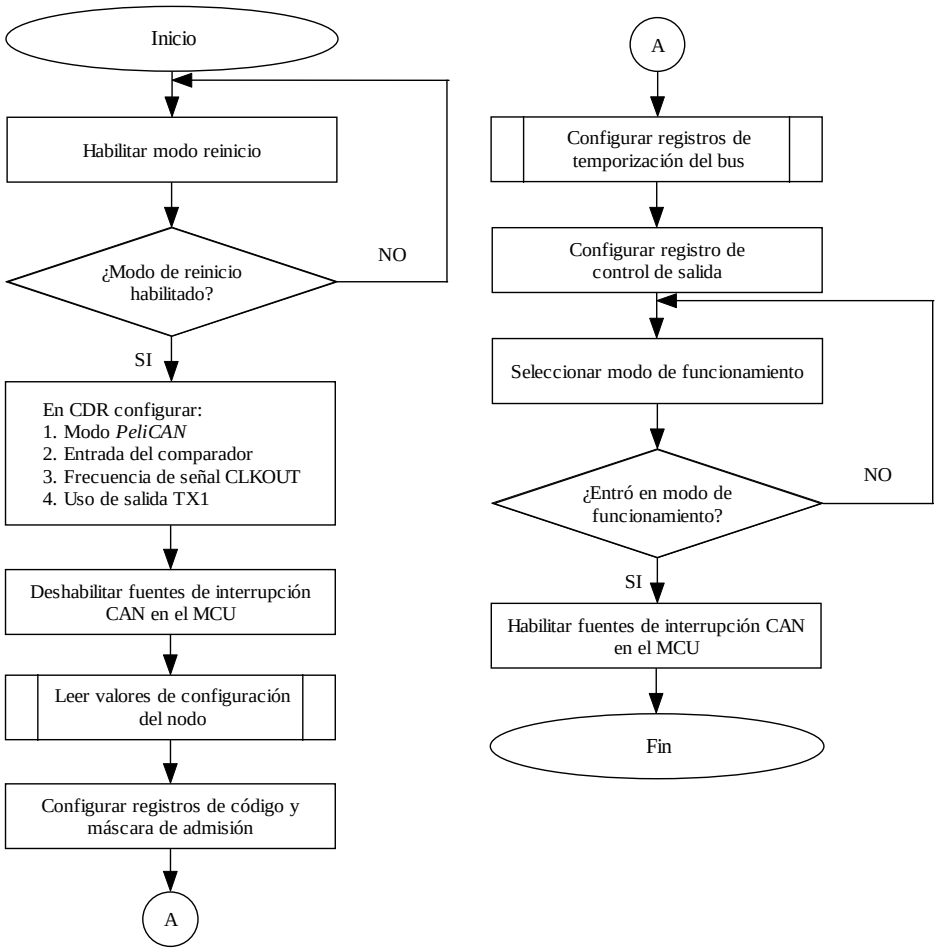


Figura 4.9. Diagrama de flujo de la subrutina de inicialización y configuración del controlador CAN.

- *Leer valores de configuración del nodo*: el administrador del nodo lee el valor correspondiente en el DIP1, que determina la velocidad de transferencia de datos, tipo de transmisión, y la configuración del filtro de mensajes.
- *Configurar los registros de código de admisión y máscara de admisión*: se configuran los valores del los registros AMR y ACR para establecer el filtro de mensajes, de acuerdo a los valores que se obtienen en la lectura del DIP1.
- *Configurar los registros de temporización del bus*: se configuran los valores de los registros BTR0 y BTR1 para establecer la velocidad de transferencia de datos, de acuerdo a los valores que se obtienen en la lectura del DIP1.
- *Configurar registro de control de salida:* se configura el valor del registro OCR para establecer el comportamiento de la salida del controlador CAN.
- *Habilitar el modo de funcionamiento del controlador CAN*: el administrador del nodo realiza la petición de ingresar al modo de funcionamiento del controlador CAN mediante la escritura del bit cero del registro MOD.
- *Verificar el ingreso al modo de funcionamiento del controlador CAN*: mediante una comparación se verifica que el registro MOD tenga el valor correspondiente al modo de funcionamiento.
- *Habilitar las fuentes de interrupción CAN en el MCU*: se habilitan las interrupciones requeridas en el registro IER del controlador CAN.

4.2.4.2.3. Inicializar visualizador LCD

El LCD se inicializa y controla mediante una secuencia de órdenes que el administrador del nodo envía al registro de control del LCD, es necesario realizar un retardo para la correcta ejecución de cada orden.

La subrutina de inicialización del LCD se ejecuta una sola vez, sin embargo se pueden ejecutar otras órdenes, como escritura de caracteres, borrado de LCD, salto de línea, sin que se altere la configuración inicial.

Una vez ejecutadas las órdenes, la configuración del LCD es la siguiente:

- Pantalla de 2 líneas de 16 caracteres cada una.
- Dimensión de caracteres de 5x7 puntos.
- Cursor encendido e intermitente.
- Desplazamiento del cursor hacia la derecha.

4.2.4.2.4. Configurar entradas y salidas del SeeCAN

La subrutina de configuración de entradas y salidas del SeeCAN se encarga de: configurar el funcionamiento de los puertos del MCU, como entradas y/o salidas; activar el indicador de estado del nodo; indicar el inicio de funcionamiento del nodo mediante el LCD; y habilitar la interrupción global. De acuerdo a la Figura 4.10 esta subrutina realiza lo siguiente:

- *Configurar puertos B, D y E*: se configuran los bits de los registros DDRB, DDRD y DDRE para habilitarlos como salidas de acuerdo a la Tabla 4.3.
- *Encender el led indicador de estado*: se escribe el valor de uno lógico en el bit cuatro del registro PORTD, para encender el led que indica el estado de error activo del nodo.
- *Habilitar la interrupción global*: se ejecuta la orden para habilitar todas las interrupciones que el administrador del nodo puede generar.
- *Inicio de funcionamiento del nodo*: el LCD indica el inicio del funcionamiento del nodo.

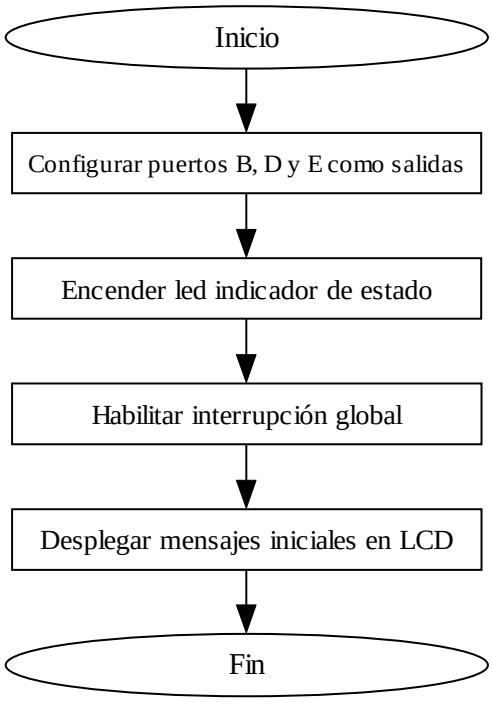
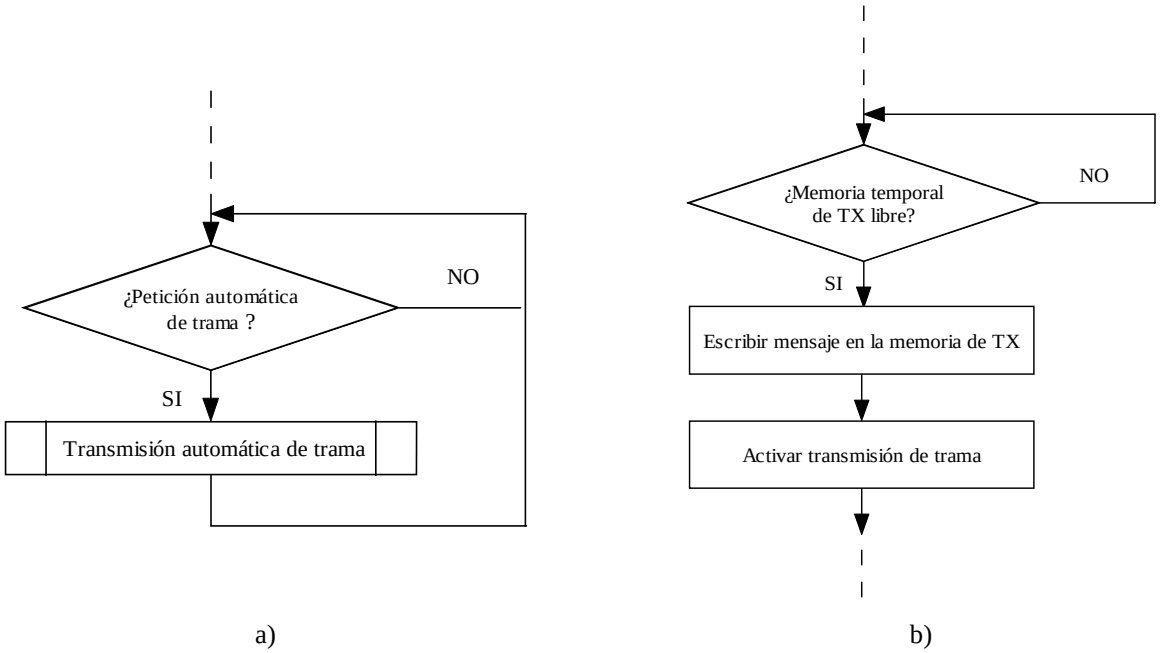


Figura 4.10. Diagrama de flujo de la subrutina de configuración de entradas y salidas del SeeCAN.



a) b)

Figura 4.11. Subrutinas del procedimiento local y de transmisión de trama automática por muestreo.

4.2.4.2.5. Procedimiento local de muestreo

El procedimiento local de muestreo es una subrutina que genera un ciclo infinito (Figura 4.11a). Dicha subrutina verifica si está activada la opción de transmisión de trama automática, si la opción es afirmativa entonces se realiza la subrutina de transmisión de trama automática (Figura 4.11b), de otra forma regresa a verificar la opción.

Está subrutina se realiza todo el tiempo y sólo se suspende si se genera alguna interrupción.

4.2.4.2.6. Rutinas de servicio a interrupción

Las rutinas de servicio a interrupción del SeeCAN son las siguientes:

- *Transmitir trama*: si el usuario presiona el botón TxM, se habilita la interrupción INT1 del MCU (Figura 4.12). Esta ISR se encarga de leer datos del DIP1, verificar la disponibilidad de memoria temporal de transmisión, leer datos del M8DES, verificar el tipo de identificador, configurar tipo de trama, longitud del campo de datos y su identificador, escribir mensaje en la memoria temporal de transmisión y activar la transmisión.
- *Señalizar errores en el bus*: se encarga de leer el código de error en el registro ECC para determinar el tipo de error, si ocurre en la transmisión o en la recepción de una trama, y desplegar en el LCD la información sobre el error detectado.
- *Señalizar pérdida de arbitraje*: esta ISR se encarga de leer el código del número de bit donde ocurrió la pérdida de arbitraje en el registro ALC, y de desplegar en el LCD la información de pérdida de arbitraje.
- *Señalizar error pasivo*: se encarga de leer el valor de los registros RXERR y TXERR, compararlos con el número de errores máximo del estado de error activo, determinar el estado actual del nodo y activar el led correspondiente en el MSE.
- *Señalizar advertencia de error*: se encarga de leer el valor de los bits 6 y 7 del registro SR, comparar los valores de los registros RXERR y TXERR con el número de errores máximo de los estados de error activo y pasivo, determinar el estado actual del nodo y activar el led correspondiente en el MSE.

- *Recibir trama*: se encarga de leer la trama de la memoria temporal de recepción y almacenarla, leer los registros de identificador y de información de trama de recepción, determinar si el tipo de trama es estándar o extendida, desplegar en el LCD el mensaje recibido, activar las salidas del M8DES, liberar la memoria temporal de recepción, y procesar el mensaje recibido (Figura 4.13).

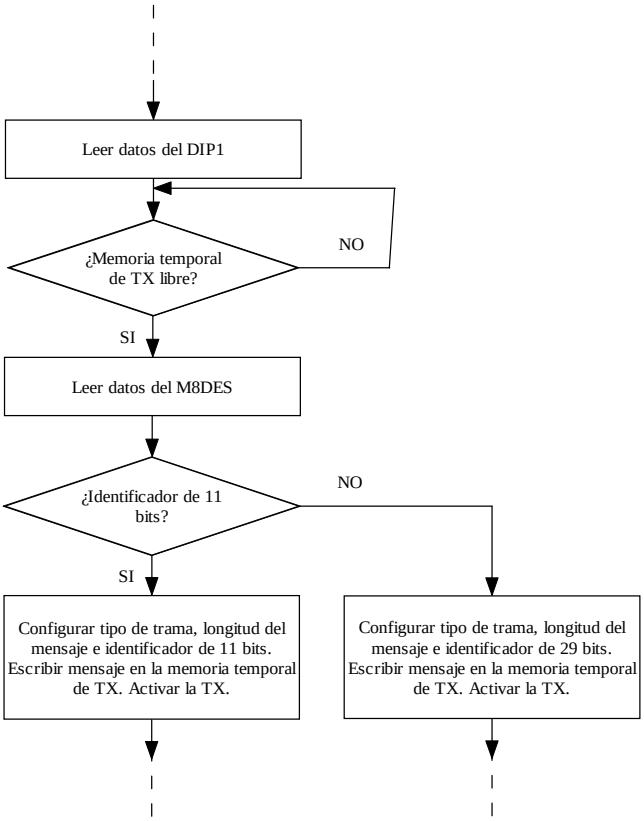


Figura 4.12. Diagrama a bloques de la ISR transmitir trama.

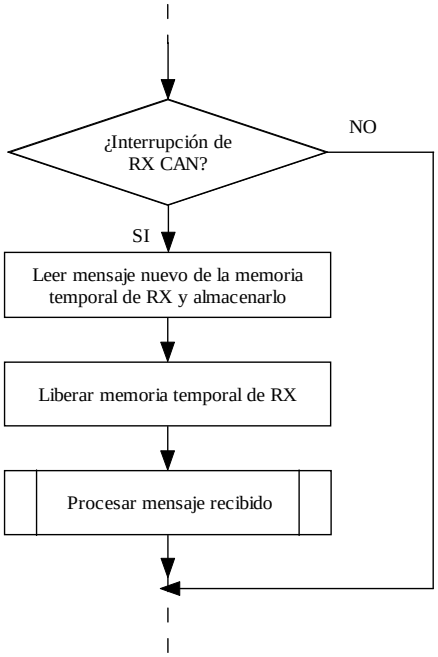


Figura 4.13. Diagrama de flujo de la ISR recibir trama.

4.2.5. Integración HW y SW del SeeCAN

La integración HW y SW del SeeCAN consistió en realizar las siguientes tareas:

- Verificar las conexiones de los componentes HW.
- Ejecutar programas de prueba a los periféricos (SJA1000, LCD, DIP1, M8DES, MSE).
- Descargar el programa principal al MCU mediante el sistema STK500.
- Ejecutar el programa principal y validar su funcionamiento.

4.2.6. Verificación del SeeCAN

Para verificar el correcto funcionamiento del sistema final, se implemento una red CAN con dos nodos SeeCAN y el sistema de entrenamiento CAN de la firma esd.

Inicialmente, la red CAN consistió de dos nodos SeeCAN y mediante sus módulos de E/S se realizaron las siguientes pruebas:

- Transmisión de tramas CAN en forma automática o manual.
- Transmisión de tramas CAN con identificadores de 11 o 29 bits.
- Transmisión del último octeto del campo de datos.
- Transmisión de tramas CAN a diferentes velocidades de tranferencia de datos.
- Configuración el filtro de admisión y de los identificadores.
- Señalización de errores (Figura 4.14 y Figura 4.15).
- Visualización de mensajes almacenados en memoria del MCU y de los mensajes de recepción (Figura 4.16).
- Visualización de las tramas CAN en un osciloscopio para su análisis.

En segundo lugar, se agregó el sistema de desarrollo CAN (*CAN Starter Kit*) de la firma esd GmbH, el cual consta de dos nodos y el SW monitor de bus CANscope [11, 12]. Las pruebas realizadas incluyen a las anteriores y se agregan las siguientes:

- Transmisión de tramas con identificadores definidos en el rango de 000h a 7FFh mediante CANscope.
- Transmisión de 1 a 8 octetos de información definidos en el CANscope (Figura 4.17)

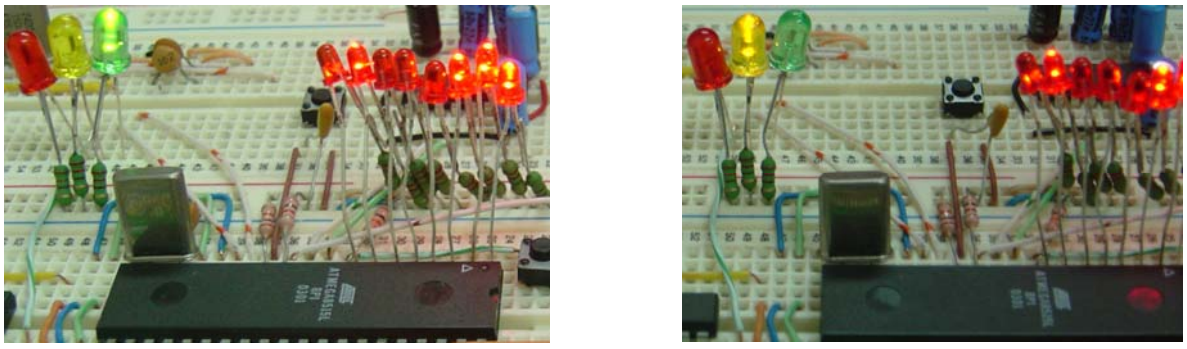


Figura 4.14. Señalización de estado de error activo y de error pasivo en el SeeCAN.

Figura 4.15. Señalización de errores en el bus CAN.

Figura 4.16. Recepción de una trama CAN.

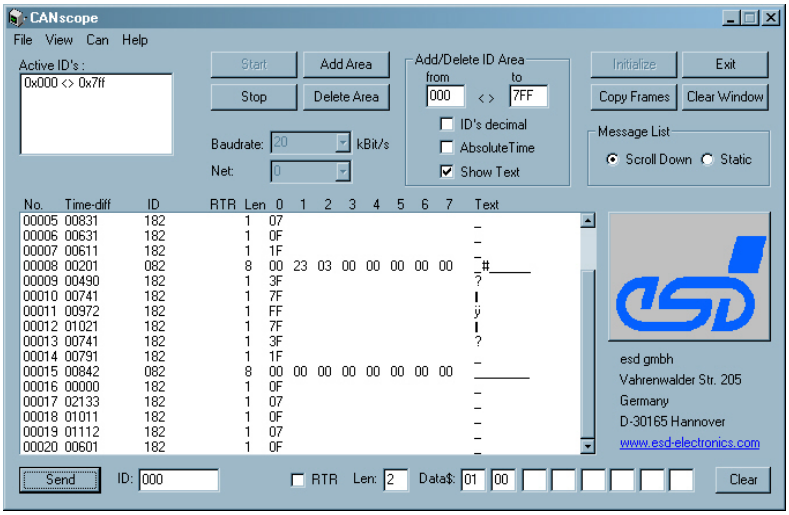


Figura 4.17. Monitor de tramas CANscope.

4.2.7. Mantenimiento y actualización del SeeCAN

La Tabla 4.7 muestra las actualizaciones realizadas al SeeCAN durante su ciclo de vida.

Tabla 4.7. Versiones de actualización del SeeCAN.

Versión	Actualizaciones
0.1	Configuración de registros del MCU para habilitar las funciones requeridas en las especificaciones iniciales. Recepción de datos mediante la interfaz serial del MCU y control del LCD. El HW del nodo CAN es: MCU, LCD, decodificador de direcciones, e interfaz serie entre el MCU y la PC.
0.2	Inicialización del SJA1000 en modo *BasicCAN*. Transmisión de tramas en formato estándar, mediante la interrupción serial del MCU; recepción de tramas por monitoreo de la RXB y despliegue de los mensajes en el LCD. Configuración del filtro de admisión para permitir la recepción de todos los mensajes. El programa determina el identificador y el campo de datos, el usuario no puede modificarlos. Velocidad de transferencia de datos de 250 Kbps. Conexión del controlador SJA1000 y el PCA82C250 al HW del nodo.
0.3	Inicialización del SJA1000 en modo *PeliCAN*. Transmisión y recepción de tramas en formato estándar y extendido.
0.4	Configuración de la INT0 del MCU para dar soporte a las interrupciones CAN de: error de bus (BEI), pérdida de arbitraje (ALI), error pasivo (EPI), advertencia de error (EI) y recepción (RI).
0.5	Configuración de la INT1 del MCU para dar soporte a la transmisión de tramas en forma manual. Programación de las subrutinas e ISRs para modificar los parámetros mediante la interfaz de usuario. Se agregó el HW correspondiente a la interfaz de usuario y se elimina la interfaz serial. Comunicación CAN a velocidades de transferencia de datos entre 20 y 250 Kbps. El SeeCAN presenta problemas con las salidas del M8DES.
0.6	Reasignación de funciones a los puertos MSE y M8DES. Se eliminan los componentes 74LS573 (*latch*) de cada nodo.
0.7	Se agregan los optoacopladores y el convertidor CD/CD.
0.8	Problemas con las tramas enviadas mediante la herramienta de monitoreo CANScope, sin embargo el nodo responde con las tramas de error correspondientes.
0.9	Ajustes en los registros BTR0 y BTR1 del SJA1000 para modificar la frecuencia del cristal de 24 a 16 MHz.
1.0	Comunicación CAN a 500 Kbps. Recepción exitosa de tramas enviadas mediante la herramienta de monitoreo CANScope.

5. Resultados

Como resultado de la investigación realizada, se obtuvo el prototipo del SeeCAN (Figura 5.1), el cual integra las especificaciones iniciales del sistema (apartado 4.2.1). En la Figura 5.1 se identifica cada uno de los dispositivos que define el diagrama a bloques de la Figura 4.3, los cuales realizan las funciones de comunicación (inciso 4.2.1.1), administración del nodo (inciso 4.2.1.2) e interfaz de usuario (inciso 4.2.1.3).

La verificación del SeeCAN se realizó según lo indicado en el apartado 4.2.6. La Figura 5.2 muestra una trama CAN con formato estándar e identificador 001h, cuyo campo de datos contiene ocho octetos con los valores 45h, 60h, 55h, 45h, 60h, 55h, 45h y 60h. La Figura 5.3 muestra una trama CAN con formato extendido e identificador 00800000h, conteniendo ocho octetos con los valores 63h, 61h, 6Eh, 32h, 2Eh, 30h, 42h y 32h.

La flexibilidad del nodo SeeCAN se basa en las opciones de configuración (Figura 4.4) definidas en las especificaciones, y permite la configuración de diferentes velocidades de transferencia de datos (Figura 5.4), el tipo de transmisión (Figura 5.5) y el tipo de trama CAN (Figura 5.6).

El filtro de admisión se configura al inicializar el nodo, y permite cuatro opciones: Todos (00), Rango 1 (01, 00XXX…), Rango 2 (10, 0101010XXX…) y Rango 3 (11, 11XXX00XXX…).

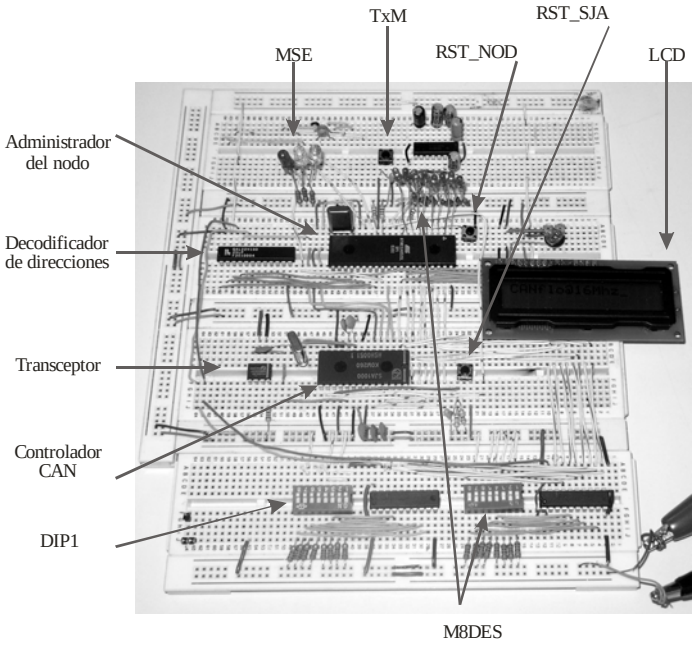
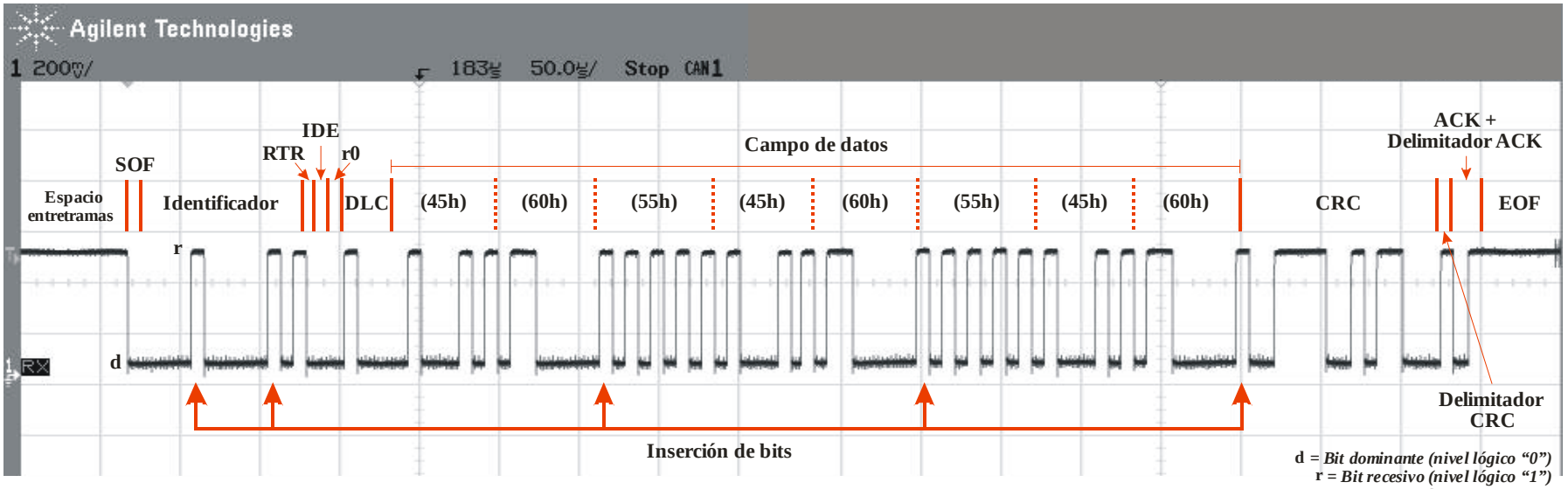
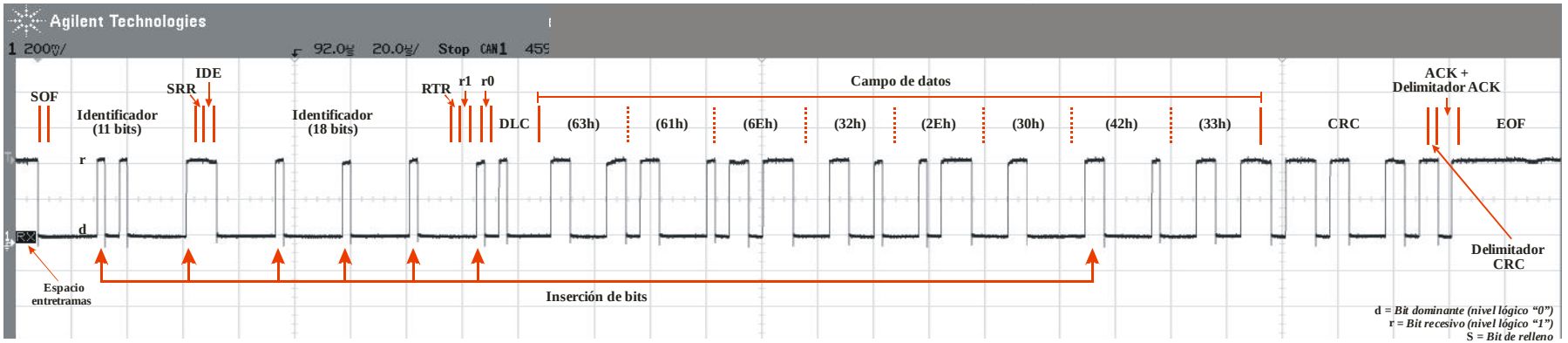


Figura 5.1. Prototipo del nodo SeeCAN.



SOF = d **Identificador (11 bits)** = ddd dSddd ddSdr (001h) **RTR** = d **IDE** = d **r0** = d **DLC** = rddd (8h)
Campo de datos = drdd drdr (45h) drrd ddddS (60h) drdr drdr (55h) drdd drdr (45h) drrd ddddS (60h) drdr drdr (55h) drdd drdr (45h) drrd ddddS (60h)
CRC = ddr rrrd drdr rddd (1E58h) **Delimitador CRC** = r **ACK** = d **Delimitador ACK** = r **EOF** = rrrrrrr

Figura 5.2. Análisis de una trama CAN con formato estándar.



SOF = d **Identificador (11 bits)** = d dddSd rddd ddS... (008h) **SRR** = r **IDE** = r **Identificador (18 bits)** = ...dd dddSd ddddS dddd dSddd (00000h) **RTR** = d **r1** = dS **r0** = d **DLC** = rddd (8h)
Campo de datos = drrd ddrr (63h) drrd dddr (61h) drrd rrrd (6Eh) ddrr ddrd (32h) ddrd rrrd (2Eh) ddrr dddd (30h) dSrdd ddrd (42h) ddrr ddrr (33h)
CRC = rdr rrdr rddd rrdr (5D8Dh) **Delimitador CRC** = r **ACK** = d **Delimitador ACK** = r **EOF** = rrrrrrr

Figura 5.3. Análisis de una trama CAN con formato extendido.

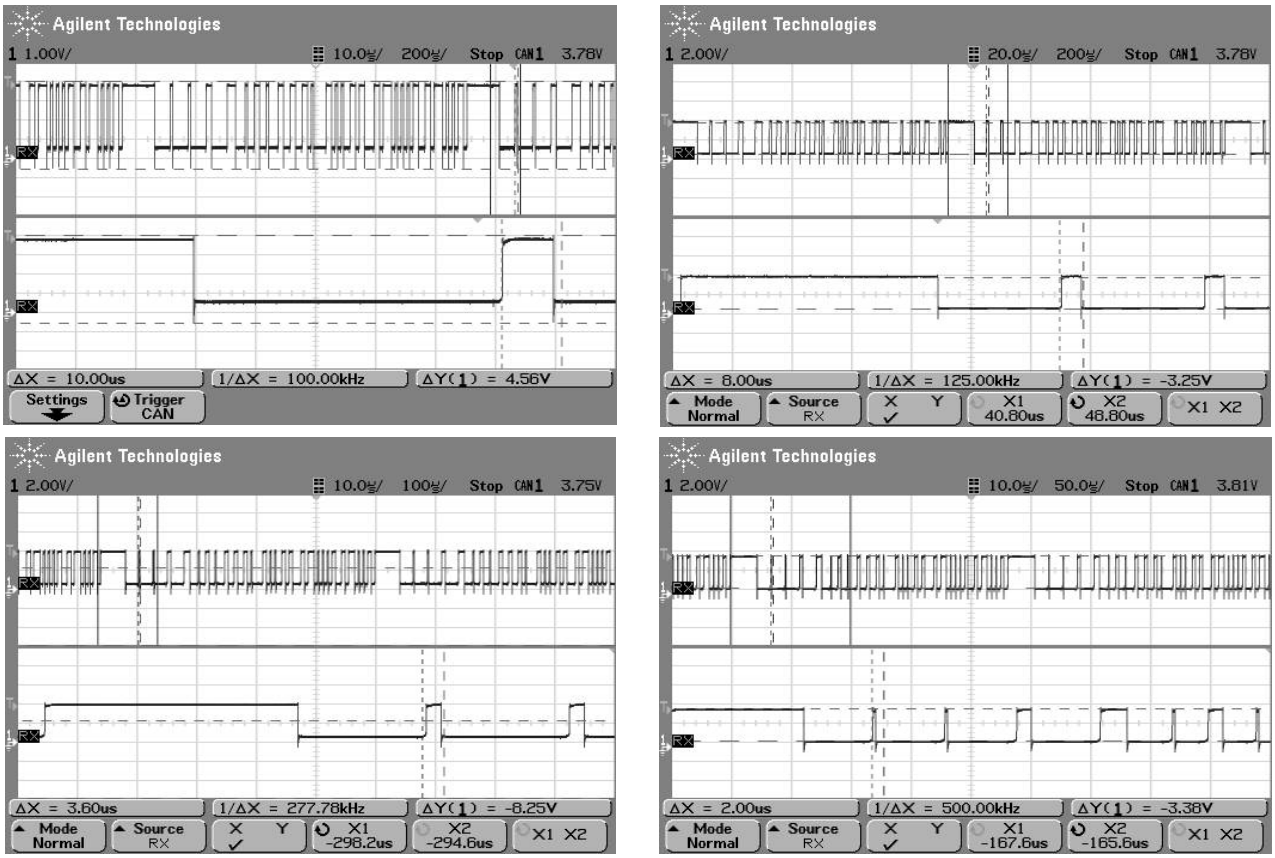


Figura 5.4. Tramas CAN a velocidades de transferencia de 100, 125, 250 y 500 Kbps.

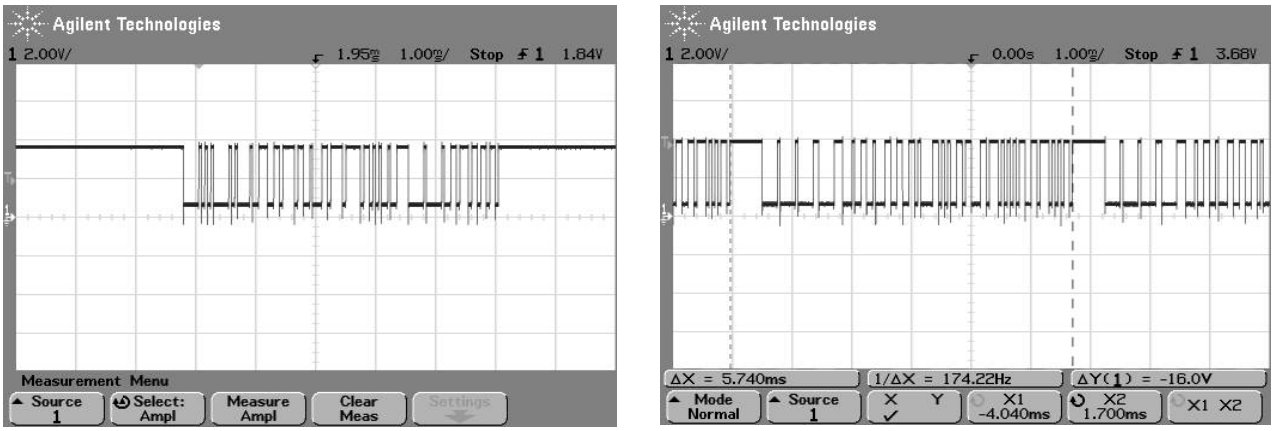


Figura 5.5. Transmisión, manual y automática, de tramas de datos CAN a velocidad de transferencia de 20 Kbps.

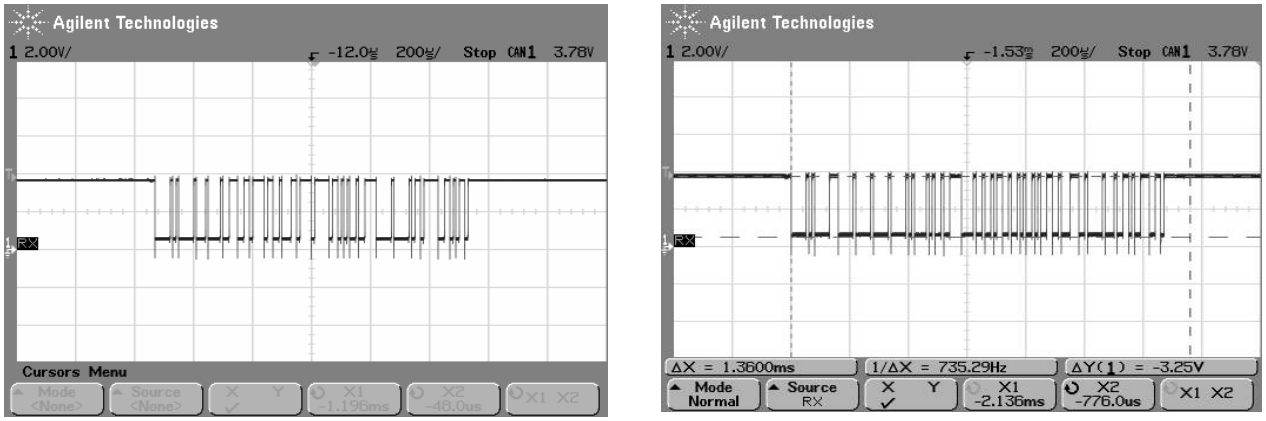


Figura 5.6. Formato de trama de datos CAN, estándar y extendido, a velocidad de transferencia de 100 Kbps.

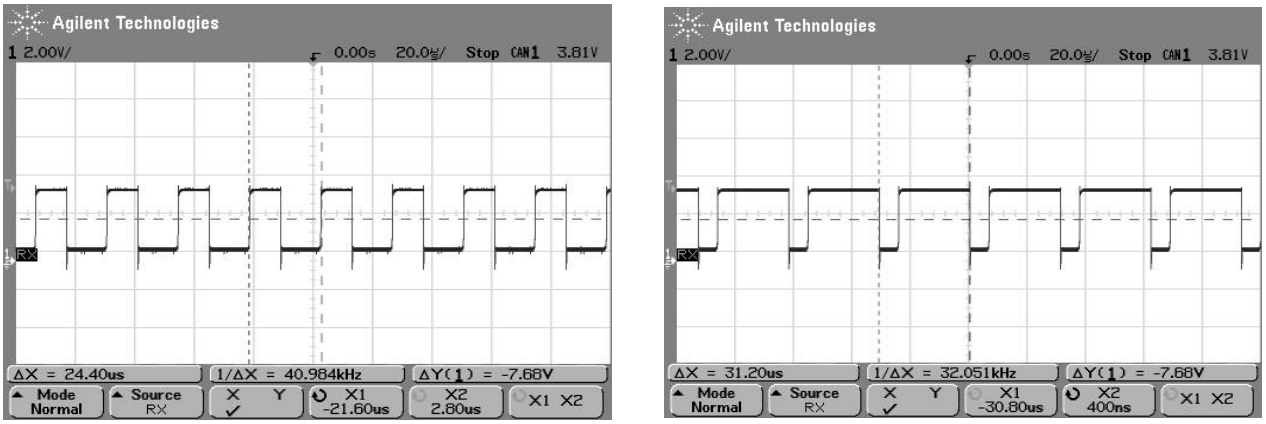


Figura 5.7. Visualización de tramas de error, de transmisión y de recepción.

La Figura 5.7 muestra las tramas de error de transmisión y de recepción, respectivamente, las cuales se indican mediante los módulos MSE y LCD del SeeCAN (Figura 4.15).

El prototipo del SeeCAN se realizo en una tarjeta de pruebas (*protoboard*) y la máxima velocidad de transferencia de datos obtenida fue de 500 Kbps. Para lograr el funcionamiento a velocidades de hasta 1 Mbps, se propone el desarrollo de la tarjeta de circuito impreso que se propone en el Anexo B.

Con respecto al desarrollo anterior [16, 17], el SeeCAN proporciona las siguientes ventajas: es autónomo y flexible, ya que permite al usuario configurar diferentes parámetros (Figura 4.4); además soporta el manejo y señalización de errores en el nodo; y su diseño permite la expansión de dispositivos periféricos. La Tabla 5.1 presenta una comparativa entre el SeeCAN y los sistemas utilizados en este trabajo de tesis.

Tabla 5.1. Comparativa de las principales características del SeeCAN respecto a otros diseños.

Características	**SeeCAN**	**Nodo [16, 17]**	**CAN-PCI/331**	**CAN-CBM-DIO8**
Soporte al estándar ISO 11898	Sí	Sí	Sí	Sí
Soporte a la especificación CAN 2.0B	Sí (*PeliCAN*)	Sí (*BasicCAN*)	Sí (*PeliCAN*)	Sí
Controlador CAN	SJA1000 (Philips)	SJA1000 (Philips)	SJA1000 (Philips)	Implementado en el MCU (SLIO)
MCU	ATmega8515 (Atmel)	87C51 (Intel)	68331 (Motorola)	SAB80C515-L (Siemens)
Interfaz externa para configuración	Módulo DIP1	Serial mediante PC	Bus PCI	DIP y bus CAN
Interfaz de usuario	Módulo M8DES (HW)	Propietaria (SW)	CANscope (SW)	8 E/S digitales (HW)
Manejo y señalización de errores	Sí (Módulo MSE)	Sí (Sólo manejo)	No	Sí (Códificación de Leds)
Filtros de admisión configurables	Sí (HW)	Sí (SW)	Sí (SW)	No
Visualización de información del nodo	Módulos LCD, MSE, y M8DES.	LCD	No	Leds
Velocidad de transferencia de datos	Hasta 500 Kbps (configurable por HW)	Hasta 1 Mbps (no acepta configuración dinámica)	Hasta 1 Mbps (configurable por SW)	Hasta 1 Mbps (configurable por HW)
Optoacoplamiento del nodo	Sí	Sí	Sí	No
Expansión para dispositivos de E/S	Sí	Sí	No	Sí
Fabricante	Presente trabajo	[16, 17]	esd GmbH	esd GmbH

6. Conclusiones y trabajos futuros

El empleo de la metodología de desarrollo de sistemas empotrados ha sido de gran ayuda en el diseño y desarrollo del SeeCAN, sobre todo en el aspecto de las especificaciones iniciales, ya que ayudan a los desarrolladores a comunicar las ideas de diseño y a plantear los procedimientos para evaluar el correcto funcionamiento del sistema durante su ciclo de vida.

Con base en la bibliografía consultada, se logró comprender el funcionamiento del protocolo de comunicaciones CAN, como resultado se presentan los capítulos 1 y 2 del presente documento de tesis. Asimismo, se propone una clasificación de los dispositivos en el capítulo 3, el cual a su vez ha sido presentado y publicado en [20, 33].

Como parte del estudio inicial del protocolo CAN se puso en funcionamiento el sistema de entrenamiento CAN de la firma esd GmbH, y se diseñó una red con los nodos desarrollados en [16, 17]. Cabe destacar que el sistema de entrenamiento señalado, fue parte importante en el análisis de tramas CAN [19], así como en la verificación del SeeCAN.

El SeeCAN puede considerarse un sistema abierto ya que cumple con los estándares del protocolo CAN y es una herramienta destinada a la enseñanza que da pauta a nuevas aplicaciones en protocolos de comunicaciones empleados en los automóviles, debido a que el protocolo CAN es la base de otros como TTCAN y LIN (*Local Interconnect Network*).

Los nodos SeeCAN obtenidos, cumplen con el objetivo inicial de este proyecto de tesis, ya que son una herramienta de enseñanza para el protocolo de comunicaciones en cuestión.

Respecto a la metodología de desarrollo, queda pendiente la fase de actualización, y se recomienda considerar las opiniones de los usuarios finales, para integrarlas al SeeCAN.

Los resultados obtenidos han dado origen a los siguientes trabajos de investigación[18]:

- Modelado en UML de un sistema SW destinado al monitoreo del SeeCAN empleando el protocolo CANopen, con lo cual se pretende lograr un sistema educativo completo. En este caso se propone añadir la comunicación USB (*Universal Serial Bus*) al SeeCAN.
- Empleo del SeeCAN para control de una estación metereológica, donde se desarrolle un módulo de acondicionamiento de señales para diferentes sensores y un SW de control desarrollado en LabVIEW.
- Considerando al SeeCAN, implementar el protocolo de comunicaciones TTCAN, asimismo se propone desarrollar una pasarela (*gateway*) para interfazar una red LIN al SeeCAN.

[18] Los cuales se basan en el diseño del SeeCAN, y por ello la tarjeta del circuito impreso para el SeeCAN propuesta en el Anexo B, no se realizó físicamente en el presente trabajo.

Bibliografía

[1] ANSI: IEC 62026-3, Low-voltage switchgear and controlgear – Controller – device interfaces (CDIs) – Part 3; DeviceNet, Ed. 1.0, ANSI, 2000.

[2] Appel, W. & Dorner, J.: "Integration of external functions in CAN-based in-vehicle networks", 2nd international CAN Conference, iCC 1995, London (United Kingdom), 1995.

[3] Atmel: ATmega8515 AVR Microcontroller, Data Sheet Rev. 2512D-AVR-02/03, Atmel Co., 2003.

[4] Ball, S.: Embedded Microprocessor Systems; Real World Design, Newnes, 2000.

[5] Berger, A.: Embedded Systems Design. An Introduction to Processes, Tools and Techniques, CMP Books, 2002.

[6] Bosch: CAN Specification Version 2.0, Robert Bosch GmbH, September, 1991.

[7] CAN in Automation: CiA Draft Standard 102 Version 2.0, CAN Physical Layer for Industrial Applications, CiA, April, 1994.

[8] CAN in Automation, CiA Draft Recommendation DR-303-1, CANopen Cabling and Connector Pin Assignment, Version 1.0, CiA, October, 1999.

[9] CAN in Automation: DS 201-207 Version 1.1, CAN Application Layer for Industrial Applications, CiA, 1996.

[10] Domínguez, M., Poza, F., Mariño, P., Cao, F., Machado, F. & Hernández, H.: "Low Cost Educational System for CAN Communications", Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, CATE 2003, pp. 217-222, Rhodes (Greece), June 30-July 2, 2003.

[11] esd GmbH: CAN-Starter-Kit for Windows NT/2000, Rev. 1.2, Electronic System Design GmbH, August, 2000.

[12] esd GmbH: Software Manual CAN Interface, Rev. 2.6, Electronic System Design GmbH, 2000-2003.

[13] Etschberger, K.: Controller Area Network, Basics, Protocols, Chips and Applications, IXXAT Automation GmbH, 2001.

[14] Farsi, M. & Barbosa, M.: CANopen Implementation, applications to industrial networks, Research Studies Press Ltd., 2000.

[15] Fredriksson, L.: "Bluetooth in Automotive Applications", KVASER AB, 1999.

[16] Guzmán, E., Hernández, H. y Urbieta, R.: "Implementación del protocolo de bus de campo CAN en un FPGA para aplicaciones de control", Actas del XI Congreso Internacional de

Computación, CIC 2002, Vol. II, pp. 127-131, México D.F. (México), 25-29 de Noviembre, 2002.

[17] Guzmán, R. y Hernández, H.: "Implementación de una red basada en el protocolo de bus de campo CAN para aplicaciones de propósito general"", Actas de la Conferencia Internacional de Dispositivos, Circuitos y Sistemas, CIDCS 2003, Veracruz (México), 25-27 de Junio, 2003.

[18] Hank, P. & Jöhnk, E.: SJA1000 Stand-alone CAN Controller, Application Note, Philips Electronics N. V., December, 1997.

[19] Hernández, H., Chamú, C. & Arias, O.: "Automation of a Robotic Arm by Analysing the CAN Protocol", Proceeding of the XIV International Conference on Electronics, Communications, and Computers, CONIELECOMP 2004, IEEE Computer Society, pp. 2-7, Veracruz (México), February 16 – 18, 2004.

[20] Hernández, H. & Chamú, C.: "Desarrollo de un Sistema Educativo para la Enseñanza del Protocolo de Comunicaciones CAN", Actas del Congreso Internacional de Electrónica, Comunicaciones y Computadoras, CONIELECOMP 2005, IEEE Computer Society, Puebla (México), Febrero 28 – Marzo 2, 2005.

[21] ISO 7498: Open Systems Interconnect Basic Reference Model, ISO, 1989.

[22] ISO 11519-2: Road Vehicles –Low-speed serial data communication – Part 2: Low-speed Controller Area Network (CAN), ISO, 1994.

[23] ISO 11898: Road Vehicles – Interchange of digital information – Controller Area Network (CAN) for high-speed communication, ISO, 1993.

[24] ISO 15765: Road Vehicles – Diagnostics on Controller Area Network (CAN), ISO, 2004.

[25] Jöhnk, E. & Dietmayer, K.: "Determination of Bit Timing Parameters for SJA1000 CAN Controller", Philips Electronics N. V., July, 1997.

[26] Lawrenz, W.: "Worldwide Status of CAN – Present and Future", 2nd international CAN Conference, iCC 1995, London (United Kingdom), 1995.

[27] Lawrenz, W.: CAN System Engineering: From Theorical to Practical Applications, Springer-Verlag, 1997.

[28] Leen, G. & Hefferman, D.: "Expanding Automotive Electronic Systems", IEEE Computer, pp. 88-93, January, 2002.

[29] Mariño, P., Nogueira, J., Hernández, H., M., Poza, F., Machado, F. & Vazquez, F.,: "Computer Engineering Education in Network Protocols", Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA403, Las Vegas, Nevada (USA), June 23-26, 2003.

[30] Mariño, P., Nogueira, J., Hernández, H., Domínguez, M., Poza, F., Machado, F. y Vazquez, F.: "Educación en Sistemas y Circuitos Electrónicos de Comunicaciones Industriales", Actas de la Conferencia Internacional de Dispositivos, Circuitos y Sistemas, CIDCS 2003, Veracruz (México), 25-27 de Junio, 2003.

[31] Mariño, P., Hernández, H., Domínguez, M., Poza, F., Machado, F. & Vazquez, F.: "State-of-the-Art in Training Systems for a Curriculum Design in Advanced Fieldbus Technology", Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, CATE 2003, pp. 615-620, , Rhodes (Greece), June 30 - July 2, 2003.

[32] Mariño, P., Hernández, H., Domínguez, M., Poza, F., Machado, F. & Vazquez, F.: "Industrial Network Technologies and Related Information Systems", Proceedings of the International Conference on Computer, Communication and Control Technologies, CCCT403, Vol. I, pp. 64-69, Orlando, Florida (USA), July 31, August 1-2, 2003.

[33] Mariño, P., Hernández, H., Domínguez, M., Poza, F., Machado, F. y Vázquez, F.: "Una Revisión de las Herramientas para las Aplicaciones Didácticas de la Ingeniería de Buses de Campo", Actas del Seminario Anual de Automática, Electrónica Industrial e Instrumentación, SAAEI 2003, Vigo, Pontevedra (España), 10-12 de Septiembre, 2003.

[34] Mariño, P., Hernández, H., Domínguez, M., Poza, F., Machado, F. & Vazquez, F., "Development Tools for Industrial Networks Design", 10th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2003, Sharjah (United Arab Emirates), December 14-17, 2003.

[35] Marsh, D.: "Automotive Design sets RTOS cost and performance challenges", EDN Europe, pp. 32 – 42, September, 1999.

[36] Oliver, J.: "Implementing the J1850 protocol", Intel Co., 1998.

[37] OVDA: DeviceNet Specifications, Volumes I and II, Revision 2.0, Open DeviceNet Vendor Association Inc., 1998.

[38] Philips: PCA82C250 CAN Controller Interface, Product Specification, Philips Semiconductors, January, 2000.

[39] Philips: SJA1000 Stand-alone CAN Controller, Product Specification, Philips Semiconductors, January, 2000.

[40] Poza, F., Domínguez, M., Mariño, P., Machado, F. y Vázquez, F.: "Sistema de iniciación y aprendizaje de redes de comunicaciones industriales", Actas de la 2ª Conferencia Internacional en Control Automático, AUT'02, p. 97, Santiago de Cuba (Cuba), Julio 2002.

[41] Quesada, J.: "Bus CAN: estado de buses industriales y aplicaciones", Zigor, 2000.

[42] SFS: EN 50325-2, Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces. Part 2: DeviceNet, SFS, 2001.

[43] Szydlowski, C.: "Tradeoff Between Stand-alone and Integrated CAN Peripherals", 1st international CAN Conference, iCC 1994, Mainz (Germany), 1994.

[44] Thompson, M.: "The Thick and Thin of Car Cabling", IEEE Spectrum, pp. 42 – 45, February, 1996.

[45] Wong, W.: "Interface Options Abound for MCU Networking", Electronic Design, pp. 45 – 50, April, 2003.

Sitios de Internet

[URL 1] http://www.ab.com
Allen-Bradley/Rockwell Automation, 2005,
"Fabricante de sistemas y dispositivos de automatización industrial".

[URL 2] http://www.can.bosch.com
Robert Bosch GmbH, 2002 – 2004,
"Desarrollador del protocolo CAN".

[URL 3] http://www.can-cia.org
CAN in Automation (CiA), 2001 – 2005,
"Organización CiA de respaldo al protocolo de comunicaciones CAN".

[URL 4] http://www.cankingdom.org
CANkingdom International Inc., 2002 –2003,
"Organización de respaldo a la capa de aplicación CANKingdom".

[URL 5] http://www.daimlerchrysler.com
DaimlerChrysler, 1998 – 2005,
"Fabricante de automóviles".

[URL 6] http://www.fh-wolfenbuettel.de/fb/i/institute/ifvs/CAN
Institut für Verteilte Systeme CAN Cube, 2002 – 2004,
"Página del Dr. Wolfhard Lawrenz, miembro del grupo de desarrollo del protocolo CAN".

[URL 7] http://www.gmodelo.com.mx
Grupo Modelo S.A. de C.V., 2003,
"Empresa cervecera".

[URL 8] http://www.hansenreport.com
The Hansen Report on Automotive Electronics, 2002 – 2004,
"Revista dedicada a la divulgación de la electrónica del automóvil".

[URL 9] http://www.hella.com
Hella KG Hueck & Co., 2004,
"Fabricante de dispositivos de luces automotrices".

[URL 10] http://www.intel.com
Intel Corp., 2005,
"Fabricante de semiconductores".

[URL 11] http://www.ixxat.de
IXXAT Automation GmbH, 2004,
"Fabricante de sistemas y dispositivos CAN, CANopen, DeviceNet".

[URL 12] http://www.kone.com
Kone Corporation, 2002 – 2003,
"Fabricante de elevadores".

[URL 13] http://www.kvaser.com
Kvaser AB, 2005,
"Fabricante de sistemas y dispositivos CAN para investigación y enseñanza".

[URL 14] http://www.lafarge.com
Lafarge Group, 2005,
"Fabricante de materiales para construcción".

[URL 15] http://www.lin-subbus.org,
LIN-subbus Consortium, 2002 – 2004,
"Organización de respaldo al protocolo de comunicaciones LIN".

[URL 16] http://www.mbegypt.mercedes-benz.com/english/technology1.htm
Mercedes-Benz, Egypt, 2002 – 2003,
"Página de divulgación tecnológica de la firma Mercedes-Benz".

[URL 17] http://www.mercedes-benz.com
DaimlerChrysler, 2004,
"Fabricante de automóviles".

[URL 18] http://www.osek-vdx.org
OSEK-VDX, 2004,
"Organización de respaldo a la especificación para sistemas abiertos OSEK-VDX".

[URL 19] http://www.semiconductors.philips.com
Philips Electronics N.V., 2003 – 2005,
"Fabricante de semiconductores".

[URL 20] http://www.siemensvdo.com
Siemens VDO Automotive AG, 2005,
“Fabricante de productos electrónicos, eléctricos y mecatrónicos para automóviles”.

[URL 21] http://www.sulzer.com
Sulzer Corp., Switzerland, 2004,
“Fabricante de máquinas textiles”.

[URL 22] http://www.toyota.com
Toyota Motors Sales, 2004 – 2005,
“Fabricante de automóviles”.

[URL 23] http://www.vag.-com-espanol.com
VAG-COM: Audi A4 (Plataforma B6/8E), 2003
“Desarrollador de software de diagnóstico para automóviles VW, Audi y Seat”.

[URL 24] http://www.yazaki-na.com
Yazaki Corp. 2003 – 2004,
“Fabricante de sistemas de suministro de energía para automóviles”.

Acrónimos

ABS (*Anti-lock Break System*, Sistema de Frenos Antibloqueo)

ACC (*Adaptive Cruise Control*, Control de Crucero Adaptable)

AMIC (*Automotive Multimedia Interface Consortium*)

API (*Application Program Interface*, Interfaz de Programación de Aplicación Específica)

AS-I (*Actuator-Sensor Interface*, Interfaz Sensor Actuador)

ASC (*Automatic Slip Control*, Control Automático de Deslizamiento)

ASIC (*Application Specific Integrated Circuit*, Circuito Integrado de Aplicación Específica)

ASR (*Anti Slip Regulator*, Regulador Antideslizamiento)

CAL (*CAN Application Layer*, Capa de Aplicación CAN)

CAN (*Controller Area Network*)

CiA (*CAN in Automation*)

CIP (*Control and Information Protocol*, Protocolo de Control e Información)

CMS (*CAN Based Message Specification*, Especificación de Mensajes Basada en CAN)

CPU (*Central Processing Unit*, Unidad Central de Proceso)

CRC (*Cyclic redundant Check*, Verificación de Redundancia Cíclica)

CSMA/CD+AMP (*Carrier Sense Multiple Access with Collision Detection and Arbitration Message Priority*, Acceso Múltiple por Detección de Portadora, con Detección de Colisiones y Arbitraje por Prioridad de Mensajes)

DBT (*Identifier Distributor*, Distribuidor de Identificadores)

DIL (*Dual In Line*)

DLL (*Data Link Layer*, Capa de Enlace de Datos)

DSP (*Digital Signal Processing*), Procesamiento Digital de Señales)

E/S (*I/O*, Entradas/Salidas)

ECU (*Electronic Control Unit*, Unidad de Control Electrónico)

EDS (*Electronic Data Sheet*, Hoja de Datos Electrónica)

EFF (*Extended Frame Format*, Formato de Trama Extendida)

EML (*Error Managemente Logic*, Lógica de Gestión de Errores)

ESP (*Electronic Stability Program*, Programa Electrónico de Estabilidad)

FB (*Fieldbus*, Bus de Campo)

FMS (*Food Machinery Sales*, Sistemas de Empaquetamiento de Comida)

HLP (*High Layer Protocol*, Protocolo de Capa Alta)

HVAC (*Heating, Ventilating and Air Conditioned*, Sistema de Calefacción, Ventilación y Aire Acondicionado)

iCC (*internacional CAN Conference*, Conferencia Internacional de CAN)

ICE (*In Circuit Emulator*)

IDB (*Intelligent Data Bus*, Bus Inteligente de Datos)

IDE (*Integrated Development Enviroment*, Entorno de Desarrollo Integrado)

IEC (Instituto de Electrónica y Computación)

ISO (*International Organization for Standarization*, Organización Internacional para la Normalización)

ISR (*Interrupt Service Routine*, Rutina de Servicio a Interrupción)

LCD (*Liquid Crystal Display*, Visualizador de Cristal Líquido)

LDSU (*Link Service Data Units*, Unidades de Datos del Servicio de Enlace)

LIN (*Local Interconnect Network*)

LLC (*Logic Link Control*, Control de Enlace Lógico)

LME (*Layer Management Entity*, Entidad de Gestión de Capa)

LMT (*Layer Management*, Gestión de Capa)

MAC (*Medium Access Control*, Control de Acceso al Medio)

MAC-LME (Entidad de Gestión de Capa MAC)

MCNet (*Mobile Communication Network*, Red de Comunicación Móvil)

MCU (*Microcontroller*, Microcontrolador)

MDI (*Medium dependent Interface*, Interfaz Dependiente del Medio)

MMI (*Man-Machine Interface*, Interfaz Hombre Máquina)

MOST (*Media Oriented System Transport*)

MSN (*The Modules are to Serves the Network*)

MVB (*Multiple Vehicle Bus*)

NMT (*Network Management*, Gestión de la Red)

NRZ (*Non Return to Zero*, No Retorno a Cero)

OD (*Object Dictionary*, Diccionario de Objetos)

OOK (*On-Off Keying*)

OS (*Operating System*, Sistema Operativo)

OSEK (*Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug*)

OSI (*Open Systems Interconnection*, Interconexión de Sistemas Abiertos)

PC (*Personal Computer*, Computadora Personal)

PDA (*Personal Digital Assistant*, Asistente Digital Personal)

PDO (*Process Data Object*, Objetos de Proceso)

PLC (*Programmable Logic Controller*, Controlador Lógico Programable)

PLD (*Programmable Logic Device*, Dispositivo Lógico Programable)

PLL (*Phase Locked Loop*)
PLS (*Physical Signalling*, Subcapa de Señalización Física)
PLS-LME (Gestión de Fallos del Bus)
PMA (*Physical Medium Attachment*, Conexión al Medio Físico)
PMS (*Philips Message Specification*, Especificación de Mensajes de Philips)
PROFIBUS-PA (*PROFIBUS Profile for Process Automation*)
PSC (*Process Segment Controllers*, Controladores de Proceso Segmentado)
RAM (*Random Access Memory*, Memoria de Acceso Aleatorio)
RJW (*Resynchronization Jump Width*)
RTOS (*Real Time Operating System*, Sistema Operativo en Tiempo Real)
RTR (*Remote Transmission Request*, Petición de Transmisión Remota)
SAE (*Society of Automotive Engineers*, Sociedad de Ingenieros Automotrices)
SBS (*Sensortronic Brake System*, Sistema de Frenado Sensortronic)
SDO (*Service Data Objects*, Objetos de Servicio)
SDS (*Smart Distributed System*)
SeeCAN (Sistema Educativo para la Enseñanza del Protocolo de Comunicaciones CAN)
SFF (*Standard Frame Format*, Formato de Trama Estándar)
SJW (*Synchronization Jump Width*)
SLIO (*Serial Linked I/O*, Controlador de E/S)
SmartLCC (*Smart Load Control Center*, Centro de Control de Carga Inteligente)
SMM (*Ship Machinery and Marine Technology*)
STZP (*Steinbeis Transfer Center for Process Automation*)
TDM (*Time-Division Multiplexing*, Multiplexación por División de Tiempo)
TTCAN (*Time Trigger CAN*, Protocolo CAN Accionado por Tiempo)
TTL (*Transistor Transistor Logic*, Lógica de Transistor a Transistor)
USB (*Universal Serial Bus*)
UTM (Universidad Tecnológica de la Mixteca)
VAN (*Vehicle Area Network*)
WCS (*Weight Classification System*, Sistema de Clasificación de Peso)
WTB (*Wide Train Bus*)

Anexo A. Manual de usuario del SeeCAN

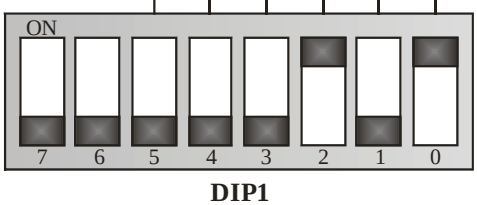
A.1. Configuración de parámetros de comunicación

La configuración de los parámetros de comunicación respecto a la red SeeCAN se realiza antes de suministrar energía al nodo, para ello se utilizan los interruptores del 0 al 5 del DIP1 (Figura A.1), como se describe a continuación:

- *Velocidad de transferencia de datos*: se activan (*on*) o desactivan (*off*) los interruptores 0, 1 y 2 para configurar la velocidad de transferencia de datos, dicha velocidad debe ser la misma para todos los nodos de la red SeeCAN.
- *Filtro de admisión*: se utilizan los interruptores 3 y 4 para determinar el rango de identificadores que cada nodo puede admitir, los cuales son: Todos (00), Rango 1 (01, 00XXX…), Rango 2 (10, 0101010XXX…) y Rango 3 (11, 11XXX00XXX…).
- *Tipo de transmisión*: se utiliza el interruptor 5 para seleccionar entre transmisión automática o transmisión manual.

Después de configurar los parámetros de comunicación, se suministra energía con una fuente de voltaje de 5 VCD y el nodo se inicializa con los valores configurados.

Tipo de transmisión		**Filtro de admisión**			**Velocidad de transferencia de datos**			
Manual	0	Todos	0	0	0	0	0	1 Mbps
Automática	1	Rango 1	0	1	0	0	1	500 Kbps
		Rango 2	1	0	0	1	0	250 Kbps
		Rango 3	1	1	0	1	1	125 Kbps
					1	0	0	100 Kbps
					1	0	1	20 Kbps



DIP1

Figura A.1. Diagrama para configuración de los parámetros del nodo SeeCAN mediante el DIP1.

A.2. Transmisión de tramas

La transmisión de tramas se puede realizar de dos formas (Figura A.2):

- *Transmisión automática*: el nodo transmite tramas con un identificador y campo de datos fijos, definidos en el programa del MCU y que el usuario no puede modificar. El administrador del nodo controla el inicio de la transmisión una vez que se ha suministrado energía al nodo.
- *Transmisión manual*: el nodo transmite tramas cada vez que el usuario lo indica mediante el botón TxM, para ello el usuario puede determinar lo siguiente:
  - *7 bits del identificador*: se utilizan los interruptores del 0 al 6 del DIP1 para configurar los 7 bits más significativos del identificador de la trama.
  - *Configurar tipo de trama*: se utiliza el interruptor 7 del DIP1 para determinar si la trama es de tipo estándar o extendida.
  - *Último octeto del campo de datos*: se utilizan los interruptores del 0 al 7 del módulo M8DES para que el usuario active o desactive las ocho entradas digitales.

Después de configurar los valores anteriores, se presiona el botón TxM para iniciar la transmisión de la trama CAN. Cada que se requiera transmitir un trama, se pueden modificar dichos parámetros.

A.3. Recepción de mensajes

La recepción de mensajes se realiza de forma automática y se puede visualizar en las salidas digitales (leds) del módulo M8DES (Figura A.3), y en los datos que se despliegan en el LCD.

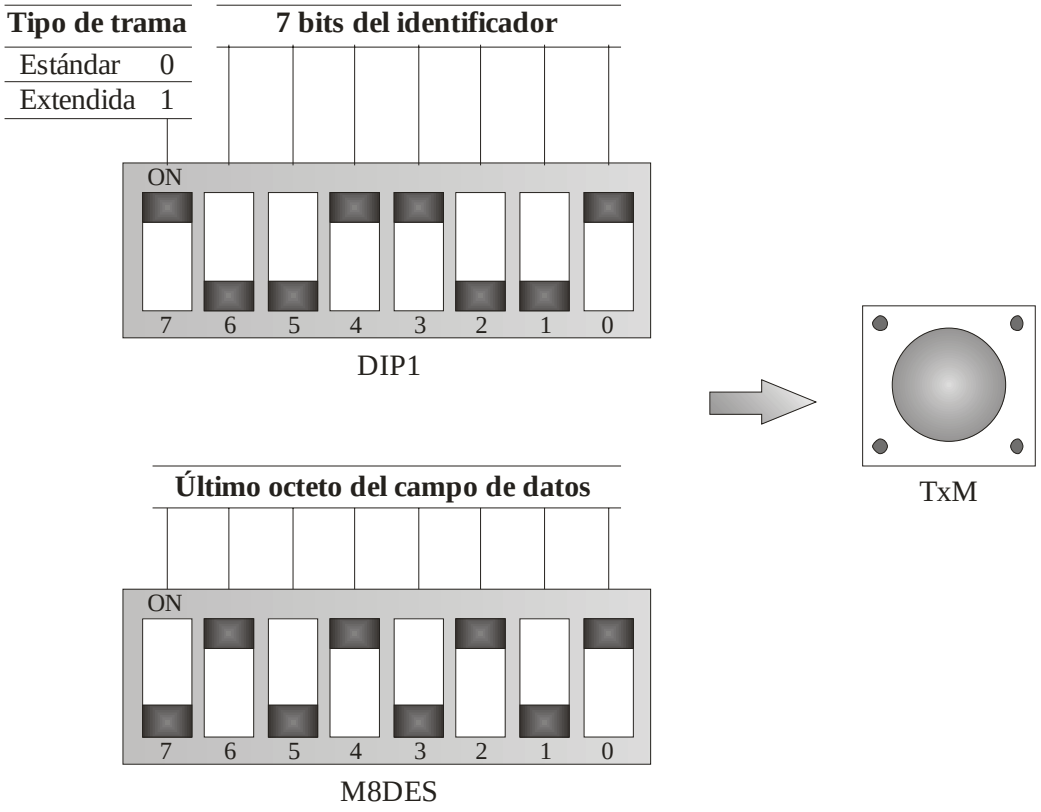


Figura A.2. Configuración para transmisión de tramas mediante los módulos DIP1 y M8DES.

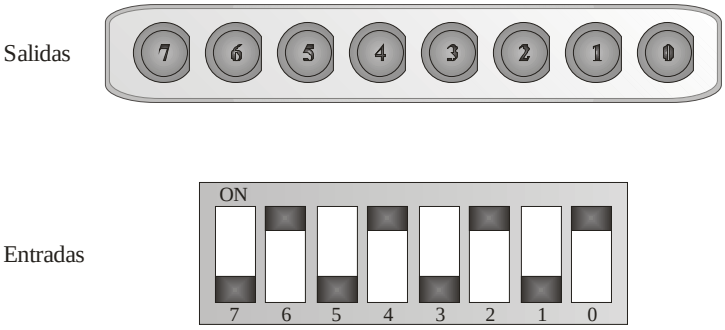


Figura A.3. Módulo M8DES para control de las entradas y salidas digitales del SeeCAN.

A.4. Señalización de errores

La señalización de errores del nodo se realiza en los siguientes módulos:

- *MSE*: indica el estado de error del nodo (Figura A.4).
- *LCD*: indica la dirección en la que ocurre un error (de transmisión o de recepción), y el tipo de error (forma, relleno, bit, otro) (Figura A.5).

EA: Estado de error activo
EP: Estado de error pasivo
D: Nodo desconectado

Figura A.4. Módulo MSE para señalización de estados de error del SeeCAN.

Figura A.5. Señalización de errores mediante el LCD.

Anexo B. Diagramas esquemático y PCB del SeeCAN

La Figura B.1 muestra el diagrama esquemático del SeeCAN diseñado en la herramienta Orcad, y las Figuras B.2-B.4 muestran el diseño del circuito impreso para el nodo SeeCAN.

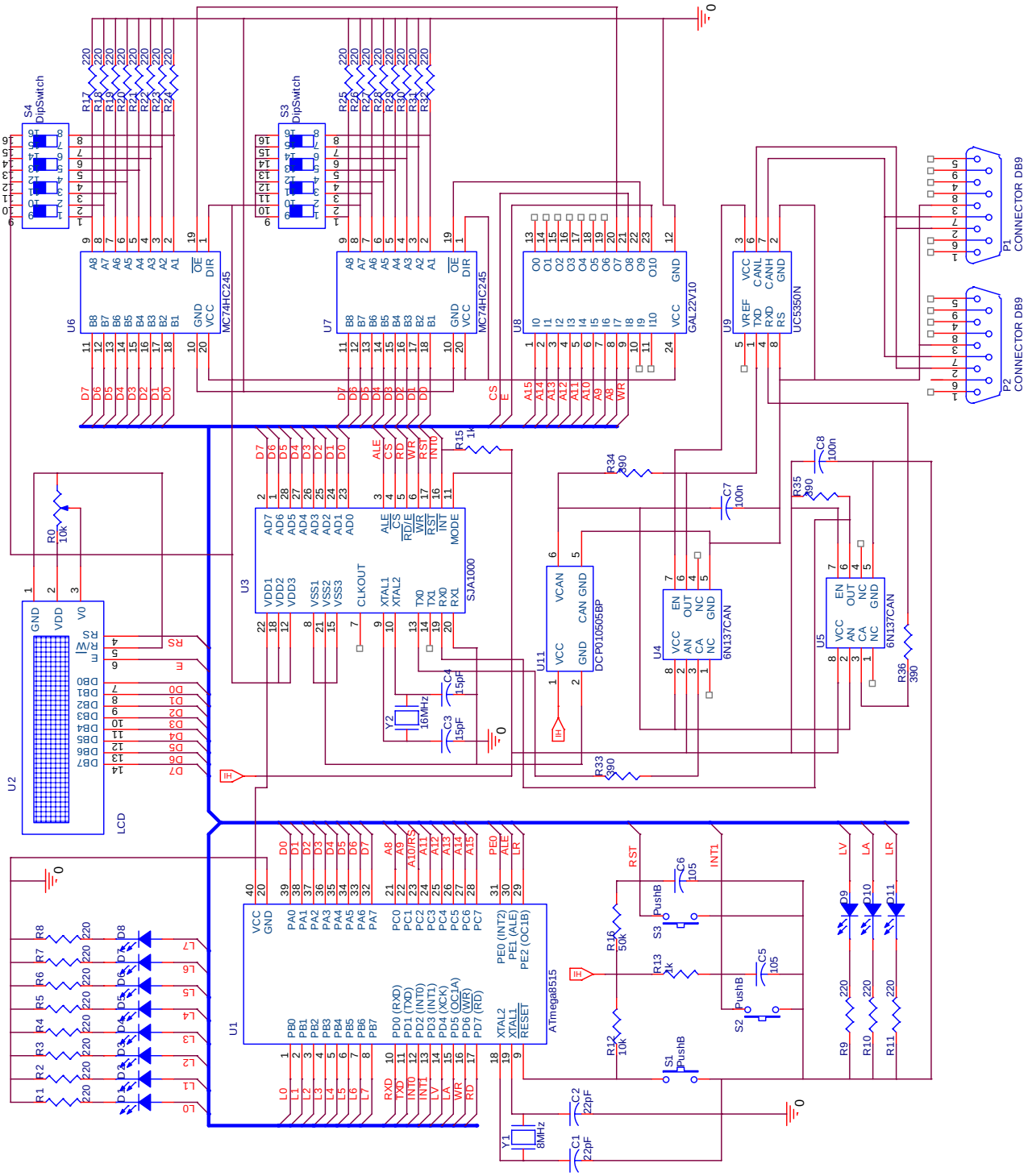


Figura B.1. Diagrama esquemático del SeeCAN.

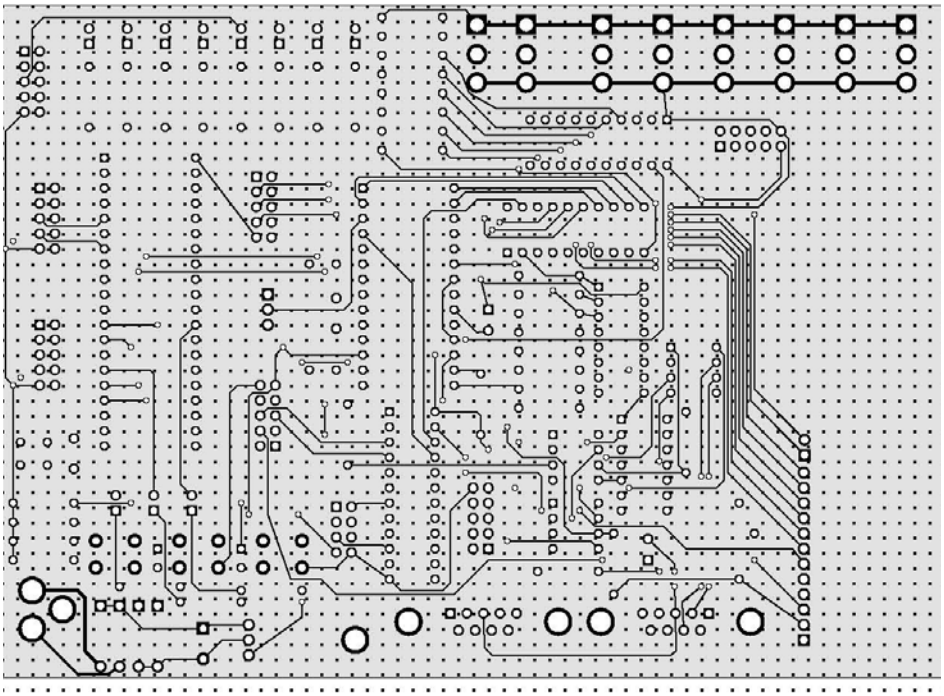


Figura B.2. Diagrama PCB de la parte de componentes del SeeCAN (*Top layer*).

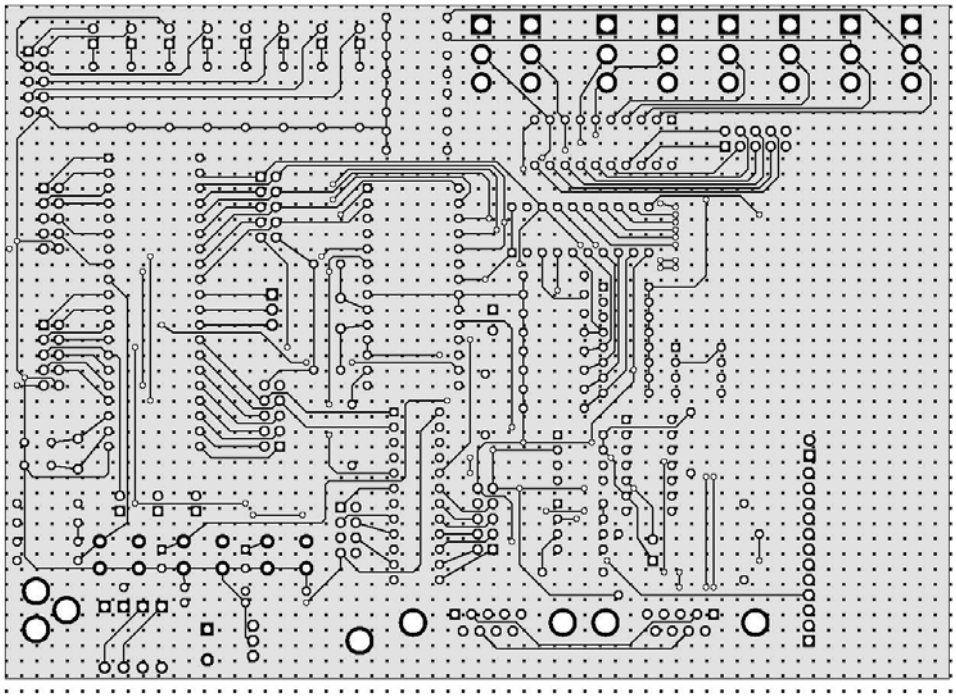


Figura B.3. Diagrama PCB de la parte de soldadura del SeeCAN (*Bottom layer*).

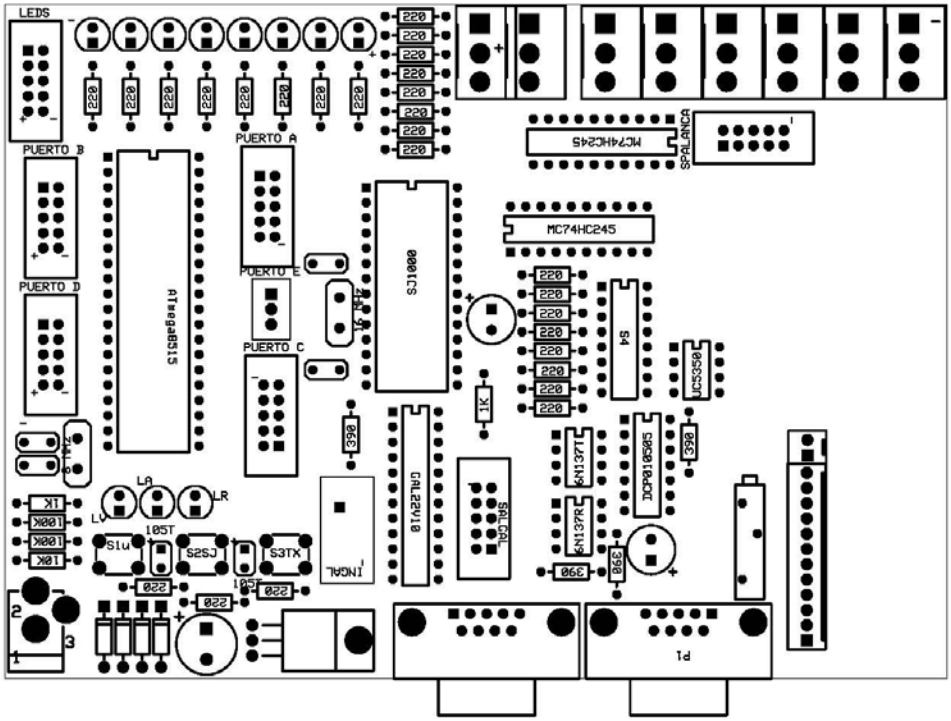


Figura B.4. Localización de los componentes del SeeCAN (*Silkscreen & pads*).