Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

GRAU D'ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

Diseño, implementación y evaluación cuantitativa del rendimiento de un prototipo mejorado de una nueva red de comunicaciones para sistemas de control distribuidos llamada Highly Discriminating Monitoring Star for CAN (HDMS-CAN)

DANIEL BUJOSA

Tutor
Manuel Barranco

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 22 de septiembre de 2017

ÍNDICE GENERAL

Índice general i

Acrónimos iii

Resumen v

1 Introducción 1
- 1.1 Antecedentes y motivación . . . 1
- 1.2 Objetivos específicos . . . 3
- 1.3 Tareas realizadas . . . 4
- 1.4 Tareas no realizadas . . . 5
- 1.5 Estructura del documento . . . 5

2 Fundamentos y trabajo previo 7
- 2.1 Control Area Network (CAN) . . . 7
  - 2.1.1 Capa física . . . 8
  - 2.1.2 Capa de enlace . . . 13
- 2.2 CANcentrate . . . 18
- 2.3 Highly Discriminating Monitoring Star for CAN, HDMS-CAN . . . 20
  - 2.3.1 Arquitectura general . . . 20
  - 2.3.2 Enmascaramiento . . . 20
  - 2.3.3 Estructura del tiempo de bit . . . 25
  - 2.3.4 Estructura interna del Hub Core . . . 26

3 Caracterización de HDMS-CAN 29
- 3.1 Dimensionamiento de los segmentos del tiempo de bit . . . 30
- 3.2 Ecuaciones para dimensionar los segmentos del tiempo de bit . . . 35
- 3.3 Restricciones topológicas de HDMS-CAN . . . 36
- 3.4 Cuantificación teórica del radio y el bit rate máximos alcanzables en HDMS-CAN . . . 37

4 Prototipado 41
- 4.1 Red de pruebas . . . 42
  - 4.1.1 Arquitectura de la red de pruebas . . . 42
  - 4.1.2 Hardware de la red de pruebas . . . 43
  - 4.1.3 Software de los nodos . . . 45
  - 4.1.4 Estrategia de debugging . . . 46

4.2 Fallos de diseño identificados y soluciones propuestas 48
4.2.1 Enmascaramiento de las tramas de error 48
4.2.2 Desincronización del hub en los ACKs 50
4.3 Problemas de implementación y soluciones 50
4.3.1 Tamaño de los contadores 51
4.3.2 Falta de hard-synchronization en el SyncSeg 51
4.3.3 Características eléctricas de los transceivers 51

5 Pruebas de rendimiento **53**
5.1 Extensión de la red de pruebas 53
5.2 Pruebas y resultados 56
5.2.1 Cobertura de HDMS-CAN 56
5.2.2 Restricciones topológicas de HDMS-CAN 59
5.2.3 Consumo de los transceivers de HDMS-CAN 59

6 Conclusiones **61**
6.1 Compendio 61
6.2 Trabajo futuro 62
6.3 Consideraciones sobre el proceso de aprendizaje 63

A Anexos **65**
A.1 Tabla de resultados experimentales 65

Bibliografía **69**

ACRÓNIMOS

TFG Trabajo Final de Grado

HDMS-CAN Highly Discriminating Monitoring Star for CAN

CAN Control Area Network

VHDL VHSIC Hardware Description Language

OSI Open System Interconnection

ISO International Organization for Standardization

TQ Time Quantum

SOF Start Of Frame

RTR Remote Transmission Request

IDE IDentifier Extension

DLC Data Length Code

CRC Cyclic Redundancy Code

EOF End Of Frame

IFS InterFrame Space

NZR Non Return to Zero

ECU Electrical Control Units

FPGA Field Programmable Gate Array

USB Universal Serial Bus

TEC Transmission Error Counter

REC Reception Error Counter

UIB Universidad de las Islas Baleares

SJW Synchronization Jump Width

RESUMEN

[1] Los buses de campo son una familia de protocolos diseñados para soportar las comunicaciones en sistemas de control distribuido, principalmente en el ámbito industrial. Uno de los buses de campo más utilizados hoy en día es el *Controller Area Network* (CAN); debido principalmente a que proporciona robustez y comunicación con ciertas garantías de tiempo real a muy bajo coste.

Sin embargo, al igual que otros buses de campo, CAN presenta una serie de limitaciones causadas por su topología de bus. Es decir, presenta limitaciones que se deben a que todos los nodos (sensores, actuadores, controladores, etc.) comparten una única línea física de comunicación para transmitir y recibir. Por ejemplo, un fallo en el bus puede dejarlo inservible de tal forma que ningún nodo se pueda comunicar. Ésta y otras limitaciones han hecho que los protocolos basados en una topología de bus no se consideren aptos para sistemas de control distribuido críticos; ya que dichos sistemas deben presentar una fiabilidad muy elevada.

En el caso particular de CAN, además, la forma en que los nodos utilizan el bus para transmitir y recibir presenta inconvenientes adicionales. Concretamente, cuando más de un nodo transmite en el bus simultáneamente, las contribuciones de éstos se acoplan irreversiblemente en el medio; lo cuál provoca que sea imposible discriminar cuál es la contribución individual (qué es lo que transmite) cada uno de ellos. Esta característica dificulta, limita o imposibilita la implementación de dispositivos, mecanismos, herramientas o redes que necesiten discernir esas contribuciones. Por ejemplo, dificulta o imposibilita la implementación de mecanismos para detectar y contener errores en la transmisión de los nodos y así evitar que dichos errores se propaguen; para realizar logs y diagnósticos basados en conocer con precisión qué es lo que transmite cada nodo en escenarios concretos de comunicación; etc.

A fin de solventar estos problemas en CAN, algunos investigadores del grupo en donde se ha realizado este TFG han inventado y patentado recientemente una topología en estrella denominada *Highly Discriminating Monitoring Star for CAN* (HDMS-CAN). En esta topología cada nodo CAN se conecta, de forma transparente para el nodo, a un concentrador (*hub*) activo que permite discriminar con una alta precisión temporal la contribución de cada nodo.

Estos investigadores implementaron un primer prototipo que permitió demostrar la viabilidad de HDMS-CAN como concepto. Sin embargo, no se realizó una depuración exhaustiva para hallar fallos ni en el diseño de HDMS-CAN, ni en la implementación de dicho prototipo. Además, tampoco se caracterizaron diferentes aspectos relacionados con la configuración de los mecanismos internos del hub, con las restricciones topoló-

[1] El contenido de este trabajo ha sido objeto de una petición de patente, con número de solicitud P201531737, y debe ser considerado como confidencial hasta la publicación de dicha patente

gicas, y con el rendimiento (radio, velocidad de transmisión y consumo eléctrico) de HDMS-CAN.

El objetivo de este TFG es depurar los posibles fallos presentes en el diseño original de HDMS-CAN; implementar un nuevo prototipo de HDMS-CAN libre de fallos; así como caracterizar, desde el punto de vista teórico y práctico, los aspectos que se acaban de mencionar sobre la configuración de los mecanismos internos del hub, las restricciones topológicas, y el rendimiento. La consecución de este objetivo es relevante para evaluar la posibilidad de explotación y/o comercialización de la patente de HDMS-CAN.

CAPÍTULO 1

INTRODUCCIÓN

1.1 Antecedentes y motivación

[1] Los buses de campo constituyen una familia de protocolos diseñados para soportar las comunicaciones de sistemas de control distribuido, principalmente en al ámbito industrial. Uno de los buses de campo más utilizados hoy en día es Control Area Network (CAN). CAN se propuso en los años 80 inicialmente para las comunicaciones dentro de los vehículos, pero rápidamente su uso se extendió a muchos otros ámbitos como, por ejemplo, los electrodomésticos, la aviación, la robótica, la automatización industrial, la domótica, etc. El éxito de CAN se debe principalmente a que proporciona robustez y comunicación con ciertas garantías de tiempo real a muy bajo coste.

Como la mayoría de buses de campo, el medio de transmisión en CAN se basa en una topología de tipo bus. Es decir, el medio en CAN es una única línea física de comunicación a la que todos y cada uno de los nodos (sensores, actuadores, controladores, etc.) están conectados. Esta topología presenta varias ventajas, como por ejemplo un coste reducido. Sin embargo, también presenta varios inconvenientes, como por ejemplo que constituye un punto único de avería (el bus es un punto tal que si falla, todo el sistema de comunicaciones se avería).

En el caso particular de CAN, además, existen otras desventajas que se derivan de la forma en que los nodos transmiten en el bus. Como se explicará más adelante, en CAN los nodos están sincronizados bit a bit (los nodos tienen una visión casi simultánea de los bits en el canal), de tal forma que el valor lógico de cada bit que se observa en el bus es el resultado de aplicar una función AND lógica sobre la contribución que cada nodo transmite para dicho bit. Dicho de otra forma, el bus acopla (mediante una función AND cableada) bit a bit la contribución de todos y cada uno de los nodos (acopla bit a bit los *streams* de datos). Este acoplamiento bit a bit da lugar a que en el bus se observe una señal resultante tal que es imposible distinguir en ella cuál es la

[1] El contenido de este trabajo ha sido objeto de una petición de patente, con número de solicitud P201531737, y debe ser considerado como confidencial hasta la publicación de dicha patente

contribución individual de cada nodo.

El hecho de que, en la señal resultante que se transmite por el bus CAN, sea imposible distinguir las contribuciones de todos y cada uno de los nodos, dificulta, limita o imposibilita la implementación de dispositivos, mecanismos, herramientas o redes que necesiten discernir esas contribuciones. Por ejemplo, en un bus CAN es muy difícil (y a veces imposible) tanto determinar qué nodos están averiados, como aislarlos para que no perjudiquen la comunicación entre los otros nodos.

Algunos miembros del grupo de investigación en donde se ha realizado este Trabajo Final de Grado (TFG) inventaron, patentaron y desarrollaron hace unos años dos soluciones, llamadas CANcentrate y ReCANcentrate [1], para superar las limitaciones que la topología de bus presenta en CAN.

CANcentrate es una topología de estrella simple, en la que cada nodo está conectado a un único concentrador activo (hub activo) mediante dos enlaces (*links*): un enlace de subida (*uplink*) y otro de bajada (*downlink*). El hub de CANcentrate acopla bit a bit las contribuciones que recibe a través de los uplinks y envía el resultado de dicho acoplamiento a través de los downlinks. Por una parte, la separación física entre la contribución de cada nodo (que el hub recibe a través de cada uplink) y la señal resultante (que se transmite por los donwlinks) le permite al hub discriminar la contribución individual de cada nodo bit a bit. Por otra parte, el hub realiza el acoplamiento y el envío de la señal resultante en una fracción de la duración de un bit, por lo que la existencia del hub es *transparente* para los nodos. Se han desarrollado varios prototipos de CANcentrate en los que, por ejemplo, su hub utiliza su capacidad de discernimiento para diagnosticar cuándo un nodo cualquiera está en fallo y desconectarlo en tal caso.

ReCANcentrate, por su parte, consiste en una topología de estrella replicada en la que los nodos se comunican a través de varios hubs activos similares al hub de CANcentrate. Además de presentar las mismas ventajas de CANcentrate que se derivan de la capacidad de discriminar de forma transparente la contribución de cada nodo bit a bit, los prototipos de ReCANcentrate permiten a los nodos tolerar el fallo de enlaces y hubs.

Sin embargo, tanto CANcentrate como ReCANcentrate presentan un inconveniente común. A pesar de que son transparentes y por tanto compatibles con cualquier software CAN existente, y a pesar de que se pueden implementar utilizando componentes hardware CAN comerciales, ambas topologías exigen que cada nodo utilice dos enlaces separados (uplink y downlink) para conectarse al hub en CANcentrate (y a cada hub en ReCANcentrate). Este hecho conlleva una serie de desventajas. Las más evidentes son un mayor coste derivado de la necesidad de utilizar más hardware, p.e. conectores y cables; una mayor dificultad para enrutar e instalar el cableado; y problemas a la hora de ubicar el hub dentro del sistema, ya que el hub aglutina un mayor número de conexiones y, por tanto, puede ocupar más espacio del que normalmente se encuentra disponible en algunos tipos de aplicaciones, p.e. en la automoción. Por otra parte, tanto en CANcentrate como en ReCANcentrate, cada nodo necesita dos *transceivers*[2] para conectarse a cada hub: uno para conectarse al uplink y otro para conectarse al downlink. Este hecho puede provocar problemas de incompatibilidad con algunos

[2] Un transceiver es un dispositivo que traduce la señal lógica que un nodo transmite a una señal que se pueda propagar a través del medio, y que traduce la señal del medio a una señal lógica que el nodo pueda recibir

nodos CAN comerciales (como las Electrical Control Units (ECU) en los vehículos) cuyo hardware no se puede modificar y que sólo disponen de un *transceiver*.

Para solventar estos problemas de CANcentrate y ReCANcentrate varios de los miembros del grupo de investigación en donde se ha realizado este TFG han propuesto y patentado recientemente una nueva solución denominada Highly Discriminating Monitoring Star for CAN (HDMS-CAN).

HDMS-CAN es una nueva topología de estrella simple con un hub activo que permite discriminar bit a bit la contribución de cada nodo, pero en la que cada nodo se conecta a dicho hub utilizando un único enlace (*link*). Es decir, HDMS-CAN permite discriminar bit a bit la contribución de cada nodo sin necesidad de separar físicamente, en cada nodo, la transmisión y la recepción. La manera en que HDMS-CAN consigue discriminar las contribuciones de los nodos se explicará más adelante.

A fin de poder patentar HDMS-CAN, sus inventores desarrollaron un primer prototipo que sirvió como prueba de concepto (*proof-of-concept*) para demostrar que HDMS-CAN era capaz de discriminar la contribución de cada nodo bit a bit. En este sentido, ese primer prototipo no permitió determinar ciertos aspectos que son relevantes desde el punto de vista de la posibilidad de explotación y/o comercialización de la patente.

En primer lugar, ese prototipo utilizó solamente links cortos (de unos 20 cm) y un único *bit rate*. Como se explicará más adelante, en CAN existe una proporción inversa entre la longitud máxima del bus y el *bit rate*. Por tanto, es necesario determinar cuál es la cobertura de HDMS-CAN en términos de su radio o su diámetro máximos en función del *bit rate*. Es más, como también se explicará, los mecanismos internos del hub de HDMS-CAN, que permiten discriminar las contribuciones de los nodos, pueden imponer ciertas restricciones topológicas; es decir, pueden imponer ciertas restricciones en la distancia a la que cada nodo se puede ubicar respecto al hub en relación al resto de nodos.

En segundo lugar, no se realizaron pruebas exhaustivas con ese prototipo a fin de depurar posibles fallos de diseño y/o implementación.

En tercer y último lugar, tampoco se realizaron pruebas de rendimiento para corroborar ni la cobertura de HDMS-CAN, ni el consumo de corriente de sus *transceivers*. Este último punto es importante, ya que en ese prototipo se detectó un consumo elevado por parte de los *transceivers*.

Así pues, el objetivo general del presente TFG es aclarar estos aspectos que son relevantes para poder explotar y/o comercializar HDMS-CAN. En el siguiente apartado se expondrán cuáles son los objetivos específicos en que se divide este objetivo general.

1.2 Objetivos específicos

El objetivo general del presente TFG se divide en los siguientes objetivos específicos:

- Caracterizar, tanto desde el punto de vista teórico como desde el punto de vista experimental, la cobertura de HDMS-CAN en términos de su radio o su diámetro máximos en función del *bit rate*.
- Caracterizar, tanto desde el punto de vista teórico como desde el punto de vista experimental, las restricciones topológicas de HDMS-CAN.

- Detectar y solucionar posibles fallos de diseño en la propuesta original de HDMS-CAN.
- Detectar y solucionar posibles fallos de implementación en el prototipo que sirvió como prueba de concepto de HDMS-CAN.
- Implementar un nuevo prototipo de HDMS-CAN, basado en el original, que sea completamente operativo; es decir, un nuevo prototipo que cumpla los siguientes requisitos:
  - Que no presente los fallos de diseño y/o implementación presentes en el prototipo original.
  - Que funcione correctamente para los *bit rates* más representativos de CAN.
  - Que permita realizar pruebas de rendimiento para poder caracterizar experimentalmente tanto la cobertura y las restricciones topológicas de HDMS-CAN, como el consumo de corriente de sus *transceivers*.

1.3 Tareas realizadas

A fin de conseguir los objetivos específicos mencionados en la sección anterior, en el presente TFG se realizaron las siguientes tareas:

- Se estudiaron los aspectos técnicos más relevantes del protocolo CAN.
- Se estudiaron todos los detalles técnicos de HDMS-CAN especificados en la patente correspondiente.
- Se caracterizó desde el punto de vista teórico la cobertura, el *bit rate* (en función de la cobertura), y las restricciones topológicas de HDMS-CAN. Como se explicará más adelante, en CAN (y por tanto también en HDMS-CAN) cada tiempo de bit se divide en varios segmentos, en los que los nodos realizan varias acciones para mantenerse sincronizados a nivel de bit entre sí, así como para poder transmitir y recibir convenientemente. El hub de HDMS-CAN divide el tiempo de bit en varios segmentos de una forma ligeramente diferente a la que lo hacen los nodos; a fin de llevar a cabo una serie de acciones que le permitan discriminar la contribución de cada nodo, acoplar convenientemente dichas contribuciones, y transmitir/recibir como si se tratase de cualquier otro nodo CAN. En este sentido, por una parte, la forma en que el hub dimensiona los diferentes segmentos del tiempo de bit es esencial para que el hub pueda discriminar correctamente la contribución de los nodos; y para que los nodos y el hub puedan comunicarse sin errores. Por otra parte, la forma de dimensionar los segmentos del tiempo de bit condiciona la cobertura y las restricciones topológicas de HDMS-CAN. Por tanto, en el presente TFG se ha propuesto cómo dimensionar los segmentos del tiempo de bit (desde el punto de vista del hub) para conseguir tanto un funcionamiento correcto, como para conseguir un rendimiento (desde el punto de vista de la cobertura, el *bit rate* y las restricciones topológicas) que sea útil a la hora de explotar HDMS-CAN. Es más, dicha propuesta se ha formalizado en forma de expresiones para calcular la duración de cada uno de los segmentos del tiempo de bit.

- Se estudiaron los aspectos técnicos de la implementación del prototipo original de HDMS-CAN que se utilizó como prueba de concepto.
- Se implementó un nuevo prototipo completamente operativo de HDMS-CAN, basado en el prototipo original. Para ello, por una parte, se realizaron diferentes pruebas que permitieron identificar fallos presentes tanto en el diseño de HDMS-CAN, como en la implementación del prototipo original. Por otra parte, se propusieron soluciones para estos fallos de diseño e implementación, y se implementaron y probaron con éxito dichas soluciones en el nuevo prototipo.
- Finalmente se realizaron una serie de experimentos a fin de caracterizar el rendimiento del nuevo prototipo de HDMS-CAN. En primer lugar se caracterizó experimentalmente su cobertura y se comparó con las previsiones teóricas. En segundo lugar se corroboraron experimentalmente las restricciones topológicas de HDMS-CAN. En último lugar se midió el consumo de corriente de los *transceivers* del prototipo.

1.4 Tareas no realizadas

Es interesante comentar que en el presente TFG se intentó abordar una tarea que finalmente quedó excluida del mismo, pero que al mismo tiempo requirió una cantidad de tiempo considerable (un mes y medio de trabajo). Esta tarea consistió en intentar migrar el prototipo original a una nueva plataforma de implementación.

Más concretamente, hay que notar que el hub del prototipo original (y finalmente del nuevo prototipo) de HDMS-CAN está sintetizado en un circuito programable de tipo Field Programmable Gate Array (FPGA) de la marca Xilinx. Sin embargo, en un principio, en este TFG se intentó migrar la implementación del hub para sintetizarlo en una FPGA de la marca Altera que se utiliza en algunas asignaturas de la Universidad de las Islas Baleares (UIB). La razón por la cual se intentó esta migración estriba en que la FPGA de Xilinx está algo obsoleta en comparación con esa placa de Altera y, por tanto, era de prever que la FPGA de Xilinx dificultaría el proceso de depuración de fallos del nuevo prototipo.

Sin embargo, por diferentes incompatibilidades entre las tecnologías empleadas por ambas FPGAs, se decidió abandonar esta tarea y continuar trabajando con la FPGA de Xilinx.

Por otra parte, también es importante destacar que otra de las tareas que se querían llevar a cabo en este TFG era escribir un artículo con los resultados del mismo, a fin de enviarlo a una revista internacional con revisión por pares. Desafortunadamente debido al tiempo invertido en intentar completar sin éxito la tarea anteriormente mencionada, no fue posible redactar dicho artículo.

1.5 Estructura del documento

La memoria de este TFG consta de varios capítulos que se corresponden en cierta medida con las diferentes tareas que se han explicado en la sección anterior. Dichos capítulos están ordenados de forma más o menos cronológica, con alguna variación de orden para mejorar la comprensión del documento.

A continuación se resume el contenido de dichos capítulos.

- Fundamentos y trabajo previo

  En este primer apartado se explicará el protocolo CAN, las configuración CAN-centrate y la patente HDMS-CAN. Además, se profundizará en aquellos aspectos fundamentales para el buen entendimiento del resto de documento.

- Caracterización de HDMS-CAN

  A continuación se explicará cómo se han caracterizado desde el punto de vista teórico la cobertura, el *bit rate*, y las restricciones topológicas de HDMS-CAN. Se hará especial énfasis en la propuesta sobre cómo se han de dimensionar los diferentes segmentos del tiempo de bit desde el punto de vista del hub.

  Además, se realizará un estudio con el fin de poder predecir cuál sería la cobertura máxima que el hub proporcionaría para diferentes bit rates.

- Prototipado

  Una vez caracterizado HDMS-CAN se describirá el nuevo prototipo que se ha implementado. En particular, se describirán los diferentes elementos *hardware* y *software* de este prototipo. También se explicarán detalladamente que elementos hardware y software adicionales se han desarrollado y/o utilizado para identificar y resolver los diferentes fallos de diseño e implementación.

- Pruebas de rendimiento

  En este capítulo se explicarán las pruebas realizadas para determinar el rendimiento de HDMS-CAN. En dichas pruebas se midió, para diferentes bit rates, la cobertura máxima, el número de errores de discriminación (cuántas veces el hub no es capaz de discriminar la contribución de algún nodo) y comunicación, y el consumo de corriente de los *transceivers* del nuevo prototipo de HDMS-CAN.

- Conclusión

  Finalmente, en el último capítulo se resume el trabajo realizado en el proyecto y cuáles son sus contribuciones; se destacan los aspectos más relevantes de los resultados obtenidos; se proponen algunos experimentos y mejoras que se podrían llevar a cabo en el futuro sobre el nuevo prototipo; y se realizan una serie de consideraciones personales.

CAPÍTULO 2

FUNDAMENTOS Y TRABAJO PREVIO

El presente capítulo tiene como objetivo explicar de forma breve aquellos aspectos tecnológicos que son necesarios para entender el trabajo realizado en este TFG.

2.1 Control Area Network (CAN)

CAN es un bus de campo desarrollado por la compañía *Robert Bosh GmbH* en los años 80, para servir como subsistema de comunicaciones a aplicaciones de control distribuido dentro de vehículos. Sin embargo, a pesar de que CAN se desarrolló inicialmente para la industria de la automoción, pronto su uso se extendió a otros contextos. De hecho, hoy en día CAN es uno de los buses de campo más utilizados en muchos ámbitos, tales como la robótica, la aviación, los automatismos, los electrodomésticos, las comunicaciones dentro de edificios, etc.

CAN contempla únicamente las dos primeras capas del modelo Open System Interconnection (OSI), es decir, la Capa de Enlace y la Capa Física. La Capa de Enlace fue definida originalmente por la propia Bosh en 1991. Posteriormente, en 1993, la International Organization for Standardization (ISO) estandarizó ambas capas mediante la especificación 11898[ISO, 1993]. Más adelante la organización *CAN in Automation* desarrolló varios estándares para la Capa Física de CAN, cada uno de ellos orientado a un tipo de aplicación específico. Por último, desde 2003, la ISO ha realizado varias extensiones del estándar original. Entre ellas destacan la especificación ISO 11898-1:2003 [ISO, 2003a,b] y, más recientemente, la especificación [ISO 11898-1, 2015].

Es importante notar que la última de estas especificaciones propone una serie de extensiones para poder alcanzar *bit rates* de más de 1 Mbit/sec. Sin embargo, estas extensiones sacrifican en parte algunas propiedades interesantes de CAN desde el punto de vista de las garantías de funcionamiento. Una de estas propiedades es la consistencia en la transmisión/recepción (*reliable broadcast*) de tramas, es decir, el hecho de que cada trama es o bien considerada como correcta por todos los nodos (el transmisor y los receptores), o bien no es aceptada por ninguno de los nodos. Además,

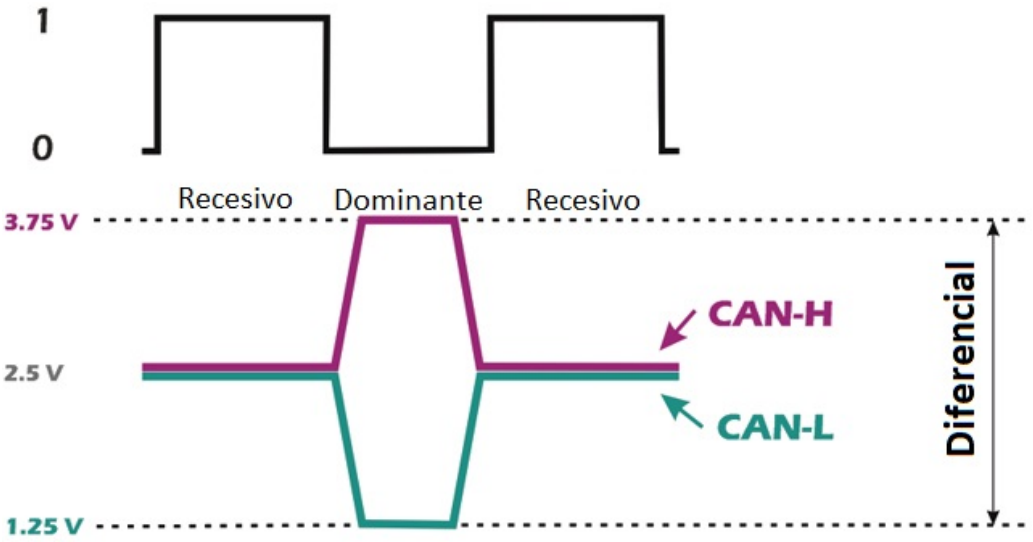

Figura 2.1: Representación de la relación entre bits y tensiones en los cables CAN-H y CAN-L. Esta imagen ha sido tomada del documento *Automotive CAN Bus System Explained*[2] y, además, ha sido editada

estas extensiones del protocolo CAN aún no han penetrado de forma considerable en la industria. Por tanto, el presente TFG se ha desarrollado para ser compatible con la especificación ISO 11898-1:2003; por lo que en principio debería preservar todas las propiedades de CAN que son deseables en términos de garantía de funcionamiento.

A continuación se presentarán aquellos aspectos de CAN que son más relevantes para este proyecto.

2.1.1 Capa física

Topología de bus

CAN está diseñado de forma que todos los nodos se conectan a un medio físico de comunicación único o bus. Normalmente este bus está formado por un par trenzado de hilos denominados CAN-H y CAN-L. El valor lógico a transmitir se codifica en el medio mediante la diferencia de tensión entre ambos hilos (Figura 2.1). El valor lógico '1' se codifica aplicando la misma tensión en ambos hilos; mientras que el valor lógico '0' se codifica aplicando una diferencia de tensión entre los dos hilos (dicha diferencia de tensión depende del tipo de *transceivers* utilizados). Este método de transmisión de bits es muy robusto frente a interferencias electromagnéticas. Éstas suelen afectar a ambos hilos simultáneamente provocando variaciones de tensión iguales en los dos hilos, lo cual no afecta a la diferencia de tensión entre ambos. Además, el hecho de ser un par trenzado también favorece la robustez frente a fuentes de interferencia externas y a la diafonía de posibles cables adyacentes. Por otro lado, para evitar el rebote de la señal, en los extremos del bus se sitúa una resistencia de 120 Ω.

Finalmente, mencionar que cada nodo se conecta al bus mediante un circuito transceptor o *transceiver*. Un *transceiver* es un dispositivo electrónico que, por una

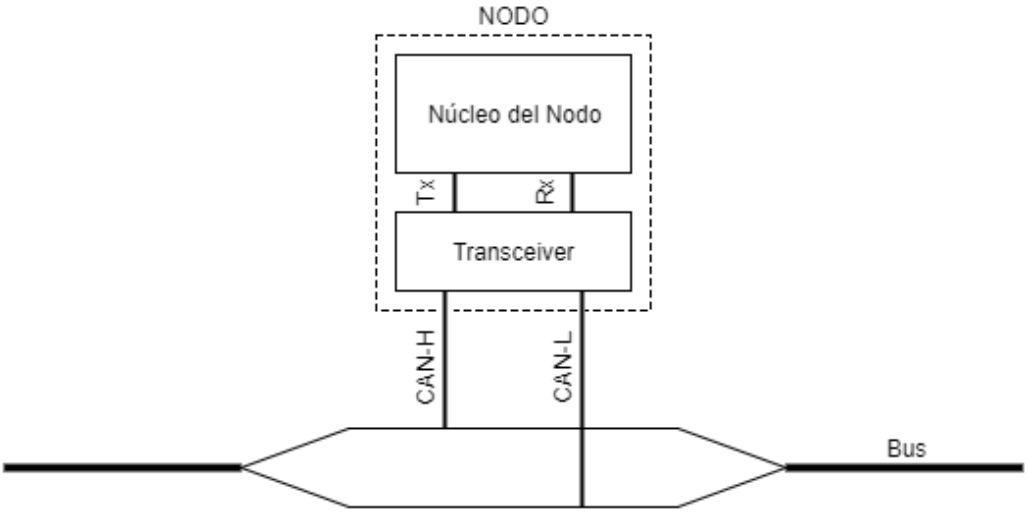

Figura 2.2: En esta imagen se muestran las diferentes conexiones entre el núcleo del nodo, el circuito *transceiver* y el bus en CAN

parte, es capaz de traducir la diferencia de voltaje entre los hilos CAN-H y CAN-L a un valor lógico, '0' o '1', que el nodo pueda recibir a través del pin Rx (señal de salida de recepción del *transceiver*); y que al mismo tiempo es capaz de traducir el valor lógico que un nodo quiere transmitir, a través del pin Tx (señal de entrada de transmisión del *trasnceiver*), a una diferencia de voltaje entre CAN-H y CAN-L. Ver Figura 2.2.

Propiedad *Wired-AND* (bits dominantes y recesivos)

Debido a la forma en que se codifican los valores lógicos en el medio físico, se dice que el medio en CAN tiene la propiedad de implementar una función lógica AND cableada (*Wired-AND*) de las contribuciones de todos y cada uno de los nodos. Esto quiere decir que basta con que haya un nodo que transmita un '0' lógico en el medio para que el valor lógico resultante en el bus valga '0'; mientras que es necesario que todos los nodos transmitan un '1' lógico en el medio para que el valor resultante del bus sea '1'. Por esta razón, se dice que en CAN un bit con valor lógico '0' es un bit *dominante*; mientras que un bit con valor lógico '1' es un bit *recesivo*.

Existen otras configuraciones en las que la operación lógica realizada es una OR. En este caso el dominante pasa a ser el '1' y el recesivo el '0'. Concretamente, para este proyecto, nosotros utilizamos la configuración con la operación lógica AND.

Para ilustrar la propiedad *Wired-AND* de CAN la Figura 2.3 muestra el valor lógico de la contribución de dos nodos, así como el valor lógico resultante que se observaría en un bus CAN en el cuál se transmitiesen únicamente esas dos contribuciones.

Es importante notar que esta propiedad *Wired-AND* permite implementar otros mecanismos relevantes de CAN, que se explicarán más adelante, p.e. el mecanismo de arbitraje y el mecanismo de señalización y globalización de errores.

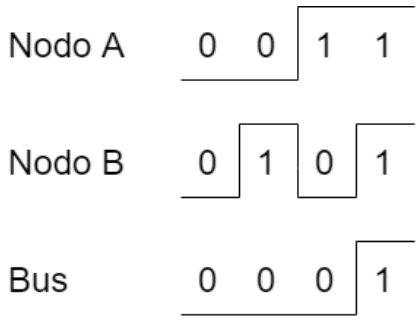

Figura 2.3: Resultado de las contribuciones de dos nodos en el bus

Propiedad *in-bit response*

Otra de las propiedades fundamentales que implementa la Capa Física de CAN es la que se conoce como *in-bit response*. Esta propiedad consiste en que todos los nodos observan casi simultáneamente todos y cada uno de los bits que se transmiten por el medio. En este sentido, un nodo no transmite un nuevo bit en el medio hasta que está seguro de que ese bit se ha propagado por todo el medio de forma que el resto de nodos han podido *samplear* (obtener) su valor lógico y de que él mismo ha sido capaz de samplear el bit transmitido por cualquier otro nodo de la red.

Como consecuencia de esta propiedad se puede decir que en CAN cada trama se transmite bit a bit, de tal forma que todos los nodos observan a la par cada uno de los bits que la constituyen.

Tal y como ocurre con la propiedad *Wired-AND*, la propiedad *in-bit response* también es necesaria para implementar el mecanismo de arbitraje y el mecanismo de señalización y globalización de errores.

Sincronización a nivel de bit y *bit timing*

Tal y como se acaba de explicar, en CAN los nodos observan casi simultáneamente cada bit de cada trama. Para conseguir este comportamiento es necesario que los nodos estén *sincronizados a nivel de bit*.

A continuación se resume la estrategia mediante la cual se consigue este tipo de sincronismo en CAN.

En primer lugar, hay que tener en cuenta que en CAN se dice que un nodo juega el papel de *leading transmitter* cuando gana el derecho a transmitir (cuando gana el acceso al medio). El resto de nodos juegan el rol de *nodos receptores*.

Dado un nodo receptor, éste debe considerar que cada bit comienza en el instante en que observa (o debería observar) el flanco que constituye el inicio de cada bit transmitido por el *leading transmitter*. Este flanco debería ser de bajada si el *leading transmitter* envía un dominante tras haber enviado un recesivo; mientras que debería ser un flanco de subida si el *leading transmitter* envía un recesivo tras haber enviado un dominante. En caso de enviar dos bits consecutivos con el mismo valor lógico, el nodo receptor no observará un flanco al inicio del segundo bit. A pesar de ello, el nodo receptor considera que el segundo bit empieza en el instante en que debería

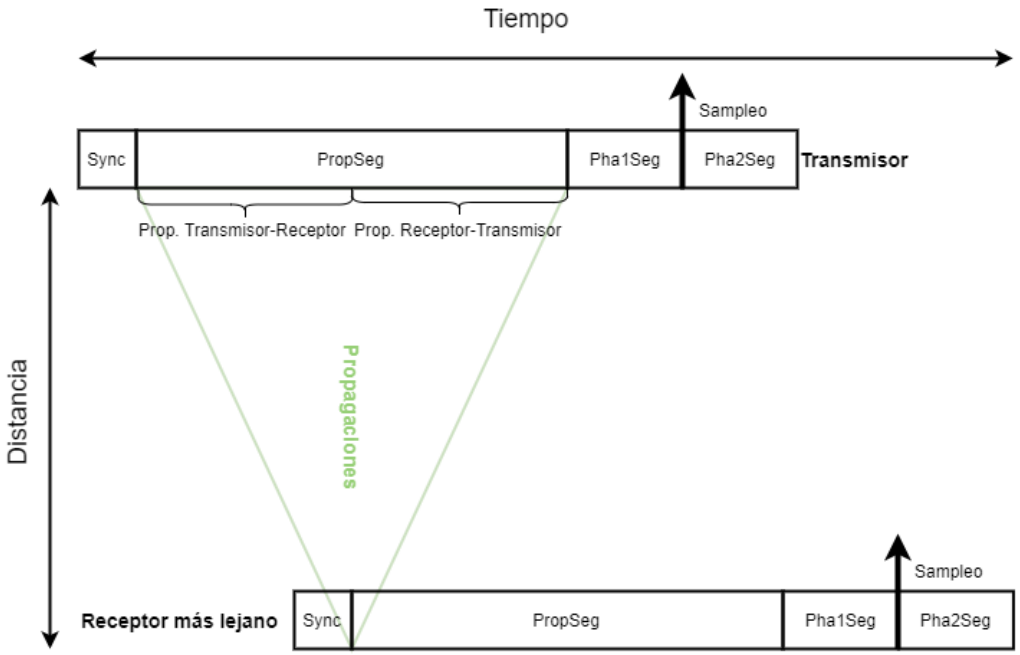

Figura 2.4: En este diagrama se aprecia como el tiempo de bit está dimensionado de tal forma que el tiempo dedicado a la propagación es suficientemente grande como para que de tiempo de que la transmisión del *leading transmitter* llegue al receptor más lejano y que la respuesta de éste llegue al transmisor antes de que ambos realicen el sampleo, de forma que todos los nodos verán lo mismo en cada bit (*in-bit responde*)

haberse producido un flanco si el valor de este segundo bit fuese el complementario del primero.

En segundo lugar, el *bit timing* (tiempo que dura el bit) se dimensiona para que dado un nodo transmisor éste sea capaz de, en cada bit, leer la contribución del bit receptor más lejano. Esto supone que el tiempo dedicado a la propagación sea dos veces el tiempo que tarda una señal en ir de un nodo al nodo más lejano. En otras palabras, el tiempo de propagación debe ser suficientemente largo como para mandar un valor y recibir la respuesta. Un ejemplo de esto se da en el bit de ACK puesto que el transmisor enviará un recesivo pero ha de ser capaz, en ese mismo bit, de leer el dominante del nodo más lejano. Ver Figura 2.4.

Cabe destacar que el *bit timing* y el tamaño del bus están directamente relacionados puesto que, como se puede ver en la Figura 2.5, reducir o alargar la distancia entre los nodos más alejados provoca cambios en el tiempo de propagación necesario, y por tanto, en el *bit timing*. Por otro lado esto implica una relación inversa entre la distancia entre nodos y la velocidad de transmisión ya que esta es inversamente proporcional al bit timing (cuanto mayor es el tiempo de bit menor es la velocidad de transmisión puesto que se necesita más tiempo para transmitir cada bit).

En las figura mencionada anteriormente y, concretamente, en la Figura 2.6 se puede apreciar como el tiempo de bit está dividido en sub secciones que llamaremos segmentos del tiempo de bit. Éstos presentan diferentes funcionalidades, más allá de la

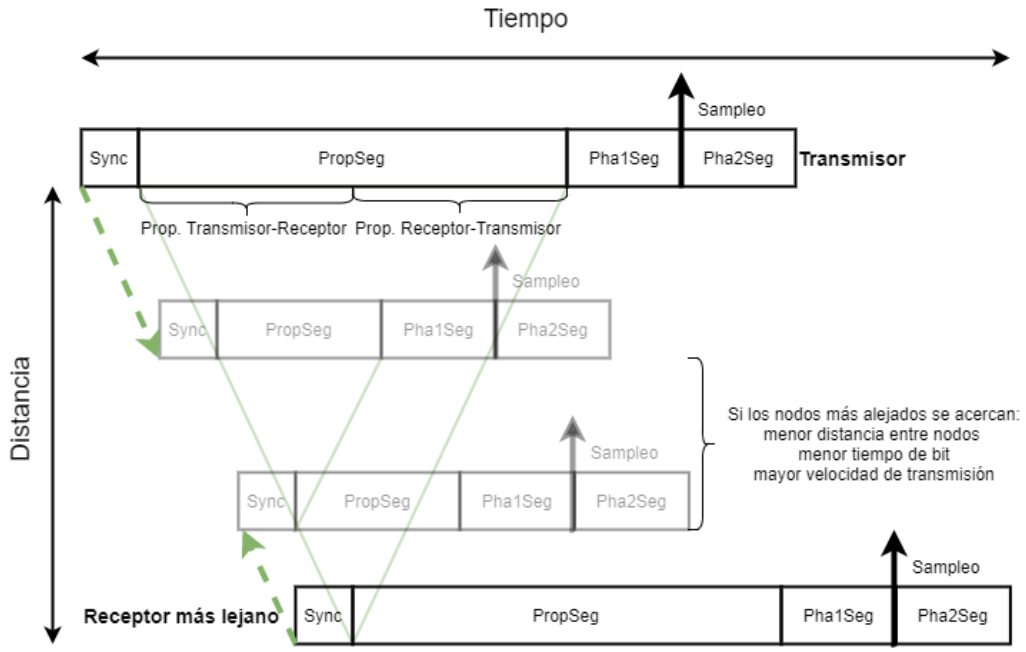

Figura 2.5: Este diagrama muestra la relación entre la distancia entre nodos, el tiempo de bit y la velocidad de transmisión

Figura 2.6: Segmentos del tiempo de bit en CAN

capacidad de sincronizar los nodos, las cuales se exponen a continuación.

- *SyncSeg*. Ésta es la primera parte del bit. La funcionalidad principal de este segmento es asegurar que los nodos están bien sincronizados a nivel de bit. Para ello, los nodos receptores deberían recibir los flancos de la transmisión en dichos segmentos.

- *PropSeg* y *Pha1Seg*. Estos dos segmentos se pueden unir en el segmento Tseg1, tal y como se puede observar en la Figura 2.6. A lo largo de este periodo de tiempo la contribución de todos los nodos de la red debe llegar a todos los nodos de la misma. Esto se debe a que, tras este tiempo, se realiza el sampleo del bus y es fundamental que todos los nodos vean exactamente lo mismo de forma que se cumpla el *in-bit response*.

- *Pha2Seg*. También llamado Tseg2, este tiempo establece un tiempo de espera entre el sampleo y la transmisión/recepción del siguiente bit.

En tercer lugar, los nodos receptores implementan un mecanismo de *resincronización* mediante el cual alargan o acortan algunas de las partes del tiempo de bit, a fin de *resincronizarse* con el *leading transmitter*. Esto es necesario porque los relojes de los diferentes nodos tienen una deriva los unos con respecto a los otros. Por tanto, si no se implementase este mecanismo de resincronización, algunos receptores podrían llegar a adelantarse o atrasarse a la hora de considerar el inicio del tiempo de bit. Es decir, un receptor podría considerar que un bit empieza demasiado pronto o demasiado tarde respecto a la visión que el *leading transmitter* tiene sobre ese bit. Si esto sucede, es posible que un nodo samplee incorrectamente el bit que resulta en el bus y, tal y como se explicará más adelante, ese nodo obligaría tarde o temprano a todos los nodos a abortar la trama que se está transmitiendo ya que detectaría inevitablemente algún error.

Dependiendo de si el nodo receptor se retrasa o adelanta, la forma de detectar la desincronización y la forma de solucionarla es distinta. A continuación se explican ambos procesos:

- *El nodo se ha adelantado*: Si durante el SyncSeg no se había producido ningún flanco pero, en el Tseg1 sí, el controlador considera que el nodo está desincronizado. Para resolver esta situación, el nodo alargará el tiempo de Pha1Seg para retrasar el SyncSeg del próximo bit haciendo que el nodo vuelva a estar sincronizado.

- *El nodo se ha retrasado*: En caso de que aparezca un flanco durante el segmento Pha2Seg el nodo considerará que está retrasado por lo que se acortará dicho segmento para hacer coincidir el flanco con el SyncSeg, recobrando así la sincronización.

Es importante, para este proyecto, mencionar que hay una máxima desincronización que se puede dar en CAN y cuyo valor es proporcionado por la especificación 11898[ISO, 1993]. Dicho valor de desincronización recibe el nombre de Synchronization Jump Width (SJW) y tiene una duración que varía en función de la velocidad de transmisión.

Aparte de las sincronizaciones anteriormente mencionadas, las cuales se producen durante la transmisión/recepción, existe una, denominada *hard-synchronization*, que se da al principio de la transmisión (en el Start Of Frame (SOF)). Esta permite comenzar la comunicación con los nodos sincronizados. Cuando los nodos detectan el primer flanco del primer bit de la trama estos pasan directamente al segmento Tseg1, independientemente del segmento del tiempo de bit en que se encontraran en el momento de recibir el flanco.

2.1.2 Capa de enlace

Formato de tramas

Existen 4 tipos de tramas en CAN: de datos, remota, de error y de overload. Sin embargo, a continuación solo se explicará el formato de la trama de datos, mientras que en la Sección 2.1.2 se expondrá el de la trama de error. Solo explicaremos éstas puesto que son las más relevantes y las utilizadas en el proyecto.

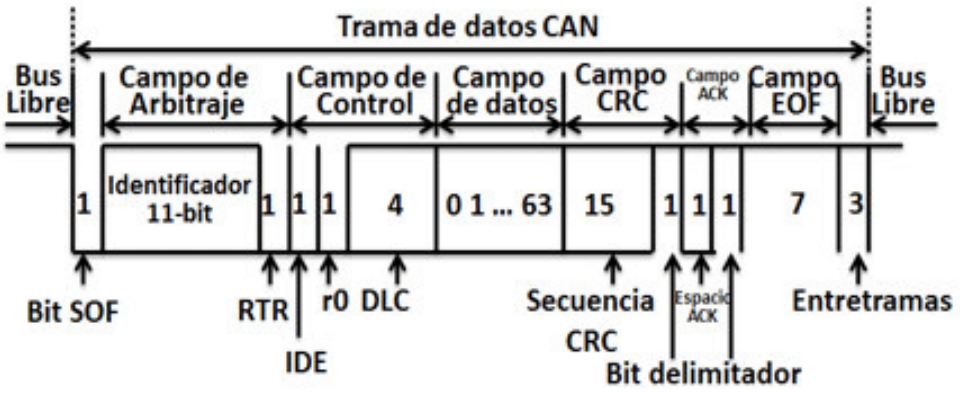

Figura 2.7: Formato de la trama de datos en CAN [3]

La trama de datos, como su nombre indica, se utiliza para enviar datos desde el nodo transmisor al nodo receptor. El formato de la trama es el mostrado en la Figura 2.7. A continuación se describen los diferentes segmentos de la trama.

Es importante mencionar que, mientras ningún nodo quiere transmitir ninguno de los 4 tipos de mensaje anteriormente mencionados, todos los nodos permanecen en estado de *idle* transmitiendo recesivos.

Las tramas comienzan con un dominante llamado SOF. Al haber recesivos en el bus, cuando éste se encuentra en estado de *idle*, el SOF genera un flanco que provoca la *hard-synchronization*, anteriormente mencionada, en todos los nodos receptores. Por otro lado, si otro nodo también pretendía transmitir, por lo que también ha transmitido el SOF, ambos deberán competir por el acceso al medio. Este proceso se realiza durante el periodo de arbitraje descrito a continuación.

El campo de arbitraje está formado por 11 bits identificadores y un bit llamado Remote Transmission Request (RTR) que básicamente vale dominante en las tramas de datos. A diferencia de otros protocolos, el identificador no representa al destinatario del mensaje, sino que da información del contenido y determina la prioridad del mensaje. Cuando hay más de un nodo intentando transmitir, éste es el campo de la trama en el que se determina al ganador del derecho a transmitir, tal y como se explica en la Sección 2.1.2.

Después del campo de arbitraje se encuentra el campo de control. Este está formado por el bit IDentifier Extension (IDE) y el bit R0, los cuales valen dominante en las tramas de datos utilizadas en el proyecto, y por 4 bits denominados Data Length Code (DLC). Estos últimos, en una trama de datos, indican el número de bytes del campo de datos.

A continuación se encuentra el campo de datos, el cual está formado por un número de bytes que puede variar entre 0 y 8. Ésta es la información que el nodo transmisor de la trama quería enviar.

Después del campo de datos se encuentra el campo de CRC. Dicho campo está formado por 15 bits que contienen un código llamado Cyclic Redundancy Code (CRC), el cual es utilizado por los nodos para asegurar la integridad de la trama. Tras estos 15

bits se encuentra un recesivo utilizado para separar este campo del siguiente, el campo de ACK.

El campo de ACK informa al nodo transmisor de que la recepción del mensaje se ha realizado correctamente. Concretamente, informa de que la trama ha superado la prueba de CRC. Este campo está formado por dos bits: el bit de ACK y un delimitador. En el bit de ACK los receptores que hayan recibido bien la trama mandan un dominante mientras que el transmisor envía un recesivo para poder leer el dominante de los receptores. Si el transmisor recibe dicho dominante interpretará que al menos un nodo ha recibido bien el mensaje. Por el contrario, si lee recesivo, considerará que ha habido error, el cual se tratará tal y como se explica en la Sección 2.1.2.

Finalmente se transmiten 7 bits recesivos del campo llamado End Of Frame (EOF) que marcan el final de la trama.

Las tramas se separan entre ellas por una serie de bits recesivos llamados InterFrame Space (IFS). Esta separación comienza con 3 bits denominados *intermission*, o entretramas, tras lo cual el bus pasa a estado de *idle* tal y como se ha mencionado al principio de este apartado.

Control de Acceso al Medio

Cuando el bus está *idle* cualquier nodo puede intentar transmitir, por lo que dos o más nodos pueden intentar transmitir a la vez. En estas situaciones es necesario un mecanismo de arbitraje que determine qué nodo gana el derecho a transmitir. Este mecanismo se lleva a cabo en el campo de arbitraje.

Cuando dos o más nodos compiten por el derecho a transmitir, ganará el arbitraje aquel que tenga el identificador más bajo.

Esta regla se cumple si en el canal se realiza una AND. Si por el contrario el bus realiza una OR de las contribuciones, el dominante pasa a ser el '1', y por lo tanto, el mensaje más prioritario será aquel que tenga el identificador más grande.

Esto se debe a que, mientras dos nodos transmiten los mismos bits no se determina el ganador, sin embargo, cuando uno de los nodos introduce un recesivo ('1' lógico en el caso de la AND) y el otro nodo un dominante ('0' lógico) aquel que haya introducido el recesivo verá el dominante en el bus y sabrá que ha perdido el arbitraje, por lo que tomará el rol de receptor. Ver Figura 2.8.

Frame encoding

CAN utiliza la codificación Non Return to Zero (NZR). Esta técnica determina el valor del bit en función de la tensión del bus. Por tanto, mientras el valor digital transmitido no cambie, el valor de tensión del bus tampoco lo hará. Sin embargo, y como ya se ha mencionado anteriormente, los nodos utilizan los flancos para sincronizarse. Esto puede provocar que los nodos se desincronicen si se transmiten muchos bits consecutivos con el mismo valor.

Para evitar esto, CAN introduce en las tramas bits de *stuff*. Cada vez que un nodo transmite 5 bits consecutivos con el mismo valor, tras estos se incluye un bit de *stuff* que toma el valor opuesto al de los 5 bits anteriores. Este bit no será procesado por los nodos mas les servirá para sincronizarse.

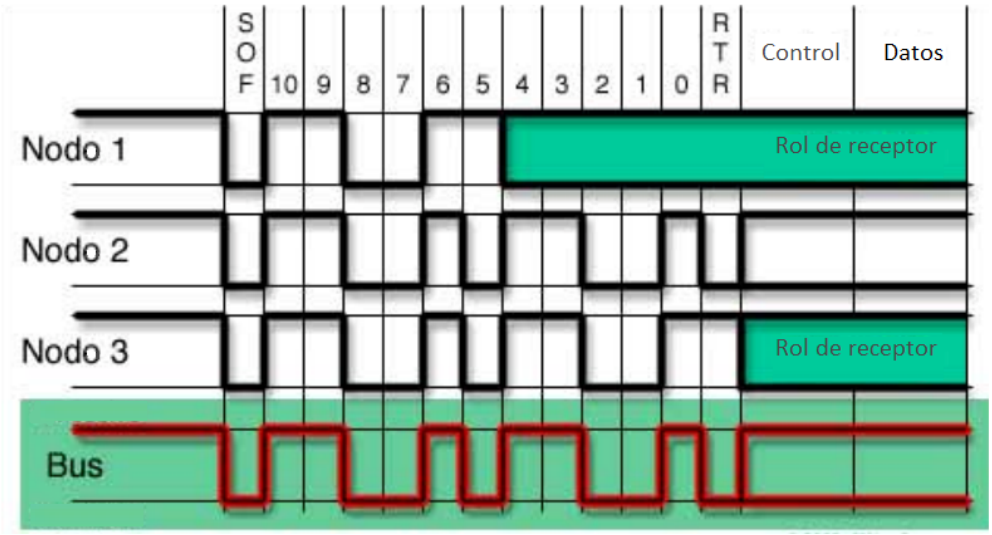

Figura 2.8: Diagrama de la ejecución del mecanismo de arbitraje realizado por 3 nodos intentando transmitir. En él se aprecia como el Nodo 1 tiene el identificador menos prioritario mientras que el Nodo 2 tiene el identificador más prioritario y, por tanto, menor. Esta imagen ha sido tomada del documento *Qué es CAN y cómo funciona el módulo del microcontrolador dsPIC30F4013*[4] y, además, ha sido editada

Esta regla tiene una serie de excepciones. Concretamente, no se aplica en los siguientes bits: delimitador de CRC, delimitador de ACK, bit de ACK, EOF e IFS. Tampoco se introducen bits de *stuff* en las tramas de error, las cuales explicaremos a continuación.

Error detection and signalling

En este apartado se expondrán los diferentes tipos de errores que CAN es capaz de detectar (*error detection*) y a continuación se explicará el proceso de señalización de dichos errores (*error signaling*).

Una de las ventajas de CAN respecto a otros buses de campo es que los nodos implementan mecanismos de detección de errores y tratamiento de fallos bastante sofisticados.

Por una parte estos mecanismos permiten a un nodo detectar errores (*error detection*) en la transmisión/recepción que afecten a la trama antes de que ésta haya terminado de transmitirse/recibirse. Más concretamente, cuando un nodo detecta un error, lo señaliza (*error signaling*) en el medio obligando al resto de nodos a detectar un error y a descartar la trama afectada. Es decir, cuando un nodo detecta un error durante una trama, lo señaliza y globaliza (*error signaling and globalization*) de tal forma que él y el resto de nodos descarten esa trama. Por este motivo se dice que las comunicaciones en CAN presentan la propiedad de *reliable broadcast*. Según esta propiedad, si un nodo transmite con éxito una trama entonces el resto de nodos la reciben correctamente. O dicho de otra forma, cada trama o bien es aceptada como válida por todos los nodos (el

transmisor y los receptores) o por ningún nodo.

Por otra parte, cada nodo incluye un contador de errores de transmisión (Transmission Error Counter (TEC)) y un contador de errores de recepción (Reception Error Counter (REC)). El nodo incrementa y decrementa estos contadores siguiendo ciertas reglas; y los utiliza para diagnosticarse a sí mismo en fallo cuando alguno de ellos (alguno de los contadores) alcanza una cota determinada. Cuando esto ocurre el nodo toma ciertas medidas (p.e. desconectarse del bus) para reducir el impacto negativo que podría ejercer sobre las comunicaciones entre los otros nodos.

En esta sección nos centraremos en explicar cuáles son los diferentes tipos de error que un nodo CAN es capaz de detectar, así como en la forma en que un nodo señaliza y globaliza un error.

CAN tiene la capacidad de detectar 5 tipos de errores:

- *Error de bit*. Este error se produce cuando el bit leído en el bus no se corresponde con el valor introducido por el transmisor. Esto no se considera error en el campo de arbitraje si un nodo transmite un recesivo y recibe un dominante. En este caso, como ya se mencionó en la Sección 2.1.2, el nodo considera que ha perdido el arbitraje. Por otro lado, tampoco se considera error en el bit de ACK ya que el transmisor espera la recepción de un dominante.

- *Error de stuff*. Este error es detectado por los nodos cuando se incumple la regla del bit de *stuff*, es decir, cuando leen 6 bits seguidos con el mismo valor. Esto será así siempre y cuando no se trate de uno de los casos especiales, mencionados anteriormente en la Sección 2.1.2, en los que la regla no se aplica.

- *Error de CRC*. Este error es detectado por los nodos receptores. El error de CRC se produce cuando el código CRC recibido no coincide con el que los nodos receptores habían calculado.

- *Error de forma*. Este error es detectado cuando los nodos receptores leen, en uno de los bits predefinidos (como pueden ser el delimitador de CRC, el de ACK, el bit R0, etc), un valor diferente al que le corresponde. Un ejemplo sería leer un dominante en los bits delimitadores de CRC o ACK ya que éstos están condicionados a ser recesivos.

- *Error de ACK*. Este error es detectado por el transmisor. Se produce cuando no recibe un dominante en el bit de ACK lo cual indica que ningún nodo ha recibido correctamente la trama.

Una vez explicados los diferentes tipos de errores que CAN es capaz de detectar, a continuación se expone el proceso de señalización de dichos errores.

En lo que respecta a la forma en que un nodo señaliza y globaliza un error cuando lo detecta, cabe recordar que, además de tramas de datos, un nodo es capaz de transmitir lo que se conoce como trama de error. Más concretamente, una trama de error está compuesta por 6 bits dominantes (el llamado *error flag*) y 8 o más bits recesivos (el conocido como *error delimiter*). Cuando un nodo detecta un error, empieza a transmitir un *error flag*. Los 6 bits dominantes consecutivos de este *error flag* fuerzan al resto de nodos a detectar, como muy tarde, un error de stuff en el sexto bit dominante de dicho

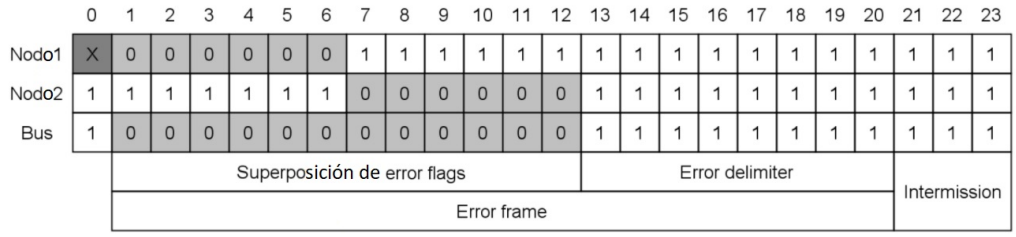

Figura 2.9: Diagrama de la superposición de las tramas de error durante el proceso de globalización de un error. Esta imagen ha sido tomada del documento *"Construction of sfiCAN: a star-based fault-injection infrastructure for the Controller Area Network"*[5] y, además, ha sido editada

flag. Esto asegura, por un lado, que el resto de nodos abortan la transmisión/recepción de la trama que se estaba transmitiendo en el bus y, por otro lado, que señalizan el error iniciando la transmisión de su propio *error flag*. Cuando un nodo termina de transmitir su *error flag*, empieza a transmitir un *error delimiter*, el cual está constituido por al menos 8 bits recesivos. Un nodo da por finalizada la señalización cuando, tras haber transmitido 8 bits recesivos, observa un bit recesivo en el bus.

Notar que desde el punto de vista de lo que se observa en el bus, la señalización de un error está formada por el solapamiento de los *error flags* y *error delimiters* transmitidos por los diferentes nodos. La Figura 2.9 muestra un ejemplo de este solapamiento en un bus al que hay dos nodos conectados. Concretamente la figura muestra un escenario en el que uno de los nodos, Nodo1, detecta un error (símbolo X) y consigue globalizarlo en el sexto bit del *error flag* que transmite. En este escenario, lo que se observaría en el bus son 12 bits dominantes (6 dominantes del nodo que detecta el error originalmente y 6 dominantes del otro nodo), seguidos de 8 bits recesivos. Al finalizar la señalización del error, los nodos consideran que el bus ingresa en el estado de *intermission*, tras el cual considerarán que el bus entra en *idle* (donde los nodos pueden intentar transmitir de nuevo).

2.2 CANcentrate

Tal y como se ha explicado en la Sección 1.1, CAN presenta varias desventajas derivadas de su medio de transmisión, el bus. En particular, en un bus CAN las contribuciones de los nodos se mezclan irreversiblemente debido a la función *wired-AND* que el bus implementa. Es decir, en el medio físico de CAN es imposible distinguir la contribución que cada nodo aporta para cada bit que se transmite por dicho medio. Tal y como se ha explicado, este hecho limita o dificulta la implementación de mecanismos que necesiten discriminar esas contribuciones, p.e. mecanismos que permitan detectar de forma precisa los errores transmitidos por cada nodo y que, por tanto, permitan diagnosticar de forma fiable qué nodos sufren de diferentes tipos de averías.

Algunos de los miembros del grupo de investigación en donde he realizado el TFG propusieron CANcentrate para poder discriminar la contribución de cada nodo para cada bit. CANcentrate utiliza una topología de estrella en lugar de un bus, en donde

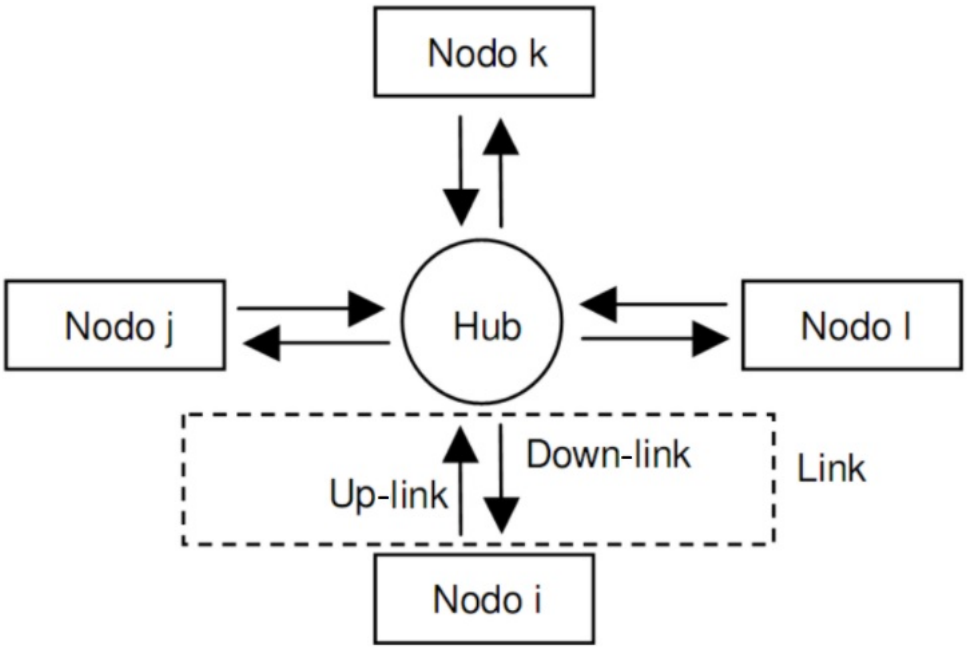

Figura 2.10: Esquema de la topología de CANcentrate. Esta imagen ha sido tomada del documento *"Construction of sfiCAN: a star-based fault-injection infrastructure for the Controller Area Network"*[5] y, además, ha sido editada

el elemento central de la estrella, un hub, puede discriminar la contribución de cada nodo de forma independiente.

CANcentrate es una infraestructura de comunicaciones formada por: una serie de nodos, que utilizan el protocolo CAN para comunicarse entre ellos, y un concentrador activo (hub). Estos elementos se conectan en una topología de estrella donde cada uno de los nodos se conecta al hub mediante dos links, uno de transmisión (*uplink*) y uno de recepción (*downlink*) tal y como se puede apreciar en la Figura 2.10. De esta forma se consigue separar físicamente la contribución del nodo y el valor resultante de la contribución de cada nodo.

Este hub se utiliza, principalmente, para poder discriminar la contribución de cada uno de los nodos de la red. En primer lugar el hub recibía la contribución individual de cada nodo a través de los *uplinks*. A continuación, en una fracción del tiempo que dura un bit, el hub realiza una AND con todas las contribuciones y transmite el resultado a través del *downlink* a todos los nodos simulando la propiedad *wired-AND* de CAN. De esta forma los nodos pueden comunicarse usando CAN de la forma habitual a la vez que el hub identifica la contribución de cada nodo. Esto permite, entre otras cosas, implementar mecanismos de contención de errores, es decir, esto le permite al hub de CANcentrate detectar errores enviados por cualquier nodo y aislar dicho nodo si fuera necesario. De hecho, el hub de CANcentrate incluye mecanismos para contar el número de errores, p.e. de stuff, de formato, etc., enviados por cada nodo, así como para aislar cualquier nodo que haya acumulado demasiados errores.

A pesar de que CANcentrate es capaz de discriminar la contribución de cada nodo,

y por tanto, permite implementar mecanismos que precisen de esta propiedad, CANcentrate aún presenta una serie de desventajas que pueden ser importantes. Por un lado está el hecho de duplicar tanto los cables como los *transceivers*, lo cual supone más peso, coste y complejidad de la instalación. Además, muchos de los productos comerciales que utilizan el protocolo CAN para las comunicaciones presentan un único *transceiver* y un único conector, impidiendo separar físicamente la transmisión de la recepción. Éste es, por ejemplo, el caso de los nodos de control utilizados en los automóviles, ECU.

2.3 Highly Discriminating Monitoring Star for CAN, HDMS-CAN

Tal y como ya se ha explicado, para solventar las desventajas de CANcentrate que se acaban de mencionar al final de la sección anterior, algunos miembros del grupo de investigación en donde he realizado el TFG propusieron una nueva infraestructura de comunicaciones denominada HDMS-CAN. Esta infraestructura de comunicaciones, como CANcentrate, también presenta un hub capaz de mantener las comunicaciones de los nodos mediante el protocolo CAN a la vez que consigue discriminar la contribución de cada uno de los nodos. Para ello se basa en una topología en estrella, con un hub activo, en la que cada nodo se conecta al hub mediante un único link, evitando tener que utilizar un *uplink* y un *downlink* separados.

En los siguientes apartados se explicará la arquitectura de HDMS-CAN brevemente introducida en este apartado. Además, se explicará el proceso de enmascaramiento, mecanismo mediante el cual es posible discriminar la contribución de cada nodo con un único link y resolver problemas como el *infinite loop-back* que explicaremos más adelante. Finalmente, se explicarán los nuevos segmentos del tiempo de bit del hub, que permiten realizar el enmascaramiento, y la estructura interna del hub.

2.3.1 Arquitectura general

La Figura 2.11 muestra la arquitectura general de HDMS-CAN. Como se puede observar, HDMS-CAN utiliza una topología en estrella donde los nodos transmiten/reciben tramas utilizando el protocolo CAN y el hub permite la comunicación a la vez que es capaz de discriminar la contribución de cada uno de los nodos.

Además, internamente, el hub incluye una serie de módulos, que se explicarán en la Sección 2.3.4, que le permiten, entre otras cosas: hacer una AND de las contribuciones de cada nodo y enviar el resultado a estos tal y como hacía CANcentrate para mantener la propiedad de *Wired-AND*; así como implementar el mecanismo de enmascaramiento.

2.3.2 Enmascaramiento

Como ya hemos dicho, este hub realiza una AND de las contribuciones de los nodos y les envía de vuelta el resultado. Sin embargo, al tener un único link por nodo, el valor resultante en cada link es el resultado de la AND y no la contribución del nodo correspondiente a ese link. Esto ocasiona dos problemas.

El primero de estos problemas es que el hub no puede identificar la contribución de cada nodo para cada bit.

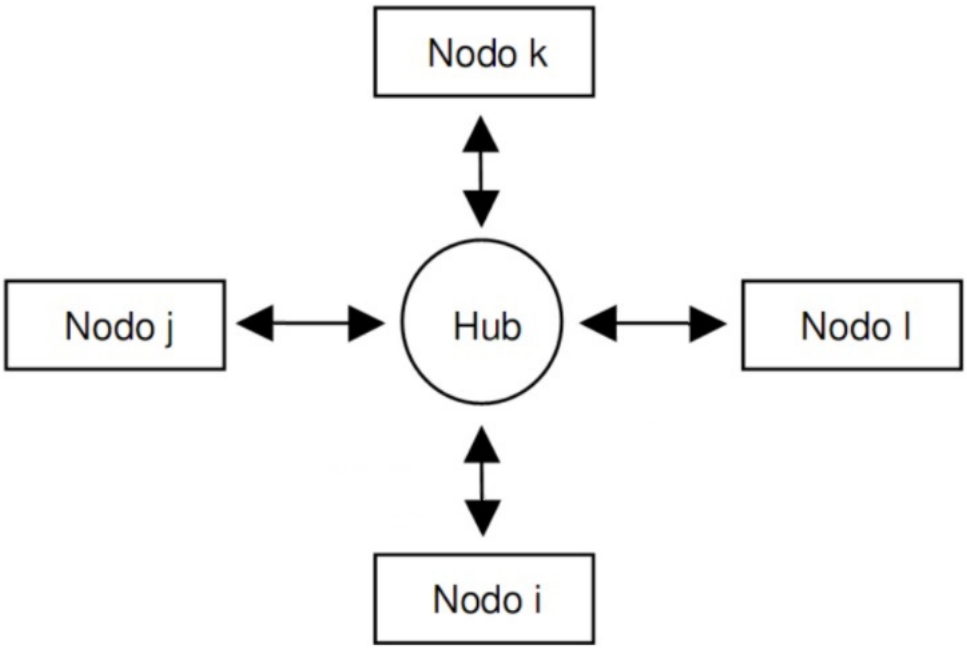

Figura 2.11: Esquema de la topología de HDMS-CAN

Para solventar este problema, durante una fracción de cada tiempo de bit el hub enmascara en cada link el resultado de la AND con un valor recesivo. Es decir, en lugar de enviar por cada link el resultado de la AND, durante dicha fracción el hub envía un valor recesivo. Notar que de esta forma, durante el tiempo que dura la transmisión de este valor recesivo, el hub puede observar en cada link cuál es la contribución real del nodo correspondiente.

La Figura 2.12 muestra un ejemplo para entender mejor este mecanismo. Supongamos que nos encontramos en una red con solo dos nodos, y que dicha red se encuentra en *idle*, es decir, todos los nodos transmiten recesivo en el canal al encontrarse en un periodo entre tramas. Imaginemos que, tras ese periodo de *idle*, sólo el nodo 1 intenta transmitir. Por tanto, durante el tiempo de bit indicado mediante la etiqueta “Bit 1” la contribución de ese nodo (“Nodo 1”) será ‘0’ (dominante), mientras que la contribución del nodo 2 (“Nodo 2”) será ‘1’ (recesivo). Sin embargo, notar que, en principio, el hub realiza una AND de ambas contribuciones y las envía de vuelta dando como resultado que la primera parte del tiempo del bit “Bit 1” el valor resultante en el link correspondiente a cada nodo (“Link 1” y “Link 2” respectivamente) es ‘0’. Esta acción que realiza el hub durante esta primera parte del tiempo de “Bit 1” está indicada mediante la etiqueta “AND” en la fila “Hub”. Ahora bien, después de esta primera fracción del tiempo de “Bit 1”, el hub envía un valor recesivo (‘1’) a través de los links. Esta acción que realiza el hub está indicada mediante la etiqueta “Mask” en la fila “Hub”. Como consecuencia, durante el intervalo de tiempo que dura esta acción, la señal resultante en el link de cada nodo es la contribución de ese nodo, por tanto, el hub será capaz de leer (“samplear”) un ‘0’ en el link del nodo que transmite el ‘0’ (“Link 1”) y un ‘1’ en el link del nodo que transmite el ‘1’ (“Link 2”). Notar que de forma genérica denominamos al valor, o pulso,

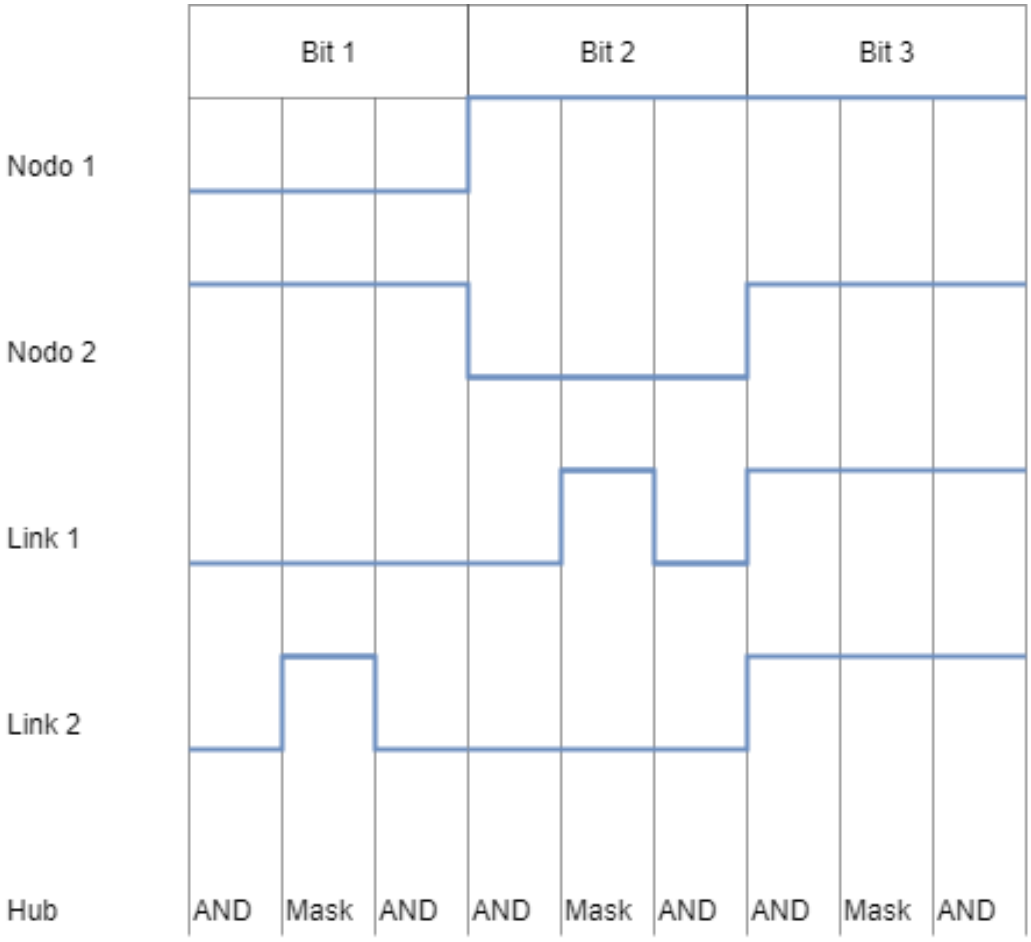

Figura 2.12: Esquema del proceso de enmascaramiento. En él se ve la transmisión de dos nodos (Nodo 1 y Nodo 2), el valor lógico en los canales de dichos nodos (Link 1 y Link 2) y la acción realizada por el hub (donde AND es cuando el hub aplica la AND y Mask es cuando el hub aplica el enmascaramiento

'1' que el hub envía en el intervalo de tiempo "Mask" como *pulso de enmascaramiento*. Más concretamente, como este pulso le permite al hub discriminar la contribución de cada nodo, nos referimos a él como *enmascaramiento de discriminación*. Por último, al final del tiempo de Bit 1, el hub deja de enviar este pulso de discriminación y, en su lugar, el hub simplemente envía de nuevo el valor que resulta de realizar la AND entre las contribuciones de ambos nodos. El resto de la Figura 2.12 muestra que el hub puede aplicar este mismo procedimiento en cada tiempo de bit para conocer cuál es la contribución de cada nodo en cada uno de esos bits.

El segundo problema de tener un único link por nodo es el que se conoce como *infinite loop-back*. A grandes rasgos este problema consiste en que una vez que el hub envía un valor dominante a través de los links, todos ellos quedan permanentemente en un valor dominante impidiendo la comunicación a partir de entonces. Para entender mejor por qué se produce este fenómeno, imaginemos que en la red se está transmitiendo

un valor recesivo y que, en un momento dado, un nodo transmite un valor dominante al hub. Cuando ese valor dominante alcance el hub, provocará que el resultado de la AND sea dominante y que el hub transmita dicho dominante por todos los links. A partir de ese momento, aunque todos los nodos envíen recesivo, el valor dominante transmitido por el hub sobrescribirá esos recesivos en todos los links. Por tanto, el hub permanecerá recibiendo dominantes (los transmitidos por él mismo) y transmitiendolos por todos los links. Dicho de otra forma, se crea un bucle tal que los links permanecerán con el valor dominante transmitido por el hub, independientemente del valor que transmitan los nodos.

En un principio parece que este problema se puede solucionar con el enmascaramiento de sampleo; ya que éste podría hacer desaparecer el dominante transmitido por el hub y, por tanto, impediría la formación del *infinite loop-back*. Sin embargo, el hub solamente debe aplicar el enmascaramiento de sampleo cuando esté seguro de que han llegado hasta él (el hub) las contribuciones de todos los nodos. En caso contrario, el hub no samplearía correctamente aquellas contribuciones que no hayan llegado hasta él. Por tanto, para resolver el problema del *infinite loop-back* el hub debe utilizar un pulso de enmascaramiento diferente al enmascaramiento de discriminación. Concretamente, como se verá a continuación, este nuevo pulso de enmascaramiento debe cumplir dos requisitos. Por una parte, debe realizarse al principio del tiempo de bit para asegurar una buena sincronización a nivel de bit entre los nodos (y el hub). Por otra parte, el hub solo debe realizar este pulso de enmascaramiento en el bit actual si el valor resultante del bit anterior fue dominante; de lo contrario, si el bit anterior fue recesivo y el hub transmite este pulso al principio del bit actual, entonces este pulso impediría a los nodos ver el flanco de bajada del bit actual en caso de que el bit actual fuese dominante.

En la Figura 2.13 se muestran los efectos de los pulsos de enmascaramiento al principio del bit y cómo estos son necesarios después de que el *leader transmitter* haya transmitido un dominante pero perjudiciales si anteriormente había transmitido un recesivo.

Describamos paso a paso el ejemplo mostrado en la Figura 2.13 para entender mejor el enmascaramiento utilizado para evitar el *infinite loop-back*. En un principio ("Bit 1") los dos nodos transmiten recesivo. A continuación, uno de los nodos , el que a partir de ahora consideraremos el transmisor ("Nodo 2"), transmite un dominante. Al principio de este bit ("Bit 2") el hub tiene dos opciones, realizar un pulso de enmascaramiento, representado mediante la etiqueta "Mask" en la fila "Hub" o realizar una AND representada por la etiqueta "AND". Dado que el resultado del bit anterior ("Bit 1") no era dominante, en principio no sería necesario aplicar el enmascaramiento puesto que no hay posibilidad de que se produzca *infinite loop-back*. Sin embargo, vamos a ver en más detalle que pasaría en cada caso. Por un lado, si se realiza dicho enmascaramiento, el flanco de bajada no se producirá en el "Link 1", el conectado al nodo receptor ("Nodo 1"), hasta que se haya producido el enmascaramiento para evitar el *infinite loop-back* y el enmascaramiento de discriminación, lo que desincronizaría al receptor. Por otro lado, si, en vez de enmascaramiento, se realiza la AND, el "Link 1" pasará a valer dominante y, por tanto, el nodo receptor recibirá el flanco correctamente, sincronizandose con el transmisor. Finalmente, el transmisor envía un recesivo, por lo que el hub vuelve a tener dos opciones, realizar o el enmascaramiento o la AND al principio del bit. En este caso, como el bit anterior había sido dominante, sí se produce el efecto de *infinite*

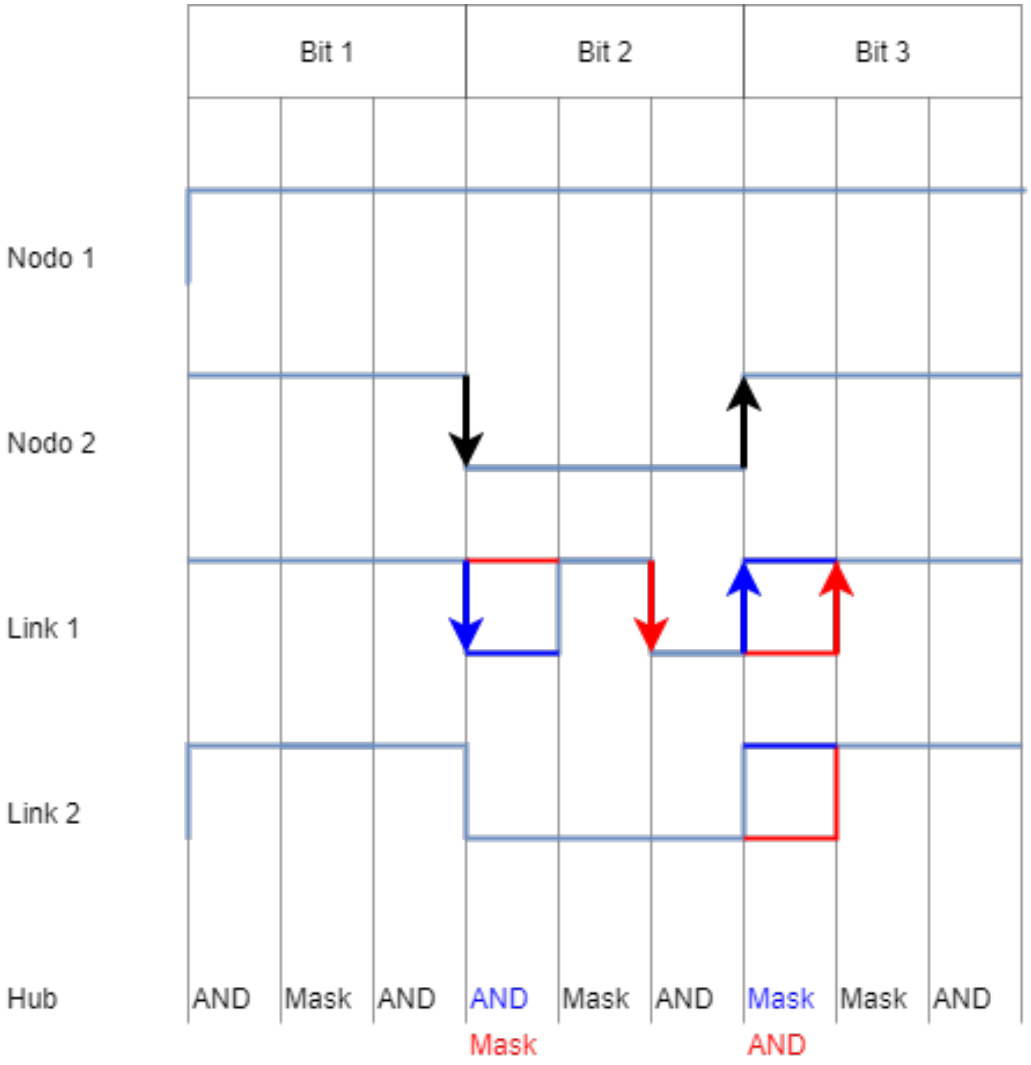

Figura 2.13: Esquema del proceso de enmascaramiento para evitar el *infinite loop-back*. Aquí se puede ver como el enmascaramiento permite que los nodos receptores se sincronicen con los flancos en vez de desincronizarse como pasaría si no tuviéramos el enmascaramiento al principio del bit tras haber transmitido un dominante. Con flechas negras se marca donde deberían situarse los flancos para sincronizar los nodos correctamente, en azul se marca donde verían los receptores los flancos si se realizan las acciones correctas (enmascaramiento solo tras un dominantes) y en rojo donde se situarían los flancos si no se realizaran dichas acciones

Figura 2.14: Segmentos del tiempo de bit en el hub de HDMS-CAN

loop-back, así que, hasta que el hub no enmascare su señal de transmisión no sabrá que los nodos están enviando recesivos. Veamos, concretamente, que es lo que ocurre. Si el hub realiza la AND ("AND") al principio del bit ("Bit 3") esta se retroalimentará, es decir, leerá el dominante de salida y lo retransmitirá, lo cual hará que en los links se mantenga el dominante, impidiendo ver el flanco de subida, aunque todos los nodos estén enviando recesivos. Sin embargo, si se realiza el enmascaramiento ("Mask"), los links pasarán a valer recesivo por lo que el nodo receptor recibirá correctamente el flanco y, por tanto, se sincronizará correctamente con el nodo transmisor.

2.3.3 Estructura del tiempo de bit

A fin de poder realizar los pulsos de enmascaramiento descritos en la sección anterior, fue necesario definir una nueva estructura del tiempo de bit desde el punto de vista del hub. Es decir, a pesar de que los nodos consideran la misma estructura del tiempo de bit que el estándar de CAN, el hub internamente considera una estructura diferente del tiempo de bit. Dicha estructura se muestra en la Figura 2.14.

A continuación explicaremos brevemente la función de cada uno de los segmentos del tiempo de bit.

El SyncSeg y el Pha2Seg tiene las mismas funcionalidades que los segmentos con el mismo nombre del estándar de CAN por lo que estos segmentos ya están explicados en la Sección 2.1.1.

Después del SyncSeg se encuentra el segmento PreMask. Éste es el segmento en el que el hub aplica el enmascaramiento necesario para evitar el *infinite loop-back*, en caso de que el valor del bit anterior haya sido dominante. Por otra parte, este segmento le permite al hub esperar un intervalo de tiempo lo suficientemente grande como para estar seguro de que la contribución de cada nodo ha llegado hasta él (el hub).

En este sentido, tras el segmento de PreMask se encuentra el segmento Mask. Es en Mask donde el hub aplica el enmascaramiento de discriminación y en donde monitoriza (samplea) la contribución de cada uno de los nodos.

Finalmente, antes del Pha2Seg se encuentra el segmento RestabSeg. En este segmento el hub deja de aplicar el enmascaramiento de discriminación a fin de que la señal resultante de acoplar las contribuciones de todos los nodos, mediante la AND, se propague de nuevo a través de todos los links hasta los nodos. Es importante notar que tanto el hub como los nodos samplearán esta señal resultante justo antes de que se inicie el segmento Pha2Seg (que en el caso de los nodos, tal y como se ha visto en la Sección 2.1.1, está situado después del segmento Pha1Seg). Por tanto, es necesario que RestabSeg sea lo suficientemente grande como para que esta señal resultante se propague hasta todos los nodos antes de que éstos (y el hub) la sampleen.

Por último, es importante aclarar que durante los segmentos PreMask y Pha2Seg el hub implementa mecanismos de resincronización a nivel de bit similares a los que implementan los nodos CAN en los segmentos PropSeg, Pha1Seg y Pha2Seg (ver Sección

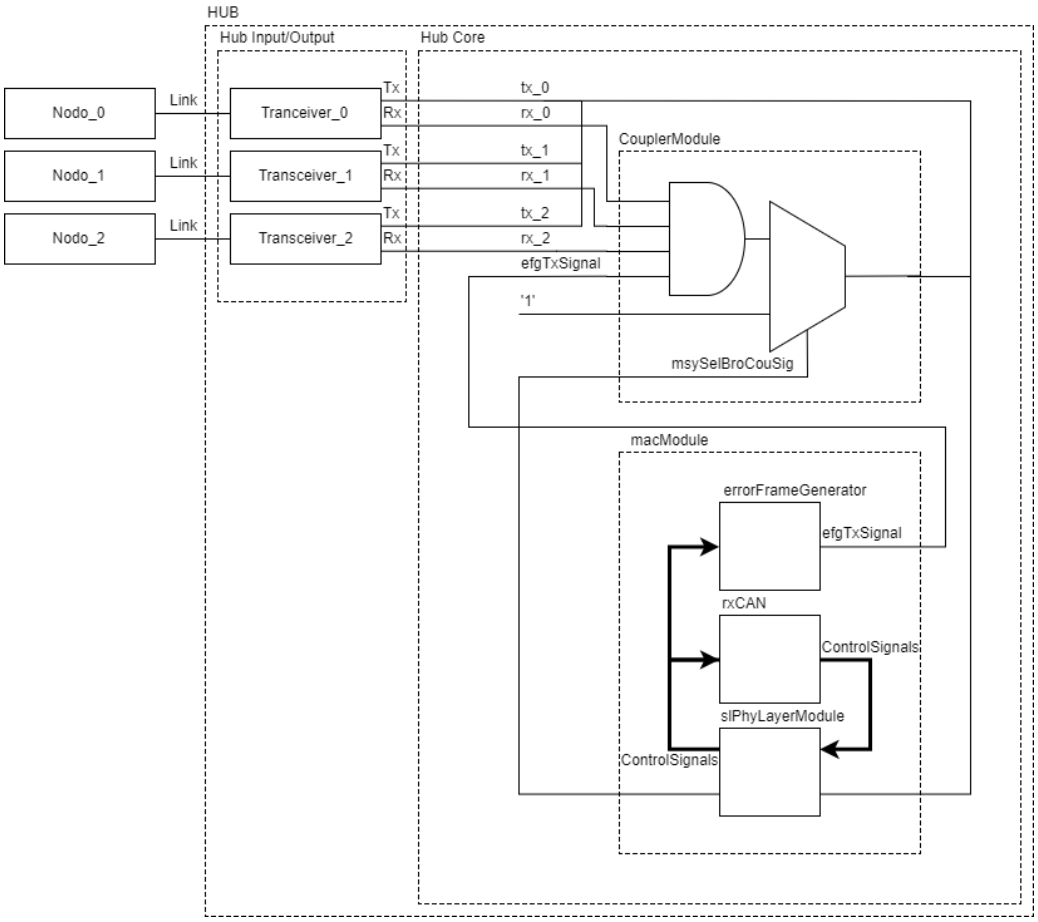

Figura 2.15: Esquema de los módulos internos del hub de HDMS-CAN

2.1.1). Esto es necesario para que el hub se resincronice correctamente a nivel de bit con el nodo transmisor (*leading transmitter*).

2.3.4 Estructura interna del Hub Core

Para que el hub pueda realizar su cometido, internamente incluye diferentes módulos con funcionalidades distintas. A continuación se presentarán los más importantes. Además de nombrar los módulos y explicar su principal funcionalidad, se hará una descripción simplificada de la estructura interna de éstos. En la Figura 2.15 se muestra un diagrama con la estructura interna del hub.

- *CouplerModule*. Este módulo es llamado acoplador puesto que une con una AND las señales recibidas. Por tanto, su principal función es realizar la AND de las transmisiones de los nodos recibidas a través de las entradas rx_X, donde X indica el número de la entrada, las cuales están conectadas a los pines Rx de los *transceivers* del hub para imitar la propiedad *wired-AND* del bus CAN. Además de éstas, en la AND también se opera la señal correspondiente a la trama de error

(*efgTxSignal*) vista en la Sección 2.1.2. Ésta se utiliza para señalizar y globalizar un error, tal y como se vio en dicho apartado, en caso de que el hub detecte un error en la trama. Por otro lado, este módulo acoplador incluye un multiplexor. Éste presenta como entradas el resultado de la AND, anteriormente mencionada, y la señal de enmascaramiento (*msyCouSig*) la cual vale normalmente recesivo ('1'). Como bit de selección se encuentra la señal *msySelBroCouSig*. Ésta vale normalmente '0' para que el hub transmita el resultado de la AND mientras que en los enmascaramientos vale '1' para transmitir el recesivo, como ya se ha mencionado en apartados anteriores. Por último, la salida del multiplexor se divide en las señales tx_X, donde X indica el número de salida, las cuales están conectadas a los pines Tx de los *transceivers* del hub.

- *macModule*. Este módulo contiene el controlador de HDMS-CAN del hub. En este módulo se encuentran los módulos *errorFrameGenerator*, *rxCAN* y *slPhyLayerModule*, los cuales se explicarán a continuación.

  - *slPhyLayerModule*. Este módulo usa una serie de contadores para determinar en que segmento del tiempo de bit se encuentra el hub. Estos contadores pueden modificar sus valores para alargar o acortar los segmentos del tiempo de bit para resincronizar los bits del hub con los del transmisor tal y como se explicó en la Sección 2.1.1. Además de implementar los mecanismos que son necesarios para que el hub se sincronice con el *leading transmitter* a nivel de bit, este módulo es el encargado de generar un conjunto de señales de sincronismo que son necesarios para los otros módulos: valor sampleado de la señal resultante en el bit actual, valor discriminado de la contribución de cada nodo para el bit actual, reloj de transmisión, reloj de recepción (cuando hay flanco ascendente indica a los otros módulos que pueden consultar los valores sampleados por él, el *slPhyLayerModule*).
  - *rxCAN*. Este módulo interno del *macModule* es el encargado, a grandes rasgos, de determinar en que parte de la trama se encuentra la transmisión. Para ello tiene un módulo que detecta los bits de *stuff* y otro con una serie de contadores que determinan el campo de la trama en la que se encuentra la transmisión. Para entender el funcionamiento pondremos un ejemplo. Cuando comienza el campo de arbitraje un contador se incrementa en 1 cada bit, siempre que no sean bits de *stuff*. Cuando este contador haya contado 11 bits, tamaño del identificador, el módulo sabrá que la trama ya no se encuentra en la sección identificador del campo de arbitraje, sino en el bit RTR.
  - *errorFrameGenerator*. Este módulo tiene unas señales de entrada provenientes de los módulos *rxCAN* y *slPhyLayerModule* y una señal de salida llamada *efgTxSignal*. A partir de sus señales de entrada determina si debe generar o no una trama de error la cual se envía a través de *efgTxSignal* al *CouplerModule* para ser transmitida tal y como ya se ha explicado. Es importante aclarar que, para asegurar que el hub no se desincroniza con el resto de nodos y que no propaga señales con errores, éste necesita señalizar y globalizar de forma activa cualquier error CAN que detecte sobre la trama

resultante (la trama que resulta de acoplar mediante la AND las contribuciones de los nodos) en curso. Esto es así porque la globalización fuerza a todos los nodos y al hub a resincronizarse y a mantener el *reliable broadcast* de la red en caso de que se produzca un error, de cualquier tipo, en la red. El módulo encargado de este proceso (de detectar los errores CAN en la trama resultante), y quien por tanto decide cuando hay que señalizar el error, es *rxCAN*. Por tanto, este módulo, el *errorFrameGenerator*, simplemente se encarga de inyectar a través del *couplerModule* (mediante *efgTxSignal*) la trama de señalización de errores demandada por el módulo *rxCAN*.

CAPÍTULO 3

CARACTERIZACIÓN DE HDMS-CAN

Después de haber explicado los conceptos necesarios para entender el trabajo realizado en este TFG, en el presente capítulo expondremos las primeras contribuciones de este proyecto; a saber: analizar cómo deben dimensionarse los segmentos en que se estructura el tiempo de bit desde el punto de vista del hub, identificar las restricciones topológicas de la red HDMS-CAN y comparar la distancia máxima entre dos nodos y la velocidad máxima de transmisión alcanzables entre HDMS-CAN y CAN.

Más concretamente, por una parte, en este capítulo calculamos qué tamaño debe tener cada uno de los segmentos del tiempo de bit, a fin de que la red funcione correctamente. En este sentido, es necesario notar que consideraremos que la red funciona correctamente cuando cumpla dos requisitos. El primero de ellos es que los nodos se comuniquen sin errores adicionales en el canal, más allá de aquellos que se puedan producir por causas ajenas a la red, tales como interferencias electromagnéticas o deterioro de los cables. El segundo de los requisitos para considerar que la red funciona correctamente consiste en que el hub sea capaz de discernir correctamente la contribución de cada nodo bit a bit. Nos referiremos al primero de estos requisitos como *funcionamiento sin errores de comunicación*, mientras que nos referiremos al segundo de ellos como *funcionamiento sin errores de discriminación*.

Por otra parte, en este capítulo se identificarán las restricciones que la red HDMS-CAN debe cumplir desde el punto vista topológico; es decir, qué restricciones existen en la relación entre las longitudes de los diferentes links de la estrella.

Finalmente, se propondrán ecuaciones para poder predecir el rendimiento de la estrella, en términos de cuál es la relación entre el radio máximo de la misma (cobertura del hub) que se puede alcanzar para una velocidad de transmisión dada (y viceversa). Así mismo, se utilizarán estas ecuaciones para comparar cuantitativamente y de forma teórica el rendimiento de HDMS-CAN respecto a CAN.

3.1 Dimensionamiento de los segmentos del tiempo de bit

En esta sección se explicará cómo hay que dimensionar los segmentos del tiempo de bit en HDMS-CAN. Es decir, se explicará qué tamaño debe tener cada uno de los segmentos del tiempo de bit, desde el punto de vista del hub, para que no haya ningún error de comunicación ni de discriminación.

Como se ha visto en la Sección 2.3.3, los segmentos que constituyen el tiempo de bit desde el punto de vista del hub son SyncSeg, PreMask, Mask, RestabSeg y Pha2Seg.

De todos estos segmentos los más fáciles de dimensionar son SyncSeg y Pha2Seg: es suficiente con asignarles la misma longitud que tienen desde el punto de vista de los nodos. Esto es así porque el hub se comporta, en parte, como cualquier nodo receptor. Más concretamente, el hub debe resincronizarse con el nodo *leading transmitter* y debe samplear el resultado de acoplar las contribuciones de todos los nodos (AND), a fin de poder interpretar qué bit se está transmitiendo en cada momento. En este sentido, al igual que cualquier nodo receptor, el hub utiliza el segmento SyncSeg simplemente para comprobar que es precisamente en ese segmento donde observa los flancos de inicio de los bits transmitidos por el nodo *leading transmitter* y, por tanto, que está sincronizado con éste. Por otro lado, como cualquier nodo receptor, el hub samplea al inicio de Pha2Seg el resultado de acoplar las contribuciones de todos los nodos. Como se ha dicho anteriormente, en CAN el segmento Pha2Seg se dimensiona para poder compensar las posibles desincronizaciones de los nodos a nivel de bit, de tal forma que un nodo receptor pueda samplear correctamente aún cuando esté adelantado con respecto al *leading transmitter*. Por tanto, el hub debe seguir el mismo criterio que en CAN a la hora de dimensionar Pha2Seg.

Otro segmento que es fácil de dimensionar es RestabSeg. En principio, simplemente tiene que ser lo suficientemente grande como para incluir dentro de sí dos intervalos de tiempo. En primer lugar, RestabSeg debe incluir la duración de Pha1Seg, que es el segmento que los nodos utilizan después del segmento de propagación (PropSeg) para asegurarse de que samplean correctamente el resultado de la AND, aún cuando están retrasados respecto al *leading transmitter*. En segundo lugar, debe incluir el tiempo necesario para que el resultado de la AND se propague de nuevo, desde el hub hasta el nodo más alejado, una vez que acaba el pulso de enmascaramiento de discriminación (segmento Mask).

Dimensionar el resto de segmentos es algo más complejo. El principal problema a la hora de dimensionarlos es, precisamente, determinar la longitud y la ubicación del pulso de enmascaramiento de discriminación (segmento Mask).

Por una parte, la longitud o duración de este pulso debe ser lo suficientemente grande como para que, al aplicarlo sobre un link cualquiera, el hub pueda monitorizar cuál es la contribución del nodo conectado a ese link. Notar que la duración del pulso debe ser lo suficientemente grande como para que el valor recesivo de ese pulso atraviese el *transceiver* desde el hub hacia el link, se acople con la contribución real del nodo, y el resultado de dicho acoplamiento atraviese el *transceiver* desde el link hasta el hub. Además, comprobamos que el principal limitante de la longitud de este segmento es el autómata, implementado en el hub, encargado de monitorizar el segmento del tiempo de bit en que se encuentra éste (el hub). Esto se debe a que dicho autómata requiere de un tiempo para pasar al segmento Mask y otro para realizar la discriminación.

Esto implica que la duración necesaria del pulso de enmascaramiento de discri-

minación depende de las características eléctricas del *transceiver*, del link y de las limitaciones temporales del autómata. Por este motivo, en este TFG lo primero que se hizo para dimensionar los segmentos del tiempo de bit fue determinar, experimentalmente, la duración necesaria del pulso de enmascaramiento. Hasta el Capítulo 4 no se explicará en detalle el prototipo de HDMS-CAN que se ha construido en este TFG. Sin embargo, en este punto basta decir que se hicieron varias pruebas para determinar la duración necesaria del pulso de enmascaramiento de discriminación. En esas pruebas se comprobó que, utilizando los transceivers, el tipo de cables escogidos para construir el prototipo y el autómata del prototipo original; esta duración debía ser de, al menos, 2 Time Quantum (TQ); siendo en nuestro prototipo la duración del TQ igual a 62,5 ns. Aclarar que los TQ son las unidades mínimas de tiempo en que se dividen los diferentes segmentos del tiempo de bit. Éstos proceden, en el prototipo de este proyecto, de un reloj con una frecuencia de 16 MHz.

Es más, se comprobó experimentalmente que esta duración es independiente de la longitud del link. Este último resultado es más importante de lo que parece a simple vista; ya que una de las dudas que se tenían antes de la realización de este TFG era si la duración del pulso de enmascaramiento de discriminación dependía de la longitud del link. El hecho de comprobar que no es así es un resultado muy positivo; ya que de lo contrario, la duración mínima del pulso debería aumentar con la longitud de los links. Esto no solo complicaría el cálculo de la duración del pulso en sí, sino que tendría un impacto negativo sobre la velocidad de transmisión; puesto que aumentar la duración del pulso implica aumentar la duración del tiempo de bit.

Por otra parte, respecto a la ubicación del pulso de enmascaramiento de discriminación, lo primero que hay que tener en cuenta es que éste tiene que comenzar cuando el hub esté seguro de que la contribución de cada nodo para el bit presente haya llegado desde cada nodo hasta el hub. En caso contrario, el hub no podría samplear correctamente la contribución de aquellos nodos cuya contribución no lo hubiese alcanzado. Para entender mejor dónde hay que ubicar el pulso de enmascaramiento de discriminación (segmento Mask), la Figura 3.1 muestra cómo la contribución de un nodo que actúa como *leading transmitter* se propaga, atravesando el hub, hasta un nodo que actúa como receptor. También muestra cómo, acto seguido, la contribución de ese nodo receptor se propaga hasta el hub. Como se puede ver, la duración del segmento PreMask tiene que ser lo suficientemente grande como para que el segmento Mask comience después de que la contribución del nodo receptor haya llegado hasta el hub.

De hecho, para decidir cuál es la duración de PreMask (y por tanto el inicio de Mask) es necesario tener en cuenta cuál es el par de nodos más alejados entre sí; a fin de que cuando uno de ellos actúe como *leading transmitter*, el hub pueda estar seguro de que ha recibido la contribución del otro nodo (que jugará el papel de receptor) antes de Mask.

Es más, la Figura 3.1 también muestra que el tamaño de PreMask tiene que ser estrictamente mayor que el tiempo necesario para que la contribución del nodo receptor llegue hasta el hub. Esto es así porque la Figura muestra un escenario en el que el nodo receptor está perfectamente sincronizado con el nodo *leading transmitter*. Sin embargo, como hemos visto, en CAN los nodos pueden desincronizarse los unos de los otros hasta que se vuelven a resincronizar. También hemos visto que el tamaño de este retraso está acotado y su valor es SJW tal y como se vio en el apartado 2.1.1. Por tanto,

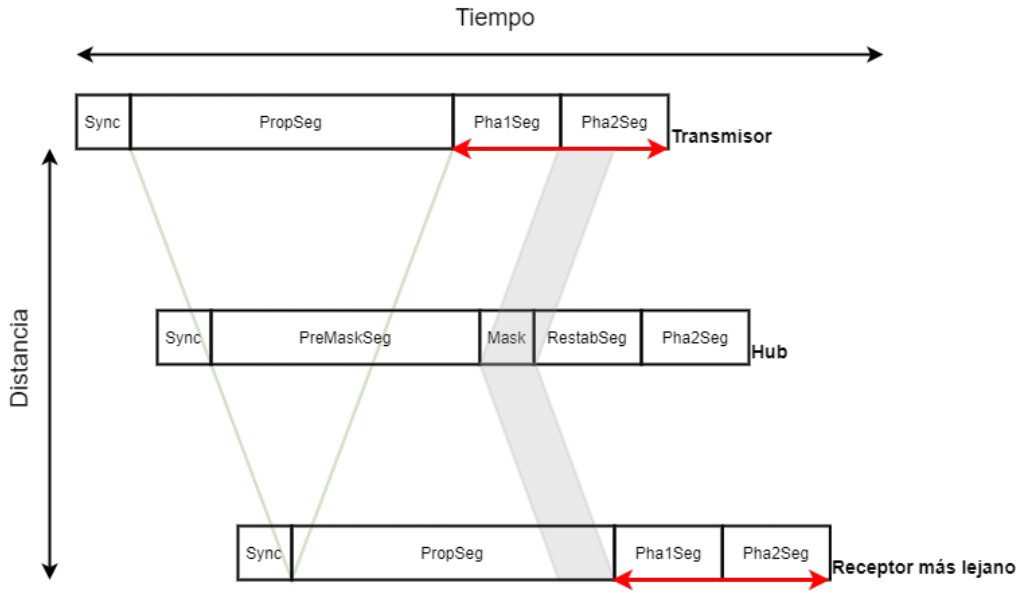

Figura 3.1: Esquema de transmisión entre dos nodos con la sombra del enmascaramiento situada detrás del sampleo de los nodos transmisores. La zona alrededor del sampleo está marcada en rojo puesto que las desincronizaciones que admite CAN podrían provocar que cualquier enmascaramiento situado en esta zona acabara cayendo sobre el punto de sampleo

PreMask tiene que ser lo suficientemente grande como para que la contribución del nodo receptor llegue al hub antes de Mask incluso cuando este nodo sufra el máximo retraso posible respecto al *leading transmitter*.

El segundo aspecto que hay que tener en cuenta respecto a la ubicación del pulso de enmascaramiento de discriminación (segmento Mask) es que, cuando el hub transmite dicho pulso, éste se propaga a cada uno de los nodos, provocando una *sombra* en la visión que cada nodo tiene sobre el valor de bit que hay en su link correspondiente. Es decir, cuando el pulso alcanza a un nodo impide que ese nodo vea en su link, durante el tiempo que dura el pulso, el resultado de acoplar las contribuciones de los nodos mediante la AND (que internamente se realiza en el hub). Este hecho plantea dos problemas.

El primer problema relacionado con la sombra de Mask es que ésta puede impedir que los nodos sampleen correctamente el resultado de la AND, en sus puntos de sampleo correspondientes (al inicio de su segmento Pha2Seg).

Una primera solución para este problema se muestra en la Figura 3.1 previamente mostrada. Dicha solución consiste en dimensionar los segmentos de tal forma que la sombra de Mask caiga después del punto donde el nodo *leading transmitter* samplea. Notar que en el caso de cualquier nodo receptor, la sombra siempre caerá delante del punto de sampleo de dicho nodo receptor. Esto se debe a que el tamaño del segmento Pha2Seg es igual en el hub y en los nodos; por lo que si Mask está situado antes de Pha2Seg en el hub, el nodo receptor observará la sombra de Mask antes de su propio Pha2Seg.

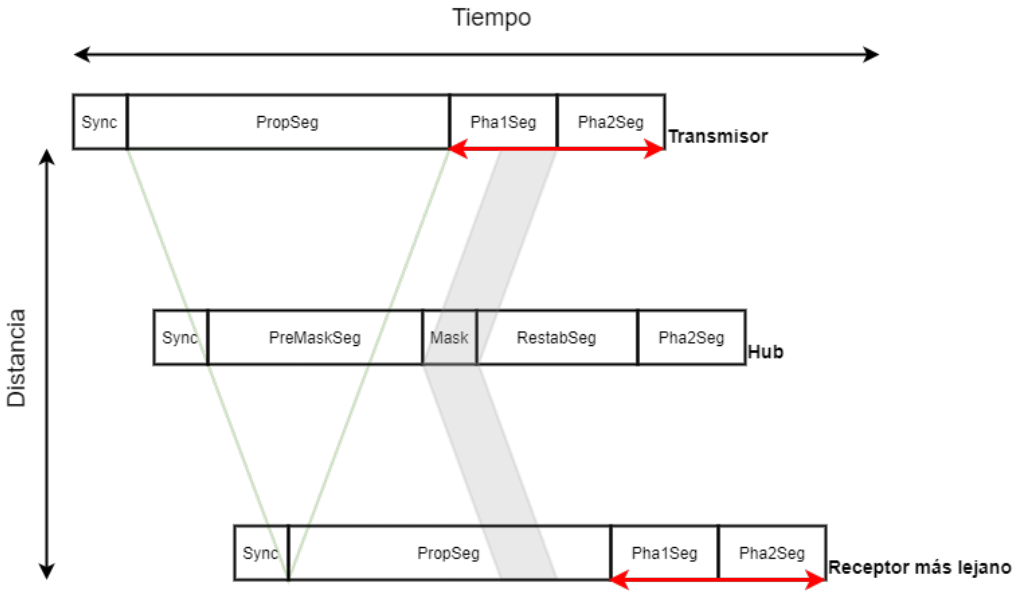

Figura 3.2: Esquema de transmisión entre dos nodos con el Mask situado de tal forma que la sombra del enmascaramiento caiga delante del punto de sampleo del transmisor. La zona alrededor del sampleo está marcada en rojo puesto que las desincronizaciónes que admite CAN podrían provocar que cualquier sombra de enmascaramiento situada en esta zona acabara cayendo sobre el punto de sampleo

Aunque esta solución podría ser viable, presenta un inconveniente importante desde el punto de vista topológico. Concretamente, notar que para asegurar que la sombra caiga después del punto de sampleo de un nodo *leading transmitter*, dicho nodo tiene que estar lo suficientemente alejado del hub. En este sentido, como cualquier nodo puede actuar en un momento dado como *leading transmitter*, esto implica que todos los nodos deben situarse lo suficientemente alejados del hub. O dicho de otra forma, en principio esta solución no permite ubicar nodos cerca del hub por debajo de una cierta distancia. Consideramos que esta restricción topológica no es deseable, ya que o bien limita la flexibilidad de la topología (impidiendo ubicar nodos cerca del hub), o bien obligaría a utilizar links largos para conectar nodos que físicamente están ubicados cerca del hub (lo que conllevaría un gasto innecesario de cableado). Por tanto, en este TFG proponemos descartar dicha solución.

Una segunda solución para que los nodos puedan samplear correctamente el resultado de la AND se muestra en la Figura 3.2. En este caso la idea es dimensionar los segmentos de tal forma que la sombra de Mask caiga antes del punto donde el nodo *leading transmitter* samplea. De nuevo, como en la solución anterior, la sombra siempre caerá delante del punto de sampleo de los nodos receptores.

Al contrario que la solución anterior, esta segunda solución no impide ubicar los nodos cerca del hub (o conectarlos con cables cortos). De hecho, cuanto más cerca esté un nodo del hub, la sombra de Mask caerá con más antelación respecto del punto donde el nodo samplea; lo cuál es deseable. Ahora bien, esta segunda solución sí puede limitar la distancia máxima a la que se puede ubicar un nodo del hub (limita

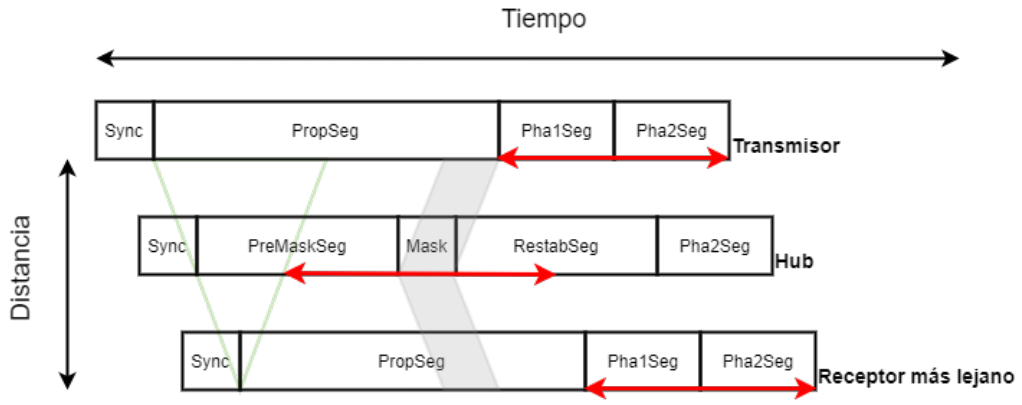

Figura 3.3: Esquema de transmisión entre dos nodos con el segmento Mask situado de tal forma que la sombra caiga antes del punto de sampleo de los nodos y a una distancia mayor que el máximo valor de desincronización posible en CAN. La zona alrededor del sampleo está marcada con una flecha en rojo puesto que las desincronizaciónes que admite CAN podrían provocar que cualquier sombra situada en esta zona acabara cayendo sobre el punto de sampleo. Además, los alrededores del enmascaramiento de discriminación también están marcados puesto que la contribución de los receptores a discriminar debe caer también a una distancia mayor que la máxima posible desincronización de CAN o estas podrían no ser bien discriminadas

la longitud máxima del link). Aun así, consideramos que ésta es la solución adecuada desde el punto de vista topológico; ya que si bien limita la longitud máxima del link, no resta flexibilidad a la hora de configurar la topología, ni obliga a utilizar cables largos innecesariamente.

De todas formas, como se ha dicho anteriormente, los nodos pueden desincronizarse los unos de los otros (y con el hub). En particular, si un nodo se adelanta respecto al hub, puede ocurrir que la sombra de Mask caiga sobre el punto de sampleo de ese nodo. Por tanto, es fundamental que al adoptar esta segunda solución, los segmentos del tiempo de bit se dimensionen de tal forma (acortando PreMask o RestabSeg) que la sombra de Mask caiga antes del inicio de Pha2Seg incluso en aquellos nodos que sufran el máximo adelanto posible (al estar desincronizados) con respecto al hub. Finalmente, teniendo en cuenta la posición de la sombra antes del sampleo y todas las distancias de seguridad para evitar problemas de comunicación y discriminación en caso de desincronización, se desarrolló una configuración de los segmentos del tiempo de bit la cual se puede apreciar en la Figura 3.3 y cuya especificación matemática se verá en el apartado siguiente, Sección 3.2.

El segundo problema relacionado con la sombra de Mask es que ésta puede provocar problemas de resincronización en los nodos receptores respecto al *leading transmitter*. Concretamente, estos problemas de resincronización pueden ocurrir sólo cuando el bit actual es dominante; ya que si es recesivo la sombra de Mask no altera el valor recesivo que los nodos observan en sus links. Para entender mejor en qué consisten estos problemas de resincronización cuando el bit actual es dominante, conviene distinguir dos escenarios.

El primero de estos dos escenarios se da cuando el bit anterior fué recesivo y el bit actual es dominante. En este caso, si el nodo está retrasado respecto al *leading transmitter*, la sombra de Mask puede impedir que el nodo detecte el flanco descendente del bit dominante actual en el momento en que realmente el flanco llegaría al nodo si no estuviese enmascarado por Mask. Para prevenir que este problema ocurra, es necesario dimensionar PreMaskSeg para que sea lo suficientemente grande como para que cualquier nodo receptor sea capaz de observar correctamente el flanco descendente de cualquier bit dominante enviado por el *leading transmitter*, aún cuando el nodo receptor esté retrasado con respecto a éste.

El segundo escenario ocurre cuando el bit anterior fué dominante y el bit actual es dominante. En este caso cabe recordar (Sección 2.3.2) que el hub no sólo transmite el pulso de enmascaramiento de discriminación durante Mask; sino que durante PreMask también transmite el pulso de enmascaramiento para prevenir el *infinite loop-back*. Por tanto, en este escenario el inicio de la sombra que proyectan conjuntamente PreMask y Mask provoca que los nodos receptores detecten un flanco ascendente (que en CAN no existiría) al principio del bit dominante. Este flanco ascendente puede hacer creer a un nodo receptor que debe resincronizarse, alargando su tiempo de bit, cuando en realidad ese flanco pertenece al inicio de la citada sombra. En la patente de HDMS-CAN ya se propuso una solución para prevenir este problema; a saber, configurar los nodos CAN para que no utilicen los flancos ascendentes para resincronizarse. De hecho, en CAN es habitual que los nodos no utilicen estos flancos para llevar a cabo acciones de resincronización. Por tanto, en este TFG no se ha hecho ninguna propuesta alternativa para prevenir este problema de otra forma.

3.2 Ecuaciones para dimensionar los segmentos del tiempo de bit

En la sección anterior se han explicado los aspectos a tener en cuenta para dimensionar los segmentos del tiempo de bit desde el punto de vista del hub. En la presente sección se tienen en cuenta dichos aspectos a fin de proponer una serie de ecuaciones que permitan calcular numéricamente el tamaño de estos segmentos.

Puesto que SyncSeg, Mask y Pha2Seg ya estaban determinados por similitudes con los segmentos del estándar de CAN o por cálculos experimentales, únicamente quedaba determinar el tamaño de PreMask y RestabSeg.

Como el tamaño de PreMask es el que determinará la posición del enmascaramiento determinaremos matemáticamente su tamaño, mientras que RestabSeg lo definiremos como el tiempo de bit restante tal y como se aprecia en la ecuación 3.4.

Comenzamos la determinación matemática del segmento PreMask con la Figura 3.3, donde se observa como el tiempo que tarda la transmisión en ir y volver forma un triángulo cuya base es PropSeg menos la suma de MaskSeg y 2*SJW, que es la *distancia de seguridad* ya que, como ya se ha mencionado con anterioridad, esta es la máxima distancia de desincronización según el estándar de CAN. A continuación calculamos que el tiempo entre el final del SyncSeg del hub y la recepción de la respuesta del receptor en el hub vale lo mismo que la mitad de la base del triángulo anteriormente mencionado. Finalmente, puesto que la distancia entre la recepción de la respuesta del receptor y el enmascaramiento de discriminación (Mask) ha de ser 2*SJW solo

queda sumar dicho valor al último resultado obtenido, consiguiendo así la fórmula 3.3 utilizada para calcular el tamaño del segmento PreMask.

A continuación se muestran las ecuaciones 3.1, 3.2, 3.3 y 3.4 utilizadas para determinar el tamaño de cada segmento del tiempo de bit. Estas son el resultado obtenido mediante el análisis geométrico de los esquemas de transmisión explicados anteriormente.

$$SyncSeg_h = SyncSeg_e \tag{3.1}$$

$$Pha2Seg_h = Pha2Seg_e \tag{3.2}$$

$$\frac{PropSeg_e - MaskSeg_h - 2 * SJW}{2} = PreMaskSeg_h - 2 * SJW \tag{3.3}$$

$$RestabSeg_h = TB_e - (SyncSeg_h + PreMaskSeg_h + MaskSeg_h + Pha2Seg_h) \tag{3.4}$$

Donde:

SJW = Tiempo máximo de desincronización

TB = Tiempo de bit

Subíndice h = Pertenece al hub

Subíndice e = Pertenece al estándar de CAN

3.3 Restricciones topológicas de HDMS-CAN

Gracias a este dimensionamiento de los segmentos del tiempo de bit existe una distancia máxima hasta la cual los nodos pueden comunicarse mediante el protocolo CAN sin problemas, a la vez que el hub puede discriminar correctamente la contribución de cada nodo. Además, este proporciona, teóricamente, el mayor rango de distancias a la que se puede situar un nodo del hub.

No obstante, en apartados anteriores ya se habló de algunas restricciones topologicas de HDMS-CAN. Además, tras determinar finalmente la estructura de los segmentos del tiempo de bit, pudimos comprender algunas caracteríticas y restricciones de HDMS-CAN adicionales.

Entre ellas pudimos observar que, a diferencia de CAN donde la distancia máxima de comunicación viene dada por la distancia máxima a la que se pueden situar dos nodos, en HDMS-CAN el tamaño de la estrella viene definido por la distancia máxima a la que un nodo puede situarse del hub. A esta distancia máxima se la ha llamado radio máximo o cobertura del hub de HDMS-CAN. En este sentido el diámetro máximo de la estrella solo se conseguirá cuando el hub esté en medio de dos o más nodos situados a la máxima distancia de él. Ver Figura 3.4.

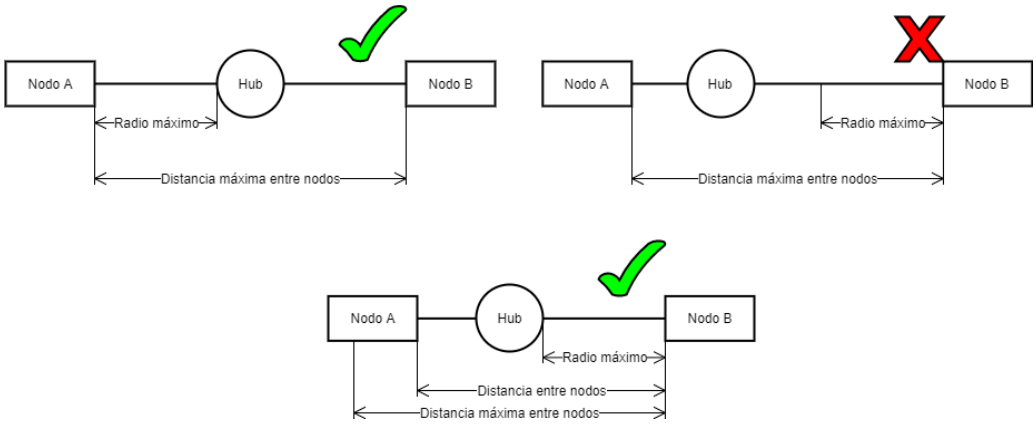

Figura 3.4: Esquema de diferentes topologías

3.4 Cuantificación teórica del radio y el bit rate máximos alcanzables en HDMS-CAN

Una vez determinado el dimensionamiento de los segmentos del tiempo de bit y después de analizar las características y restricciones de HDMS-CAN procederemos a cuantificar estas últimas. En otras palabras, intentaremos calcular el impacto real de dichas restricciones. Para ello comenzaremos creando ecuaciones que nos permitan determinar el radio máximo de la estrella de HDMS-CAN a partir de comparar esta topología con un bus CAN estándar. Finalmente calcularemos la cobertura del hub mediante las ecuaciones obtenidas y analizaremos los resultados.

Aunque existen diferencias entre la longitud máxima de un bus CAN estándar y el diámetro máximo de estrella que se puede alcanzar con esta topología, analizando los esquemas de transmisión (Figura 3.3) podemos crear ecuaciones que nos permitan predecir el diámetro máximo alcanzable por la estrella de HDMS-CAN.

Suponiendo que la distancia máxima entre nodos es proporcional al tiempo de PropSeg del estándar de CAN podemos predecir que el diámetro máximo de la estrella será un porcentaje de la distancia máxima del estándar. Dicho porcentaje se calcula dividiendo el tiempo utilizado por el hub para la transmisión entre el tiempo utilizado en CAN estándar para la transmisión. A continuación se muestran las ecuaciones 3.5 y 3.6 utilizadas para predecir los diámetros máximos de la estrella de HDMS-CAN en función del tamaño de los segmentos del tiempo de bit, es decir, de la velocidad de transmisión.

$$TT = \frac{PropSeg_e - MaskSeg_h - 2 * SJW}{PropSeg_e} \tag{3.5}$$

$$DM_h = DM_e * TT \tag{3.6}$$

Donde:

TT = Porcentaje de tiempo de transmisión

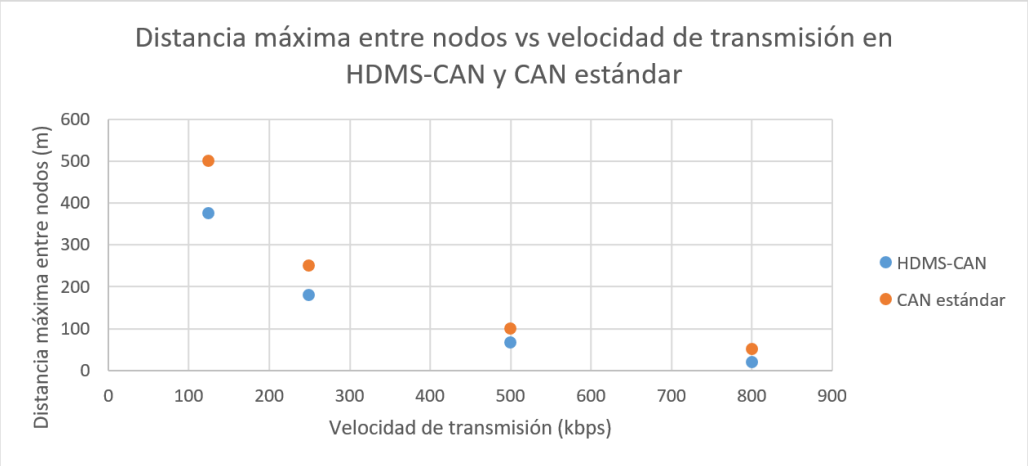

Figura 3.5: Gráfica que muestra la distancia máxima entre nodos para distintas velocidades de transmisión en HDMS-CAN y CAN

SJW = Tiempo máximo de desincronización

DM = Diametro máximo

Subíndice h = Pertenece al hub

Subíndice e = Pertenece al estándar de CAN

Una vez que podíamos calcular el diámetro máximo de la estrella, el cual se muestra en la Figura 3.5, solo necesitábamos dividir el resultado entre 2 para obtener el radio máximo alcanzable, o lo que es lo mismo, la cobertura del hub.

En la gráfica de la Figura 3.6 se muestra el radio máximo al que se puede situar un nodo del hub en función de la velocidad de transmisión. Se puede observar como la cobertura disminuye a medida que aumenta la velocidad de transmisión. Esto se debe a que la velocidad de transmisión y el tamaño de los bits en CAN presentan una relación inversamente proporcional; mientras que el tamaño de los bits y la distancia presentan una relación directamente proporcional, como ya se ha mencionado anteriormente.

Por otro lado, en la gráfica de la Figura 3.7 se puede apreciar como el impacto del enmascaramiento es mayor a medida que aumenta la velocidad de transmisión, lo cual se ve representado en una disminución del porcentaje de tiempo de transmisión útil para la propagación de las señales. Esto se debe a que el tiempo de transmisión perdido en HDMS-CAN, debido al enmascaramiento, se mantiene más o menos constante mientras que el tiempo de bit disminuye con el aumento de la velocidad de transmisión. En la gráfica se ve como el rendimiento de HDMS-CAN respecto al estándar de CAN es del 70 % para una velocidad de transmisión de 125 kbps y como este rendimiento se ve reducido a medida que aumenta la velocidad de transmisión hasta alcanzar un rendimiento del 40 % para una velocidad de transmisión igual a 800 kbps.

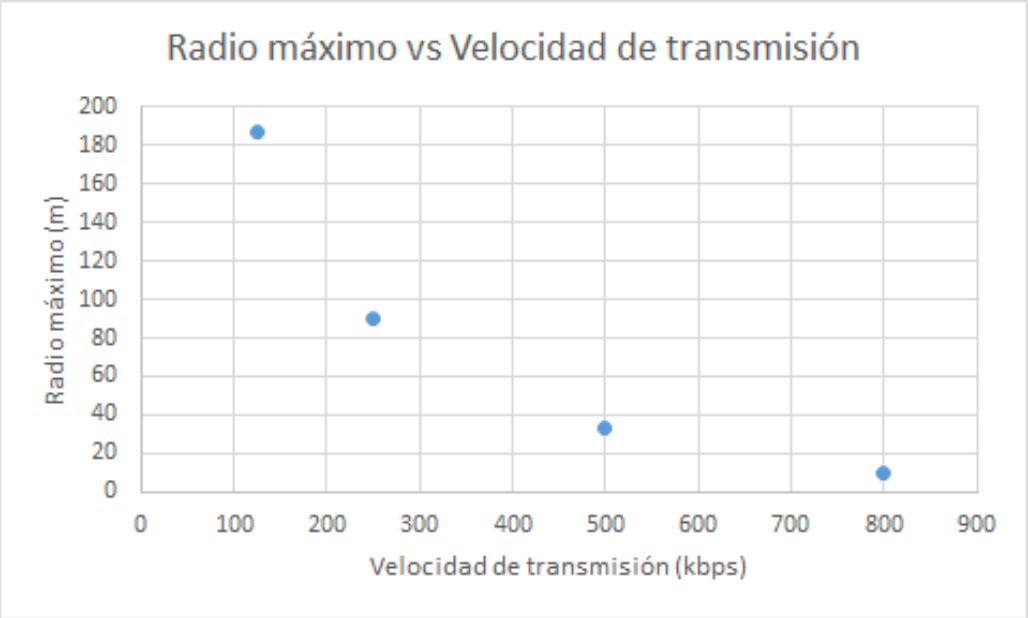

Figura 3.6: Gráfica que muestra la cobertura del hub para distintas velocidades de transmisión

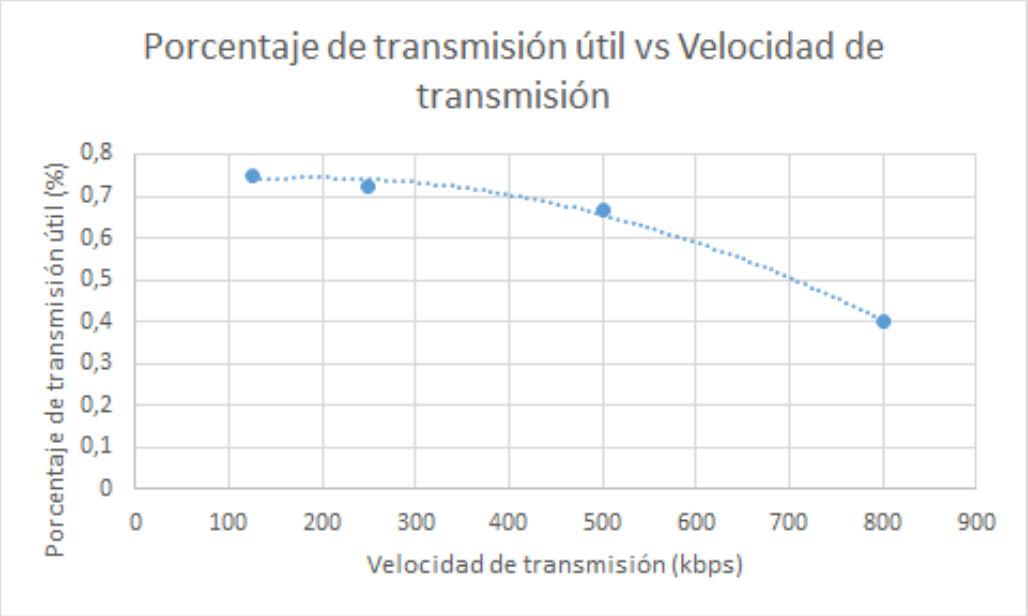

Figura 3.7: Gráfica que muestra el porcentaje de transmisión útil en HDMS-CAN, es decir, el porcentaje de distancia máxima del estándar de CAN que HDMS-CAN admite, en función de la velocidad de transmisión

CAPÍTULO 4

PROTOTIPADO

En este capítulo se describe el segundo conjunto de contribuciones más relevantes de este TFG. Una de estas contribuciones es la puesta en marcha de un nuevo prototipo de HDMS-CAN que sea *totalmente operativo*.

Para entender a lo que nos referimos por prototipo *totalmente operativo* hay que recordar que, como se ha dicho anteriormente, ya existía un prototipo anterior de HDMS-CAN. Sin embargo, el objetivo de ese prototipo era servir de prueba de concepto a fin de poder patentar HDMS-CAN. Por esa razón, por una parte, ese prototipo se utilizó para demostrar que el hub podía monitorizar las contribuciones de cada nodo teniendo en cuenta un único *bit rate* y links cortos (de unos 20 cm). Por otro lado, no se realizaron pruebas exhaustivas con ese prototipo a fin de depurar posibles fallos de diseño y/o implementación.

En este TFG, por el contrario, se ha implementado un prototipo que funciona correctamente para diferentes bit rates y longitudes de los links, y sobre el que se han realizado pruebas para detectar y solucionar fallos tanto en el diseño de la propuesta original de HDMS-CAN, como en la implementación del prototipo utilizado para la prueba de concepto.

Es importante destacar que la detección y solución de los fallos de diseño e implementación son contribuciones importantes del presente TFG; ya que los cambios propuestos en este sentido son indispensables para que una red HDMS-CAN funcione correctamente.

También es importante destacar que la obtención de este nuevo prototipo totalmente operativo es lo que nos ha permitido caracterizar experimentalmente, por primera vez, el rendimiento de HDMS-CAN. Los detalles de esta caracterización se presentarán, sin embargo, en el Capítulo 5.

El presente capítulo se organiza de la siguiente forma. En primer lugar se muestra la implementación de lo que nosotros llamamos la *red de pruebas*. Esta red incluye el nuevo prototipo de red HDMS-CAN en sí, así como otros elementos necesarios para poder monitorizar su actividad a fin de depurar su funcionamiento. En segundo y tercer lugar se explican los problemas detectados en el diseño de HDMS-CAN y

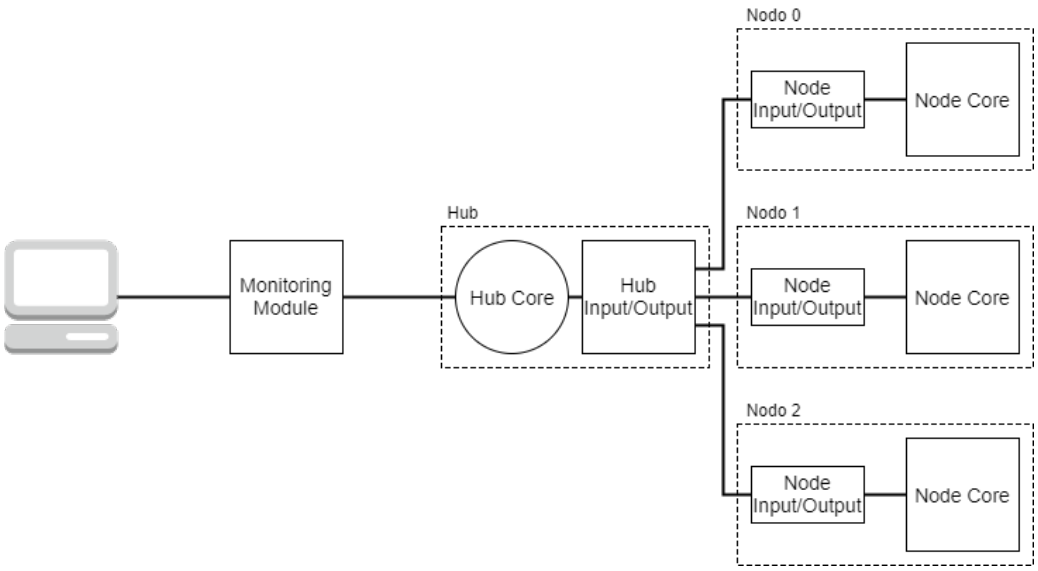

Figura 4.1: Arquitectura general de la red de pruebas implementada

en la implementación del prototipo original respectivamente, así como los cambios propuestos para solucionarlos.

4.1 Red de pruebas

Como se acaba de explicar arriba, entendemos por red de pruebas al montaje que incluye, por una parte, el nuevo prototipo de HDMS-CAN en sí y, por otro lado, una serie de elementos necesarios para monitorizar su funcionamiento. A continuación se explica la arquitectura de esta red de pruebas, se explican brevemente sus características desde el punto de vista del hardware y del software, y se presenta cuál ha sido la estrategia que se ha seguido para depurar (*debug*) el funcionamiento del nuevo prototipo de HDMS-CAN.

4.1.1 Arquitectura de la red de pruebas

La Figura 4.1 muestra la arquitectura de la red de pruebas, mientas que la Figura 4.2 es una foto del aspecto real de dicha red.

Como se puede observar en estas figuras, por una parte, la red de pruebas incluye el nuevo prototipo de HDMS-CAN. Este prototipo está formado por un hub HDMS-CAN, el cual está constituido por lo que nosotros llamamos el *Hub Core* y por el *Hub Input/Output*. Como se ha visto en la Sección 2.3.4, el Hub Core contiene toda la lógica y memoria interna del hub en sí; mientras que el Hub Input/Output está formado por una serie de *transceivers*, uno por link. En particular, el nuevo prototipo incluye 6 *transceivers*, aunque sólo se han utilizado 3 para conectar 3 nodos CAN al hub. La razón por la que se han conectado 3 nodos CAN se explicará más adelante, en la Sección 4.1.3. Notar también que cada nodo CAN está formado por lo que nosotros llamamos *Node*

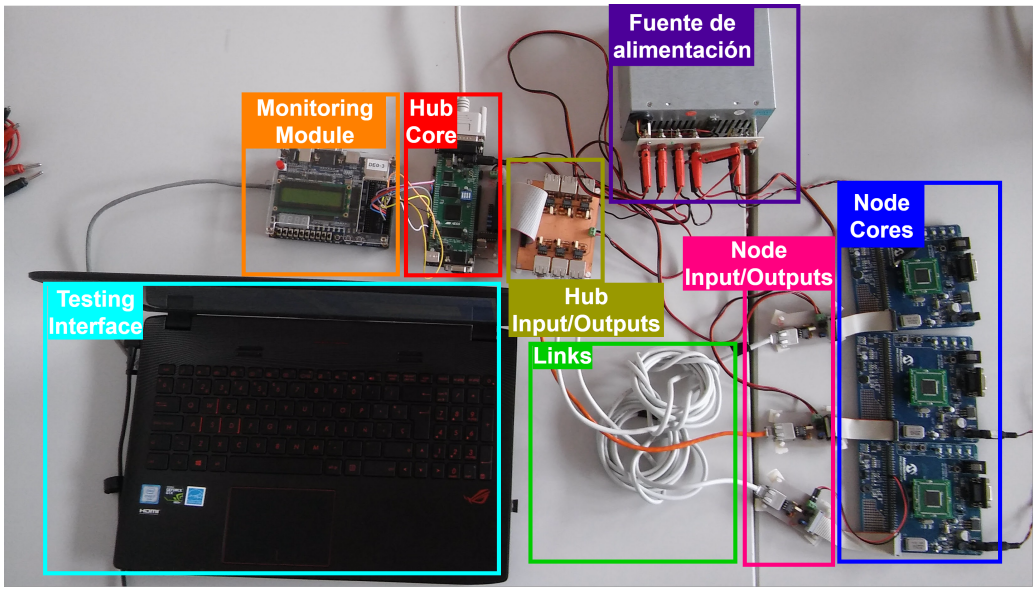

Figura 4.2: Foto del aspecto real de la red de pruebas implementada

Core y *Node Input/Output*. De forma similar a lo que ocurre con el hub, el Node Core engloba la lógica interna y los elementos de memoria del nodo CAN, mientras que el Node Input/Output es básicamente un *transceiver*.

Por otra parte, la Figura 4.1 muestra como la red de pruebas incluye también lo que nosotros llamamos el *Monitoring Module* y la *Testing Interface*. El Monitoring Module es un dispositivo conectado a diferentes pines del Hub Core con el fin de adquirir y registrar el valor lógico de diferentes señales internas que el Hub Core expone en esos pines. Tal y como se verá en detalle en la Sección 4.1.2, el Monitoring Module captura información útil para depurar el prototipo de HDMS-CAN. Por ejemplo, es capaz de registrar el porcentaje de tramas en las que han tenido lugar errores de discriminación; es decir, el porcentaje de tramas en las que el hub no ha discriminado correctamente la contribución de alguno de los nodos. En lo que respecta a la Testing Interface, ésta es un computador (*laptop*) conectado al Monitoring Module mediante Universal Serial Bus (USB), que permite visualizar en una pantalla las señales adquiridas y registradas por dicho módulo.

4.1.2 Hardware de la red de pruebas

A continuación se describe brevemente el hardware que se ha utilizado para implementar los diferentes elementos de la red de pruebas (Figura 4.2).

- *Hub Core*. El Hub Core está implementado en VHSIC Hardware Description Language (VHDL) y está sintetizado en una FPGA modelo Spartan-3 XC3S1000, la cual está integrada en una placa comercial de prototipado llamada XSA-3S1000. Tanto la descripción VHDL del hub, como la placa de prototipado se han heredado del prototipo original de HDMS-CAN. Sin embargo, como se ha comentado y se explicará en detalle más adelante, en el presente TFG se ha modificado el

código VHDL para solventar varios fallos de diseño e implementación presentes en la propuesta e implementación originales de HDMS-CAN.

- *Hub Input/Output*. El Hub Input/Ouput consiste en una placa de circuito impreso que contiene 6 *transceivers* CAN comerciales modelo PCA82C250. Además, presenta 6 conectores RJ45 Cat5 y 6 terminaciones de 120 Ω (un conector y una terminación por cada *transceiver*). Este modelo de *transceiver* es capaz de soportar una comunicación de hasta 1 Mbit/sec. Es importante notar que esta placa de circuito impreso *no* se ha desarrollado en este TFG, sino que se ha aprovechado del prototipo original de HDMS-CAN.

- *Node Core*. Cada nodo CAN se ha implementado utilizando exclusivamente componentes comerciales. Concretamente, el Node Core está formado por una placa de prototipado de la marca Microchip, que básicamente contiene el microprocesador dsPIC30F6014A. A grandes rasgos este microprocesador incluye dos controladores CAN, aunque en este proyecto solo se utilizó uno de ellos. Además, la placa proporciona 80 pines que permiten un fácil acceso a los pines del microcontrolador, así como a los pines de los controladores CAN.

- *Node Input/Output*. Por su parte, cada Node Input/Output consiste en una placa de circuito impreso que contiene un *transceiver* CAN comercial, del mismo modelo que el utilizado en el Hub Input/Output, un conectore RJ45 Cat5 y una terminación de 120 Ω. Al igual que en el caso del Hub Input/Output, esta placa de circuito impreso *no* se ha desarrollado en este TFG, sino que se ha aprovechado del prototipo original de HDMS-CAN.

- *Link*. Cada link está implementado mediante un par trenzado incluido dentro de un cable Ethernet UTP Cat5 sin blindaje. El motivo por el cual se ha decido utiliar cableado y conectores típicos de Ethernet y no de CAN se debe a una cuestión prática; ya que este tipo de componentes típicos de Ethernet son muy fáciles de conseguir y son baratos. De hecho, esta decisión ha sido impuesta por el director del TFG, ya que este tipo de componente fue el que se utilizó para implementar el prototipo original de HDMS-CAN. Como veremos en la Sección 5.2, se han utilizado links de hasta 50 m en las pruebas de rendimiento. Sin embargo, como se explicará más adelante dentro del presente capítulo, para depurar el prototipo de HDMS-CAN se han utilizado links cortos de 20 cm.

- *Monitoring Module*. El Monitoring Module está implementado en VHDL y está sintetizado en una FPGA EP3C16F484C6 de la marca Altera, la cuál está integrada en una placa comercial de prototipado llamada DE0. Esta placa contiene un conector con varios pines de entrada. Algunos de estos pines se conectan, uno a uno y directamente, con pines de salida de la placa donde se ha implementado el Hub Core. A cada una de estas conexiones la llamamos *señal de monitorización*. La implementación VHDL del Monitoring Module procesa estas señales de monitorización para proveer un conjunto de *señales de debug* que permitan visualizar, en la Testing Interface, información relevante sobre el funcionamiento del hub. Las señales de monitorización y debug más importantes se explicarán en la Sección 4.1.4. Es importante notar que la implementación VHDL del Monitoring Module se ha realizado *integramente* en este TFG.

- *Testing Interface*. Como se ha indicado más arriba, la Testing Interface consiste simplemente en un *laptop* conectado al Monitoring Module mediante un cable USB, a través del cual recibe el valor de las señales de debug. Este *laptop* ejecuta un software comercial de Altera, llamado *Signal Tap II*, que permite controlar cuándo y cómo visualizar en la pantalla señales provenientes del Monitoring Module, concretamente las señales de debug.

4.1.3 Software de los nodos

Para entender el software que se ha desarrollado para ser ejecutado en los nodos CAN, primero es necesario recordar que mantener una buena sincronización a nivel de bit es esencial en cualquier red CAN. De lo contrario, los nodos CAN pueden samplear incorrectamente la señal resultante de acoplar las contribuciones de los nodos (AND) en el bus CAN.

Esta necesidad también está presente en HDMS-CAN, ya que tanto los nodos como el hub tienen que ser capaces de samplear correctamente el resultado de acoplar la contribución de los nodos. Además, como se ha explicado anteriormente en la Sección 3.1, el hub de HDMS-CAN debe realizar los enmascaramientos de tal forma que, por una parte, pueda samplear correctamente la contribución de cada nodo (discriminar cada contribución) y, por otra parte, permita que tanto los nodos como el hub puedan resincronizarse con el *leading transmitter* para manterner una sincronización adecuada a nivel de bit.

En este sentido, hay que notar que, tanto en CAN como en HDMS-CAN, la situación en la que es más probable que falle el mecanismo de sincronización a nivel de bit CAN es cuando los dos nodos más alejados entre sí arbitran para ganar el acceso al medio. Por una parte esto se debe a que un nodo CAN que intenta ganar el acceso al medio se considera a sí mismo como *leading transmitter* y, por tanto, no lleva a cabo acciones de resincronización a nivel de bit; a no ser que pierda el arbitraje y, como consecuencia, cambie su rol por el de nodo receptor. Por otra parte, el caso más difícil para conseguir que un nodo que pierde el arbitraje se resincronice con el *leading transmitter* es aquel en el que el nodo que pierde esté desincronizado, respecto al que gana, una cantidad de tiempo ("tamaño de desincronización") igual al tiempo de propagación entre ambos nodos. Esto último sólo puede ocurrir cuando se consideran los dos nodos más alejados entre sí.

Es más, teniendo en cuenta lo que se acaba de decir, el escenario de arbitraje más pesimista desde el punto de vista de la sincronización a nivel de bit es aquel en el que, estando arbitrando los dos nodos más alejados entre sí, uno de ellos pierde el arbitraje en el penúltimo bit del campo *identificador* (ver Figura 2.7). Esto es así porque este escenario es aquel en el que los dos nodos han estado desincronizados el uno con respecto al otro durante más tiempo.

En este punto es donde se puede entender por qué hemos decidido utilizar 3 nodos CAN en nuestro prototipo. Los dos nodos más alejados entre sí de estos tres nodos se programan de tal forma que arbitran para ganar el acceso al medio y, además, para ganar el acceso de forma alterna (en una trama gana uno de ellos, y en la siguiente el otro). Nos referiremos a estos nodos como *nodos transmisores*. El tercero de los tres nodos siempre juega el papel de nodo receptor (nunca arbitra), y nos referimos a él como *nodo receptor*. Este último nodo no es del todo necesario, pero es importante notar

que este nodo puede sincronizarse al principio de una trama con un nodo transmisor que finalmente pierda el arbitraje. Esto nos permite testear que este nodo receptor puede resincronizarse correctamente con el nodo transmisor que finalmente gana el arbitraje, a pesar de haber estado sincronizado durante todo el campo de arbitraje con el nodo transmisor que finalmente pierde el arbitraje. Es más, como los dos nodos transmisores ganan el acceso al medio de forma alterna, podemos testear que el nodo receptor no tiene problemas para sincronizarse correctamente con esos dos nodos de forma alterna, de trama a trama.

Para poder forzar estos escenarios de arbitraje y resincronización, programamos uno de los nodos transmisores (*Transmitter 0*) para que siempre transmitiese la misma trama con un valor de identificador igual a 11010011100. El otro nodo transmisor (*Transmitter 1*) lo programamos para que transmitiese de forma alterna 2 tramas, una de ellas con identificador igual a 11010011110 y la otra con identificador igual a 11010010100. El nodo receptor se programó para no intentar transmitir nunca, sino para actuar sólo como receptor.

Más concretamente, la secuencia de tramas que se transmiten en nuestros tests sigue el siguiente patrón: (1) Transmitter 0 y Transmitter 1 intentan acceder al medio con identificadores 11010011100 y 11010011110 respectivamente, ganando el acceso el Transmitter 0; (2) Transmitter 0 y Transmitter 1 intentan acceder al medio con identificadores 11010011100 y 11010010100 respectivamente, ganando el acceso el Transmitter 1; (1) Transmitter 0 y Transmitter 1 intentan acceder al medio con identificadores 11010011100 y 11010011110 respectivamente, ganando el acceso el Transmitter 0; y así sucesivamente.

4.1.4 Estrategia de debugging

Como se ha explicado al principio de este capítulo, una de las contribuciones relevantes de este TFG es la construcción de un nuevo prototipo de HDMS-CAN totalmente operativo; es decir, un prototipo que funcione correctamente para bit rates y longitudes de links que sean adecuados según los resultados de la cuantificación teórica y del análisis realizado en la Seccion 3.4.

En este TFG se ha considerado que la estrategia más conservadora para conseguir este nuevo prototipo totalmente operativo es depurar (debug) HDMS-CAN utilizando links cuya longitud no comprometa su correcto funcionamiento. De lo contario, sería difícil saber si el mal funcionamiento de HDMS-CAN se debe a un fallo de diseño/implementación, o a un fallo provocado por haber utilizado links demasiado largos. Concretamente, depuramos HDMS-CAN para los siguientes bit rates de transmisión: 125, 250, 500, 800 y 1000 kbps. Notar que por encima de 125 kbps se considera que se está utilizando CAN de alta velocidad (*high-speed CAN*). Respecto a la longitud de los links, para depurar HDMS-CAN se han utilizado links de 20 cm de longitud (como en el caso del prototipo original).

Esta estrategia resultó ser exitosa. Nos permitió depurar HDMS-CAN y, posteriormente, utilizar links más largos en las pruebas de rendimiento. Como se verá en el Capítulo 5, esas pruebas de rendimiento arrojaron resultados similares a las predicciones teóricas que se han realizado en las Secciones 3.3 y 3.4.

Es importante recordar en este punto a lo que nos referimos cuando decimos que HDMS-CAN funciona correctamente. Según explicamos al principio del Capítulo 3,

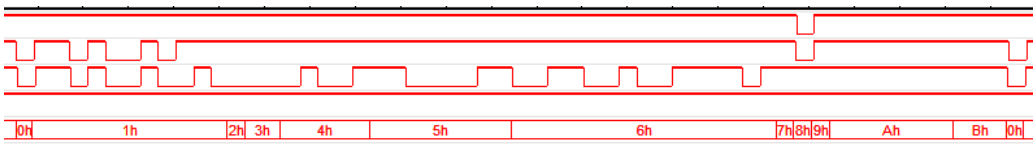

Figura 4.3: Ejempo de visualización de señales de debug

consideramos que HDMS-CAN funciona correctamente cuando cumple dos requisitos. El primero de ellos es que los nodos se comuniquen sin errores adicionales en el canal (los errores sólo deben ser ocasionados por circunstancias externas al prototipo, como interferencias electromagnéticas o deterioro de los cables). El segundo de los requisitos consiste en que el hub sea capaz de discernir correctamente la contribución de cada nodo bit a bit. Tal y como dijimos, nos referimos al primero de estos requisitos como *funcionamiento sin errores de comunicación*, mientras que nos referimos al segundo de ellos como *funcionamiento sin errores de discriminación*.

Para poder evaluar el correcto funcionamiento de HDMS-CAN es necesario monitorizar y visualizar un gran número de señales; tanto externas al hub (p.e. valor de la señal de transmisión en la salida del controlador CAN de un nodo determinado), como internas del hub (p.e. su reloj de transmisión).

Para visualizar señales externas al hub se ha usado un osciloscopio digital de dos canales. Por otra parte, a fin de poder monitorizar las señales internas del hub se ha utilizado el Monitoring Module. Como ya se ha explicado, este módulo no sólo provee y registra el valor lógico de un gran número de señales internas del Hub Core, sino que también provee información más elaborada que se calcula en base a esas señales, p.e. el porcentaje de errores de discriminación. Adicionalmente, existían casos en los que era importante ver la sincronización entre señales internas y externas del hub, para ello se reservaron algunos pines de entrada del Monitoring Module para conectar a las señales externas del hub que se requirieran observar junto a otras señales internas del mismo.

Además, se ha utilizado la Testing Interface para controlar cuáles son las señales que provee el Monitoring Module que deben visualizarse en la pantalla (pantalla del *laptop*), así como en qué momento deben visualizarse y/o procesarse éstas señales. La Figura 4.3 muestra un ejemplo de visualización en la pantalla de la Testing Interface (*laptop*). Cada una de las tres primeras señales se corresponde con la contribución de un nodo concreto discriminada por el hub. La cuarta es una señal interna del Hub Core que indica cuando éste ha detectado un errores de comunicación; es decir, cuando, desde el punto de vista de un receptor más, el Hub Core ha detectado un error en la trama actual (notar que cuando el Hub Core detecta un error en la trama, la globaliza como si fuese un nodo receptor). La quinta señal es un vector de bits cuyo valor numérico (representado en hexadecimal en la figura) codifica el campo de la trama al que pertenece el bit que se está transmitiendo actualmente en el medio, p.e. 0 es el SOF, 1 es el identificador del campo de arbitraje, 2 es el RTR, etc.

A continuación se especifican brevemente las señales monitorizadas más relevantes.

- *ClkR* y *ClkT*. ClkR y ClkT son respectivamente el reloj de recepción y el reloj

de transmisión generadas por el Hub Core. ClkR indica los instantes en que el Hub Core samplea la señal resultante de acoplar las contribuciones de los nodos (AND). ClkT indica los instantes en que el Hub Core considera que empieza cada tiempo de bit. Estas dos señales son muy útiles, ya que permiten saber respectivamente si el hub samplea en los instantes correctos, y si el hub está bien sincronizado con el *leading transmitter*.

- *msySelBroCouSig*. La señal msySelBroCouSig indica el valor de la señal de selección del multiplexor del módulo acoplador (ver Sección 2.3.4 y Figura 2.15). Concretamente, ésta es la señal que determina si el hub transmite el resultado de acoplar las contribuciones de los nodos (AND), o el valor recesivo que constituye alguno de los pulsos de enmascaramiento. Gracias a esta señal podemos saber en qué instantes el hub aplica los enmascaramientos; y además, cuando el Hub Core samplea la contribución de cada nodo para discriminarla (el hub samplea la contribución de cada nodo justo antes de acabar el pulso de enmascaramiento de sampleo).

- *pmuRx0*, *pmuRx1* y *pmuRx2*. Estas señales son muy importantes, puesto que cada una de ellas contiene el valor lógico sampleado de la contribución discriminada de un nodo concreto de HDMS-CAN.

- *gfmIniErrorFrame*. La señal gfmIniErrorFrame (cuarta señal de la Figura 4.3) indica cuándo el hub detecta un error de comunicación y, por tanto, comienza a transmitir una trama de error para globalizar dicho error.

- *gfmGlobalFrameState*. La señal gfmGlobalFrameState es la que especifica a qué campo de la trama pertenece el bit que se está transmitiendo actualmente (quinta señal de la Figura 4.3).

- *msyEstatBin*. La señal msyEstatBin es un vector de bits cuyo valor numérico indica a qué segmento del tiempo de bit pertenece el instante de tiempo actual; es decir SynSeg, PreMaskSeg, etc.

4.2 Fallos de diseño identificados y soluciones propuestas

A continuación se explican cuáles han sido los fallos de diseño de HDMS-CAN que se han encontrado al depurar el nuevo prototipo, así como las modificaciones que se ha propuesto en este TFG para solventarlos.

Es importante aclarar de nuevo que entendemos por fallos de diseño a aquellos presentes en la concepción inicial de HDMS-CAN, y que se deben a escenarios que no se tuvieron en cuenta a la hora de proponer esta nueva red de comunicaciones basada en CAN.

4.2.1 Enmascaramiento de las tramas de error

Este primer problema de diseño era resultado de mantener los enmascaramientos, tanto de prevención del *infinite loop-back* (PreMask) como de discriminación (Mask), durante las tramas de error.

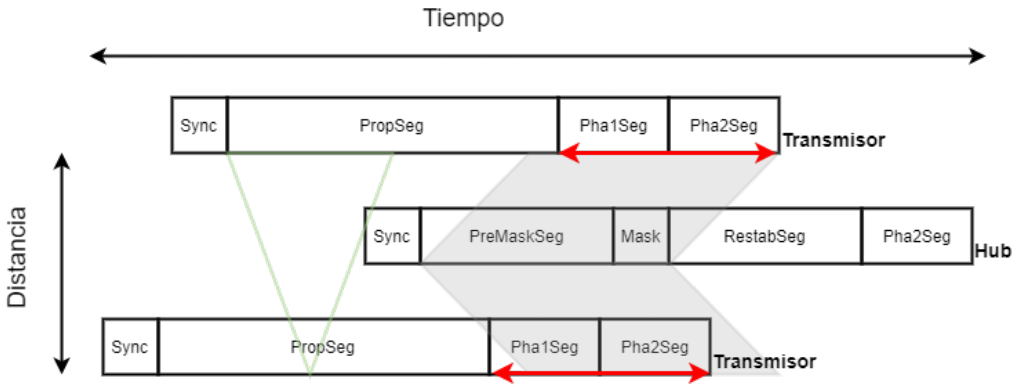

Figura 4.4: Esta imagen muestra como una gran desincronización puede provocar que la sombra del enmascaramiento haga que los nodos vean continuamente recesivos en el canal. Además, si se produce este efecto durante las tramas de error ni los nodos ni el hub podrán resincronizarse

Para entender el problema que supone esto recordemos que cuando cualquier nodo de la red, incluido el hub, detecta un error, lo globaliza para mantener la *reliable broadcast* de la red, como ya se explicó en la Sección 2.1.2. Dicho error se globaliza mediante una trama de error, formada por el *error flag* y el *error delimiter*. La primera parte de la trama de error está formada por 6 dominantes consecutivos que provocará alguno de los 5 tipos de errores, que CAN puede detectar, en los otros nodos. De esta forma todos los nodos irán abortando la recepción, o la transmisión en el caso del *leading transmitter*, del mensaje y enviando sus propias tramas de error. Esto debería conseguir que todos los nodos aborten la transmisión/recepción y que el canal vuelva al estado de *idle*, de tal forma que los nodos puedan intentar transmitir de nuevo.

Recordar que el hub globaliza los errores como si fuese un nodo más, transmitiendo tramas de error cuando detecta un error. Sin embargo, hay que tener en cuenta que su trama de error incluye enmascaramientos. Por tanto, cuando el hub envía los dominantes del error flag, los nodos no observan 6 dominantes "puros", sino una serie de pulsos dominantes y recesivos, donde los pulsos recesivos se corresponden con los enmascaramientos de prevención del *infinite loop-back* (PreMask) y de discriminación (Mask).

Para algunos *bit rates* cuando los nodos observaban esa secuencia de pulsos se desincronizaban de tal forma que la sombra de los enmascaramientos cubría sus puntos de sampleo tal y como se puede observar en la Figura 4.4. Como consecuencia, cada nodo siempre sampleaba un valor recesivo a no ser que él mismo enviase un valor dominante. Por tanto, cada nodo sólo veía la trama que él mismo transmitía de forma que no detectaba ningún problema y permanecía transmitiendo tramas, una tras otra, constantemente. Por su parte, el hub observaba que los nodos no señalizaban error como él esperaba y, por ello, el transmitía la trama de error una y otra vez.

Para solucionar el problema, en este TFG se ha propuesto eliminar los enmascaramientos en los *error flags* de las tramas de error del hub. Para ello, en el módulo acoplador (CouplerModule), se ha incluido una AND a la entrada de la señal de selec-

ción del multiplexor. La AND presenta como entradas las señales msySelBroCouSig y efgTxSignal, mientras que la salida esta conectada a la entrada de selección del multiplexor. De esta forma, cuando se transmite el *error flag* de la señal de error , es decir, los 6 dominantes ('0'), el enmascaramiento no se produce. Esto se debe a que la señal de selección del multiplexor tiene que valer '1' para hacer el enmascaramiento y, mientras se transmite el *error flag*, efgTxSignal vale '0' por lo que la salida de la AND tambien valdrá '0' evitando así el enmascaramiento y el error.

Esto soluciona el problema, pero conlleva una limitación: el hub no puede discernir la contribución de cada nodo mientras él esté enviando los 6 bits dominantes que constituyen su error flag. Respecto a esto, nosotros consideramos que el hecho de no poder discernir la contribución de cada nodo durante los 6 bits del *error flag* no es una limitación relevante debido al poco porcentaje de tiempo que supone en una red de comunicaciones.

4.2.2 Desincronización del hub en los ACKs

Recordemos que tanto nodos como hub deben sincronizarse con el *leading transmitter* y que, para ello, los nodos y el hub utilizan los flancos descendentes que observan en el canal tras haber recibido un bit recesivo. En este sentido, en el diseño original de HDMS-CAN, el hub se resincronizaba no sólo con los flancos descendentes provocados por los bits dominantes transmitidos por el transmisor, sino también con el flanco descendente que provoca la transmisión, por parte de los nodos receptores, del bit de ACK.

El hecho de que el hub se resincronizase con el bit de ACK provocaba que el hub se desincronizase con el *leading transmitter* y, por tanto, también con el resto de nodos (los cuáles no realizan ninguna resincronización en el bit de ACK). Esta desincronización del hub respecto al *leading transmitter* y al resto de nodos ocasionaba con cierta frecuencia que algún nodo o el hub detectasen errores.

De hecho, este fue uno de los últimos problemas de diseño que se identificaron y solucionaron. Esto se debió a que la citada desincronización provocaba que algún nodo o el hub detectasen errores, como mucho, cada 15 minutos. Esto suponía que, por ejemplo a 500 Kbps, se produjese un error cada 7.5 millones de tramas más o menos.

La solución a este problema fue enrutar la señal GlobalFrameState hasta el módulo MaskSynchroUnit para que, de esta forma, MaskSynchroUnit pudiese saber cuándo el bit actual era el bit de ACK y, por tanto, no llevara a cabo acciones de resincronización en ese bit.

4.3 Problemas de implementación y soluciones

Gracias al visualizador y a las señales de debug también pudimos detectar todos los errores de implementación.

Entendemos por error de implementación a un error en el hardware o software que no es consecuencia de un mal diseño o planteamiento.

4.3.1 Tamaño de los contadores

Las pruebas a 1000 y 800 kbps dieron buenos resultados, aptos para las pruebas de rendimiento, a pesar de que dichas pruebas no se realizarían para estos bit rates. Seguramente esto se debía a que el prototipo previo de red HDMS-CAN se creó para funcionar, específicamente, a estas velocidades. Sin embargo, las pruebas con una velocidad de transmisión de 500 kbps dieron errores de comunicación.

En este caso la comunicación se realizaba con normalidad hasta que, de repente y sin ningún patrón, aparecían ráfagas de errores. Debido a esto, y puesto que la única diferencia entre las velocidades altas y bajas de transmisión es el tamaño de los diferentes segmentos del tiempo de bit, se determinó que el problema debía estar en el tamaño de los contadores utilizados para determinar la longitud de cada segmento de bit.

Para solucionar el problema únicamente fue necesario aumentar el tamaño de los contadores del módulo maskSynchroUnit. De esta forma, tras comprobar el buen funcionamiento de la red, determinamos que el mal funcionamiento se debía a un desbordamiento de los contadores. Una vez solucionado el problema solo fue necesario ir reduciendo el tamaño de dichos contadores para evitar el uso innecesario de memoria, manteniendo el buen funcionamiento del hub.

4.3.2 Falta de hard-synchronization en el SyncSeg

Recordar que cuando un nodo receptor (y el hub) detecta el primer flanco descendente cuando el canal estaba en *idle*, asume que ese flanco se corresponde con el inicio del bit del SOF enviado por el *leading transmitter*. Por tanto, cuando dicho flanco ocurre, el nodo (y el hub) adapta los segmentos de su tiempo de bit para hacer coincidir dicho flanco con su SyncSeg. Este tipo de acción de sincronización, ya explicada en la Sección 2.1.1, se llama *hard-synchronization*.

En el prototipo no se había implementado la *hard-synchronization* en el SyncSeg del hub.En principio, esa implementación podría considerarse como correcta, ya que se supone que si el flanco de SOF cae en el SyncSeg quiere decir que el hub está bien sincronizado con el *leading transmitter*. Sin embargo, esta falta de *hard-synchronization* en el SyncSeg provocaba a posteriori una *hard-synchronization* en otro segmento del tiempo de bit que desincronizaba el hub y, por tanto, generaba errores de comunicación en la red.

Esta falta de *hard-synchronization* fue uno de los fallos más difíciles de detectar, ya que solo se producía si el flanco del SOF caía en el segmento del tiempo de bit SyncSeg del hub.

La solución a este problema fue implementar la *hard-synchronization* también en el SyncSeg del hub. Concretamente, el hub se ha modificado para que, estando en *idle* y al recibir un flanco descendente en el SyncSeg, registre que está bien sincronizado con el *leading transmitter* y que, por tanto, no realice una *hard-synchronization* erronea.

4.3.3 Características eléctricas de los transceivers

Uno de los errores de discriminación que encontramos consistía en que, en el link conectado a uno de los nodos, el hub, mediante el enmascaramiento de discriminación, sampleaba un valor recesivo a pesar de que el nodo introducía un dominante en el link. Esto se debía a que el hub sampleaba su propio recesivo en vez del dominante que

transmitía el nodo, provocando así un error de discriminación. Esta es una limitación de los transceivers utilizados. Para evitarla, habría que o bien utilizar otros transceivers o implementar transceivers adhoc. Sin embargo, a pesar de la limitación que suponen los *transceivers*, en este TFG conseguimos mejorar el comportamiento de éstos modificando algunos aspectos eléctricos como la alimentación.

Además, creemos que el mismo nodo tenía un reloj muy poco estable. Esto, combinado con el problema del *transceiver*, explica las grandes diferencias entre número de errores de discriminación de un nodo y otro que se expondrán en la Sección 5.2.

CAPÍTULO 5

PRUEBAS DE RENDIMIENTO

En el presente capítulo expondremos el tercer conjunto de contribuciones de este TFG, a saber la caracterización experimental del rendimiento de una red real HDMS-CAN.

Concretamente, hemos utilizado el prototipo completamente operativo de HDMS-CAN descrito en el capítulo anterior para caracterizar experimentalmente, y en función del bit rate, su cobertura, sus restricciones topológicas y el consumo de intensidad de sus *transceivers*. Como se ha comentado en el Capítulo 1, caracterizar estos aspectos es importante para ayudar a determinar la utilidad de la patente de HDMS-CAN desde el punto de vista de su posible explotación y comercialización.

En el presente capítulo se explicará en primer lugar cómo se ha ampliado el Monitoring Module, visto en la Sección 4.1.2, para facilitar la realización de las pruebas de rendimiento. Posteriormente, se mostrarán y explicarán los resultados obtenidos en los experimentos realizados.

5.1 Extensión de la red de pruebas

Para poder determinar la cobertura de HDMS-CAN y el consumo de sus *transceivers*, fue necesario añadir algunos mecanismos a la red de pruebas descrita en la Sección 4.1.

Por una parte, para poder determinar la cobertura en función del bit rate y corroborar las restricciones topológicas de HDMS-CAN explicadas en la Sección 3.3, fue necesario dotar a la red de pruebas de la capacidad de contar automáticamente el número de errores de comunicación y de discriminación. Para ello, por un lado se creó un autómata dentro del Hub Core, visto en la Sección 2.3.4, capaz de detectar los errores de comunicación y de discriminación; por otro lado, se creó un autómata dentro del Monitoring Module capaz de leer las señales de monitorización de los errores detectados en el autómata del Hub Core anteriormente mencionado y calcular el porcentaje de errores de comunicación y discriminación.

El autómata implementado en el Hub Core diferencia dos tipos de errores de discri-

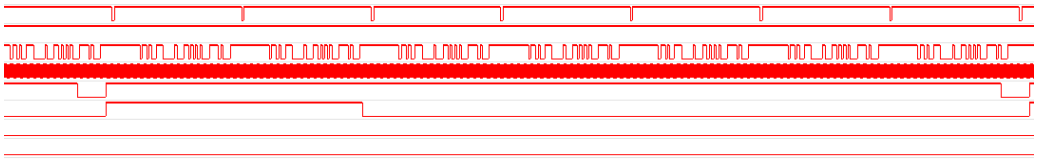

Figura 5.1: En esta captura de pantalla podemos ver las señales mostradas por el Testing Interface en una de las pruebas de rendimiento

minación de las transmisiones de los nodos transmisores (Transmitter 0 y Transmitter 1), *errores de discriminación de la trama* y *errores de discriminación del SOF de la trama*. Este último error de discriminación se identificó por separado dado que en las pruebas de debugging parecía ser un error bastante recurrente. Para detectar los errores, el autómata comprueba que la trama discriminada se corresponde con la trama transmitida por los transmisores. Si hay alguna diferencia entre la trama discriminada y la real, el hub transmite un pulso a través de las *señales de monitorización de los errores de discriminación*. Dichas señales, junto a una señal de monitorización que indica al Monitoring Module cuando comienza cada trama, permiten a un autómata del Monitoring Module calcular el porcentaje de errores de discriminación. El porcentaje calculado se transmite mediante una señal de debug al Testing Interface como una señal periódica cuyo ciclo de trabajo se corresponde con el porcentaje de errores de discriminación. Por último el Testing Interface representa dichas señales, en la pantalla del *laptop*, permitiendo visualizar el porcentaje de errores de discriminación medido en la prueba. Es importante remarcar que las pruebas se inician desde el Testing Interface y que el porcentaje de errores de discriminación se calcula contando el número de errores de discriminación que se producen a lo largo de 10000 tramas.

En cuanto a los errores de comunicación el Monitoring Module crea una señal de debug cuyo periodo tiene una duración proporcional al número de errores de comunicación producidos desde que se inicia cada prueba, las cuales se han descrito en el párrafo anterior. Concretamente, el Monitoring Module cuenta el número de errores de comunicación contando el número de flancos ascendentes de la señal de monitorización gfmIniErrorFrame vista en la Sección 4.1.4.

En la Figura 5.1 se muestra un ejemplo del resultado obtenido en una de las pruebas anteriormente mencionadas. Las 3 primeras señales mostradas en la figura se corresponden con la contribución discriminada de los nodos Receptor, Transmitter 1 y Transmitter 0 respectivamente. La siguiente señal indica el número de errores de comunicación que ha habido y las 4 siguientes señales indican el porcentaje de errores de discriminación de los transmisores (un 100 % de errores de discriminación de la trama en el Transmitter 1 ya que no estaba transmitiendo nada y un 40 % de Transmitter 0).

Por otra parte, en lo que concierne al consumo, se utilizó un multímetro para medir el consumo en los *transceivers* del hub y de los nodos. En el caso de los nodos se midió el consumo con y sin el hub, a fin de comparar el consumo de estos *transceivers* en CAN y HDMS-CAN. Esto es fundamental para aclarar si la presencia del hub incrementa el consumo de los *transceivers* de los nodos comparado con CAN. Este era uno de los objetivos principales del proyecto dado que, en HDMS-CAN, la presencia de los pulsos

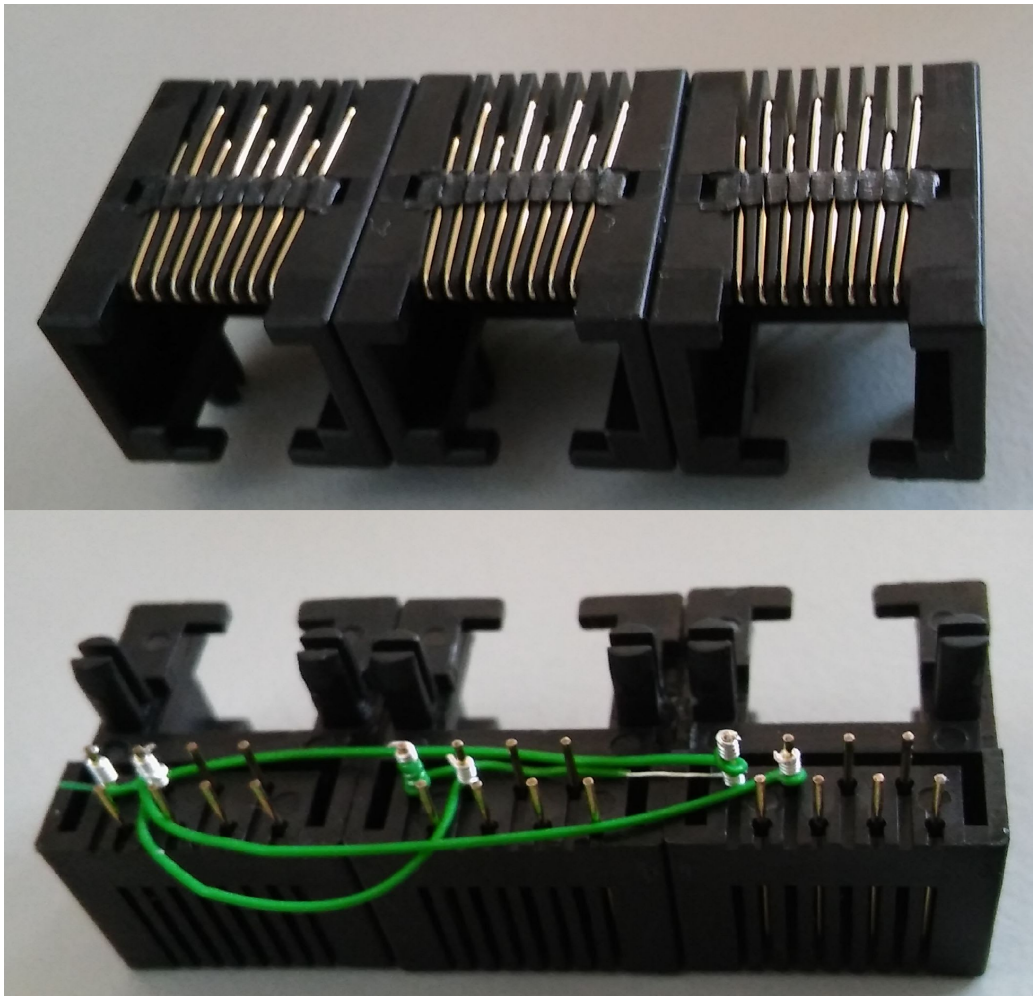

Figura 5.2: Conector central de un bus, con forma de estrella, utilizado para sustituir al hub en las pruebas de consumo de los *transceivers* de los nodos

de enmascaramiento durante el tiempo de bit obliga a los *transceivers* a conmutar más veces por bit que en CAN. Por esta razón no estaba claro antes de la realización del TFG si HDMS-CAN incrementaba dicho consumo y, de ser así, en qué medida lo incrementaba. Para medir el consumo de los *transceivers* de los nodos sin el hub, construimos un medio de transmisión con topología de bus, el cual se muestra en la Figura 5.2. Esta parte del bus está formado por 3 conectores RJ45 con sus pines cableados de tal forma que el CAN-H y CAN-L de los links se unieran en un único medio físico, en un bus. De esta forma se podía mantener la misma "estructura" de estrella de la red de pruebas, pero utilizando una topología de bus. Concretamente, este dispositivo nos permitía hacer las pruebas con y sin el hub manteniendo los mismos nodos y los mismos links creando "estrellas" idénticas, en las que solo cambiaba el elemento central que podía variar entre el hub de HDMS-CAN y el segmento de bus mencionado y mostrado en la Figura 5.2.

5.2 Pruebas y resultados

A continuación se explicarán las pruebas y los resultados que se han obtenido en ellas.

Las pruebas de rendimiento se realizaron con la red de pruebas transmitiendo a velocidades de 125, 250 y 500 kbps. Éstas permitieron comprobar experimentalmente aspectos como: (1) la cobertura máxima que podía abarcar el hub para distintas velocidades de transmisión, (2) comprobar las restricciones teóricas de la topología de HDMS-CAN, expuestas en la Sección 3.3, y (3) el impacto que supone la presencia del hub en el consumo de los *transceivers*, sobre todo en los nodos.

5.2.1 Cobertura de HDMS-CAN

El objetivo de esta prueba era calcular experimentalmente la máxima distancia a la que podía situarse un nodo del hub sin que se produjeran errores ni de comunicación ni de discriminación. A continuación se explica la estrategia seguida.

La prueba comenzaba situando los dos nodos transmisores a la máxima distancia que CAN admite (500 m para una velocidad de 125 kbps, 250 m para una velocidad de 250 kbps y 100 m para una velocidad de 500 kbps). Teniendo en cuenta que el hub debía colocarse en medio de los dos nodos más alejados para encontrar el diámetro máximo entre nodos, como ya se vio en la Sección 3.3, los nodos debían colocarse a una distancia del hub igual a la mitad de la distancia máxima entre nodos (250 m para una velocidad de 125 kbps, 125 m para una velocidad de 250 kbps y 50 m para una velocidad de 500 kbps). Una vez hecho esto, se realizaba una prueba en la que se calculaba el número de errores de comunicación y discriminación que se producían. Si se producía una cantidad de errores mayor que la que se produciría en CAN, se reducía 5 m la distancia máxima entre los nodos (se reducía 2.5 m la distancia de cada nodo al hub) y se repetía la prueba. Esto se realizaba hasta encontrar una distancia entre nodos tal que los errores de comunicación y discriminación fueran prácticamente 0. De esta forma se conseguía encontrar experimentalmente la máxima distancia a la que se podían colocar dos nodos del hub y, por tanto, la cobertura de éste (el hub). Por otro lado mencionar que el nodo receptor se había situado a una distancia del hub mucho menor que los dos nodos transmisores (la presencia de este nodo receptor esta explicada en la Sección 4.1.3).

En principio, como ya se ha mencionado anteriormente, nuestra intención era determinar la cobertura máxima del hub para 125, 250 y 500 kbps. Sin embargo, para 125 y 250 no pudimos determinar el radio máximo. La razón es que, para ello, deberíamos haber partido de un diámetro de la estrella igual a 500 y 250 metros respectivamente (ya que esas son las longitudes máximas del bus que CAN admite en ambos casos) y por restricciones de material no nos fue posible hacerlo. A pesar de ello, comprobamos que la red de pruebas a dichas velocidades de transmisión y una distancia entre nodos de 100 m funcionaban correctamente ya que se producía un 0 % de errores de comunicación, un 0 % de errores de discriminación en el Transmitter 0 y menos de un 0,1 % de errores de discriminación en el Transmitter 1. Este último dato se debía a las características eléctricas del *transceiver* de este nodo las cuales ya se explicaron en la Sección 4.3.3. A continuación procederemos con las pruebas de cobertura máxima realizadas a 500 kbps.

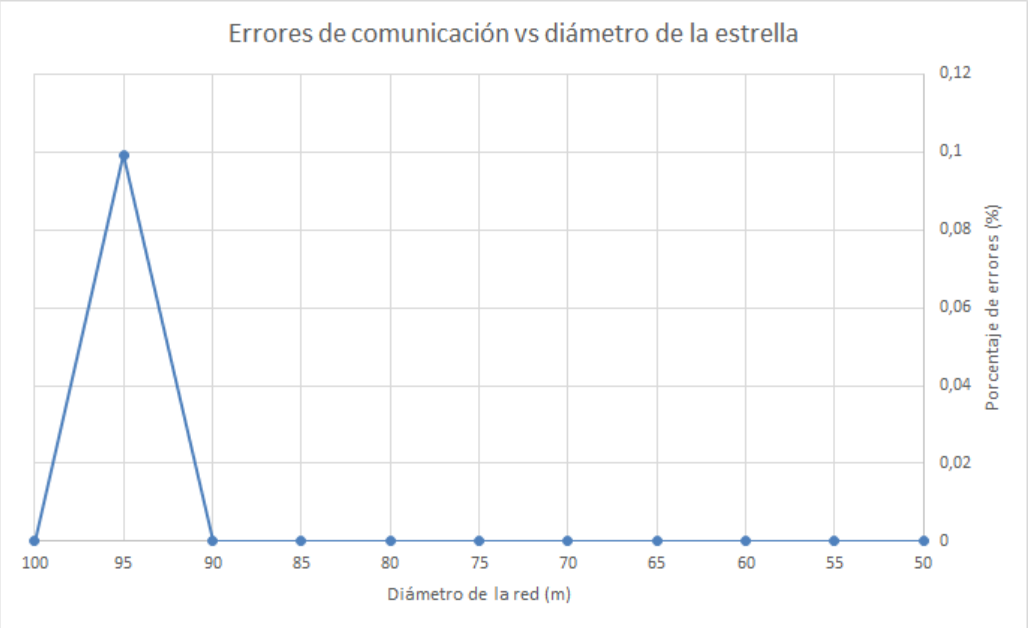

Figura 5.3: En esta gráfica se puede apreciar la evolución del número de errores de comunicación en función del diámetro de la estrella para una velocidad de transmisión de 500 kbps

Comenzaremos el análisis observando los *porcentajes de errores de comunicación* obtenidos en las pruebas a distintas distancias entre nodos, los cuales están graficados en la Figura 5.3. Seguidamente, analizaremos los *porcentajes de errores de discriminación* obtenidos en las mismas pruebas, los cuales están graficados en la Figura 5.4.

En cuanto a los porcentajes de errores de comunicación observamos que a una velocidad de transmisión de 500 kbps y una distancia entre nodos de 100 m (la máxima admitida por CAN) nos encontramos con que la red apenas presentaba errores de comunicación. Éste fue un resultado sorprendente ya que los nodos podían comunicarse con normalidad a la máxima distancia que admite CAN, para esa velocidad de transmisión, a pesar de haber incluido en la red un hub que incluye delays y modifica las señales recibidas por los nodos con enmascaramientos.

En la Figura 5.3 también se puede observar como a una velocidad de 500 kbps y 95 m de diámetro (los dos nodos transmisores situados a 47,5 m del hub) se produjo un pico de errores de comunicación y como a partir de 90 m, entre nodos, los errores de comunicación desaparecieron. Esta distribución de porcentajes de error concuerda con lo esperado y a continuación explicaremos por qué.

Cuando los nodos transmisores estaban situados a 50 m del hub (100 m entre nodos) la sombra del enmascaramiento caía detrás del punto de sampleo de los transmisores y por ese motivo no entorpecía la comunicación. Sin embargo, a medida que los nodos se aproximan al hub la sombra del enmascaramiento avanza en el tiempo de bit, tal y como se puede deducir observando la Figura 3.3. De esta forma, cuando los nodos se situaron a 47,5 m del hub la sombra del enmascaramiento cayó sobre el punto de sampleo produciendo el pico de errores, mientras que, cuando los nodos se situaron

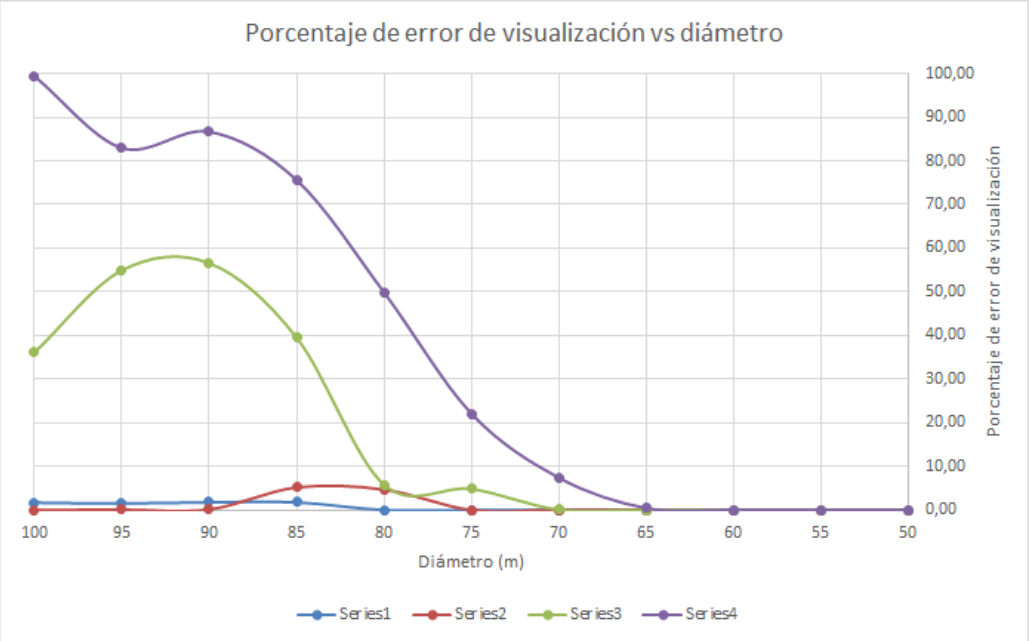

Figura 5.4: En esta gráfica se puede apreciar la evolución del porcentaje de errores de discriminación (visualización de la contribución de los nodos) en función del diámetro de la estrella para una velocidad de transmisión de 500 kbps. Por orden las series indican: porcentaje de errores de discriminación del SOF del Transmitter 0, porcentaje de errores de discriminación del SOF del Transmitter 1, porcentaje de errores de discriminación del Transmitter 0 sin contar los errores de discriminación del SOF en el Transmitter 0 y porcentaje de errores de discriminación del Transmitter 1 sin contar los errores de discriminación del SOF del Transmitter 1

a 45 m del hub la sombra del enmascaramiento avanzo suficiente en el tiempo de bit como para, de nuevo, no entorpecer las comunicaciones.

A continuación analizaremos los *porcentajes de errores de discriminación* obtenidos en las pruebas a distintas distancias entre nodos, los cuales están graficados en la Figura 5.4.

Sobre estos resultados cabe destacar la gran diferencia de errores de comunicación entre los dos nodos, el que presentaba problemas eléctricos en el *transceiver* y el que no, y que la distancia máxima a la que se dejaron de producir errores de comunicación y discriminación fue 65 m entre nodos (32,5 m entre los nodos transmisores y el hub). Es importante mencionar que este resultado concuerda con el obtenido de forma teórica mediante las ecuaciones de la Sección 3.4 lo cual es un indicio de que dichas ecuaciones modelan correctamente la realidad.

En cuanto al conjunto de resultados mencionados anteriormente vemos como en la red se consiguió un 0 % de errores de comunicación mucho antes que el 0 % de errores de discriminación. Esto podría indicar que se puede conseguir una configuración del tiempo de bit que mejore los resultados obtenidos en este proyecto. Esto se conseguiría situando la sombra del enmascaramiento más próximo al punto de sampleo de los

nodos reduciendo la distancia máxima a la que fallan las comunicaciones pero aumentando la distancia a la que desaparecen los errores de discriminación. Esto se debe a que, como retrasar el segmento MaskSeg en el tiempo de bit disminuye la distancia máxima sin errores de comunicación pero aumenta la distancia máxima sin errores de discriminación, el diseño de los segmentos del tiempo de bit óptimo haría que ambas distancias fueran la misma. Este desfase entre la máxima distancia sin errores de comunicación y discriminación seguramente se debe a una caracterización de los segmentos del tiempo de bit demasiado conservadora que estableció distancias de seguridad mayores a las realmente necesarias.

Además de encontrar la cobertura máxima del hub calculando los errores de comunicación y discriminación cuando los nodos arbitran en cada trama, como se explicó en la Sección 4.1.3, quisimos medir los errores de comunicación y discriminación cuando el mecanismo de resincronización a nivel de bit no se estresa al máximo, es decir, cuando los nodos no arbitran y, por tanto, no hay desincronización entre nodos que intentan transmitir a la vez. Este dato es importante ya que normalmente, a no ser que la red tenga una utilización muy alta, lo más habitual es que no se produzcan arbitrajes. En este sentido observamos que, cuando no se produce arbitraje, la red presenta un 0 % de errores de comunicación y discriminación con todas las velocidades de transmisión y todas las distancias entre nodos, siempre que estas no superen la distancia máxima de bus que admite CAN. Éste es un gran resultado que reafirma el buen funcionamiento de la red de HDMS-CAN.

5.2.2 Restricciones topológicas de HDMS-CAN

Además de la cobertura máxima del hub, quisimos comprobar que las restricciones topológicas de HDMS-CAN mencionadas en la Sección 3.3 se correspondían con la realidad. Concretamente buscábamos demostrar que el factor relevante, en cuanto a topología se refiere, es la distancia de los nodos al hub. Para ello analizamos los errores de comunicación y discriminación obtenidos a 500 kbps con distintos tamaños de links.

La estrategia consistía en calcular los errores cuando los nodos estaban situados a la misma distancia del hub y, a continuación, se reducía la distancia al hub de *uno* de los nodos y se volvían a calcular los errores.

En este experimento se observó como los errores del nodo que se había mantenido a la misma distancia del hub se mantenían iguales, mientras que los del nodo que se había situado más cerca del hub se reducían. Esto demostró que los errores no dependen de la distancia entre nodos, sino que dependen de la distancia de los nodos al hub, tal y como se había deducido.

5.2.3 Consumo de los transceivers de HDMS-CAN

Como ya se ha mencionado anteriormente en este capítulo, la presencia de los pulsos de enmascaramiento durante el tiempo de bit obliga a los *transceivers* a conmutar más veces por bit que en CAN. Por esta razón decidimos estudiar el consumo de los *transceivers* para ver en que sentido esta característica les afecta.

Con este objetivo se calculó el consumo de corriente alterna y contínua de los *transceivers*, tanto del hub como de los nodos, con y sin el hub, en todas aquellas

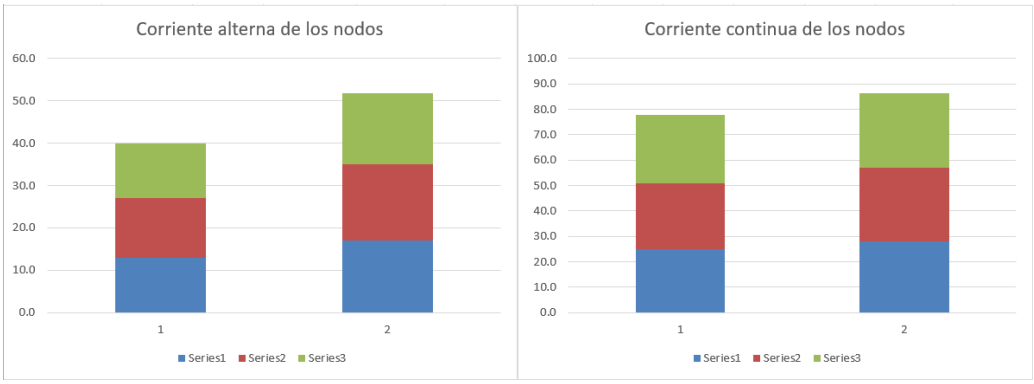

Figura 5.5: Las gráficas muestran el consumo de corriente continua y alterna respectivamente en los *transceivers* (Node Input/Output) de los nodos. La columna 1 de ambas gráficas indica el consumo con el hub y la columna 2 muestra el consumo sin el hub. Las series se corresponden con las velocidades 125, 250 y 500 kbps respectivamente

pruebas que no se dieron errores ni de comunicación ni de discriminación, es decir, a 125 y 250 kbps con un diámetro de estrella de 100 m y a 500 kbps con un diámetro de 60, 55 y 50 m.

El consumo de los *transceivers* del hub supone un consumo adicinal inevitable. Sin embargo, dependiendo de la aplicación, consideramos que vale la pena el consumo extra que supone si tenemos en cuenta la gran cantidad de mejoras que aporta HDMS-CAN. En cuanto a la corriente consumida, hemos comprobado que el consumo de los *transceivers* del hub es muy similar a la corriente de los *transceivers* de los nodos, incluso inferior.

Por otro lado, el consumo de los *transceivers* de los nodos resultó ser menor en presencia del hub, tal y como se puede apreciar en la Figura 5.5. En esta gráfica se puede observar como el consumo de los *transceivers*, tanto de corriente alterna como continua, en presencia del hub es menor que cuando éste (el hub) era sustituido por el bus mostrado anteriormente en la Figura 5.2. Éste fue un resultado sorprendente, y muy favorable, ya que en un principio pensábamos que el aumento de conmutaciones supondría también un aumento del consumo. Esta variación de corriente se debe a que la longitud del link es más relevante para el consumo de los *transceivers* que la propia frecuencia de conmutación. De esta forma, como en presencia del hub los *transceivers* de los nodos ven reducido el tamaño del link al que están conectados, éstos también ven decrecer su consumo de corriente.

CAPÍTULO 6

CONCLUSIONES

En este último capítulo de conclusiones resumiremos todas las tareas realizadas y los resultados obtenidos en su ejecución. Además, propondremos futuras acciones a realizar que podrían enriquecer mucho el prototipo desarrollado en este proyecto, muchas de las cuales son consecuencia de los resultados obtenidos a lo largo de este proyecto. Finalmente se incluirá un apartado más personal en el que comentaré mi experiencia vivida a lo largo de este TFG y, a grandes rasgos, mencionaré todo lo que he aprendido.

6.1 Compendio

A continuación, en este apartado, se listarán todas las tareas realizadas, se referenciarán los capítulos del documento en los que se trata dicha tarea y se resumirán brevemente los resultados obtenidos en su ejecución.

- *Estudio previo.* El primer objetivo consistía en entender y profundizar en el protocolo CAN y en la patente de HDMS-CAN, así como en el código del prototipo previo a mi TFG. Aunque no hay un resultado específico de esta labor, el proyecto no se habría podido realizar sin esta acción previa. Por otro lado, todos los conocimientos adquiridos a lo largo de este proceso de familiarización con el proyecto se pueden ver en el Capítulo 2.

- *Diseño de los segmentos del tiempo de bit de HDMS-CAN y restricciones de la invención.* Esta tarea se corresponde con el proceso de caracterización de los segmentos del tiempo de bit y el estudio de las restricciones topológicas de HDMS-CAN mostrados en el Capítulo 3.

  Mediante este proceso de diseño se propusieron unas ecuaciones para el dimensionamiento de los segmentos del tiempo de bit que, aunque experimentalmente se descubrió que no eran óptimas, proporcionaron muy buenos resultados. Además descubrimos que lo que limita el tamaño de la red es la distancia de los

nodos al hub, aspecto al que llamamos cobertura del hub, y creamos ecuaciones capaces de predecir dicha cobertura, para el diseño de segmentos del tiempo de bit propuesto, a distintas velocidades de transmisión. Otro descubrimiento remarcable fue comprobar la independencia de la longitud de los pulsos de enmascaramiento de discriminación con la longitud de los links lo cual habría supuesto un empeoramiento del rendimiento de las redes de HDMS-CAN, además de un aumento de la complejidad del dimensionamiento de los segmentos del tiempo de bit.

- *Revisión del prototipo*. Esta tarea se corresponde con el proceso de revisión y puesta a punto, del prototipo, mostrado en el Capítulo 4.

  A pesar de los problemas de diseño e implementación que presentaba el prototipo inicial conseguimos implementar un hub totalmente operativo para diferentes bit rates y tamaños de los links, sin errores ni de comunicación ni de discriminación, apto para las pruebas de rendimiento y una posible futura comercialización.

- *Estudio del rendimiento del prototipo*. Esta tarea se corresponde con el proceso de determinación del rendimiento del prototipo mostrado en el Capítulo 5.

  En este último apartado del proyecto se realizaron análisis de la red de pruebas que permitieron caracterizar en detalle la invención. Entre otras cosas se confirmaron las ecuaciones que preveían la cobertura máxima del hub para diferentes *bit rates*, se comprobó el buen funcionamiento del invento para diferentes tamaños de estrella y velocidades de transmisión, descubrimos que los nodos consumían menos corriente en presencia del hub y también comprobamos que si la red no realizaba arbitrajes la probabilidad de error se volvía cero. Además, como ya se mencionó en el apartado 5.2.1, descubrimos que habíamos sido demasiado conservadores con las distancias de seguridad en los segmentos del tiempo de bit y que, por lo tanto, el prototipo aun podía mejorar su ya buen rendimiento.

6.2 Trabajo futuro

En este apartado propondremos algunos trabajos futuros que se deberían realizar para seguir mejorando HDMS-CAN. Estas propuestas surgen de los resultados obtenidos en este TFG.

Enlazando con el último punto del apartado anterior, una tarea interesante podría ser encontrar experimentalmente la posición óptima del segmento Mask en los tiempos de bit de HDMS-CAN. Para ello únicamente sería necesario ir retrasando dicho segmento, dentro del tiempo de bit, de TQ en TQ e ir realizando las pruebas de rendimiento hasta encontrar la posición óptima. Incluso, podría no ser necesario realizar todas las pruebas de rendimiento para cada desplazamiento del segmento. Suponiendo que la distancia máxima sin errores de comunicación y la distancia máxima sin errores de discriminación varían linealmente con el desplazamiento del segmento Mask, bastaría con realizar las pruebas solo dos veces más. En la primera prueba se situaría MaskSeg en una posición diferente a la utilizada en este proyecto y se calcularían las nuevas distancias máximas sin errores ni de discriminación ni de comunicación. Una

vez hecho esto tendríamos, para dos posiciones del segmento Mask, las distancias máximas sin errores de discriminación y sin errores de comunicación con las cuales podríamos crear dos ecuaciones, una que relacione la distancia máxima sin errores de discriminación con la posición de Mask y otra que relacione la distancia máxima sin errores de comunicación también con Mask. A partir de estas ecuaciones podríamos encontrar la posición de Mask para la que las dos distancias coinciden y solo quedaría comprobar experimentalmente que dicha posición de Mask es la posición óptima.

Adicionalmente se podrían crear *transceivers* especiales para el hub de HDMS-CAN lo cual también mejoraría su rendimiento. Esto debería realizarse antes de ejecutar la tarea de optimización de los segmentos del tiempo de bit, expuesto anteriormente, puesto que el cambio de *transceivers* modificaría la posición óptima del segmento Mask.

Por último la inserción de un concentrador activo en el centro de la estrella supone un único punto de ruptura más complejo que solo un bus. Por este motivo sería interesante intentar replicar lo conseguido en la infraestructura de comunicaciones reCANcentrate, la cual duplicaba los hubs utilizados en CANcentrate, para eliminar el punto único de ruptura.

6.3 Consideraciones sobre el proceso de aprendizaje

Además de los objetivos cumplidos, este proyecto también ha contribuido a enriquecer mis conocimientos y habilidades en diferentes ámbitos como son la programación, la electrónica o la lectura y redacción de documentos técnicos. A continuación se listarán los principales beneficios obtenidos en este TFG.

- *Comprensión de una patente.* Para este proyecto debía leer y comprender la patente de HDMS-CAN. Éste fue un proceso difícil debido a lo específico que es el lenguaje en este tipo de documentos.

- *Comprensión de un extenso y complejo código VHDL.* Hasta el momento había trabajado con códigos de hasta 3 módulos. Sin embargo, en este proyecto debía entender el funcionamiento de un código VHDL con cientos de módulos internos muy complejos lo cual supuso todo un reto.

- *Depuración de un código VHDL complejo.* Encontrar y corregir los problemas de diseño e implementación en un código con las características anteriormente mencionadas requirió un gran esfuerzo. Incluso fue necesario la creación de un visualizador de señales específico para este proyecto (Monitoring Module y Testing Interface) para poder visualizar la gran cantidad de señales involucradas en cada uno de los mecanismos internos del hub.

- *Caracterización de los segmentos del tiempo de bit y su descripción matemática.* Para poder determinar las dimensiones que debían tener los segmentos del tiempo de bit fue necesario, no solo comprender bien la patente, sino analizar procesos de transmisión y propagación de tramas en CAN mediante esquemas como los mostrados a lo largo de este documento p.e. la Figura 4.4. Además tenía que ser capaz de traducir los resultados gráficos en expresiones matemáticas genéricas, que sirvieran independientes a la velocidad de transmisión de la red.

- *Realización de pruebas de rendimiento.* En cuanto a las pruebas de rendimiento lo más difícil no fue la realización sino la planificación de éstas. Buscamos un proceso de análisis de la red de pruebas que minimizara los recursos tanto materiales como humanos creando, por ejemplo, códigos que determinaran directamente los porcentajes de errores de discriminación.

- *Organización del proyecto.* En cuanto a la organización de un proyecto, sobre todo, he aprendido pautas que se deben evitar. Entre ellas destaca buscar cumplir metas que no estaban marcadas previamente, quitando mucho tiempo a los objetivos principales. Un ejemplo de esto es el tiempo invertido en el cambio de interfaz expuesto en la Sección 1.4 que no se pudo realizar.

- *Realización de la memoria.* Finalmente dedicamos mucho tiempo a la elaboración de una memoria comprensible. Esto se debe a que el proyecto incluye una serie de elementos muy novedosos difíciles de entender a priori.

APÉNDICE A

ANEXOS

A.1 Tabla de resultados experimentales

En la tabla A.1 se muestran todos los resultados obtenidos a lo largo de las pruebas de rendimiento.

A continuación se explicará la tabla, concretamente, que contiene cada una de las columnas:

1. *BitRate*. Indica la velocidad de transmisión a la que se ha realizado la prueba.
2. *Diámetro*. Indica la máxima distancia entre nodos de la red de pruebas.
3. *ICTH*. Muestra la componente continua de la corriente en los *transceivers* del hub.
4. *IATH*. Muestra la componente alterna de la corriente en los *transceivers* del hub.
5. *ICNH*. Muestra la componente continua de la corriente en los *transceivers* de los nodos en presencia del hub.
6. *IANH*. Muestra la componente alterna de la corriente en los *transceivers* de los nodos en presencia del hub.
7. *nºe/15mill*. Indica el número de errores de comunicación que se han producido a lo largo de 15 millones de tramas.
8. *e%*. Indica el porcentaje de error que supone el valor de la columna anterior (nºe/15mill).
9. *ICN*. Muestra la componente continua de la corriente en los *transceivers* de los nodos sin el hub.
10. *IAN*. Muestra la componente alterna de la corriente en los *transceivers* de los nodos sin el hub.

11. *T1eSOF%*. Indica el porcentaje de errores de discriminación del SOF en uno de los nodos transmisores (Transmitter 0).

12. *T1eSOF%*. Indica el porcentaje de errores de discriminación del SOF en el otro nodo transmisor (Transmitter 1).

13. *T1eSAMP%*. Indica el porcentaje de errores de discriminación en el Transmitter 0.

14. *T2eSAMP%*. Indica el porcentaje de errores de discriminación en el Transmitter 1.

15. *ADN/ADF*. Esta columna indica si la prueba se ha hecho con los transceivers alimentados mediante salidas de tensión de las placas en las que estaban sintetizados los Node Core (ADN) o a través de la fuente de alimentación (ADF).

BirRate(kbps)	Diámetro(m)	ICTH(mA)	IATH(mA)	ICNH(mA)	IANH(mA)	nºe/15mill	e%	ICN(mA)	IAN(mA)	T1eSOF%	T2eSOF%	T1eSAMP%	T2eSAMP%	
125	100,0	131,0	20,0	25,0	13,0	0	0	28,0	17,0	0,00	0,00	0,00	3,540	ADN
250	100,0	132,0	17,0	26,0	14,0	0	0	29,0	18,0	0,00	0,00	0,00	2,630	ADN
250	100,0	-	-	-	-	0	0	-	-	0,00	0,00	0,07	0,000	ADF
250	100,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	100,0	133,2	17,6	28,7	12,5	1	6,67E-06	28,6	17,3	23,26	0,00	36,08	99,460	ADN
500	100,0	-	-	-	-	1	6,67E-06	-	-	1,68	0,00	34,84	66,860	ADF
500	100,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	95,0	-	-	-	-	14903	0,099353	-	-	1,50	0,08	54,91	82,990	ADF
500	95,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	90,0	-	-	-	-	0	0	-	-	1,75	0,13	56,60	86,700	ADF
500	90,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	85,0	-	-	-	-	0	0	-	-	1,75	5,17	39,50	75,640	ADF
500	85,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	80,0	-	-	-	-	0	0	-	-	0,00	4,68	5,79	49,880	ADF
500	80,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	75,0	-	-	-	-	0	0	-	-	0,00	0,01	4,81	22,080	ADF
500	75,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	70,0	-	-	-	-	0	0	-	-	0,00	0,00	0,07	7,380	ADF
500	70,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	67,5	-	-	-	-	0	0	-	-	0,00	0,00	0,00	7,210	ADF
500	65,0	-	-	-	-	0	0	-	-	0,00	0,00	0,00	0,430	ADF
500	65,0	-	-	-	-	0	0	-	-	-	0,00	-	0,000	ADF
500	62,5	-	-	-	-	0	0	-	-	0,00	0,00	0,04	0,010	ADF
500	60,0	129,0	16,8	26,4	13,0	0	0	28,4	15,9	0,00	0,00	0,00	0,007	ADF
500	55,0	135,5	18,3	26,8	12,8	0	0	29,8	17,2	0,00	0,00	0,00	0,000	ADF
500	50,0	135,5	18,2	26,7	12,8	0	0	29,8	17,2	0,00	0,00	0,00	0,000	ADF

Figura A.1: Tabla de resultados

BIBLIOGRAFÍA

[1] M. Barranco, J. Proenza, and L. Almeida, "Boosting the Robustness of Controller Area Networks: CANcentrate and ReCANcentrate," *Computer*, vol. 42, no. 5, pp. 66–73, may 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5070041/ 1.1

[2] Kiril Mucevski, "Automotive CAN Bus System Explained | Kiril Mucevski | Pulse | LinkedIn." [Online]. Available: https://www.linkedin.com/pulse/automotive-can-bus-system-explained-kiril-mucevski 2.1

[3] "Literature Review: Ciencias de la Tecnología e Innovación, Handbook T-I." [Online]. Available: https://www.researchgate.net/publication/313666204{_}Ciencias{_}de{_}la{_}Tecnologia{_}e{_}Innovacion{_}Handbook{_}T-I 2.7

[4] A. Soriano, H. Nazario Giménez Espinosa de los Monteros, H. Jordá Fernández, J. Manuel Ripoll Gisbert, F. Saval Sendra, P. Miguel Serrano Reche, J. Talens Noguera, and J. Vicente, "Qué es CAN y cómo funciona el módulo del microcontrolador dsPIC30F4013." [Online]. Available: http://server-die.alc.upv.es/asignaturas/PAEEES/2005-06/A03-A04-BusCAN.pdf 2.8

[5] A. Ballesteros, J. Proenza, and M. Barranco, "Construction of sfiCAN: a star-based fault-injection infrastructure for the Controller Area Network," Ph.D. dissertation, UIB, 2012. 2.9, 2.10