SIMULADOR FERROVIARIO DEL ELEMENTO DE CAMPO BALIZA

Jorge Fernández Ortihuela, Ricardo de la Torre González

PROYECTO DE FIN DE CARRERA

DEPARTAMENTO ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

FACULTAD DE INFORMÁTICA

TRABAJO PROYECTO DE FIN DE CARRERA

Junio 2014

Directora: Matilde Santos Peñas

Autorización de difusión y utilización

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Jorge Fernández Ortihuela

Madrid, 10 de Junio de 2014

Ricardo de la Torre González

Madrid, 10 de Junio de 2014

A Ramón Galán López por sus enseñanzas aportadas y su ayuda en el proyecto

Agradecimientos

Gracias a los miembros del Centro de Automática y Robótica de la Universidad Politécnica de Madrid por ofrecer el proyecto y las instalaciones. Agradecemos también a Matilde Santos su aportación y colaboración como directora del proyecto, así como al departamento de Arquitectura de Computadoras y Automática de la Universidad Complutense de Madrid. Asimismo, agradecemos a la empresa THALES ESPAÑA GRP, S.A.U. su colaboración en el proyecto y el darnos la oportunidad para poder llevarlo a cabo.

Por último nos gustaría agradecer a nivel personal a todas aquellas personas que han hecho posible que hayamos podido realizar este proyecto, familiares, amistades, compañeros de trabajo y carrera y profesores.

Índice

Índice de ilustraciones

Índice de Tablas

Índice de abreviaturas

Abreviatura	Nombre
BD	Balise Driver
BDSim	Balise Driver Simulator
BTM	Balise Transmission Module
CCS	Control Command and Signaling
CEC	Centralised ECTS Controller
CRC	Comprobación Redundancia Cíclica
CW	Onda Continua
ECSim	ECTS Controller Simulator
ECTS	European Train Control System
ERTMS	European Railway Traffic Management System
EVC	European Vital Computer
KER	Sistemas KVB, Ebicab, RSDD ...
LEU	Lineside Electronic Unit
RBC	Radio Block Centre
SO	Sistema Operativo
STM	Specific Transmission Module
SW	Software
HW	Hardware
I/F	Interface

Resumen

En el presente proyecto se desarrolla una aplicación informática de simulación del elemento de campo baliza para un sistema de control de tráfico ferroviario. Los requisitos de seguridad de los elementos ferroviarios requieren baterías de pruebas complejas de realizar en los elementos físicos reales, por lo que se diseñan simuladores software para llevar a cabo dichas pruebas.

La aplicación BDSim se ha desarrollado en el marco de un acuerdo de colaboración entre la Universidad Politécnica de Madrid y la empresa THALES S.A., en el que han participado los alumnos del proyecto de la Facultad de Informática de la Universidad Complutense de Madrid.

La aportación fundamental del trabajo es el diseño de un software de simulación en un entorno de tiempo real que cubre los siguientes objetivos:

- La aplicación simula el comportamiento de controladoras de elementos de campo balizas.
- El usuario puede simular cualquier escenario, incluyendo situaciones de error en comunicaciones, anomalías de las señales de campo, retardos en el envío de señales, etc., pudiendo garantizar que el software de control responde correctamente frente a cualquier situación posible.
- Reduce el coste de los equipos de verificación que quedan reducidos al hardware y software de simulación, no siendo necesario disponer de elementos reales como se venía haciendo hasta ahora. Asimismo también se reduce el tiempo de pruebas.

Palabras clave: Control de tráfico ferroviario, Baliza, Ingeniería Software, Simulador elementos de campo, Sistemas de tiempo real

Abstract

In this project, it's been designed and developed a computer simulation application for a Balise element field in railway systems. The safety requirements of railway items require performing complex tests in the case of the physical elements, so software simulators are designed to carry out such tests.

The BDSim application was developed under a collaboration agreement between the Polytechnic University of Madrid and the company THALES SA in which the authors of the project from the Complutense University of Madrid have participated.

The key features of the work are marked by the design simulation software on a real time environment covering the following objectives:

- The application simulates the behavior of controller boards Balise field elements.
- The user can simulate any scenario, including communications error situations, abnormalities field signals, delays in sending signals, etc.., Can ensure that the control software responds correctly to any possible situation.
- The cost of verification teams are reduced to the hardware and software simulation, not being necessary to have real elements as it's been done so far. It is also reduced the testing time.

Keywords: Railway Traffic Control, Balise, Balise, Software Engineering, simulation, Real Time Systems

CAPÍTULO 1

INTRODUCCIÓN

1. INTRODUCCIÓN

El presente proyecto, denominado BDSim (Balise Driver Simulator), ha sido realizado en el Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid. Se encuentra enmarcado dentro de un acuerdo de colaboración entre el departamento de Automática, Ingeniería Electrónica e Informática Industrial de la Universidad Politécnica de Madrid (UPM) y la empresa THALES S.A. Desde el año 2000 se vienen desarrollando diferentes programas informáticos para emular el comportamiento de elementos de campo ferroviarios y conseguir de esta manera una validación del software que incorporarán en su emplazamiento final.

A partir de Septiembre de 2013 los alumnos a cuya autoría corresponde este proyecto fin de carrera han participado en la elaboración de un simulador para la validación del software de un elemento de campo en concreto: la baliza electrónica.

El proyecto es un ejemplo de la colaboración que se puede llevar a cabo entre la empresa privada y la universidad, quedando patente la alta competencia que tiene esta última en proporcionar I+D y aportar valor en el ámbito académico del proyecto.

Los elementos de campo situados en tramos ferroviarios, tradicionalmente basados en relés, han sufrido un cambio radical con la incorporación de la alta velocidad. La empresa THALES S.A. ha apostado por la utilización de buses tipo CAN BUS para conectar estos elementos a sus enclavamientos. En el caso que nos ocupa, el elemento baliza, se trata de una herramienta esencial que permite incrementar la seguridad de los tramos ferroviarios y a la vez proporcionar información de los vehículos que la utilizan en tiempo real.

Como elemento esencial de recepción y emisión de información se utiliza una redundancia doble en todas sus comunicaciones con el fin de establecer un sistema seguro de transmisión de información. Hay que tener en cuenta que, al incluir un bus en la transmisión de información, la

complejidad del sistema aumenta y la tolerancia a fallos es sensiblemente menor, pues ha de estar preparado para simular los distintos fallos en la transmisión que se puedan producir.

A esto contribuyen los exigentes sistemas de validación de software que ha de pasar el simulador para poder ser implementado, y que vienen determinados por las empresas responsables de la gestión del proyecto ferroviario de modernización de infraestructuras que está siendo llevado a cabo en toda Europa.

Antes de su instalación en una estación determinada se realizan todo tipo de pruebas para verificar que el comportamiento del software desarrollado y su configuración específica para la estación reacciona de forma segura. THALES S.A. había solicitado a la Universidad simuladores que tuvieran el mismo comportamiento electrónico que los elementos de campo y, a la vez, forzar comportamientos anómalos: lámparas fundidas, cortocircuitos, motores bloqueados, falta de reacción frente a mandos, etc.

En la Ilustración 1 se muestran los diferentes sistemas que se han utilizado en la elaboración de este simulador así como los elementos a simular, las balizas electrónicas.

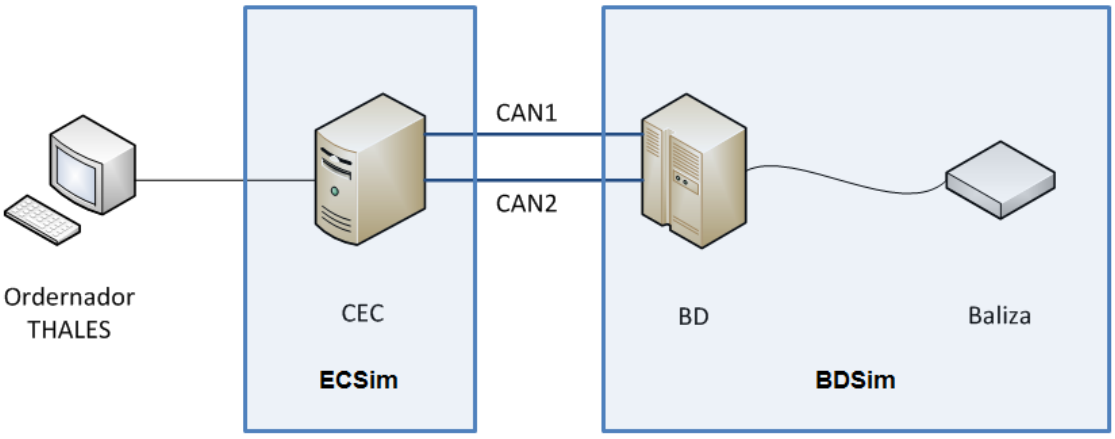

Ilustración 1 - Sistema a simular en el proyecto BDSim

Los principales elementos de los que consta el sistema simulado son:

- Ordenador de la empresa THALES: donde se visualizan los datos del CEC
- CEC (Controlador de Elementos de Campo): Es el controlador de la baliza, que manda las órdenes que las tarjetas controladoras deben ejecutar.
- BD (Balise Driver): Transforma el mensaje (orden) del CEC en señal eléctrica para señalizar la baliza en los distintos estados de funcionamiento. Estas órdenes pueden variar desde la configuración en un estado en concreto hasta la lectura de datos de un fichero de texto para la configuración automática de la propia baliza.

2. OBJETIVOS DEL PROYECTO

El objetivo de la simulación de este sistema consiste en modelizar el comportamiento del driver y la baliza (BD) en un programa informático. El marco del proyecto al que está vinculado este trabajo consta de tres partes bien diferenciadas:

1) Implementar el programa de simulación de la BD
2) Verificar el correcto funcionamiento del CEC introduciendo errores en los mensajes. Dado un error se genera una cierta respuesta. Para todos los errores y sus correspondientes respuestas existe un documento técnico que define las pruebas que debe superar el programa para validar su correcto funcionamiento.
3) Realizar esas pruebas que serán programadas en cierto lenguaje para ser interpretado por el simulador BDSim, de forma que se ejecuten automáticamente sin necesidad de operaciones manuales.

3. ESTRUCTURA DE LA MEMORIA

La memoria se ha estructurado de la siguiente manera:

Capítulo 1: se ofrece un marco introductorio al proyecto, presentando brevemente los objetivos del mismo y resumiendo el contenido de la memoria.

Capítulo 2: breve exposición del estado del arte en lo relativo a señalización ferroviaria en el sistema ERTMS/ETCS. Se exponen los diferentes niveles de funcionamiento del sistema y se explica brevemente el elemento baliza y los simuladores de elementos ferroviarios.

Capítulo 3: relación de requerimientos del sistema software elaborado por los integrantes del proyecto fin de carrera para poder simular el elemento baliza. Consta de diferentes tipos de requisitos, gráficos, generales y elementos propios del protocolo de comunicación.

Capítulo 4: se ofrece una descripción de la solución implementada por el equipo del proyecto para el simulador, incluyendo sus paquetes software y un resumen de sus atributos y métodos más relevantes.

Capítulo 5: un caso práctico que ofrece una visión general de la solución implementada y sirve para realizar un test de validación de los requisitos iniciales.

Capítulo 6: breve resumen de las conclusiones del proyecto y una relación de las posibles líneas de trabajo que pueden surgir a partir del mismo.

La memoria termina con la bibliografía del proyecto.

4. ASIGNATURAS RELACIONADAS

En el proyecto fin de carrera que se presenta en esta memoria se han trabajado materias relacionadas, entre otras con las siguientes asignaturas de la carrera de Ingeniería en Informática:

- Ingeniería del Software: para la implementación de la solución software se han utilizado metodologías estudiadas en esta asignatura.
- Redes: se han utilizado conocimientos relativos a esta asignatura en lo concerniente a protocolos de comunicación y estudio de puertos CANBUS.
- Sistemas operativos: al tratarse de un proyecto software implementado en el lenguaje de programación C, se han utilizado diferentes métodos y estructuras de datos estudiadas en el ámbito de sistemas operativos UNIX.
- Laboratorios de Programación: se ha utilizado los conocimientos adquiridos en estas asignaturas para el desarrollo en lenguaje C++.

CAPÍTULO 2

SISTEMA DE SEÑALIZACIÓN FERROVIARIA

2. SEÑALIZACIÓN FERROVIARIA: SISTEMA ERTMS/ETCS

2.1. Sistema ERTMS/ETCS

A continuación se ofrece una explicación del sistema ERTMS/ ETCS, que incluye las balizas como uno de sus elementos principales para mostrar su funcionalidad. Para obtener una información pormenorizada del sistema, sus elementos y funciones ver [1], [2] y [3].

2.1.1. Introducción

ERTMS son las siglas de European Rail Traffic Management System (Sistema Europeo de Administración de Tráfico Ferroviario). Dentro de él se encuentra el ETCS, European Train Control System o Sistema Europeo de Control de Trenes, ideado y desarrollado conceptualmente por un equipo de expertos europeos integrados en el ERTMS Users Group.

Según sus impulsores sus múltiples ventajas nacen del hecho de ser un sistema diseñado en su totalidad para hacer frente a las demandas presentes y futuras del transporte ferroviario. Es decir, por primera vez en un proyecto de este tipo se parte del conocimiento y la experiencia acumulada de años de trabajo en la gestión de líneas ferroviarias, y se realiza el diseño con la única restricción de obtener un producto con la suficiente modularidad como para poder hacer una implantación progresiva. De este modo durante el tiempo que se tarde en instalar la totalidad de los equipos necesarios en los trenes, las vías, las estaciones, etc., pueden coexistir con la tecnología convencional usada hasta ahora. Existe no obstante una importante dificultad que consiste en poner de acuerdo a toda la industria de señalización

para conseguir alcanzar los principales objetivos del proyecto ERTMS/ETCS, que pueden resumirse en lo siguiente, y que se han basado en [4]:

- Estandarizar las funciones de control de trenes y el intercambio de información.
- Permitir una implementación gradual
- Permitir diferentes niveles de aplicación de acuerdo con las necesidades de actuación.
- Lograr mayor seguridad que los actuales sistemas de señalización.
- Definir:
    - las interfaces externas.
    - los estándares de comunicación.
- Crear interfaces con los sistemas nacionales actuales de modo que sea posible recibir información de ellos y traducirla al formato ERTMS/ETCS.

Como podemos comprobar diariamente la liberalización existente en el mundo del transporte obliga a que el futuro del ferrocarril dependa de la competitividad de este medio de transporte frente a otros. El ERTMS/ETCS ofrece para ello una serie de ventajas económicas basadas en su versatilidad, que permitirá operar en una misma línea con diferentes niveles creando diferentes escenarios de ocupación y capacidad entre los que podremos optar, tratando siempre de adaptarnos de la forma más rentable a los análisis de coste–beneficio. Entre las diferentes posibilidades del ERTMS/ETCS podemos destacar las siguientes:

- puede funcionar como un sistema de protección sin necesidad de señalización en cabina y sin sistemas de seguridad.
- puede funcionar como un sistema de control, con sistemas de seguridad y con señalización en cabina.
- permite reemplazar parcial o completamente la señalización a lo largo de la línea por señalización en cabina.
- puede equiparse en líneas secundarias, principales y de alta capacidad en diferentes niveles de acuerdo con las necesidades de operación.
- permite una instalación modular sobre una línea no equipada.

Por otra parte, la adopción de un sistema de control y protección ferroviario común en toda Europa conseguirá reducir costes de adquisición y mantenimiento debido al aumento en el número de unidades del mismo tipo que se fabricarán e instalarán. No debemos olvidar que en la actualidad cada fabricante tiene su propio sistema, con mucha frecuencia incompatible con los del resto. Se evitará tener que cambiar de unidades de tracción en las fronteras (produciendo incómodas y contraproducentes pérdidas de tiempo), o instalar en éstas varios equipos diferentes capaces de funcionar en cada uno de los países por los que circulará el tren (aumentando significativamente su coste). Si pensamos en estas dificultades: aumento de costes, incomodidad de los pasajeros, pérdida de tiempo en las fronteras, necesidad de personal dedicado a estas operaciones,..., vemos claramente que se pondrá en tela de juicio la viabilidad de una red de alta velocidad europea si no se toman medidas al respecto.

Por último, desde un punto de vista técnico, otro de los graves inconvenientes actuales con los que se encuentra la industria del sector ferroviario es que normalmente el equipo de tierra necesario fue diseñado con una larga vida útil y, además, es utilizado en una amplia zona geográfica, por lo que es siempre difícil introducir nuevos sistemas de control en el tráfico ferroviario. Además los nuevos equipos tienen unos costes de desarrollo muy elevados.

2.1.2. Objetivos del ECTS

El ETCS es un moderno sistema unificado de control de trenes elaborado en el marco de la cooperación internacional que implica al ferrocarril y a la industria, así como expertos de CCS (Control Command and Signaling). Como resultado, el ETCS implica el uso de tecnología punta y de vanguardia en lo que a transmisión de señales se refiere dentro del espacio de aire que comprende el tren y la vía. Esto lleva a una mejora en la seguridad y el rendimiento del sistema ferroviario en su conjunto.

1) Mejora en la Seguridad – Límites de funcionamiento precisos aportados por Autoridades de Movimiento

La señalización ferroviaria tradicional se basa principalmente en sistemas de enclavamiento y bloqueo, excluyendo rutas convergentes y asegurando el espaciamiento entre trenes. Los mensajes elaborados por esos sistemas se muestran por medio de señales en tierra como los códigos de color que han de ser respetados por los conductores. Los sistemas de control de trenes se han ideado para garantizar la recepción de esos mensajes y que la conducción cumpla con la distancia recibida y las limitaciones de velocidad.

De este modo, la transmisión vía-tren se utiliza para la transferir información de limitaciones de movimiento (velocidad máxima permitida, información de próximos puntos de ruta con peligro, etc.). Esta información de limitaciones de movimiento se envía a un tren para un espacio determinado de la ruta que este va a seguir (por ejemplo, para los 300 metros siguientes a la baliza que le ha enviado la información). A este espacio de la ruta se le llama Autoridad de Movimiento. Por lo tanto, una Autoridad de Movimiento define un lugar en una pista que no debe ser rebasado por un tren. Además otro tipo de información, como los perfiles de velocidad, también se transmite a bordo.

2) Mejora en la Seguridad - Supervisión de conducción de trenes

Los datos recibidos son utilizados por los equipos ETCS de a bordo para supervisar a los conductores de trenes. Para este fin el equipo de a bordo tiene que conocer no sólo los datos relacionados con la ruta sino también los datos relacionados con el tren. Por tanto, estos datos del tren pueden ser introducidos por el conductor del vehículo antes de iniciar el viaje.

En base a los datos de la pista y a los introducidos por el conductor, los equipos de a bordo calculan un conjunto de curvas de frenado para la supervisión del movimiento del tren.

3) Aumento de Rendimiento - Aumentar la velocidad y capacidad

Los sistemas basados en la transmisión vía-tren que supervisan los movimientos de trenes utilizan información precisa sobre los límites de funcionamiento de los vehículos. Por lo tanto, estos sistemas pueden supervisar un tren continuamente para evitar que se excedan dichos límites de velocidad.

Se utiliza el conocimiento de los equipos de a bordo acerca de los límites de funcionamiento del tren para informar a los conductores a través de pantallas, y asegurar así que pueden ver los límites independientemente de las condiciones meteorológicas. Además esto permite que los ferrocarriles aumenten las velocidades sin preocuparse por el acortamiento en el período de tiempo para la observación de la señal de tierra. Los ferrocarriles suelen utilizar este tipo de sistemas teniendo en cuenta el hecho de que el aumento de velocidad resulta también en un alargamiento de las distancias de frenado.

La transmisión de datos en los sistemas más avanzados se produce tanto en la dirección pista-tren como en la contraria. Esto se utiliza, por ejemplo, para informar a los sistemas de tierra sobre la posición del vehículo.

4) Interoperabilidad – Capacidad para operar en diferentes países

Como se puede ver en [5], los sistemas de señalización basados en señales a un lado de la pista utilizan diferentes códigos de color, dependiendo de las distintas normas nacionales que se apliquen (Ilustración 2). Asimismo se han diseñado diferentes sistemas de control de trenes con paso de información desde el vehículo a la pista para diferentes países, e incluso para diferentes líneas, existiendo actualmente más de veinte sistemas diferentes en uso.

El ETCS funciona para las líneas pertenecientes a diferentes administraciones ferroviarias. Como resultado, ofrece una amplia gama de funciones de protección, diferente aplicación de niveles y diferentes configuraciones. Los valores nacionales garantizan la coherencia entre el comportamiento del sistema y las normas y reglamentos de los diferentes países.

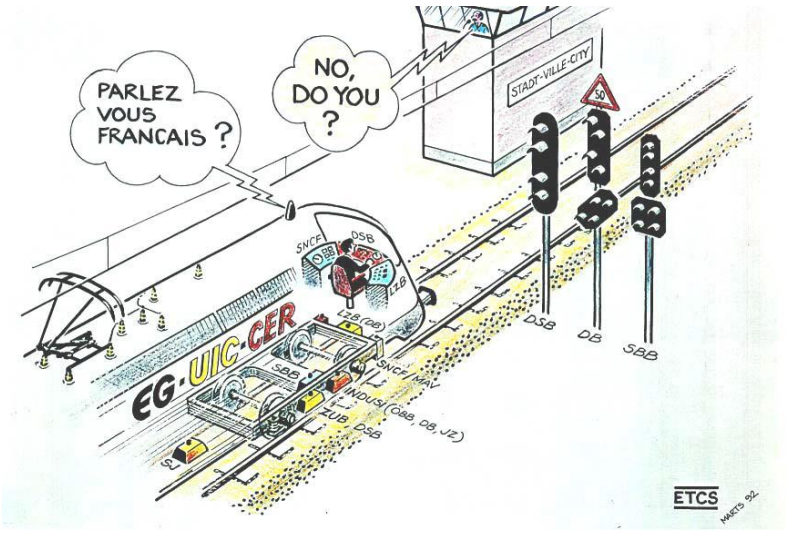

Ilustración 2 - Interoperabilidad

2.1.3. Funcionamiento básico del sistema

Como ya hemos dicho, ERTMS/ETCS es un sistema de control de trenes. Todo sistema de control de trenes debe incorporar un sistema de protección que asegure que:

- El tren no excede los límites de la ruta que le ha sido asignada.
- El tren no excede la velocidad máxima permitida en ningún punto de dicha ruta.

La velocidad máxima permitida a un tren se determina a partir de un conjunto de factores que influye en ella y están asociados a las condiciones de la vía: radios, gradientes,... y a las propiedades de los vehículos. Podemos obtener un perfil de velocidades asociado a cada uno de los parámetros que influyen en ella.

El ETCS debe asegurar que ningún tren excederá sus límites de velocidad. En términos generales, los valores de la velocidad máxima permitida se toman con márgenes superiores a los realmente necesarios, principalmente por razones de comodidad para el pasajero. No obstante, estos márgenes varían en función de las condiciones: cambios de vía en posición inversa o restricciones temporales de la velocidad.

Además se deben tener en cuenta las condiciones efectivas de frenado de cada tren. Estas condiciones determinan tanto las transiciones entre diferentes niveles de velocidad como la curva de frenado al final de la ruta. Al final de la autorización de movimiento de un tren, la velocidad permitida es cero.

El principio de funcionamiento básico para cualquier sistema de control de ferrocarriles es asegurar (mediante la comparación continua de las condiciones nominales con las condiciones reales) que la velocidad actual de cada tren no excede la velocidad permitida por encima de una tolerancia.

En los sistemas de protección más rudimentarios los perfiles de velocidad permitida se especifican en instrucciones escritas y/o carteles de velocidad que se colocan a lo largo de la vía. Todo el trabajo de comparar los valores de velocidad permitida y velocidad real recae en el conductor, si bien en zonas críticas como la aproximación a una señal con aspecto de peligro se requiere monitorización técnica.

Con ETCS el perfil de velocidad permitida se calcula desde los datos de la infraestructura y del tren para la totalidad de la ruta establecida. Cuantos más datos estén disponibles, más acertada será la utilización de las propiedades de la vía.

Por todo ello, ETCS es un sistema de protección con la capacidad de incorporar control automático de trenes utilizando sistemas convencionales o sistemas de bloqueo móvil. Tiene componentes y funciones especialmente diseñados para cumplir con los requerimientos de seguridad de la señalización. Está diseñado para ayudar al conductor a conducir el tren de forma segura. Debe funcionar a una velocidad máxima de 500 km/h [5].

No todas las funciones que proporciona son necesarias en todas las redes de ferrocarriles, dependerá de las diferentes aplicaciones del sistema.

2.1.4. Niveles de funcionamiento

Dentro de la filosofía general del ERTMS/ETCS está el deseo de posibilitar que las distintas administraciones que opten por este sistema puedan elegir entre diferentes alcances en su implantación para adaptarse tanto a las necesidades de diferentes líneas como a las diferentes posibilidades de inversiones. Por ello se establecen cinco posibles niveles de operación del ERTMS/ETCS: 0, 1, 2, 3 y STM (Ilustración 3).

Las diferencias entre ellos son sustanciales y hacen referencia tanto a los equipos embarcados como a los equipos en vía, pero tienen en común las características necesarias para su absoluta compatibilidad que sólo está limitada por razones técnicas insalvables.

Haremos una breve descripción de las características de cada uno de los niveles así como de los equipos necesarios para su implantación, dentro de la concepción integral del ERTMS/ETCS como sistema de control y protección de ferrocarriles.

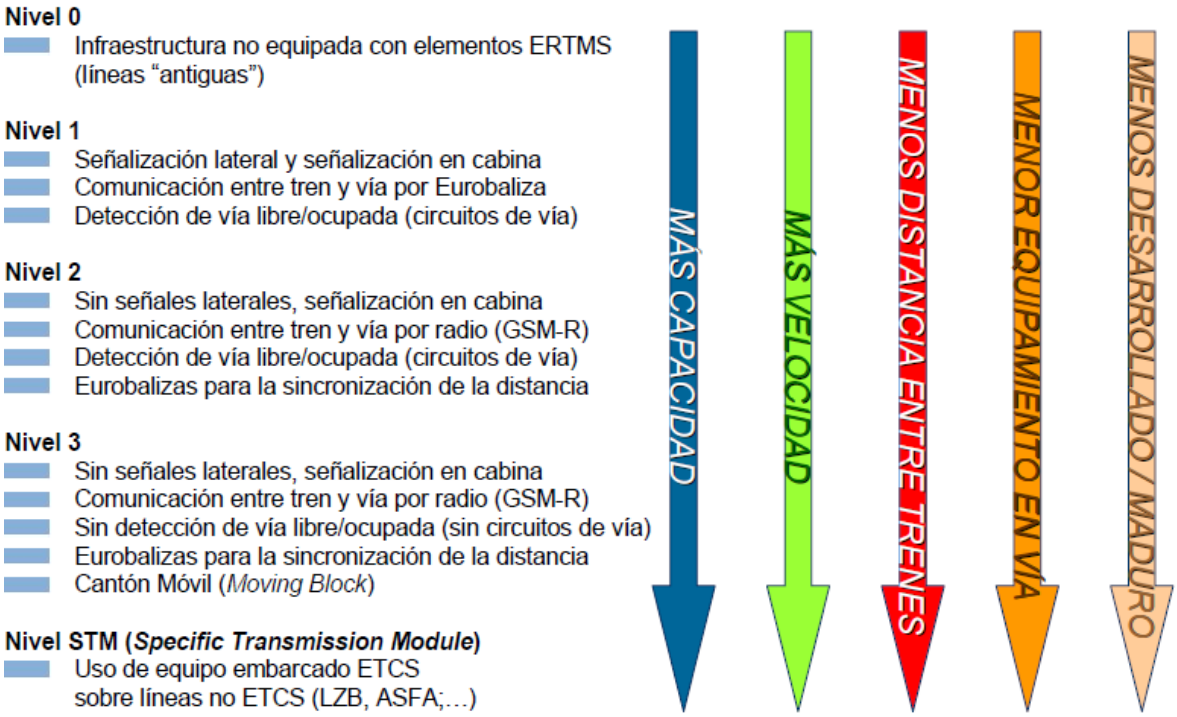

Ilustración 3 - Resumen niveles ERTMS/ ETCS

Nivel 0

El nivel 0 se utiliza en situaciones nada corrientes dentro del contexto ERTMS/ETCS. Más bien se refiere a situaciones en las que no nos encontramos por entero dentro del ERTMS/ETCS. Así, por ejemplo, algunas de las situaciones en las que este nivel tiene utilidad pueden ser las siguientes:

a) Tren equipado con el ERTMS/ETCS que circula por una línea que no esté equipada con el ERTMS/ETCS. Y en el caso de estar la línea equipada con cualquier otro

sistema de señalización, el equipo embarcado no dispone del STM compatible con el citado sistema de señalización.

b) Tren equipado con el ERTMS/ETCS que circula por una línea que está equipada con el ERTMS/ETCS, pero en la cual por cualquier razón (obras, mantenimiento,. . .) las funciones necesarias del ERTMS/ETCS no están operativas.

Las autorizaciones de movimiento necesarias para que el conductor conozca en cada momento el punto hasta el cuál puede avanzar y con qué velocidades se facilitan utilizando señalización externa, normalmente señales o carteles colocados en la vía.

Cuando el tren circula con el equipo embarcado del ERTMS/ETCS funcionando en este nivel su velocidad se encuentra limitada por un valor máximo que es un valor nacional, es decir, cuyo valor no está fijado por las especificaciones ERTMS/ETCS sino por cada una de las administraciones ferroviarias. El equipo embarcado del ERTMS/ETCS supervisa que el tren en su circulación en este nivel no supere esta velocidad cuyo valor por defectos es 100 km/h.

La detección del tren y la supervisión de su integridad la realizan los equipos de vía habituales, que se sitúan fuera del alcance del ERTMS/ETCS, pero con los cuales interacciona estrechamente: enclavamientos y circuitos de vía. Los únicos dispositivos para la transmisión de datos que se utilizan son las *eurobalizas*, necesarias para anunciar transiciones de nivel; el resto de informaciones que pudieran llegar al equipo embarcado del ERTMS/ETCS se ignorarán.

Así pues podemos resumir diciendo que el equipo en vía del ERTMS/ETCS consiste exclusivamente en eurobalizas para transmitir hipotéticas órdenes de transición de nivel. Por su parte el equipo embarcado del ERTMS/ETCS, además del EVC (European Vital Computer), debe disponer al menos de una antena para la recepción de los datos enviados por las *eurobalizas* y se encarga de la supervisión de la velocidad máxima permitida, que es el mínimo de la velocidad máxima del tren y la velocidad máxima en nivel 0.

Nivel 1.

El nivel 1 del ERTMS/ETCS es un sistema de control de ferrocarriles con transmisión puntual y que funciona como soporte de un sistema de señalización subyacente.

Las autorizaciones de movimiento se generan en el equipo de vía del ERTMS/ETCS y se transmiten a los distintos trenes a través de las *eurobalizas*. La supervisión de la velocidad máxima permitida y del final de la autorización de movimiento es continua.

La detección del tren y la supervisión de su integridad se siguen realizando fuera del alcance del ERTMS/ETCS por los equipos habituales: enclavamientos y circuitos de vía. El equipo de vía del ERTMS/ETCS no conoce en ningún momento la identidad del tren para el cual está elaborando las autorizaciones de movimiento y el resto de informaciones sobre la vía. Todos los datos se elaboran para colocarlos en una *eurobaliza* y que los reciban los trenes capaces de procesar esa información cuando pasen por encima de ella antes de que dicha información cambie.

Una consecuencia importante que se deriva de la transmisión discontinua utilizada en este nivel es la necesidad de prever una *Release Speed* o velocidad de aproximación a un punto de parada. Para comprender esta necesidad pensemos en un tren que recibe una autorización de movimiento que le obliga a parar en la siguiente señal en rojo. El tren se verá obligado a detenerse en esta señal antes de llegar a la baliza que tiene asociada debido a ese error odométrico. Cuando el aspecto de esta señal cambie a verde el tren debería seguir avanzando para entrar en comunicación con la baliza, pero el equipo embarcado no permite avanzar al tren por encontrarse ante un punto de parada. Para que el movimiento sea posible se utiliza la velocidad de aproximación a un punto de parada o *Release Speed*, por debajo de la cual se suprime la supervisión. De esta forma el conductor puede, bajo su responsabilidad, pasar por encima de la baliza a una velocidad superior a la permitida, pero inferior a la *Release Speed*.

Resulta evidente que durante el tiempo en que el conductor esté haciendo uso de la *Release Speed* la supervisión desaparece por debajo de esta velocidad y la responsabilidad recae por entero en el conductor. Esto puede resultar de algún modo inseguro y por ello se plantean otras soluciones que permiten prescindir de la *Release Speed* e incluso de la señalización en vía. Se consigue utilizando señalización semi-continua, que se proporciona mediante *eurolazos* o equipos de radio *in-fill*. Ambos dispositivos permiten disponer de una zona en la cual la comunicación es continua en las proximidades de la señal.

Así pues, los equipos instalados en vía del ERTMS/ETCS serán las *eurobalizas* con información fija o variable (conectadas al LEU, *Lineside Electronic Unit*) y, en caso de disponer de transmisión semicontinua, los equipos de comunicación necesarios. La principal función que realiza el equipo en vía del ERTMS/ETCS es la elaboración de las autorizaciones de movimiento en función de los datos facilitados por el sistema de señalización subyacente, y enviar estas autorizaciones de movimiento junto con los datos de la vía necesarios a los trenes. Por su parte el equipo embarcado del ERTMS/ETCS debe disponer de los dispositivos de transmisión capaces de recibir la información de los equipos de la línea. Las principales funciones de este equipo son la recepción de las autorizaciones de movimiento y de los datos de la vía, la elección del valor más restrictivo de la velocidad para cada punto de la vía, el cálculo del perfil dinámico de velocidades a partir de las características de frenado del tren y de los datos de la vía, la supervisión continua de la velocidad comparándola con la máxima velocidad permitida y, en caso de que sea necesario, la aplicación de los frenos y por último la señalización en cabina (Ilustración 4).

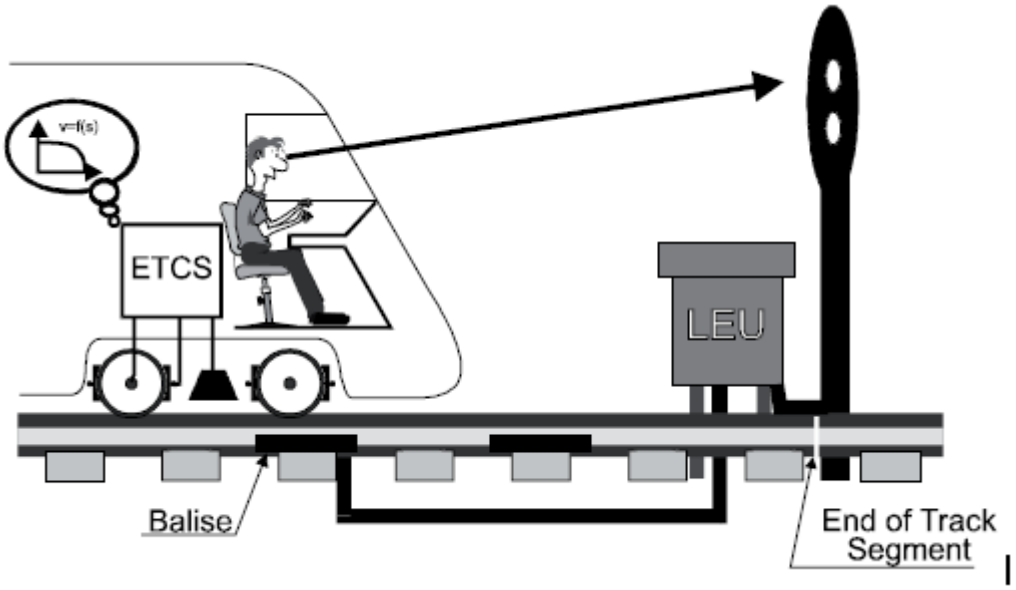

Ilustración 4 - ERTMS/ ETCS: Nivel 1

Nivel 2

El nivel 2 del ERTMS/ETCS es un sistema de control de ferrocarriles con transmisión continua por radio que funciona como soporte de un sistema de señalización subyacente.

Las autorizaciones de movimiento se generan en el equipo de vía del ERTMS/ETCS y se transmiten a los distintos trenes vía *euroradio*. La supervisión de la velocidad máxima permitida y del final de la autorización de movimiento es continua. Además de la *euroradio* para la comunicación entre tren y vía también se utilizan *eurobalizas*, principalmente para ser utilizadas como puntos de referencia en el posicionamiento de los trenes.

La detección del tren y la supervisión de su integridad se siguen realizando fuera del alcance del ERTMS/ETCS por los equipos habituales: enclavamientos y circuitos de vía.

El RBC o Centro de Bloqueo por Radio conoce individualmente cada uno de los trenes que circulan dentro de su área de influencia bajo el control del sistema ERTMS/ETCS, de forma que toda la información que proporciona a los trenes: autorizaciones de movimiento, datos de la vía, etc., son elaborados expresamente para el tren al cuál se van a transmitir. El RBC identifica a dada uno de estos trenes por su identificador ERTMS/ETCS.

Por supuesto con l a comunicación continua la señalización lateral deja de ser necesaria, y tampoco lo es la *Release Speed*.

Concretando, los equipos del ERTMS/ETCS instalados en vía deben incluir: RBC, equipos para la comunicación por radio bidireccional (GSM-R), y *eurobalizas* para ser utilizadas como referencia en el posicionamiento de los trenes (Ilustración 5). Las principales funciones que desarrollan estos equipos son: identificar a cada uno de los trenes que circulan bajo ERTMS/ETCS por su identificador ERTMS/ETCS, seguir la posición de cada tren que se encuentre dentro del área de influencia del RBC, determinar las distintas autorizaciones de movimiento de acuerdo con el sistema de señalización, transmitir estas autorizaciones junto a

los datos de la vía a cada uno de los trenes y, por último, controlar la circulación de los trenes en las zonas de transición entre RBCs.

Por su parte los equipos embarcados deben incluir los dispositivos para la comunicación vía *euroradio* y *eurobaliza*. Las principales funciones que desempeña este subsistema son: leer las *eurobalizas* y enviar su posición relativa a las balizas detectadas al RBC, recibir las autorizaciones de movimiento y de los datos de la vía, la elección del valor más restrictivo de la velocidad para cada punto de la vía, el cálculo del perfil dinámico de velocidades a partir de las características de frenado del tren y de los datos de la vía, la supervisión continua de la velocidad comparándola con la máxima velocidad permitida y, en caso de que sea necesario, la aplicación de los frenos y la señalización en cabina al conductor.

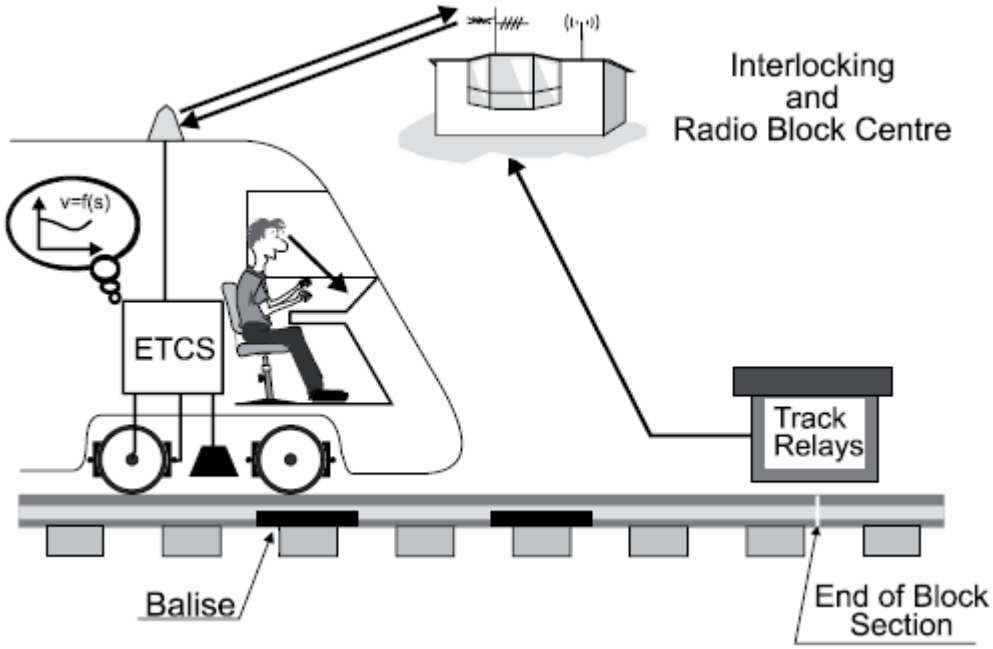

Ilustración 5 - ERTMS/ETCS: Nivel 2

Nivel 3

El nivel 3 del ERTMS/ETCS es un sistema de control de ferrocarriles con transmisión continua por radio que no necesita para su funcionamiento un sistema de señalización subyacente. La supervisión de la velocidad máxima permitida y del fin de la autorización de movimiento sigue siendo continua.

Las autorizaciones de movimiento se generan en el equipo de vía del ERTMS/ETCS y se transmiten a los distintos trenes vía *euroradio*. Además de la *euroradio* para la comunicación entre tren y vía también se utilizan *eurobalizas*, principalmente para ser utilizadas como puntos de referencia en el posicionamiento de los trenes.

La posición del tren y la supervisión de su integridad la realiza el RBC en cooperación con el tren que le envía informes de posición e información de la integridad del tren.

El RBC o centro de bloqueo por radio conoce individualmente cada uno de los trenes que circulan dentro de su área de influencia bajo el control del sistema ERTMS/ETCS, de forma que toda la información que proporciona a los trenes: autorizaciones de movimiento, datos de la vía, etc., son elaborados expresamente para el tren al cuál se van a transmitir. El RBC identifica a dada uno de estos trenes por su identificador ERTMS/ETCS.

Es decir, los equipos del ERTMS/ETCS instalados en vía coinciden con los citados para el nivel 2. Las principales funciones que desarrollan estos equipos son también idénticas, pero además el RBC debe encargarse del bloqueo y liberación de rutas utilizando para ello la información recibida de los trenes. Por su parte los equipos embarcados añaden a los dispositivos necesarios en nivel 2 un sistema que permita monitorizar la integridad del tren y enviársela al RBC (Ilustración 6). Este dispositivo se encuentra fuera del alcance del ERTMS/ETCS.

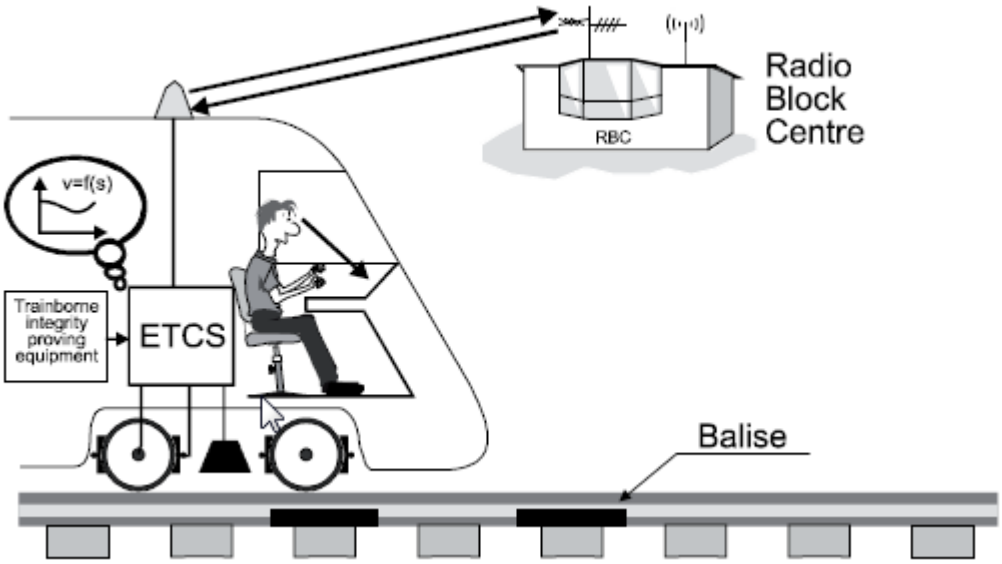

Ilustración 6 - ERTMS/ETCS: Nivel 3

Nivel STM

El nivel STM sirve para circular en una línea equipada con un sistema nacional con un tren equipado con el ERTMS/ETCS. El STM permite reconocer los datos facilitados por el sistema nacional y traducirlos a información que pueda manejar el ERTMS/ETCS. Evidentemente depende muy estrechamente del sistema de que se trate.

2.2. Elementos de campo: Baliza

A continuación se expondrá brevemente el sistema denominado baliza, más concretamente el llamado *eurobaliza*.

Hay que tener en cuenta que una baliza contemplada de una forma aislada no es más que un dispositivo físico (Ilustración 7) con una serie de prestaciones electromagnéticas, que es necesario integrar dentro de un sistema que se encargue de recibir, decodificar, procesar y ejecutar órdenes en función de la información recibida. Para elaborar parte de la documentación de este apartado se ha utilizado [6] y [7].

Ilustración 7- Eurobaliza en posición de vía

2.2.1. Arquitectura del Sistema

El sistema de transmisión por *eurobalizas* es un sistema de transmisión puntual que transporta la información de forma segura entre la infraestructura de la vía y el tren. Más información de otros sistemas de señalización, entre ellos las balizas, se puede encontrar en [8] y [9].

La información transmitida por una baliza hacia el equipo de transmisión de a bordo de un tren puede ser fija o variable, según las necesidades concretas de la aplicación, y siempre haciendo referencia a la transferencia de información desde la baliza al tren (enlace de subida o ascendente).

La información se suministra al tren únicamente por la Unidad de Antena, por ejemplo por microondas [10]. La longitud del tramo en la que la información se transmite y se recibe es aproximadamente de un metro por baliza.

El sistema de transmisión de la *eurobaliza* se plantea para el uso en todos los niveles de aplicación definidos dentro de la norma del ERTMS/ETCS (nivel 0, 1, 2, 3 y nivel STM respectivamente).

Unidades que componen el sistema y Funciones

El sistema de transmisión por *eurobaliza* consiste en la baliza de vía y en el equipo de transmisión de a bordo. Las balizas pueden ser fijas o controladas (de información variable).

El equipo de transmisión de a bordo está formado por la Unidad de Antena y las funciones del BTM (Balise Transmission Module).

La señalización de vía consiste en el LEU (Lineside Electronic Unit) y otros equipos externos involucrados en el proceso de señalización lateral (Ilustración 8).

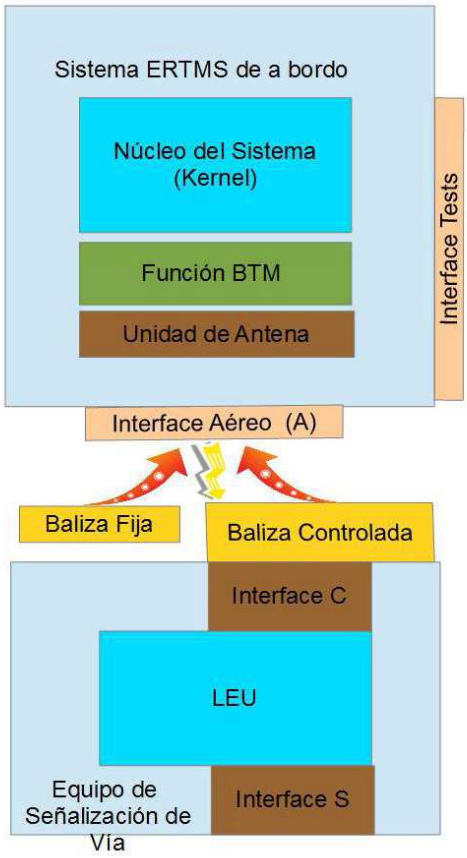

Ilustración 8 - Sistema Eurobaliza

2.2.2. Funciones Básicas

El sistema de transmisión con *eurobalizas* comprende las funciones siguientes básicas:

Equipo transmisor de a bordo con las funciones:

- Generación de la señal de Telealimentación.
- Control de la señal de Telealimentación y de detectabilidad de la baliza.
- Detección de balizas transmisoras.
- Demodulación y filtrado de la señal de subida
- Protección física contra el *Cross-talk* (diafonía)
- Prevención física de la transmisión de los lóbulos laterales y/o la gestión de los efectos de estos.
- Inmunidad al ruido ambiente.
- Comprobación de entrada de datos de subida respecto de los requerimientos de la codificación.
- Detección y decodificación del tipo de telegrama.
- Extracción de los datos de usuario
- Filtrado de telegramas
- Control del telegrama de subida con la baliza de contenido conmutado

- Validación de los datos de salida con los del temporizador y del odómetro
- Soporte para localización de balizas. Vitales y no vitales
- Gestión del temporizador y del odómetro.
- Detección de errores de bits
- Test de inicio

Funcionalidades opcionales en el equipo de transmisión de a bordo:

- Recepción de Señal KER *Up-link*
- Decodificación y comprobación de los datos de subida del KER
- Reporte de datos de subida del KER
- Conmutación del modo de TeleAlimentación (CW, modulación *toggling*)
- Control de la protección de *Cross-talk* lógico
- Auto test

Funcionalidades de subida de la señal desde la baliza:

- Recepción de Señal de TeleAlimentación
- Generación de la señal de subida
- Gestión de Datos
- Selección de modo al inicio
- Limitación del campo en subida (por ejemplo, corriente en la Baliza)
- Soporte a la programación y a la gestión de modo operacional/programación.
- Recepción de datos desde el interface 'C'
- Control de características de Entrada/Salida
- Protección de *Cross-talk* frente a otros cables
- Generación de señal de bloqueo de conmutación de telegramas (opcional)

2.3. Simuladores

Los simuladores de elementos de campo ferroviarios son un conjunto de aplicaciones software que tienen un largo recorrido. Así se pueden encontrar aplicaciones que simulan ciertos elementos en casi todas las áreas del sistema ETCS. Un ejemplo de ello se encuentra en [11], donde se puede observar un simulador enmarcado en el mismo proyecto que el que se presenta en esta memoria pero de otro tipo de sistemas de enclavamiento ferroviarios.

CAPÍTULO 3

REQUERIMIENTOS DEL SISTEMA

3. REQUERIMIENTOS DEL SISTEMA

Este capítulo recoge y elabora los requisitos marcados por la empresa THALES S.A. para el desarrollo del software de simulación Se han clasificado en dos grandes apartados en función de sus características: Generales y Específicos.

3.1. Requerimientos generales

Los requerimientos generales vienen derivados del resto de simuladores de otros elementos ferroviarios de campo que están siendo o han sido realizados en el marco del proyecto de colaboración con la empresa. Se han numerado para facilitar posteriormente su identificación.

3.1.1. Requerimientos generales

- R1.1: La aplicación estará dividida en cuatro partes diferenciadas: la vista gráfica de BDs, el Espía de mensajes, las Propiedades de simulación y el Filtro de trazas.

- R1.2: La vista gráfica de BDs mostrará la monitorización de las tarjetas y para cada una de ellas sus canales A/B, los comandos y las indicaciones.

- R1.3: El Espía de mensajes permitirá guardar un log de los mensajes monitorizados en un fichero de texto plano.

- R1.4: En Propiedades de la simulación se mostrará la conexión de CEC y/o BD, permitiendo aplicar un filtro a los mensajes y cerrar la recepción de información desde los Canales A/B del Can Bus.

- R1.5: Se permitirá ejecutar acciones sobre las tarjetas tales como apagarlas o desconectar sus canales correspondientes.

- R1.6: El Filtro de trazas mostrará las trazas de los mensajes que hayan sido filtrados.

3.1.2. Requerimientos gráficos

- R2.1: En la vista gráfica de BDs, se representará mediante leds simulados las acciones de las tarjetas (si están conectadas, apagadas o en proceso de conexión).

- R2.2: En la vista gráfica de BDs, el usuario puede seleccionar cualquiera de las tarjetas representadas y esto determinará qué BD será simulada y qué trazas se visualizarán en el Espía de mensajes.

- R2.3: En Propiedades de la simulación, el usuario podrá activar o desactivar los canales A/B y, en consecuencia, el envío y la recepción de mensajes de todas las tarjetas del proyecto. Por defecto, estos valores estarán activos.

- R2.4: La representación de las BDs para los elementos de tipo Balise debe mostrar qué canales están activos mediante la implantación de 2 LED simulados.

3.1.3. Requerimientos del espía de mensajes

- R3.1: Los mensajes recibidos desde los CEC serán automáticamente almacenados en un buffer interno con una capacidad para 200 mensajes.

- R3.2: Si un mensaje corresponde con la selección del usuario de la vista gráfica será automáticamente mostrado en la ventana de vista de mensajes.

- R3.3: El formato de los mensajes registrados identificará los siguientes campos :
  - **Canal:** canal de comunicación en uso
  - **Origen:** origen del mensaje
  - **Fecha y hora:** Momento en el que el mensaje es enviado/recibido
  - **Tamaño del mensaje:** número de *bytes* que componen el mensaje
  - **Datos del mensaje:** los datos que contiene el mensaje en su totalidad
  - **Nombre del mensaje:** nombre del mensaje

3.1.4. Requerimientos del filtro de trazas

- R4.1: En el filtro de trazas podremos elegir si queremos ver los mensajes correspondientes a todas las BD's, a una en concreto, o no ver ningún mensaje.

- R4.2: En el filtro de trazas podremos filtrar aquellos mensajes que cumplan unos valores concretos en sus bits.

3.1.5. Requerimientos de propiedades de la simulación

- R5.1: Permitirá simular el fallo físico de una tarjeta determinada y ésta se parará, rechazando todos los mensajes entrantes, no enviando más mensajes, no estando disponible y actualizando los correspondientes LED's en la vista gráfica de BD's.

- R5.2: Permitirá reiniciar un canal y en la vista gráfica se simulará la recuperación de este fallo físico.

- R5.3: Permitirá cambiar la dirección de una tarjeta determinada variando el valor de las direcciones de sus canales A y B.

3.2. Requerimientos específicos

En cuanto a requisitos específicos, se han tenido en cuenta aquellos que han venido dados por el propio funcionamiento físico de la comunicación entre el CEC y el BD (Ilustración 1), así como el protocolo de dicha comunicación y sus características particulares. Es decir, se presentan en esta sección los requisitos propios del protocolo de comunicación que se ha implementado en el programa realizado para el proyecto.

Estos requisitos derivan del manual principal de referencia relacionado con el funcionamiento de la comunicación entre CEC y BD [12], por lo que se ha optado por mantener el original del texto en inglés, ofreciendo una breve explicación en castellano cuando sea necesario.

3.2.1. Variables *timer*

Los *timer* son la decisión de implementación que se ha tomado en el software para simular los *timeout*, tiempos que definen ciertas acciones dentro del protocolo tales como: tiempo de espera para obtener una confirmación de cierto mensaje, tiempo de envío de tramas, etc. En general suelen determinar *timeouts* de espera en el envío de mensajes entre CEC y BD.

En la tabla 1 se presentan los valores que toman cada uno de las variables de tiempo presentes en el protocolo.

Variable	Value (local BD)	Value (remote BD)	Definition
t_aspect	1,25s	1,25s	Sending period of a new aspect frame from the CEC to BD
to_aspect	3s	5s	Timeout for BD which defines the time to receive an aspect frame
t_status	t_roundtrip	1,25s	Sending period of a new status frame from the BD to CEC
to_status	5s	5s	Timeout for CEC which defines the time to receive an status frame
to_newmode	5s	5s	Timeout for BD to switch to the new mode
to_con_BD	25s	25s	Timeout to receive BD id, version and first/last error ID.
BD_timeout	5s	5s	Timeout in CEC to receive upload frame from BD in upload or acknowledgement from BD in download
CEC_timeout	5s	5s	Timeout in BD to receive download frame from CEC in download or acknowledgement from BD in upload
to_ack_frames	t_roundtrip	1,25s	Period of time for sending the acknowlege to CEC / BD during the download /upload process
to_DL_send	t_roundtrip	1,25s	Sending period of "frames_to_send" new DL frames
to_UL_send	t_roundtrip	1,25s	Sending period of "frames_to_send" new UL frames
DL_frames_to_send	4..7	4..7	Frames to send when to_DL_send expires
UL_frames_to_send	4..7	4..7	Frames to send when to_UL_send expires
max_not_acked_messages	12	12	Maximal number of not acknowledged messages to be in up / download process.
max_user_data	6 Byte	6 Byte	Maximum value how many bytes are transferred in one download or upload frame
max_number_mode_request	4	4	Maximum number of mode requests till BD is considered as missing

Tabla 1 - Variables *timer*

3.2.2. Estados de CEC

Los estados del CED se muestran en la Ilustración 7. Se han enumerado a continuación 19 transiciones de estado con una pequeña descripción de las mismas.

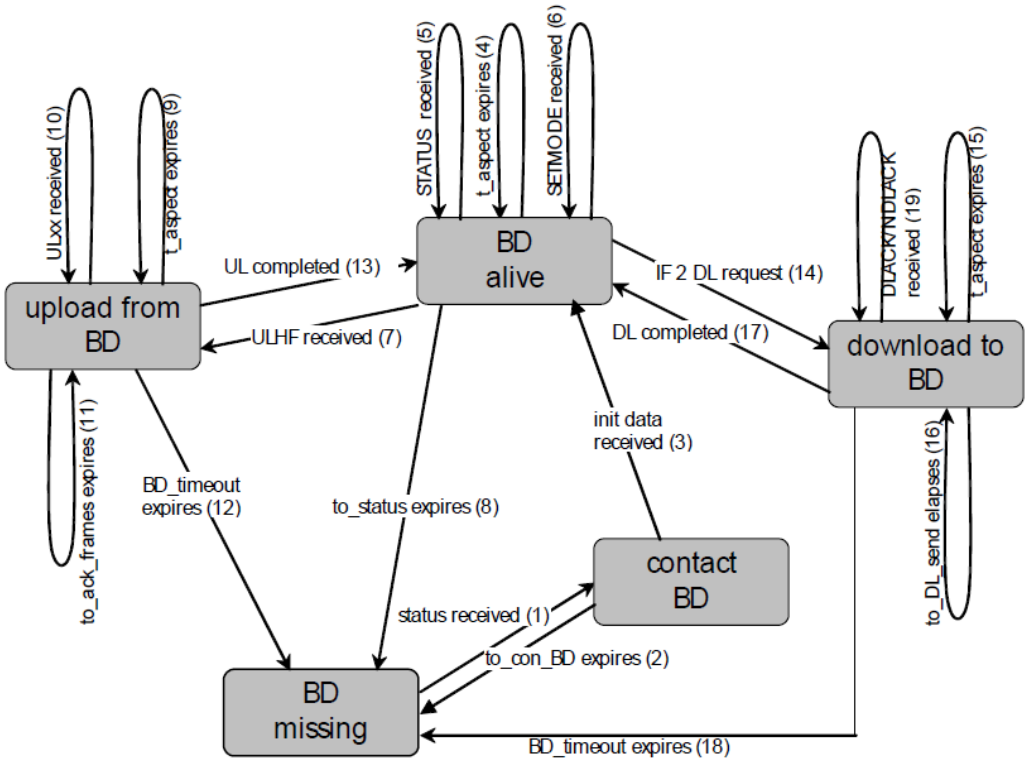

Ilustración 7 - Diagrama estados CEC

(1) **BD missing → contact BD (*status* recibido)**
(2) **Contact BD → BD missing (expira to_con_BD)**
(3) **Contact BD → BD alive (recibido datos de inicio)**
- t_aspect se pone a 0
- to_status se pone a 0
- CEC manda un mensaje de diagnostic por el IF2 para informar del cambio de estado de la BD a "BD *alive*"

(4) **BD alive □□BD alive (t_aspect expira)**
(5) **BD alive → BD alive (recibido trama STATUS)**
(6) **BD alive → BD alive (habiendo recibido trama SETMODE)**
(7) **BD alive → upload from BD (habiendo recibido trama ULHF)**
(8) **BD alive → BD missing (expira to_status)**
(9) **Upload from BD → upload from BD (*timer* t_aspect expira)**
(10) **Upload from BD → upload from BD (recibida trama ULxx)**
(11) **Upload from BD → upload from BD (trama to_ack expira)**
(12) **Upload from BD → BD missing (expira BD_timeout)**
(13) **Upload from BD → BD alive (se completa UL)**
(14) **BD alive → download to BD (petición de IF2 DL)**
(15) **Download to BD → download to BD (expira t_aspect)**
(16) **Download to BD → download to BD (transcurre to_DL_send)**
(17) **Download to BD → BD alive (descarga completada)**
(18) **Download to BD → BD missing (expira BD_timeout)**
(19) **Download to BD → download to BD (recibida trama DLACK/NDLACK)** -

3.2.3. Estados de BD

A continuación se muestran en la ilustración 8 los diagramas de estado de BD, que se comentan brevemente.

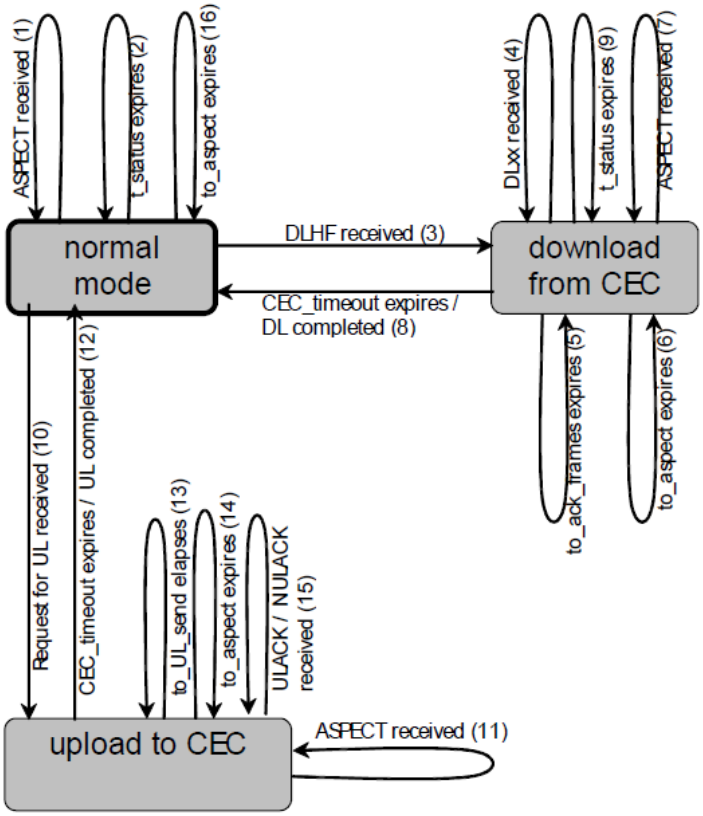

Ilustración 8 - Diagrama estados BD

(1) **Normal mode → normal mode (recibida trama ASPECT)**
(2) **Normal mode → normal mode (expira t_status)**
(3) **Normal mode → download from CEC (recibido DLHF)**
(4) **Download from CEC → download from CEC (DLxx recibido)**
(5) **Download from CEC → download from CEC (expira trama to_ack_frmaes)**
(6) **Download from CEC → download from CEC (expira to_aspect)**
(7) **Download from CEC → download from CEC (trama ASPECT recibida)**
(8) **Download from CEC → normal mode (DL completado/ CEC_timeout expira)**
(9) **Download from CEC→ Download from CEC (t_status expira)**
(10) **Normal mode → upload to CEC (petición para UL recibida)**
(11) **Upload to CEC → upload to CEC (trama ASPECT recibida)**
(12) **Upload to CEC → normal mode (UL completado / CEC_timeout expira)**
(13) **Upload to CEC → upload to CEC (to_UL_send transcurre)**
(14) **Upload to CEC → upload to CEC (to_aspect expira)**
(15) **Upload to BD → upload to BD (ULACK/NULACK recibida)**
(16) **Normal mode → normal mode (to_aspect expira)**

3.2.4. Modos de comunicación entre CEC y BD

En esta sección se mostrarán los principales modos de funcionamiento que pueden darse entre el CEC y la BD en condiciones óptimas. Debido a que existen una multitud de posibilidades de modos de operación se ha decidido ilustrar tres de ellos que comprenden distintas casuísticas: modo normal, fallo en un extremo, y modo normal en carga/descarga.

Aplicando variaciones a las posibilidades de funcionamiento mencionadas anteriormente se puede obtener una visión completa de las distintas formas de comunicación entre el CEC y la BD. Además se ofrece una explicación gráfica del importante papel que juegan los *timers* en la implementación del protocolo. Se han tomado algunas referencias de anomalías de funcionamiento de [13].

Modo normal de operación

La ilustración 9 muestra el modo de operación en condiciones normales que adquieren los elementos de BD y CEC. Se puede observar como el CEC envía periódicamente el aspecto (mensaje), que ha sido calculado de la información recibida previamente de IF2. El intervalo del *timer* se define en *t_aspect*. También la BD envía el estado propio de forma periódica al CEC. Este intervalo se encuentra definido en la variable *t_status*.

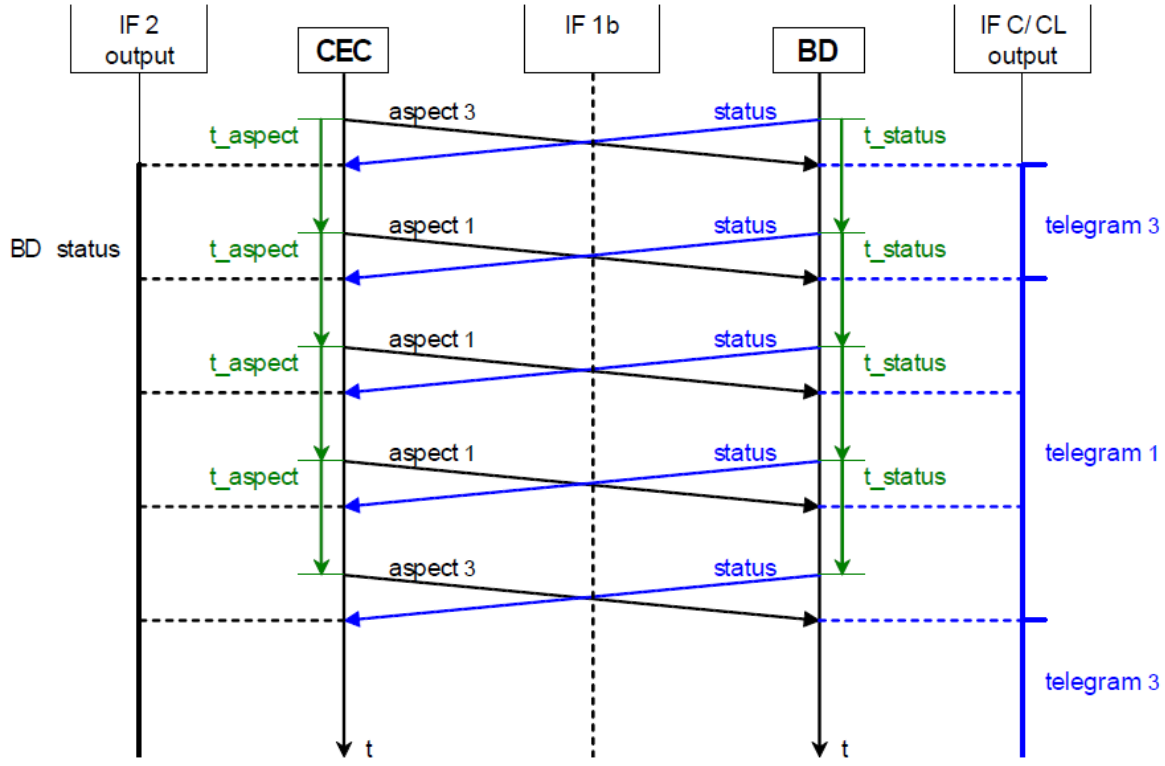

Ilustración 9 - Modo normal de operación

Modo de operación con caída de CEC

El CEC y la BD se encuentran funcionando de manera correcta al principio pero, después de un tiempo indeterminado, se produce una caída en el CEC (Ilustración 10).

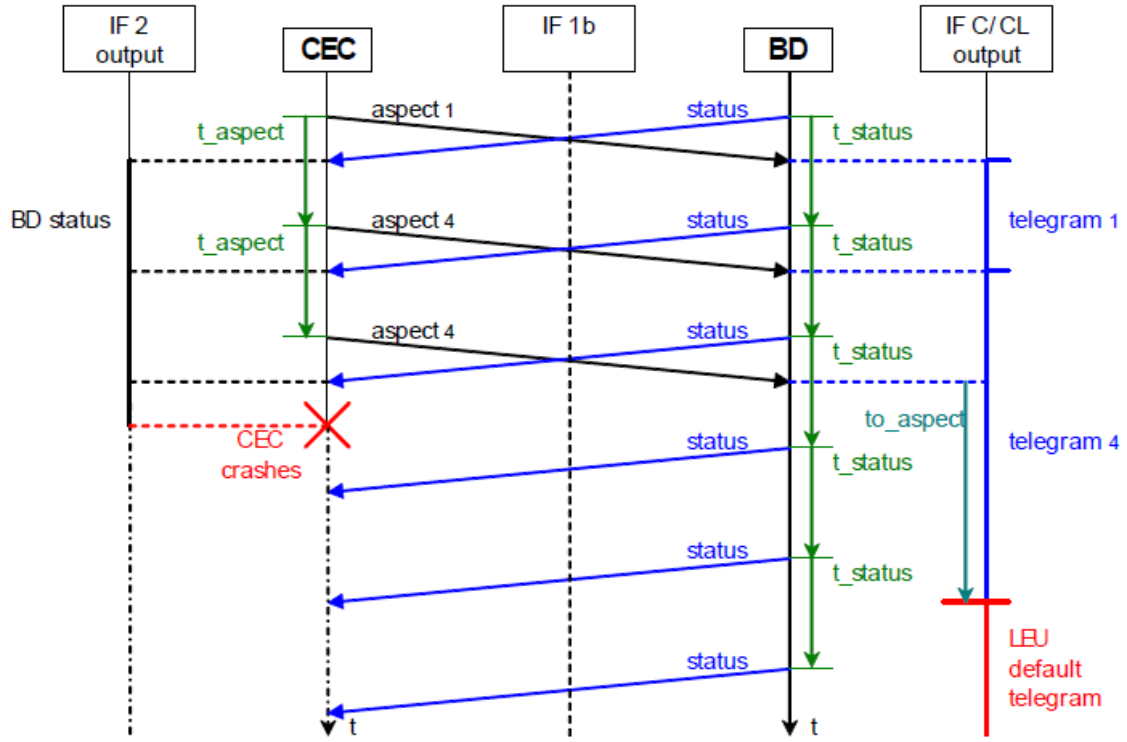

Ilustración 10 - Modo de operación con caída de CEC

Si la BD no recibe un *frame* durante un tiempo mayor que el definido en *t_aspect* desde el CEC, enviará el mensaje “LEU default telegram” a la salida IF C/CL y vuelve a mandar de forma periódica su status.

Descarga del CEC a BD

Se transfiere información de CEC a BD. Además se implementa un mecanismo de recuperación de información perdida durante la transmisión (Ilustración 11).

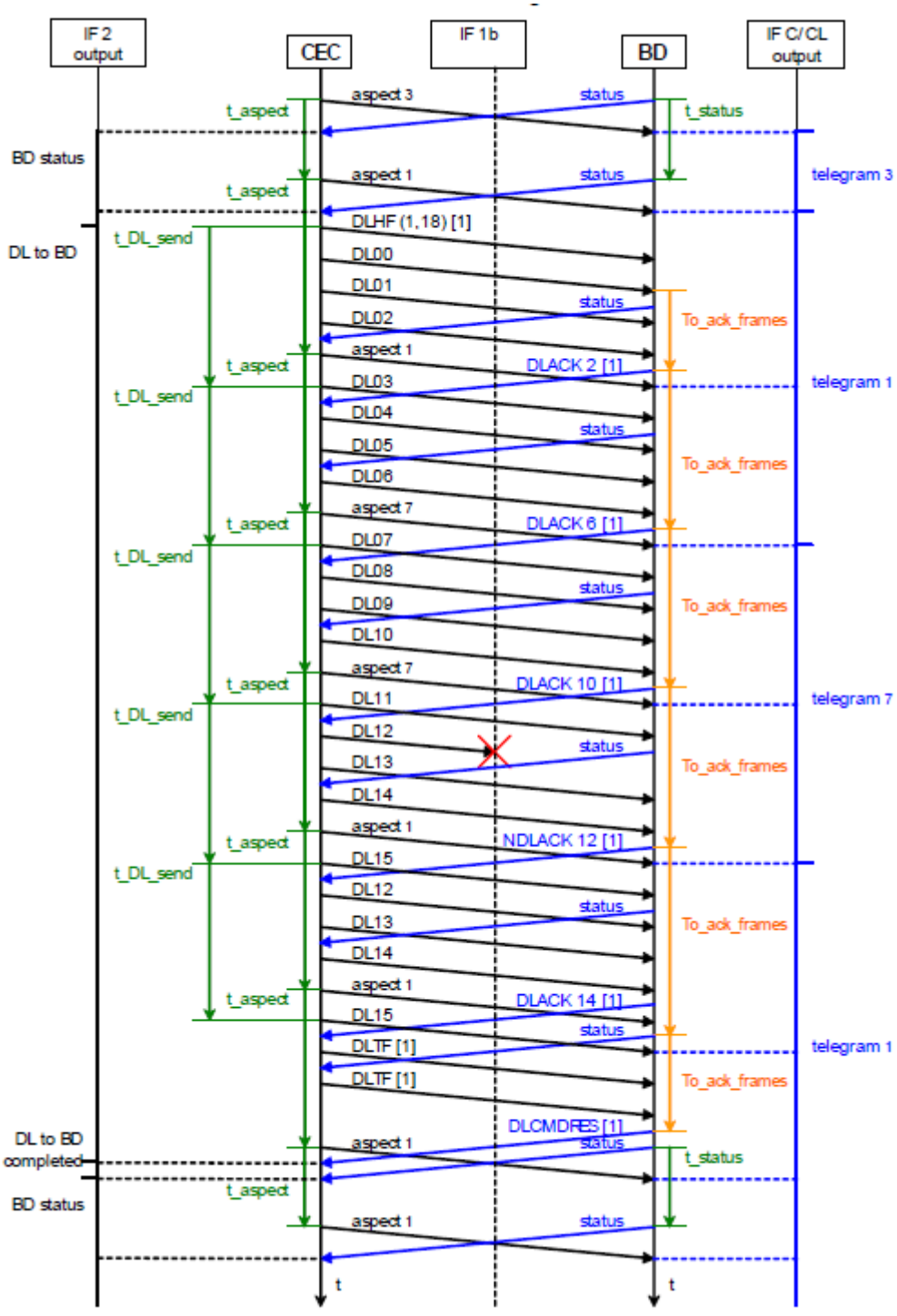

Ilustración 11 - Modo descarga

El CEC inicializa la conexión de descarga mandando la trama DLHF (*DownLoad Header Frame*), tras la que se procederá a mandar periódicamente las tramas de descarga estándar (DLxx). Durante la descarga se mandan mensajes de status desde la BD, a la vez que son mandadas tramas de aspecto por el CEC para permitir el correcto funcionamiento en modo operativo

"normal" una vez haya terminado la descarga en modo "transición". Si la descarga se produce en modo "mantenimiento", el C-Interface se apaga y el envío de los aspectos se torna irrelevante. Tras recibir la última trama de descarga el BD vuelve al modo "normal" de operación y envía de forma periódica su status.

3.2.5. El mensaje

El mensaje utilizado en el protocolo consta de 8 bytes repartidos de la manera siguiente.

1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte
BDID	message type	message data	message data	message data	message data	message data	message data

- **BDID:** es el ID de la controladora de baliza (BD). Este byte puede ser usado también para dirigir el mismo mensaje a todas las BD's (00xF), con la restricción de que sólo se puede dirigir el mensaje a un máximo de 255 BD's de forma directa.
- **Message type:** clasifica el mensaje enviado
- **Message data:** información propia del mensaje a transferir en la comunicación.

Tipos de mensajes

A continuación se muestra una relación de los posibles mensajes que pueden producirse en la comunicación entre la BD y el CEC (Tabla 2). La totalidad de estos mensajes han sido implementados en el software desarrollado en el proyecto, así como las posibles acciones que puedan disparar en los dispositivos que toman parte en dicha comunicación.

Type	Description	Length
0x00	SIF1BCONF ..... Set IF1b configuration parameters	8 Bytes
0x01	SMODE ..... Set BD mode	7 Bytes
0x02	GID ..... Request LEU ID	5 Bytes
0x03	GLERRID..... Request last error ID which is available	5 Bytes
0x04	GERRIDA ..... Request one error by ID from channel A	8 Bytes
0x05	GERRIDB ..... Request one error by ID from channel B	8 Bytes
0x06	GVER..... Request SW version	5 Bytes
0x07	GTELVA ..... Request (major) telegram version from channel A	4 Bytes
0x08	GTELVB ..... Request (major) telegram version from channel B	4 Bytes
0x09	TRANSCMD ..... Transaction handling	7 Bytes
0x0B	GTELCRC ..... Request CRC of telegram storage	5 Bytes
0x0C	GVARCRC..... Request CRC of variable storage	5 Bytes
0x0D	GHWCONF..... Get the hardware configuration	5 Bytes
0x0E	GFERRID ..... Request first error ID which is available	5 Bytes
0x0F	STIMEA ..... Set time on BD from channel A	8 Bytes
0x10	STIMEB ..... Set time on BD from channel B	8 Bytes
0x11	GVARA ..... get variable from channel A	6 Bytes
0x12	GVARB ..... get variable from channel B	6 Bytes
0x13	GFFVER ..... get the file format version from both channels	5 Bytes
0x14	SFFVERA ..... set the file format version of channel A	6 Bytes
0x15	SFFVERB ..... set the file format version of channel B	6 Bytes
0x16	GFTELVA ..... Request full (major and minor) telegram version from channel A	4 Bytes
0x17	GFTELVB ..... Request full (major and minor) telegram version from channel B	4 Bytes
0x18	DYNTELCMD ..... Dynamic telegram command	8 Bytes
0x19	DYNTELEXE ..... Dynamic telegram execution	8 Bytes
...	*** UNUSED ***	
0x20	ASPECT00 ..... New aspect to be transmitted on the CIF (implicit aspect id 0)	8 Bytes
0x21	ASPECT01 ..... New aspect to be transmitted on the CIF (implicit aspect id 1)	8 Bytes
0x22	ASPECT02 ..... New aspect to be transmitted on the CIF (implicit aspect id 2)	8 Bytes
0x23	ASPECT03 ..... New aspect to be transmitted on the CIF (implicit aspect id 3)	8 Bytes
0x24	ASPECT04 ..... New aspect to be transmitted on the CIF (implicit aspect id 4)	8 Bytes
0x25	ASPECT05 ..... New aspect to be transmitted on the CIF (implicit aspect id 5)	8 Bytes
0x26	ASPECT06 ..... New aspect to be transmitted on the CIF (implicit aspect id 6)	8 Bytes
0x27	ASPECT07 ..... New aspect to be transmitted on the CIF (implicit aspect id 7)	8 Bytes
0x28	ASPECT08 ..... New aspect to be transmitted on the CIF (implicit aspect id 8)	8 Bytes
0x29	ASPECT09 ..... New aspect to be transmitted on the CIF (implicit aspect id 9)	8 Bytes
0x2A	ASPECT10 ..... New aspect to be transmitted on the CIF (implicit aspect id 10)	8 Bytes
0x2B	ASPECT11 ..... New aspect to be transmitted on the CIF (implicit aspect id 11)	8 Bytes
0x2C	ASPECT12 ..... New aspect to be transmitted on the CIF (implicit aspect id 12)	8 Bytes
0x2D	ASPECT13 ..... New aspect to be transmitted on the CIF (implicit aspect id 13)	8 Bytes
0x2E	ASPECT14 ..... New aspect to be transmitted on the CIF (implicit aspect id 14)	8 Bytes
0x2F	ASPECT15 ..... New aspect to be transmitted on the CIF (implicit aspect id 15)	8 Bytes
0x30 - 0x5E	used for inter BD communication	
0x5F	DLHF ..... download header frame	7 Bytes
0x60	DL00 ..... download frame (implicit sequence number 0)	8 Bytes
0x61	DL01 ..... download frame (implicit sequence number 1)	8 Bytes
0x62	DL02 ..... download frame (implicit sequence number 2)	8 Bytes
0x63	DL03 ..... download frame (implicit sequence number 3)	8 Bytes
0x64	DL04 ..... download frame (implicit sequence number 4)	8 Bytes
0x65	DL05 ..... download frame (implicit sequence number 5)	8 Bytes

0x66	DL06 ..... download frame (implicit sequence number 6)	8 Bytes
0x67	DL07 ..... download frame (implicit sequence number 7)	8 Bytes
0x68	DL08 ..... download frame (implicit sequence number 8)	8 Bytes
0x69	DL09 ..... download frame (implicit sequence number 9)	8 Bytes
0x6A	DL10 ..... download frame (implicit sequence number 10)	8 Bytes
0x6B	DL11 ..... download frame (implicit sequence number 11)	8 Bytes
0x6C	DL12 ..... download frame (implicit sequence number 12)	8 Bytes
0x6D	DL13 ..... download frame (implicit sequence number 13)	8 Bytes
0x6E	DL14 ..... download frame (implicit sequence number 14)	8 Bytes
0x6F	DL15 ..... download frame (implicit sequence number 15)	8 Bytes
0x70	DLTF ..... download tail frame	6 Bytes
0x71	ULCMDRES..... result of uploaded command	5 Bytes
...	*** UNUSED ***	
0x7E	ULACK..... upload acknowledgement frame	5 Bytes
0x7F	NULACK ..... upload acknowledgement frame	5 Bytes

Tabla 2 - Tabla de mensajes

Una vez mostrado a modo de resumen el funcionamiento de las transiciones en el sistema BD – CEC, los mensajes y las tramas que forman parte de la comunicación interna entre los mismos, se procederá a explicar la solución llevada a cabo por los integrantes del presente proyecto fin de carrera para implementar la simulación *software* de este sistema.

CAPÍTULO 4

SOLUCIÓN IMPLEMENTADA

4. DESCRIPCIÓN DE LA SOLUCIÓN IMPLEMENTADA

A continuación se explicará en detalle la solución que se ha implementado para el software de simulación *Balise Driver Simulator*, que responde a los requisitos detallados en el capítulo anterior.

El simulador se divide en dos paquetes: BDSim y ECSim (Ilustración 12).

- BDSim: se encarga de la gestión y simulación de los elementos formados por las tarjetas *Balise Driver* (BD) que actúan de controlador de las balizas físicas situadas en los distintos puntos de la vía. Ya que estas balizas son elementos físicos inertes, la simulación del comportamiento que se ha implementado en el programa se realiza enteramente en las tarjetas controladoras.

- ECSim: se encarga de la generación de mensajes que procesarán las tarjetas BD. Este simulador se creó debido a la falta de acceso proporcionada por la empresa THALES a sus instalaciones. Dado que es imposible obtener el funcionamiento correcto de la comunicación entre el CEC y BD de forma física, se optó por la creación de un programa software que emulara el comportamiento de esta comunicación.

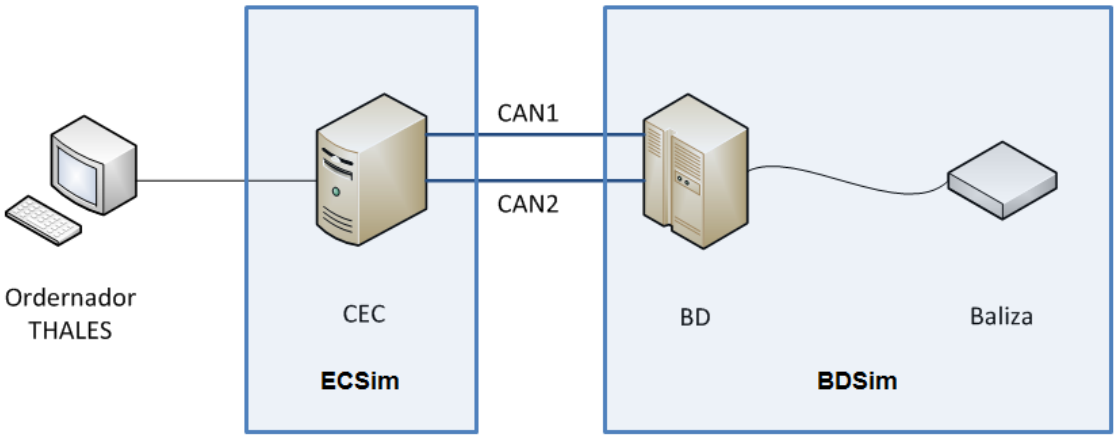

Ilustración 12- Sistema comunicación CEC y BD

A continuación se dividirá la definición de este desarrollo en las distintas partes de las que está formado: por un lado el hardware y su simulación en un entorno software; por otro los paquetes, detallando para cada uno el funcionamiento, las partes de las que está formado, sus atributos y métodos más relevantes.

4.1. Metodología software

Una metodología de desarrollo de software es un entorno usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información. A lo largo del tiempo se han desarrollado una gran cantidad de métodos.

El marco para metodología de desarrollo de software que hemos utilizado en este trabajo consiste en:

- Una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software
- Herramientas, modelos y métodos para asistir al proceso de desarrollo de software

En concreto se ha utilizado una metodología de desarrollo iterativo y creciente, debido a la disponibilidad de requisitos desde la primera etapa del desarrollo y a la exigencia de fiabilidad del programa.

Esta metodología trata de un proceso de desarrollo creado en respuesta a las debilidades del modelo tradicional de cascada [14]. Es uno de los más utilizados en los últimos tiempos ya que, como se relaciona con novedosas estrategias de desarrollo de software y *extreme programming*, se emplea en metodologías diversas.

El modelo consta de diversas etapas de desarrollo en cada incremento, las cuales se inician con el análisis y finalizan con la instauración y validación del sistema.

El proceso en sí mismo consiste en:

- Etapa de inicialización
- Etapa de iteración
- Lista de control de proyecto

4.1.1. Ventajas e inconvenientes del desarrollo incremental

A continuación se enuncian algunas de las ventajas y debilidades de este tipo de metodología de desarrollo.

- En este modelo los usuarios no tienen que esperar hasta que el sistema completo se entregue para hacer uso de él. El primer incremento cumple los requerimientos más importantes de tal forma que pueden utilizar el software al instante.
- Los usuarios pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.
- Existe muy pocas probabilidades de riesgo en el sistema. Aunque se pueden encontrar problemas en algunos incrementos, lo normal es que el sistema se entregue sin inconvenientes al usuario.
- Ya que los sistemas de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es muy poco probable que los sistemas más importantes sean a los que se les hagan más pruebas. Esto quiere decir que es menos probable que los usuarios encuentren fallos de funcionamiento del software en las partes más importantes del sistema.

Además, otras ventajas derivadas del desarrollo incremental son:

- En el desarrollo del modelo se da la retroalimentación muy temprano a los usuarios.
- Permite separar la complejidad del proyecto, gracias a su desarrollo por parte de cada iteración o bloque.
- El producto es consistente y puntual en el desarrollo.
- Los productos desarrollados con este modelo tienen una menor probabilidad de fallar.
- Se obtiene un aprendizaje en cada iteración que es aplicado en el desarrollo del producto y aumenta las experiencias para próximos proyectos.

Sin embargo existen una serie de inconvenientes o debilidades como son:

- La entrega temprana de los proyectos produce la creación de sistemas demasiados simples que a veces se perciben como incompletos o con falta de detalle a ojos del que los recibe.
- La mayoría de los incrementos se harán en base de las necesidades de los usuarios. Los incrementos en sí ya son estipulados desde antes de la entrega del proyecto. Sin embargo hay que ver si el producto necesita otros cambios además de los estipulados antes de la entrega final del proyecto. Este problema no se ve frecuentemente ya que la mayoría de las veces los incrementos estipulados suplen satisfactoriamente al usuario.
- Los incrementos no deben constar de muchas líneas de código ya que la idea de estos es

agregar accesorios al programa principal (o funcional); extender los incrementos provocaría que se perdiera la objetividad o base de lo que se trata en el desarrollo incremental.

- Requiere de un cliente involucrado durante todo el curso del proyecto. Hay clientes que pueden no estar dispuestos a invertir el tiempo necesario.
- El trato con el cliente debe basarse en principios éticos y de colaboración mutua, más que trabajar cada parte independientemente defendiendo sólo su propio beneficio.
- La entrega de un programa que es parcial pero funcional puede hacer vulnerable al programa debido a la falta de robustez en su sistema, provocando que agentes ajenos puedan interferir con el correcto funcionamiento del programa en sí.
- Infunde responsabilidad en el equipo de desarrollo al trabajar directamente con el cliente, requiriendo de profesionales.
- Sufre fuertes penalizaciones en proyectos en los cuales los requerimientos están previamente definidos, o para proyectos "todo/nada" en los cuales se requiere que se completen en un 100% (por ejemplo, licitaciones). Otro punto importante es asegurar que el trabajo se pueda cumplir tomando en cuenta los costes de los propios recursos [14].

4.2. Hardware

Los componentes físicos que forman parte de este elemento de campo y que son simulados en el proyecto software son (Tabla 3):

Elemento	Paquete	Subpaquete	Función
Centralized ETCS Controller (CEC o CEC)	ECSim	CEC	Controlador centralizado del sistema ETCS. Se encarga de la comunicación desde la parte de la vía.
Balise Driver (BD)	BDSim	BDCard	Se encarga de realizar la comunicación con el CEC a las balizas correspondientes. El componente es una tarjeta controladora que monitoriza la baliza situada en la vía.
IFCan A y B (CAN 1 y CAN 2)	BDSim	NicanPort	Son los puertos de comunicación que conectan el CEC y las BD. Son dos puertos que se encargan de realizar la misma comunicación por motivos de seguridad (redundancia).

Tabla 3 – Elementos Hardware simulados

El hardware del que se compone la BD comprende una configuración en el que las LEUs se conectan directamente al CEC, o indirectamente a través de un modem.

Existen dos canales de comunicaciones (CAN1-2/IF1a-b), por los que se distribuye la misma información debido a la redundancia. Los datos enviados por la CAN2 (IF1b) son enviados automáticamente de forma interna por el interface CAN1 de la BD (Ilustración 13).

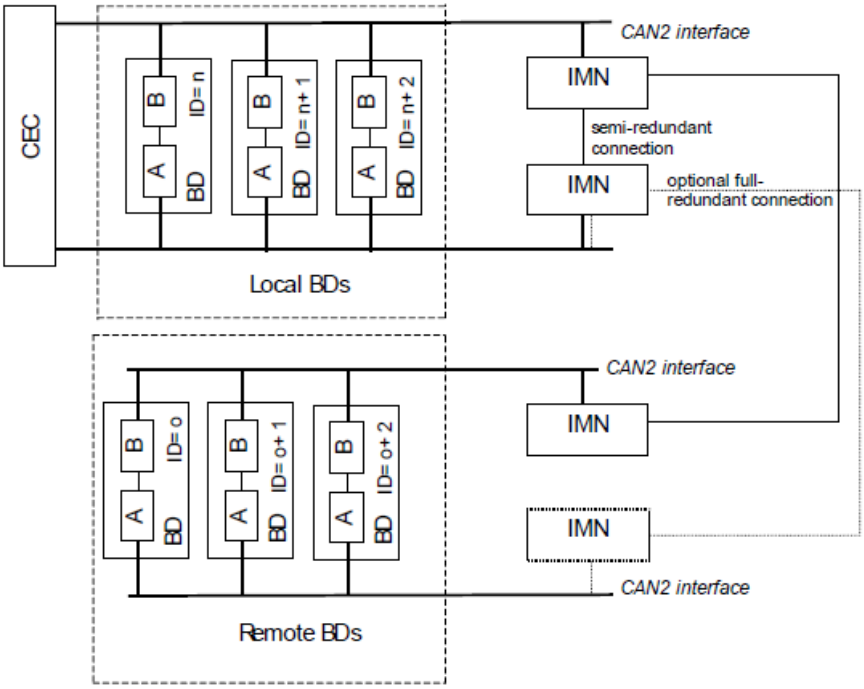

Ilustración 13 - Esquema SF de comunicación CEC con BD

En las siguientes secciones se explicarán de forma detallada los métodos utilizados para la simulación de cada componente físico del sistema en los subpaquetes correspondientes.

4.3. Paquete BDSim

El paquete BDSim es el paquete que se encarga de simular el funcionamiento de la transmisión y procesamiento de mensajes entre el CEC y la BD (Ilustración 14).

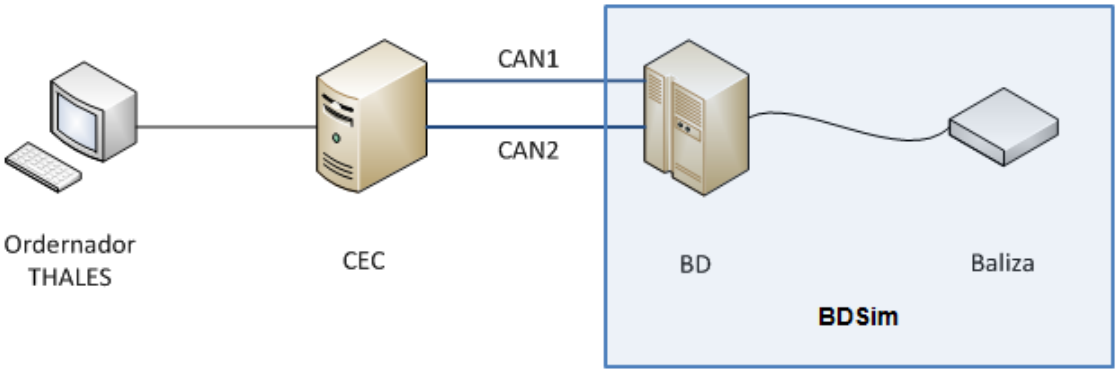

Ilustración 14 - Parte BDSim del sistema

Este módulo contiene todos los elementos necesarios para cumplir los requisitos mencionados anteriormente en la memoria (capítulo 3), tanto gráficos como de software, respecto a la simulación de las tarjetas controladoras de balizas.

A continuación se detallarán los subpaquetes de los que está formado y la información concerniente a los mismos en cuanto a métodos utilizados, estructuras de datos, atributos relevantes y aspectos destacados de la implementación.

4.3.1. Subpaquete BDSimDlg

Propósito

El subpaquete BDSimDlg actúa como elemento principal del programa, aunando las clases y métodos principales que permiten el funcionamiento del simulador BDSim al completo y como programa independiente.

Implementación del paquete

Mediante la implementación de un *thread* continuo que alimenta los canales de envío y transmisión de datos, el programa simula el funcionamiento de la comunicación entre la BD y el CEC.

Este *thread* realiza el envío y recepción de los mensajes que permiten la comunicación, que se realiza por los dos puertos de cada tarjeta (A y B) de forma simultánea e idéntica cuando se encuentran operativos. Es necesario indicar que la existencia de redundancia de estos puertos es debida a la extrema seguridad que deben llevar estos dispositivos en el mundo físico pues no puede haber fallos en el envío de mensajes.

Asimismo el subpaquete procesa los mensajes en el método principal de ejecución del programa, indicando a la tarjeta el funcionamiento a realizar dependiendo del mensaje que haya sido transmitido.

Es importante añadir que estas comunicaciones las realiza para cada una de las tarjetas que se encuentren en el "*rack*" físico. Dado que se trata de una simulación de dichas tarjetas controladores, se ha optado por simular un conjunto completo de 16 tarjetas, el máximo permitido por CEC.

La práctica totalidad del funcionamiento del paquete funciona bajo *timers*, debido a motivos estrictamente derivados de la utilización del protocolo a simular y sus restricciones de tiempos explicados en apartados anteriores.

Estructura de datos

a) Atributos relevantes

Nombre	Descripción
*CNiCanPort** Ports[2]	Puertos de comunicaciones utilizados por las tarjetas para la transmisión de mensajes con el CEC. Se trata de un vector con dos elementos de la clase NicanPort que representa cada uno de los dos puertos por tarjeta.
*BDCard** cards[16]	Vector de 16 objetos de la clase BDCard que modelizan las 16 tarjetas controladoras de balizas.
vector<int> timers	Array de *timers* utilizados para las funciones que requieren el uso de contadores en la implementación del protocolo de funcionamiento de las tarjetas controladoras.

b) Métodos relevantes

Nombre	Descripción
int Prepare(MsgData *Msg)	Función utilizada en la modificación de mensajes, cuyo uso se ejemplifica en el cambio de los mensajes según el filtro utilizado en el módulo de filtrado de mensajes.
void ProcessMsg(MsgData Msg)	Método principal del subpaquete y esencial para el correcto funcionamiento del simulador. Al indicar como argumento de la función un mensaje válido, ésta lo procesará en función de su contenido y ejercerá las acciones correspondientes a dicho mensaje. Generalmente, provoca la llamada de los métodos correspondientes al mensaje procesado en el objeto BDCard correspondiente, es decir, provocará una acción en la tarjeta controladora simulada. Ciertos mensajes provocarán la creación de *timers* cuya función radicará en la medición de tiempos establecidos en el protocolo (*timeouts*).
void Enqueue(MsgData Msg)	Prepara el mensaje para su envío por los canales CAN1 y CAN2 de la tarjeta seleccionada.

virtual void OnTimer(UINT nIDEvent);	Método utilizado en la inicialización de todos los *timers* que incluye el sistema a modo de contadores de tiempos establecidos por el protocolo de comunicación implementado.

c) Subpaquetes y clases relacionadas

Nombre	Descripción
TimesDlg	Utiliza los objetos y métodos incluidos en la misma para la definición y uso de *timers* en el programa principal.
BDCard	Es la clase encargada de la manipulación de objetos de tipo BDCard, usado en BDSimDlg para la definición del vector de 16 tarjetas controladoras.
NicanPort	Necesario para la utilización de los dos puertos redundantes de envío y recepción de mensajes con el CEC.

Retos de implementación

- Investigando el modo de implementar la comunicación entre los sistemas de la BD y el CEC, y mediante el estudio de simuladores previos, llegamos a la conclusión de que la forma óptima de implementación de este mecanismo sería mediante el uso de hilos de comunicación (*threads)*.
- Durante la implementación del SW se observó que la utilización de un *thread* por cada mecanismo (envío y recepción) aumentaba de forma alarmante el consumo de recursos de la máquina. Al tratar de implementar esta solución en nuestro sistema observamos que el envío y recepción de mensajes se realizaba de manera secuencial, con lo que con un único hilo que agrupara la funcionalidad necesaria mejoraríamos de manera inmediata el empleo de recursos por parte del SO a la hora de la ejecución del programa en una máquina. Esto podría tener beneficios tanto económicos como estructurales en la implantación de estos sistemas en entornos reales.
- En cuanto a la estructura de datos, nos encontramos el problema de saber qué estructura de datos utilizar para modelizar la ristra de tarjetas a simular. Se observó que, dado que el sistema admitirá como máximo 16 tarjetas por controlador en un entorno real, lo óptimo sería utilizar un vector estático de 16 objetos BDCard.
- En el subpaquete se encuentra la inicialización de varios *timers*, además del método *OnTimer*, cuya definición es obligatoria en VC++. Es un único método que define el comportamiento de todos los *timers* del sistema. Dado que es un software cuyo

funcionamiento depende en gran medida de contadores temporales para el protocolo de comunicación (*timeouts*), se trata de un método básico en el paquete BDSimDlg.

- Durante la ejecución del programa hay muchos *timers* con muy diferentes propósitos. Uno de los retos que surgió fue el de gestionar el inicio y fin de cada *timer*. Debido a la alta cantidad de *timers* se debió optar por estructuras especiales (vectores dinámicos) debido a que, durante la ejecución del programa, existían variaciones en el número de estos.
- A la hora de realizar el log de trazas se encontró otro reto en su implementación. La sintaxis era importante para poder examinar el fichero y sacar conclusiones de la simulación, que es el objetivo principal de este programa. También era importante para la auto-evaluación del programador a la hora de implementar el simulador.
- El módulo para el filtrado de mensajes dentro de las trazas se creó para poder visualizar los mensajes que nos interesaran dentro de todo el log de trazas.
- Se debió crear un método único, debido a la creación de un único hilo de ejecución, que procesase todos los mensajes sin excepción. Evidentemente este es el método más crítico del sistema, y es en el cual se inicializan la mayor parte de los *timers*. A la hora de procesar los mensajes hay unos más fáciles que otros, pero se encontraron ciertas dificultades serias en el procesamiento de algunos de ellos.
- Debido a la complejidad de ciertos mensajes se hubo de utilizar métodos del subpaquete BDCard para el procesamiento de estos mensajes, ya que era necesario realizar procesamientos a nivel de tarjeta para un correcto funcionamiento motivado por los mensajes enviados.
- Se encontraron retos a la hora de modelizar el comportamiento de los canales de comunicación, de tipo Canbus. Se eligió finalmente utilizar la clase NicanPort, que se encontraba implementada en otros elementos de campo, para la simulación de los puertos.

4.3.2. Subpaquete BDCard

Propósito

El subpaquete BDCard contiene los métodos y clases necesarios para la modelización de los elementos ferroviarios correspondientes a las tarjetas controladoras de las balizas. Este subpaquete será utilizado por el programa principal para simular el funcionamiento de dichas tarjetas y obtener el comportamiento definido en los requisitos anteriormente expuestos.

Implementación del paquete

El paquete BDCard se ha implementado como una clase complementaria a BDSimDLg que utilizaría el objeto BDCard para las operaciones con tarjetas controladoras.

De este modo, el paquete BDCard estaría compuesto de un método constructor que inicializaría todos los atributos relevantes de la tarjeta. Se implementa también un método encargado del

aspecto gráfico del programa, en el que se desarrolla el código de dibujado de estas tarjetas en sus ranuras correspondientes de la interfaz gráfica.

Adicionalmente, y como parte nuclear de la funcionalidad de la clase, se han implementado los métodos de manejo y envío de mensajes, que constituyen la razón de ser del subpaquete descrito.

Por último se tienen los métodos de procesado de mensajes de descarga y telegramas, que se encargan de la funcionalidad de descarga de las tarjetas controladoras descritas anteriormente en el protocolo de comunicación.

Estructura de datos

a) Atributos relevantes

Nombre	Descripción
int mode	Atributo mediante el cual se almacena el estado de la tarjeta. Mediante los estados en los que se encuentra la tarjeta controladora, se podrá informar al CEC de los posibles mensajes que puedan comunicarse a la misma.
int m_StateAB[2]	Vector de estado de los dos canales A/B asociados a NicanPort. Dependiendo de estos estados se indicará si se encuentran ocupados/desocupados y por tanto si se puede iniciar la comunicación entre el CEC y BD.
vector<unsigned char> telegrama;	Es el atributo más importante de la clase, dado que almacena el mensaje que se está transmitiendo desde la tarjeta al CEC.

b) Métodos relevantes

Nombre	Descripción
void BDCard::Status(*int* iChannel)	Función que se encarga de transmitir al CEC el estado completo de la tarjeta. Este estado está compuesto por las variables más relevantes que identifican a la misma, y se comunica al CEC mediante el envío de un mensaje de protocolo.

void BDCard::SendULHF(int iChannel, byte requestid, unsigned char type)	Función que se encarga de enviar los datos de carga a el CEC que construyen las balizas. La carga es un proceso que debe ser realizado mediante pasos y por eso dicho proceso se realiza por pasos.
void BDCard::PrepareULTF(int iChannel, byte requestid,unsigned char *crc_data,int size, int a_b)	Función utilizada desde SendULHF y que conforma ciertos pasos para la realización del proceso de carga de datos de las balizas al CEC.
void BDCard::SetMsgMode(MsgData Msg[10],int index)	Función que dado el mensaje recibido en la tarjeta decide o no enviar el estado del mismo y/o realizar diversas tareas adicionales.
void BDCard::ProcessDownload(int num_msg,unsigned char *data,int iChannel)	Hay varias funciones implicadas en el proceso de descarga de datos del CEC a las tarjetas, y esta función en concreto mete los datos de descarga en un único vector para poder así ser tratado de manera más cómoda
void BDCard::ProcessTelegram()	Esta función construye el valor de ciertas variables de las tarjetas a partir de los valores obtenidos en la descarga, y además trata el vector de descarga para poder operar sobre él mismo y así obtener a su vez los valores de seguridad CRC necesarios.
int BDCard::sdoRead(int numero_fichero)	A esta función se acude cuando o bien la descarga de datos falla y es necesario cargar la tarjeta con unos valores predeterminados, o bien cuando de manera voluntaria se decide cargar con estos datos a la misma.

c) Subpaquetes y clases relacionadas

Nombre	Descripción
BDSimDlg	Conforma el programa principal y contiene las funcionalidades más básicas y aspectos más generales de la aplicación.

Retos de implementación

- A la hora de modelizar una tarjeta física se han tenido que manejar aspectos de la simulación que no se conocían previamente. Ha sido necesario tomar decisiones de

implementación de atributos para que la simulación de la tarjeta controladora y sus propiedades sean idénticas a la realidad.

- En lo relativo a modelización de modos de funcionamiento, estados de la tarjeta y atributos que tengan que ver con estos estados, ha sido necesario un proceso de prueba-error para determinar que las estructuras de datos en las que se almacenaban los mencionados atributos eran correctos en cada caso.
- En los casos en los que la complejidad de procesamiento de los mensajes a nivel de tarjeta superaba el procesamiento que podía manejar BDSimDlg, se ha optado por implementar parte de ese procesamiento a nivel de tarjeta.
- Los atributos de las tarjetas se rellenan de dos maneras: mediante los mensajes del CEC (hacen que cambien de estado o modo) y mediante la lectura de ficheros (sdo). Hubo que desarrollar un módulo de configuración de tarjetas mediante ficheros externos.

4.3.3. Subpaquete NicanPort

Propósito

El subpaquete NicanPort controla la recepción y el envío de mensajes, además de la comprobación de dicha comunicación que se realiza mediante los puertos Nican, que simulan los canales IFb1 e IFb2 del sistema de señalización físico de la baliza.

Esta clase es la implementación a más bajo nivel que se encuentra en el sistema ya que se podría considerar una librería software que soporta la gestión de los puertos CANBUS.

También se puede encontrar en este paquete una importante presencia de métodos destinados a la comprobación a nivel de bit de la comunicación puerto-a-puerto.

Implementación del paquete

La implementación del paquete se ha realizado mediante la inclusión de métodos utilizados en la mayoría de librerías de conexión que se encuentran en el mercado software. Estos métodos se caracterizan, a grandes rasgos, por la inclusión de al menos:

- Inicialización del puerto
- Enviar
- Recibir
- Comprobación CRC

A continuación se muestran los métodos que implementan estas funcionalidades en el subpaquete NicanPort, que utiliza la librería Nican a bajo nivel para el manejo de los puertos físicos con el mismo nombre.

Estructura de datos

a) Atributos relevantes

Nombre	Descripción
int status	Su función es la de indicar el estado en el que se encuentra el puerto. Los posibles estados en los que se puede encontrar el puerto son CONNECTED y NOT_CONNECTED.
int port	Indica el puerto usado por la aplicación. En nuestro caso se han utilizado un máximo de dos puertos para simular los presentes entre el CEC y la BD (IFb1 e IFb2).
CCriticalSection cs_trigger	Es el mensaje utilizado por el protocolo de comunicación para indicar el inicio y fin de la misma entre los puertos.
MsgData messages_pipe[10]	Es la cola de mensajes utilizada para canalizar de manera ordenada los mensajes que se tratarán en la comunicación entre los puertos. Si por alguna razón el CRC fallase sería posible recuperar los mensajes desde esta estructura de datos.

b) Métodos relevantes

Nombre	Descripción
int Receive(MsgData* f)	Lee un único mensaje del puerto indicado y lo almacena en el pipe de mensajes.
int Send(MsgData* f, int resFilter)	Envía el mensaje procedente del pipe de datos almacenado.
int InitPort(int lport)	Inicializa el puerto indicado. En la aplicación SW desarrollada existen dos posibles puertos a inicializar.
int BuildCRC(MsgData *msg)	Método utilizado para aplicación del algoritmo de comprobación de errores de comunicación. Existe la posibilidad de realizarlo de 8 y 16 bits.

bool CheckCRC(MsgData *msg)	Aplica el algoritmo de forma inversa para la comprobación de errores del mensaje.

c) Subpaquetes y clases relacionadas

Nombre	Descripción
Nican.h	Librería utilizada para el manejo a bajo nivel de los puertos de comunicación Nican. Esta librería está proporcionada por el fabricante y no ha sido desarrollada en este proyecto.

Retos de implementación

- El objetivo fundamental de la comunicación en el sistema es que no se produzcan pérdidas de información en la transmisión de mensajes. Tiene que existir una verificación fiable e ininterrumpida que permita la comprobación de manera simultánea en los dos canales. Se utiliza para ello la implementación de BuildCRC y ChecCRC en la clase del sistema.
- Para que existiese una comunicación de forma continuada y no se produjesen interrupciones hubo que crear una estructura de paso de mensajes que diese una holgura suficiente a la comunicación. Se decidió implementar un *buffer* de un máximo de 10 mensajes por puerto para este fin.

4.3.4. Subpaquete TimesDlg

Propósito

El protocolo físico de comunicación entre el CEC y la BD exige la utilización de tiempos que marcan el envío y recepción de los mensajes. En términos generales estos tiempos definen la exigencia de comunicación que han de tener los dos sistemas, siendo de menor exigencia, es decir permitiendo una comunicación con más retrasos cuanto mayor sean los tiempos del protocolo. Así, cuanto menor sean los tiempos definidos en la clase *TimesDlg*, la comunicación se verá provista de una mayor calidad.

Implementación del paquete

Este paquete es un caso excepcional dentro del sistema pues se trata de una implementación destinada casi en exclusiva a cubrir los requisitos gráficos. Así en esta clase se encuentran desarrollados métodos que permiten la lectura/escritura de variables numéricas que representan el valor de los tiempos de la comunicación.

Estructura de datos

a) Atributos relevantes

Nombre	Descripción
*int t*_roundtrip	Cada uno de estos atributos corresponde a un tiempo definido en el manual del protocolo. A nivel de mensaje su implementación es sencilla pues representan un entero en el que se almacena el tiempo transcurrido entre ciertos mensajes en *ms*. A nivel semántico, por el contrario, estos mensajes pueden indicar aspectos más profundos del protocolo de comunicación, tales como confirmación de recepción de mensajes, etc. En el apartado de Requisitos Específicos de la presente memoria se hace una presentación de los más importantes a nivel de protocolo.
int to_aspect	
int CEC_timeout	
int to_ack_frames	
int UL_frames_to_send	
int max_not_acked_messages	
int max_user_data	

b) Métodos relevantes

Los métodos que se han utilizado en esta clase son puramente gráficos, utilizándose en su mayor parte para la entrada de datos en la interfaz. Por este motivo, no procedemos a su explicación.

c) Subpaquetes y clases relacionadas

No existe una relación con otras subclases debido las características aisladas de la misma.

Retos de implementación

Pese a la importancia de esta clase, no se han producido retos destacables en su implementación.

4.4. Paquete ECSim

El desarrollo del paquete ECSim (Ilustración 15) surge de la necesidad de testear nuestra aplicación fuera de los laboratorios de la empresa con la que se ha realizado la colaboración. De

este modo, procederemos a explicar el propósito del paquete en más detalle y ofreceremos una visión general de la implementación del mismo.

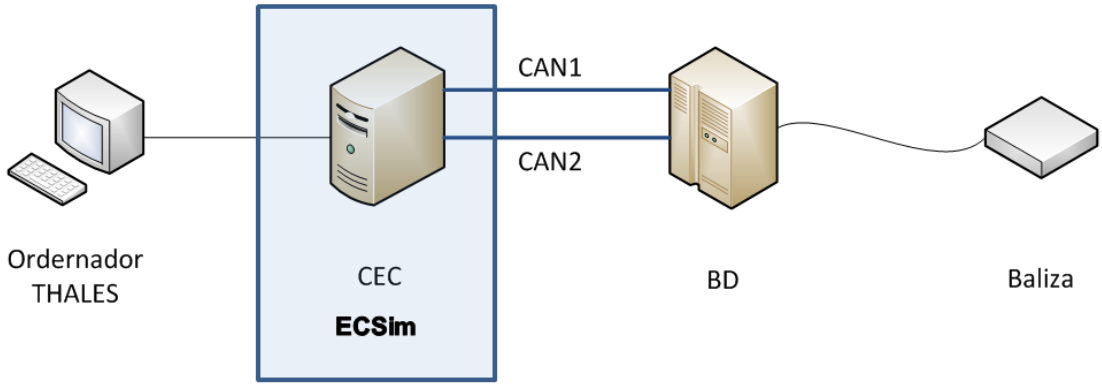

Ilustración 15 - Paquete ECSim

Propósito

El objetivo del paquete es la simulación del comportamiento del controlador CEC para la interacción del mismo con la aplicación BDSim (Ilustración 15). De esta manera seríamos capaces de ofrecer una solución de pruebas que pudiese ser portable fuera de las instalaciones de la empresa.

Características de la implementación

De la misma manera que en BDSim se realiza una comunicación de dos canales, el ECSim será el punto de envío y recepción que se encontrará al otro lado de dicha comunicación. En nuestro caso el ECSim no es un sistema que incorpore una lógica pareja a BDSim ya que se trata de un mero simulador de envío y recepción de mensajes. El CEC físico, que se encuentra situado dentro de las instalaciones de la empresa THALES, es el elemento más complejo de todo el sistema de elementos ferroviarios ya que, como hemos indicado en el apartado “Elementos de campo” del capítulo 2, es el encargado del control de una multitud de elementos, siendo la baliza uno de ellos.

Por esta razón la simulación que hemos realizado del ECSim se limitará a los aspectos más prácticos de la misma, es decir, al envío y recepción de mensajes que corresponden a la verificación a nivel de protocolo de los mensajes enviados y recibidos entre BD y CEC, obviando todas aquellas características tanto a nivel manual como a nivel automático de toma de decisiones de control del comportamiento de sistemas de baliza.

En definitiva, la implementación de este simulador es relativamente sencilla por lo que no parece necesaria una descripción detallada a nivel de desarrollo. No obstante será utilizado en capítulos posteriores para ofrecer una visión del funcionamiento del sistema BDSim y aportar a modo de validación una prueba de la correcta implementación del sistema.

CAPÍTULO 5

RESULTADOS Y VALIDACIÓN

5. RESULTADOS Y VALIDACIÓN

En este apartado de la memoria se realiza un recorrido guiado por la aplicación desarrollada para el proyecto BDSim, atendiendo al correcto cumplimiento de los requisitos tanto funcionales como gráficos que se exponen en el capítulo 3 de la misma.

Para ello se explica de manera detallada el funcionamiento de la aplicación mediante el uso de la misma para un caso que se describe paso a paso.

5.1. Caso práctico de utilización del programa

Para iniciar el programa se pulsa el icono de la aplicación situado en el escritorio que muestra la siguiente pantalla (Ilustración 16).

Seguidamente es necesario abrir un cuadro de diálogo en *Project -> Open* y seleccionar un escenario de tarjetas de BDSim que interese simular. Dicho escenario está codificado en un fichero de tipo XML (Ilustración 17)

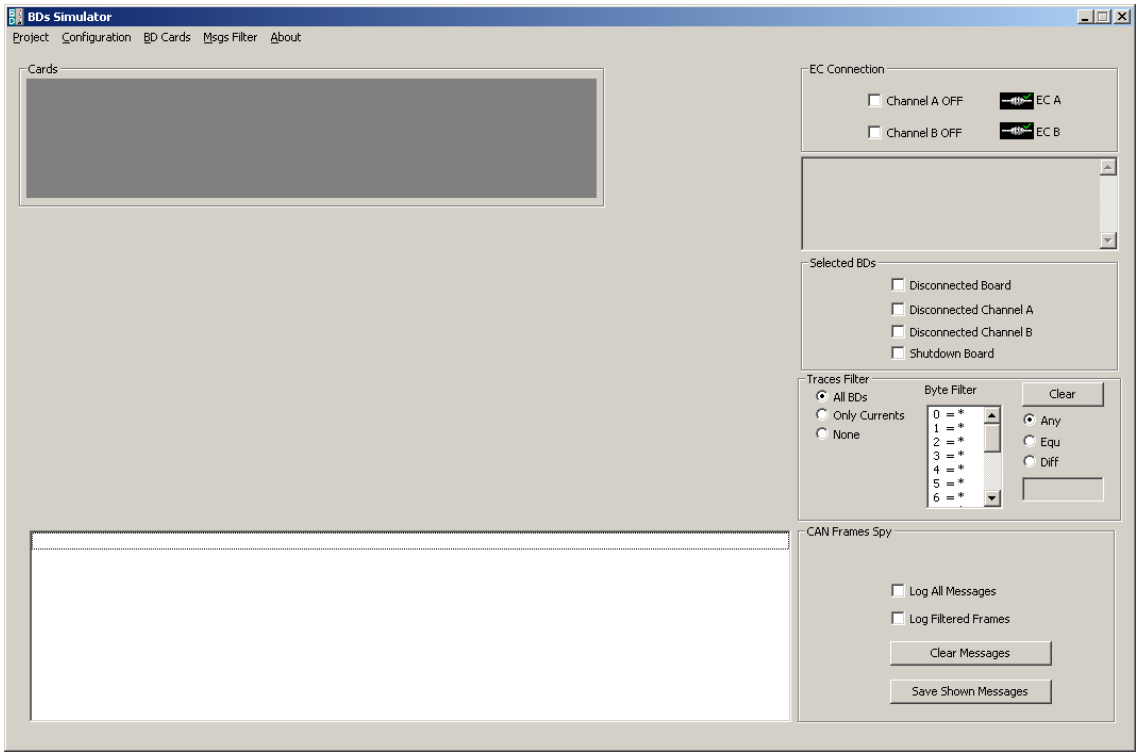

Ilustración 16 - Pantalla inicial del programa

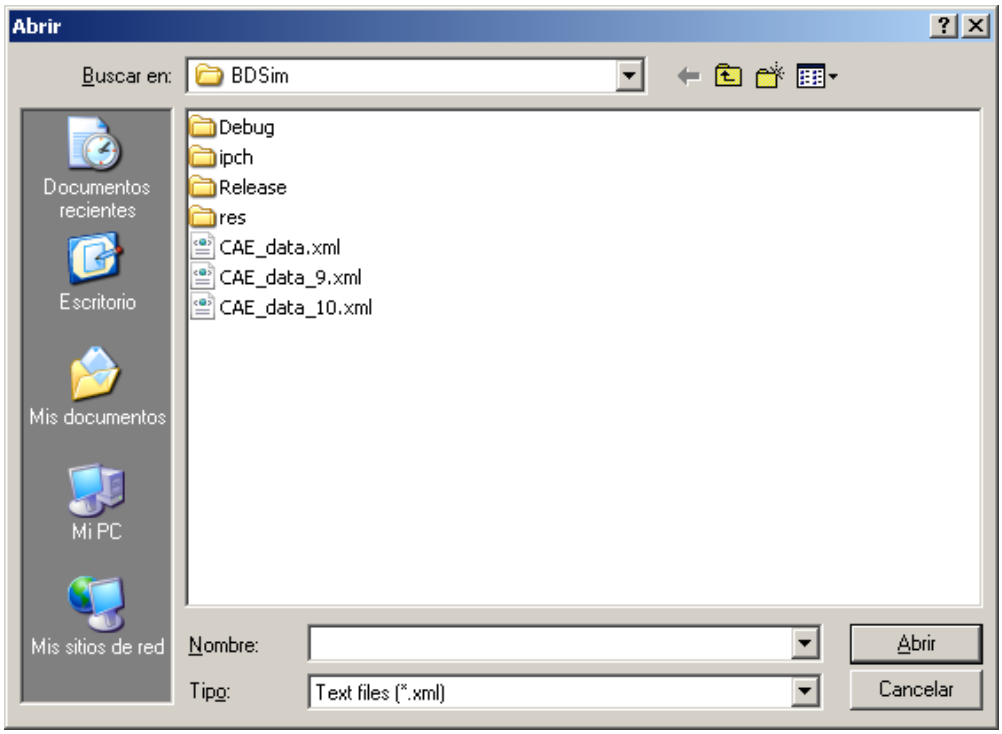

Ilustración 17 - Cuadro de diálogo de selección y apertura del fichero

Tras seleccionar el fichero XML se carga el escenario. Para este ejemplo se puede ver que se han cargado 3 tarjetas consecutivas en los rack 9, 10 y 11. También se puede observar el inicio de la comunicación entre el simulador desarrollado y el EC (Ilustración 18).

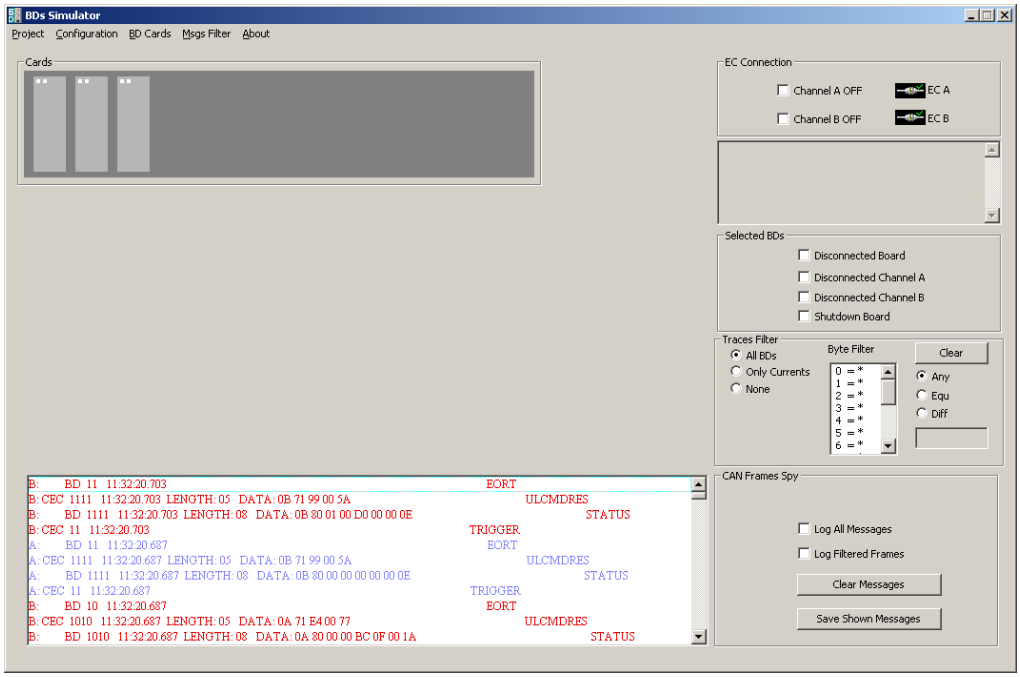

Ilustración 18 - Escenario cargado en la aplicación

Una vez iniciado el modo de simulación se pueden seleccionar las tarjetas haciendo un *click* sobre cada una de ellas. En este caso se ha seleccionado la tarjeta del rack 9. Además en este punto se puede desconectar dicha tarjeta, apagarla o simplemente desconectar alguno de sus canales mediante las opciones de "Selected BDs". En este caso se ha seleccionado la opción de apagado de la tarjeta (Ilustración 19).

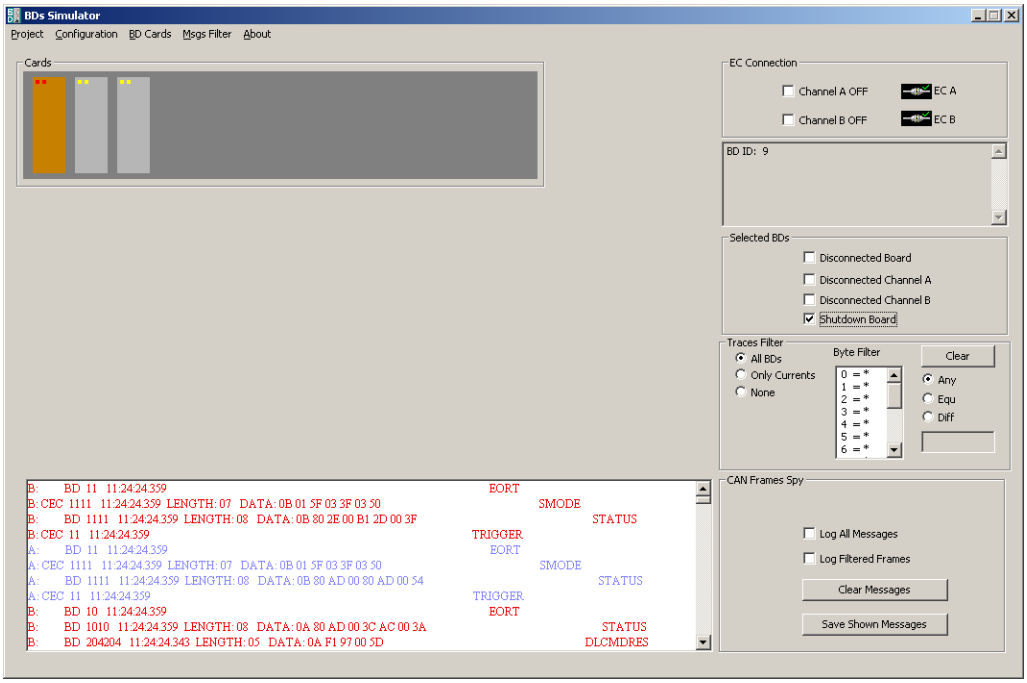

Ilustración 19 - Tarjetas 9, 10 y 11 seleccionadas y apagado de tarjeta del rack 9

Las tarjetas pueden estar en distintos estados, en función de la conexión o no de sus canales, como también los canales están en estado de carga o a la espera de recibir información. Durante la mayor parte del tiempo lo normal es que estén desconectados o conectados, siendo canal A y B independientes entre sí (Ilustración 20).

Ilustración 20 - Detalle de la tarjeta con los dos canales A y B

Se puede utilizar el filtro de trazas para visualizar sólo los mensajes que interesan, filtrando su vista en términos de tarjetas de interés y/o de coincidencia de ciertos bytes de mensaje según el objetivo (Ilustración 21).

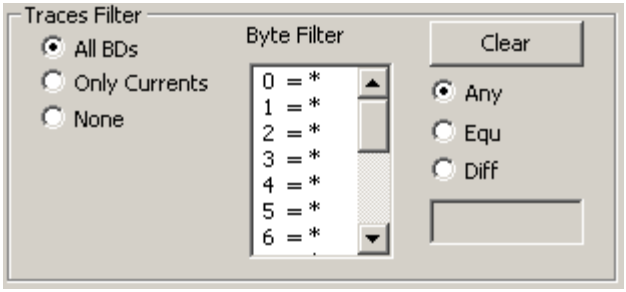

Ilustración 21 - Filtro de trazas

En la aplicación existe la opción de hacer un log de los mensajes durante el intercambio de mensajes con el EC. Se pueden mostrar todos los mensajes o sólo los mensajes resultantes de aplicar el filtro de trazas. Un ejemplo del archivo resultante de aplicar esta opción se muestra en la Ilustración 22.

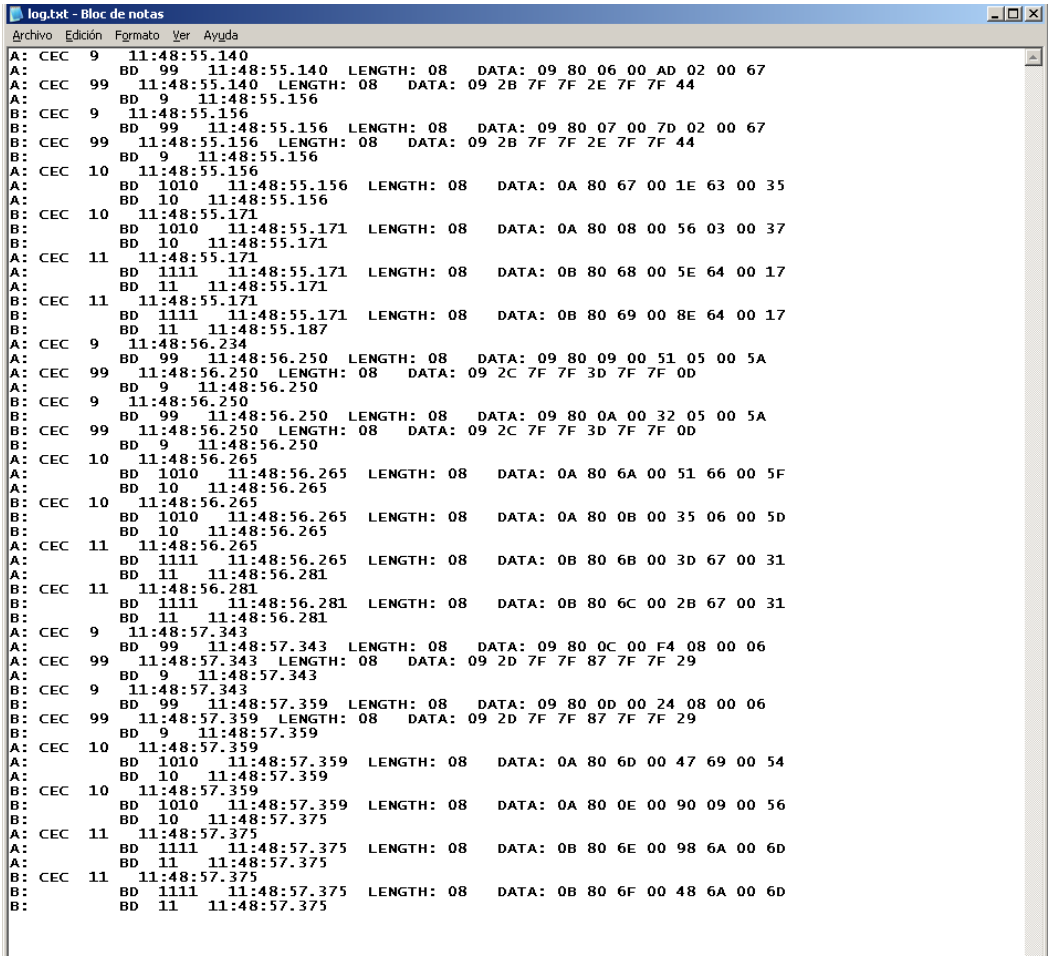

Ilustración 22 - Log de trazas

Este archivo se crea si no existe o puede ser seleccionado si ya existe (Ilustración 23).

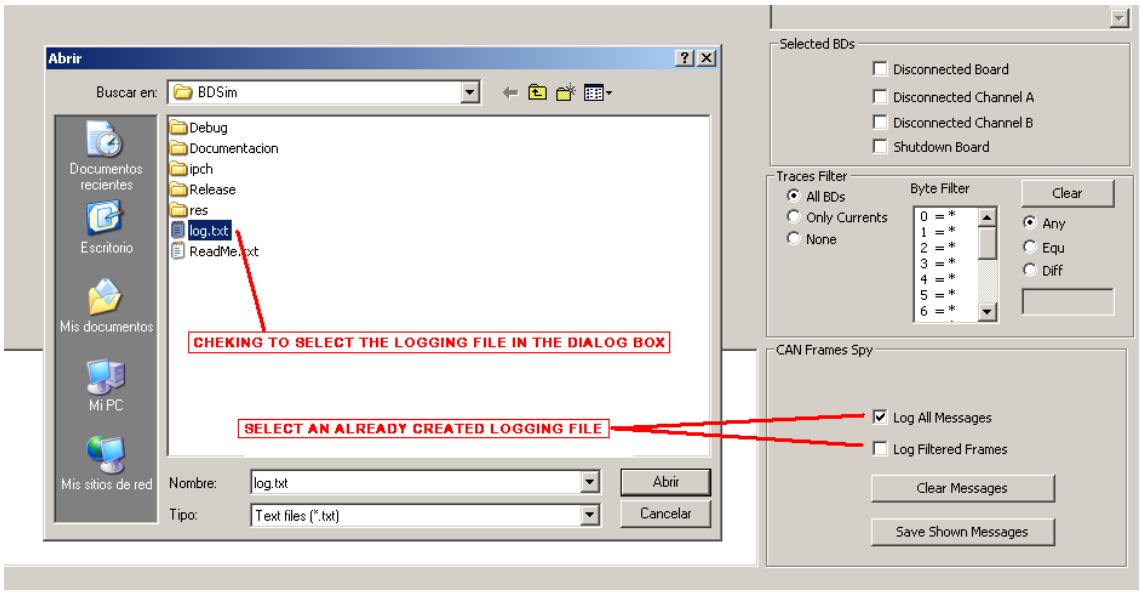

Ilustración 23 - Selección de archivo de mensajes

Como se ha dicho, hay dos canales de comunicación entre BDSim y el EC, y por ambos canales fluye la misma información. Parte del propósito de la simulación es tener la opción de seleccionar qué canales están activos en cada momento y así poder observar el comportamiento del sistema (Ilustración 24).

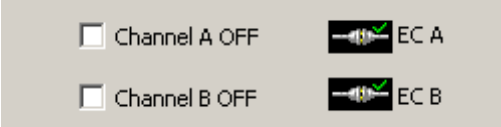

Ilustración 24 - Selección de canales

En la pestaña Msgs Filter se muestra la opción de modificar los mensajes que se mandan de forma predeterminada al EC según el protocolo de comunicación, alterando dicho protocolo. De este modo se puede observar cuál es el comportamiento del EC en dichas condiciones. Se pueden aplicar las mismas opciones que en el filtrado pero en esta ocasión modificando el contenido de los mensajes enviados, al igual que los canales y la frecuencia de envío de dichos mensajes (Ilustración 25).

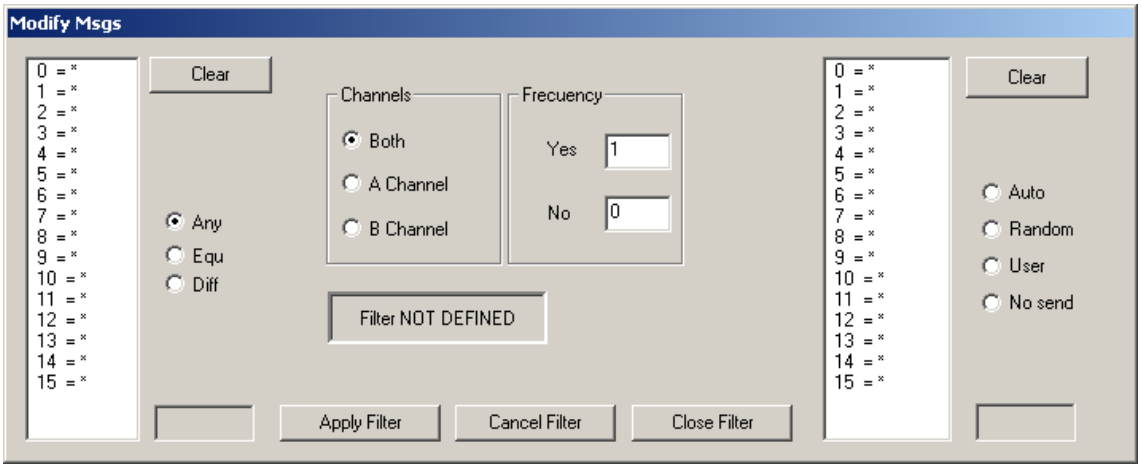

Ilustración 25 - Modificación de mensajes

BDSim simula un entorno real en el que, como es evidente, existen retardos. El valor de dichos retardos debe ser configurable. Para ello en Configuration->Time Definition se introducen los valores que interesan para la simulación de retardos (Ilustración 26).

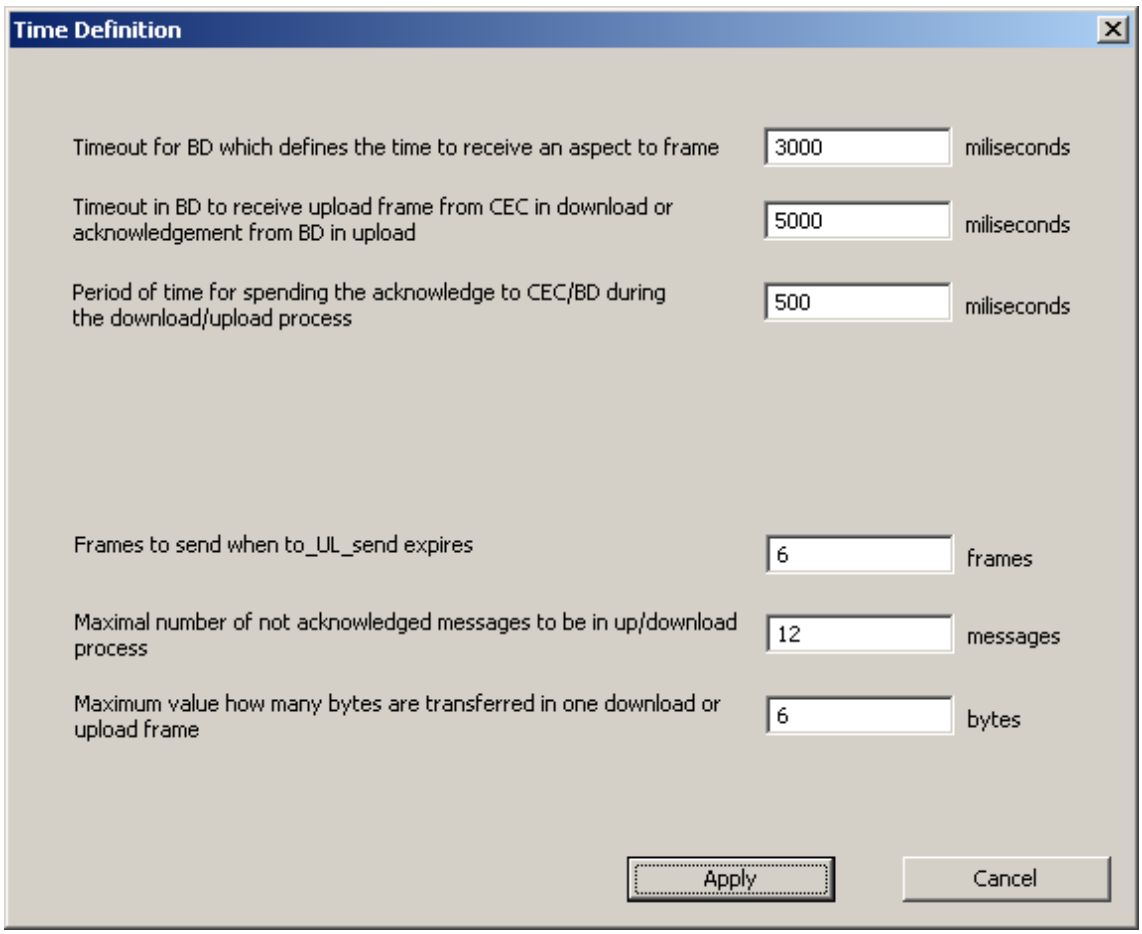

Ilustración 26 - Definición de tiempo de retardo

Los valores de los parámetros internos los asigna el EC a cada una de las tarjetas en el proceso de descarga (del EC al BDSim). Posteriormente se debe comprobar de forma automática que dichos valores son correctos. En caso de no serlo, dichos valores se cargarán de unos ficheros de texto que existen como seguridad, y que son creados a mano.

En cualquier caso, algunos de estos valores pueden ser modificados a mano pulsando encima de cada tarjeta, por ejemplo en la tarjeta del rack 9 (Ilustración 27).

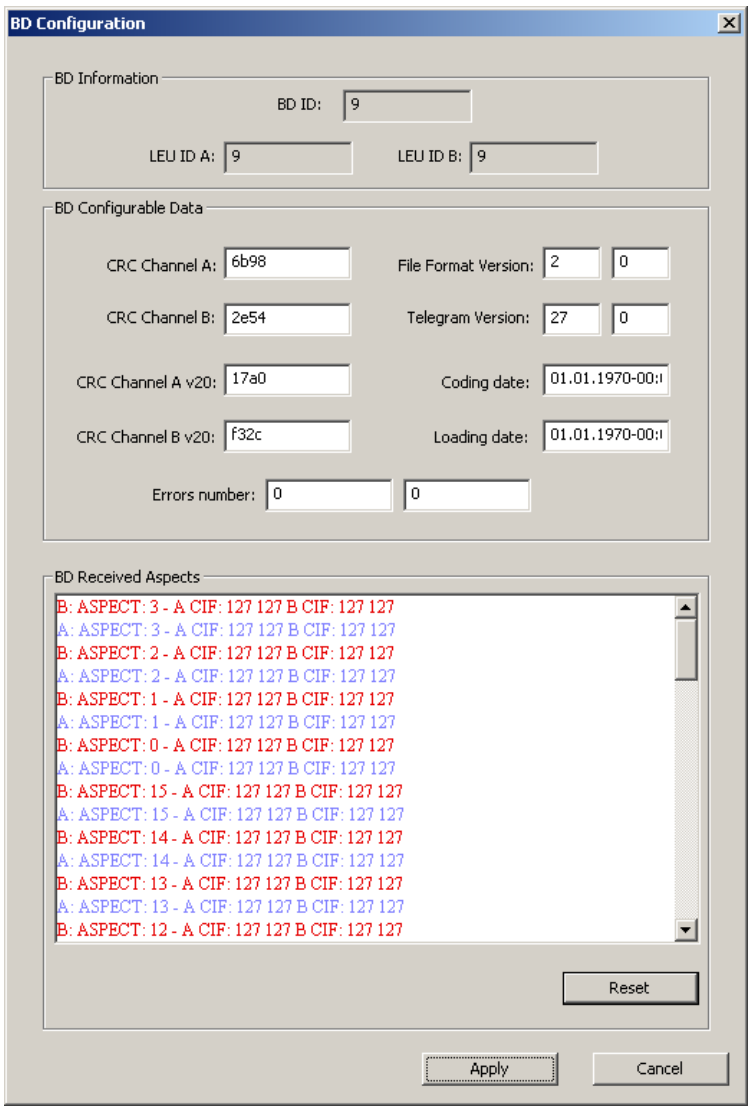

Ilustración 27 - Modificación de los parámetros de la tarjeta del rack 9

Además, como se puede observar, es posible ver un filtrado por tarjeta de los mensajes en los que se encuentra involucrada.

5.2. Validación

Una vez mostrado el proceso de aplicación del simulador se procede a realizar un resumen de la validación llevada a cabo para confirmar la correcta implementación de los requisitos.

5.2.1. Requerimientos generales

Requisito	Descripción	Evidencia	Resultado
R 1.1	La aplicación estará dividida en cuatro partes diferenciadas: la vista gráfica de BDs, el Espía de mensajes, las Propiedades de simulación y el Filtro de trazas.	*Ilustración 19*	✓
R 1.2	La vista gráfica de BDs mostrará la monitorización de las tarjetas y para cada una de ellas sus canales A/B, los comandos y las indicaciones.	*Ilustración 21*	✓
R 1.3	El Espía de mensajes permitirá guardar un log de los mensajes monitorizados en un fichero de texto plano.	*Ilustración 22, 25 y 26*	✓
R 1.4	En Propiedades de la simulación se mostrará la conexión de CEC y/o BD, permitiendo aplicar un filtro a los mensajes y cerrar la recepción de información desde los Canales A/B del Can Bus.	*Ilustración 22*	✓
R 1.5	Se permitirá ejecutar acciones sobre las tarjetas tales como apagarlas o desconectar sus canales correspondientes.	*Ilustración 21 y 22*	✓
R 1.6	El Filtro de trazas mostrará las trazas de los mensajes que hayan sido filtrados	*Ilustración 24*	✓

5.2.2. Requerimientos gráficos

Requisito	Descripción	Evidencia	Resultado
R 2.1	En la vista gráfica de BDs se representará mediante leds simulados las acciones de las tarjetas (si están conectadas, apagadas o en proceso de conexión).	*Ilustración 23*	✓
R 2.2	En la vista gráfica de BDs el usuario puede seleccionar cualquiera de las tarjetas representadas y esto determinará qué BD será simulada y qué trazas se visualizarán en el Espía de mensajes.	*Ilustración 22*	✓
R 2.3	En Propiedades de la simulación el usuario podrá activar o desactivar los canales A/B y, en consecuencia, el envío y la recepción de mensajes de todas las tarjetas del proyecto. Por defecto estos valores estarán activos.	*Ilustración 27*	✓
R 2.4	La representación de las BDs para los elementos de tipo Balise debe mostrar qué canales están activos mediante la implantación de 2 LED simulados.	*Ilustración 23*	✓

5.2.3. Requerimientos de espía de mensajes

Requisito	Descripción	Evidencia	Resultado
R 3.1	Los mensajes recibidos desde los CEC's serán automáticamente almacenados en un buffer interno con una capacidad para 200 mensajes.	*Código que implementa BDSimDlg*	✓
R 3.2	Si un mensaje corresponde con la selección del usuario de la vista gráfica será automáticamente mostrado en la ventana de vista de mensajes.	*Ilustración 21*	✓
R 3.3	El formato de los mensajes registrados identificará los siguientes campos: ○ Canal: canal de comunicación en uso ○ Origen: origen del mensaje ○ Fecha y hora: cuando el mensaje es enviado/recibido ○ Tamaño del mensaje: número de bytes del mensaje ○ Datos del mensaje: los datos que contiene el mensaje ○ Nombre del mensaje: nombre del mensaje	*Ilustración 29*	✓

5.2.4. Requerimientos de filtro de trazas

Requisito	Descripción	Evidencia	Resultado
R 4.1	En el filtro de trazas podremos elegir si se quiere ver los mensajes correspondientes a todas las BD's, a una en concreto o no ver ningún mensaje.	*Ilustración 24*	✓
R 4.2	En el filtro de trazas podremos filtrar aquellos mensajes que cumplan unos valores concretos en sus bits.	*Ilustración 21*	✓

5.2.5. Requerimientos de propiedades de simulación

Requisito	Descripción	Evidencia	Resultado
R 5.1	Permitirá simular el fallo físico de una tarjeta determinada y ésta se parará, rechazando los mensajes entrantes, no enviando más mensajes, y actualizando los correspondientes LED's en la vista gráfica de BD's.	*Ilustración 22*	✓
R 5.2	Permitirá reiniciar un canal y en la vista gráfica se simulará la recuperación de este fallo físico.	*Ilustración 22*	✓
R 5.3	Permitirá cambiar la dirección de una tarjeta determinada variando el valor de las direcciones de sus canales A y B.	*Ilustración 30*	✓

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

6. CONCLUSIONES

Se ha desarrollado una aplicación software que simula el comportamiento del sistema de comunicación CEC-BD en un entorno ferroviario.

Una vez realizada la implementación del proyecto, llegamos a las siguientes conclusiones acerca del mismo, valorando la total consecución de los requisitos presentados en el apartado 3 de la memoria:

1. Se ha desarrollado un simulador de utilización sencilla y potente que cubre todos los requisitos planteados en el proyecto.
2. Se ha creado una herramienta segura y fiable que cubre los requerimientos de seguridad establecidos para los elementos ferroviarios de tipo baliza.
3. En caso de que hubiera una ampliación o modificación del protocolo, es relativamente sencillo llevarla a cabo mediante la aplicación de una metodología software escalable.
4. Mediante la experiencia del desarrollo de este simulador hemos concluido que es posible simular un sistema físico aplicado a elementos ferroviarios con elementos software.
5. El alto rendimiento exigido para este simulador requiere de la aplicación de un lenguaje que permita una alta velocidad de proceso y a la vez ofrezca una estructura orientada a objetos. Por esa razón se ha realizado la implementación del programa en C++.
6. Se ha concluido que la programación de un simulador ferroviario ligada a su seguridad y estabilidad con C++ permite aprovechar las APIs y herramientas que ya existen para un mejor desarrollo y escalabilidad.

6.1. Trabajos futuros

La aplicación del simulador del elemento de campo baliza tiene recorrido más allá de la simulación de este elemento en concreto. Analizando las necesidades actuales del mundo ferroviario, las líneas de trabajo que podrían ampliar el campo que se ha presentado en la memoria son, entre otras:

1. BDSim es un simulador de balizas; no obstante podría ampliarse para simular un mayor número de elementos de campo.
2. Una posible y aconsejable mejora de BDSim sería poder leer de un fichero pruebas escritas en una nomenclatura dada y así poder ejecutar dichas pruebas automáticamente.
3. Observando los intercambios de mensaje entre los distintos elementos del sistema y dado que siempre se presupone factible la mejora, se podría proponer una forma de comunicación que en vez de basarse en 3 puntos de intercambio esté basada en 2 (CEC-Baliza), con el controlador BD incluido en el CEC. De esta manera la cantidad de cableado necesaria para la instalación sería menor y además la comunicación entre estos puntos sería directa, lo que redundaría en una cantidad de retardos inferior.

BIBLIOGRAFÍA

[1] J. M. Vera y C. Mera, «Introducción al ERTMS/ECTS,» Escuela Técnica Superior de Ingenieros Industriales (Universidad Politécnica de Madrid), Madrid, 2003.

[2] P. Winter, Compendium on ERTMS, 2009.

[3] R. Bloomfield, Fundamentals of European Rail Traffic Management System ERTMS, 2006.

[4] F. A. N. Li-Jun, «Application of ERTMS/ETCS in Train Control System,» *China Railway Science,* vol. 3, p. 20, 2003.

[5] J. C. Lorenzo Villanueva y J. I. de Santiago Cid, «El sistema ERTMS: el primer estándar paneuropeo para señalización feroviaria orientado a la interoperabilidad,» de *Anales de Mecánica y Electricidad*, 2007, pp. 56-63.

[6] A. A. Sanz, Fundamentos y Técnicas del sistema de Eurobalizas, 2014.

[7] F. Montes Ponce de Leon, «Los sistemas de señalización en el ferrocarril, su evolución,» de *Anales de mecánica y Electricidad*, Madrid, 2007, pp. 30-39.

[8] F. Montes Ponce de León, Los sistemas de control de tráfico y señalización en el ferrocarril, Madrid: Universidad Pontificia de Comillas, 2011.

[9] ENYSE Enclavamientos y Señalización Ferroviaria S.A., «Sistemas de seguridad, enclavamiento electrónico».

[10] S. Hiraguri y N. Nishibori, «Computer and microwave balise train control,» *International Railway Journal and Rapid Transit Review,* vol. 44, nº 1, 2004.

[11] D. Galán Vicente y F. González Aribas, Simulador de elementos de campo para el control de tráfico ferroviario, Madrid: Universidad Complutense de Madrid, 2010.

[12] THALES S.A., *6413 ALTRAC Detailed Design Document Centralised LEU Interface 1b Specification,* 2009.

[13] P. di Tomasso, F. Flammini, A. Lazzaro y R. &. S. A. Pellechia, «The simulation of anomalies in the functional testing of the ERTMS/ETCS trackside ssystem,» *High Assurance Systems Engineering,* pp. 131-139, 2005.

[14] I. Sommerville, Ingeniería del software, Pearson Educación, 2005.