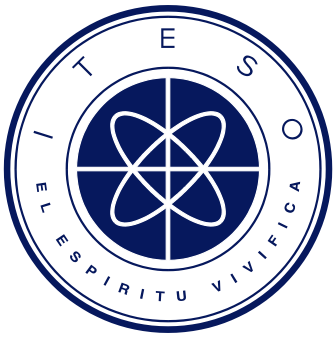
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

ESPECIALIDAD EN SISTEMAS EMBEBIDOS



ACTUALIZACIÓN DE SOFTWARE EN UNIDADES DE CONTROL EMBEBIDO POR MEDIOS INALÁMBRICOS

Trabajo final que para obtener el diploma de

ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presenta: José Luis Díaz Muñoz
Presenta: José Eduardo Luquin Ortiz
Presenta: Isidro Santos Lechuga

Asesor: Héctor Antonio Rivas Silva
Asesor: Raúl Campos Rodríguez

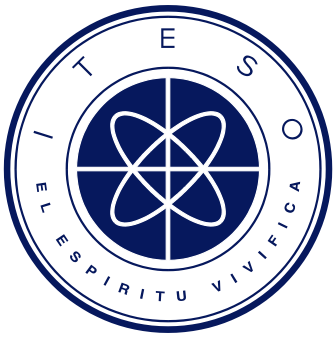
Tlaquepaque, Jalisco, Noviembre de 2016.

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

ESPECIALIDAD EN SISTEMAS EMBEBIDOS



ACTUALIZACIÓN DE SOFTWARE EN UNIDADES DE CONTROL EMBEBIDO POR MEDIOS INALÁMBRICOS

Trabajo final que para obtener el diploma de

ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presenta: José Luis Díaz Muñoz
Becario Conacyt No. 424492
Presenta: José Eduardo Luquin Ortiz
Becario Conacyt No. 424539
Presenta: Isidro Santos Lechuga
Becario Conacyt No. 424546

Asesor: Héctor Antonio Rivas Silva
Asesor: Raúl Campos Rodríguez

Tlaquepaque, Jalisco, Noviembre de 2016.

AGRADECIMIENTOS

Los autores de este trabajo agradecen sinceramente a:

- Isidro: Agradezco a mis hermanas y padres por siempre apoyarme en los retos que me he propuesto a lo largo de mi preparación profesional, a Larissa Santos Lechuga, Gilda Santos Lechuga, Hilda Lechuga Ramos y Jose Alberto Edmundo Santos Barrios;

- Jose Eduardo: Mis más sinceros agradecimientos a mis padres Jose Antonio Luquin López y Josefina Ortiz Villaseñor, por siempre apoyarme y motivarme a seguir adelante en todos los ámbitos de mi vida, también por mostrarme los resultados que se pueden alcanzar trabajando duro todos los días;

- José Luis: Agradezco a mi hermana Nelida Díaz Muñoz, por el apoyo que me ha dado para poder llevar este proyecto adelante y a mi madre Ma. Ignacia Muñoz Mejía, quien fue un ejemplo de vida tanto en lo familiar como en lo personal y que espero que siga orgullosa de mí en donde quiera que se encuentre;

- Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca otorgada a los autores de este trabajo, José Luis Díaz Muñoz, Beca No. 424492; Isidro Santos Lechuga, Beca No. 424546 y José Eduardo Luquin Ortiz, Beca No. 424539;

- A los asesores Héctor Rivas Sílva y Raúl Campos Rodríguez, por el apoyo técnico y en la revisión y redacción de este documento;

- Al ITESO por proporcionar las herramientas, laboratorios y facilidades para el desarrollo de este Proyecto de grado;

RESUMEN

El presente documento describe el diseño y el desarrollo del proyecto llamado “OTA Update Unit” (On the Air Update Unit en inglés). Este proyecto desarrolla una técnica de actualización para computadores (ECU, electronic control units en inglés) que son utilizadas dentro en los vehículos para diferentes funcionalidades, tanto críticas como de confort. Se supone que dichas actualizaciones son transmitidas mediante un enlace inalámbrico, que puede ser señales de radio digital, WiFi u otro. Una particularización de este proyecto se enfoca en actualizaciones enviadas mediante señal de Radio Digital (HD Radio).

El documento describe el diseño propuesto, su arquitectura e implementaciones en el desarrollo de la prueba de concepto en esta etapa del proyecto. Además, pretende ser una guía para la próxima generación que tenga como objetivo continuar el desarrollo de este proyecto. Pretende ilustrar los principales retos y las decisiones de diseño más importantes tomadas en el desarrollo del prototipo propuesto.

El documento se divide en cuatro capítulos fundamentales, donde se aborda con detalle el diseño, la implementación, las pruebas realizadas y sus resultados, así como una sección final con los anexos que incluyen el código fuente de las principales funciones implementadas.

El capítulo dedicado al diseño describe las arquitecturas propuestas en los módulos principales de la solución, así como una descripción del funcionamiento de las interfaces diseñadas y su flujo de ejecución.

En la sección dedicada a la implementación se puede encontrar una descripción sobre el desarrollo de cada módulo implementado. Así mismo, se puede encontrar información específica del prototipo desarrollado, como una descripción detallada de los archivos e interfaces utilizados, el flujo del protocolo comunicación y aspectos de su implementación.

En la sección dedicada a los resultados, se encuentran descritas un conjunto de pruebas realizadas para validar funcionalmente el prototipo propuesto. Debido a la complejidad del proyecto, fue necesario realizar diferentes pruebas para validar cada fase del desarrollo.

En los anexos se encuentra información relevante acerca del desarrollo del prototipo. En esta sección se puede encontrar el código fuente de la aplicación actual, así como documentación "en código" para describir las funciones principales funciones, sus parámetros de entrada utilizados y valores de retorno, cuando aplica.

Un elemento de gran importancia agregado en los anexos es la convención de nombres usada en el desarrollo del software. Esto proporciona una formidable ayuda para la lectura y comprensión del código desarrollado y su futuro desarrollo. Igualmente, se incluye una serie de nomenclaturas que sirven como referencia para los desarrolladores.

Debido a que mucho del material de tecnología de punta en el área automotriz se encuentra en inglés, muchos de los términos se conservan en dicho idioma y se colocan entre comillas. Por ejemplo "message key" de deja de esta manera en lugar de su versión en español como clave de mensaje. Esto aplica para acrónimos, nombres de funciones, variables, constantes en código, acrónimos y nombres comunes en el campo de las aplicaciones automotrices. Esto para facilitar la lectura del documento y su explicación de manera más intuitiva. De este modo se posible revisar, en el código proporcionado en los apéndices, la implementación de alguna función descrita en el texto del documento.

ABSTRACT

This document describes the design and development of the project called "OTA Update Unit" (On the Air Update Unit). This project develops a technique for the embedded software update of electronic control units (ECU) that are used inside in vehicles for different functionalities, both critical and comfort. It is assumed that such updates are transmitted over a wireless link, which may be digital radio, Wi-Fi or other signals. At this stage, this project focuses on updates sent by Digital Radio signal (HD Radio).

The document describes the proposed design, its architecture and implementations in the proof of concept development at this stage of the project. In addition, it aims to be a guide for the next team generation which purposes is to continue the development of this project. It intends to illustrate the main challenges and the most important design decisions taken in the development of the proposed prototype. The document is divided into four fundamental chapters, which discusses in detail the design, implementation, tests performed and their results, as well as a final section with the annexes that include the source code of the main functions implemented.

The chapter on design describes the architectures proposed in the main modules of the solution, as well as a description of the operation of the designed interfaces and their execution flow. In the section dedicated to implementation you can find a description of the development of each module implemented. Likewise, specific information about the developed prototype can be found, such as a detailed description of the files and interfaces used, the flow of the communication protocol and aspects of its implementation.

In the section devoted to the results, a set of tests performed to functionally validate the proposed prototype are described. Due to the complexity of the project, it was necessary to perform different tests to validate each phase of development. Relevant information about the development of the prototype is found in the annexes. In this section you can find the source code of the current application as well as documentation "in code" to describe the main functions, their input parameters used and return values, when applicable. An element of great importance added in the annexes is the

naming convention used in software development. This provides a formidable help for reading and understanding the code developed and its future development. Also included is a series of nomenclatures that serve as a reference for developers.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ACRONIMOS

ACRONIMO	SIGNIFICADO
ECU	Electronic Control Unit
FAT	File Allocation Table
MCU	Microcontroller Unit
CAN	Controller Area Network
OTA	Over the Air
USB	Universal Serial Bus
OTG	On the Go
MSC	Mass Storage Class
UART	Universal asynchronous Receiver-Transmitter
BSP	Board Support Package
RTOS	Real Time Operating System
RAM	Random Access Memory
AES	Advanced Encryption Standard
GCC	GNU Compiler Collection
OEM	Original Equipment Manufacturer

1. INTRODUCCION

Resumen: *Este capítulo proporciona una introducción general al proyecto de grado descrito en este documento.*

1.1. INTRODUCCIÓN

Radio de Alta Definición (HD Radio en inglés) es un estándar de radio digital el cual permite la trasmisión de una señal digital sobre las frecuencias de radio tradicionales FM y AM, admitiendo de esta manera agregar contenido adicional a la información trasmitida de manera convencional. El estándar ofrece un sin fin de beneficios tales como una gran calidad de sonido, y la capacidad de trasmitir datos como títulos de las canciones, artistas e incluso imágenes de los álbumes de la canción, por mencionar algunos. Su protocolo es capaz de enviar grandes cantidades de información permitiendo la transmisión de uno o múltiples canales audio en la misma frecuencia, lo que se conoce como "multicasting". Es importante resaltar que tales beneficios se hacen sin cargos adicionales al cliente final, exentándolo de cuotas o suscripciones mensuales, como otros servicios.

Adicionalmente la red de datos de Radio HD permite trasmitir de forma digital los reportes de tráfico, condiciones climáticas, los últimos resultados de marcadores de diversos deportes e incluso mensajes de emergencia. Debido a la gran cantidad de información que puede ser transmitida atreves de este estándar, surge la oportunidad de poder enviar paquetes de datos los cuales contengan las últimas actualizaciones de software para los dispositivos ECU dentro de los automóviles. Dichas actualizaciones permitirán, por ejemplo, que los OEM's actualicen el software de los automóviles usando las ideas propuestas en este trabajo.

1.2. PROGRAMACIÓN POR EL AIRE (OTA PROGRAMMING)

OTA por sus siglas en inglés (On-the Air programing) se refiere a el proceso para distribuir nuevas versiones de software, actualizaciones o nuevas configuraciones a una amplia gama de dispositivos atreves del aire, este método de distribución ha sido principalmente explotado en el área telefonía móvil aprovechando su conectividad a las redes inalámbricas. Esta posibilidad se utiliza para hacer actualizaciones al firmware de dispositivos conectados a la red, cuyo procedimiento de actualización es denominado FOTA (Firmware on-the air).

En términos generales, una actualización de software por aire usualmente se da en tres etapas. En la primera, se requiere de una forma confiable y segura de diseminar la nueva imagen de firmware que se desea actualizar. En este punto, típicamente, se hace necesario que la aplicación provea un método de autentificación a través de una llave compartida y la encriptación de paquetes para evitar que los datos sean interceptados con un potencial uso malintencionado.

En la segunda etapa, una vez que el software fue completamente recibido, se debe llevar a cabo un proceso de reprogramación a prueba de fallos para aquellos nodos para los que está dirigido el software descargado previamente. Esta es una de las parte más complejas etapas de este proceso de actualización, debido a que los protocolos de un nuevo firmware pueden ser incompatibles con los más viejos.

Finalmente, en la tercera y última etapa, para hacer frente a una posible carga software erróneo es necesario algún mecanismo de recuperación, ya que incluso aun habiendo implementado una extensa cantidad de pruebas es posible que el nuevo software no trabajare de manera correcta bajo ciertas condiciones. En este escenario, es posible que la mejor opción sea volver a la versión anterior del software.

1.3. DIAGRAMA CONCEPTUAL PROGRAMACIÓN POR EL AIRE (OTA PROGRAMMING)

La Figura 1 muestra un diagrama conceptual del OTA. En el nivel más general, el diagrama considera un nodo, que representa el objetivo de la actualización de software. En este caso el nodo representa un auto, camino de pasajeros, o camión de carga, por mencionar algunos. En este mismo nivel de abstracción se encuentra el administrador de red. Este se encarga de la gestión y distribución de los paquetes de actualización para los nodos.

En este diagrama conceptual, el OTA manager recibe actualizaciones atreves de una red inalámbrica. Este diagrama captura las generalidades de la idea que se pretende desarrollar en este trabajo. De este modo, el diagrama funciona de manera adecuada para el diseño de los diferentes bloques que formaran el software y su funcionalidad a implementar.

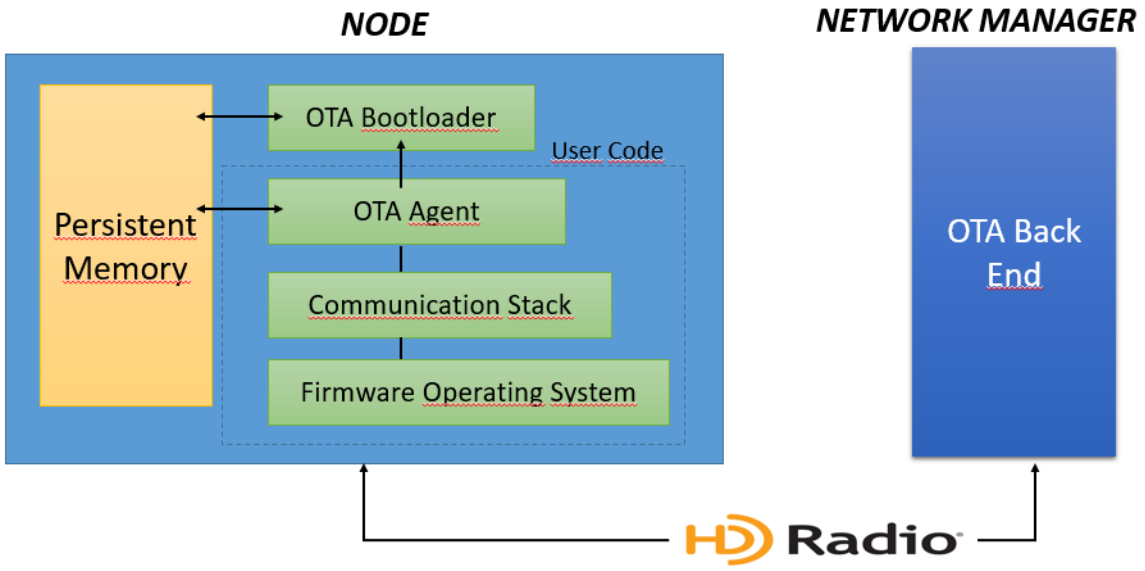


Figura 1. Diagrama Conceptual OTA [9]

En el contexto de aplicaciones automotrices, este tipo de distribución de software tiene múltiples beneficios, ya que permite la actualización no solo de software sin también actualizaciones de seguridad. Esto abre la posibilidad a los OEM's realizar actualizaciones en segundo plano, evitando costosos proceso de revisión, conocidos como "recalls" en el argo de la industria automotriz. De esta manera se mejorar la satisfacción del cliente, la distribución de nuevas funcionalidades y se incrementar el tiempo de vida útil de los productos.

Figura 2. Funcionalidades del Auto que se pueden actualizar mediante el concepto OTA [7]

Por medio del envió de archivos binarios mediante el concepto OTA se pueden cargar códigos para actualización de ECU's y actualizaciones de firmware en general. De hecho, se puede hacer la actualización del conjunto de datos de aplicaciones como mapas para la navegación abordo, archivos personales de música, videos, fotos e incluso actualización de paquetes de software de aplicaciones de confort y seguridad y del mismo sistema operativo del automóvil.

Algunas de las características de los servicios basados en el concepto OTA que se deben tener en cuenta son el manejo de persistencia para los diferentes usuarios, roles, componentes y paquetes de software que pudieran existir en los diferentes escenarios. Además deben ser escalables y tener una infraestructura que les permita soportar altos volúmenes de información. También es sumamente importante que los servicios OTA cuenten con múltiples niveles de seguridad y autentificación, así como un mecanismo de reversión para volver atrás a un punto funcional del software en caso de problemas.

La Figura 2 muestra de manera gráfica un posible conjunto de servicios de actualización con potenciales usos en la industria automotriz que se pueden implementar mediante la implementación del concepto OTA.

2. ESTADO DE LA TECNICA

Resumen: *Este capítulo proporciona una breve revisión de antecedentes y el estado de la técnica en la actualización de software en la industria automotriz.*

2.1. ANTECEDENTES

Las actualizaciones de software se están convirtiendo en una nueva tendencia dentro de diversas industrias, incluida la automotriz. Como consecuencia de un "coche conectado", se obtienen ventajas útiles para los fabricantes de automóviles además del solo hecho te tener Internet dentro del coche. Por ejemplo, los mapas de navegación pueden ser descargados y actualizados por el aire. Entretenimiento en línea también es susceptible para ser descargado, así como juegos en línea mientras que la familia está en un viaje. Algunos modelos de coche son capaces de compartir su conexión a Internet a una serie de nodos inalámbricos, como laptops, tabletas o teléfonos móviles.

Por otro lado, debido a que cada vez más funcionalidades en los automóviles se están trasladando a complejas computadoras automotrices, o ECU's por su siglas en inglés, un sistema de actualizaciones de software se hace algo muy atractivo y necesario para la industria. De este modo, se abre una ventana de oportunidades completamente nueva y prometedora para los nuevos modelos de autos del futuro.

En este esquema, los fabricantes de automóviles pueden lanzar algún software básico para sus nuevos automóviles y luego actualizar dicho software mediante liberaciones posteriores. De esta manera, los coches pueden mantener al día sus funcionalidades. Al mismo tiempo, los fabricantes pueden vender algunas funcionalidades adicionales en una especie de "tienda de aplicaciones", donde los usuarios pueden comprar "skins" para el tablero de sus autos, música, películas o juegos, por mencionar algunos contenidos que se pueden vender.

Tal vez más importante que lo antes mencionado es que, con un sistema de actualizaciones como el que se discute en este documento, los fabricantes de automóviles pueden ser capaces de corregir errores de software de ECU's dentro del automóvil mediante el envío en el aire de paquetes de actualización y corrección, sin la costosa necesidad de llamar un coche a revisión de taller.

Incluso, puesto que la tendencia es que los coches incluyan más sensores, radares y cámaras, es desarrollo de un sistema de conducción automática que responde a condiciones de carga en tiempo real es posible. Así, nuevos algoritmos para el reconocimiento de la señal de tráfico pueden desarrollarse en el laboratorio y luego ser entregados a los automóviles mediante actualizaciones con un sistema OTA.

En conclusión, los coches conectados, con una gran flexibilidad en sus funcionalidades, y de característica actualizables parecen ser el futuro de la industria automotriz en los años próximos. La subsección siguiente revisa brevemente algunos de los sistemas que se presentan en varios modelos de automóviles modernos.

2.1. TESLA

El fabricante Tesla es uno de los principales protagonistas en las actualizaciones de software para sus coches eléctricos de lujo. Tesla suele enviar actualizaciones a su sistema operativo automotriz, y principalmente a su subsistema de piloto automático. Proporcionan nueva funcionalidad y solucionan problemas con este sistema de actualizaciones de software.

Cuando se dispone de una actualización para el sistema Tesla, la pantalla principal informa de ello y solicita una instalación inmediata o una actualización programada por el conductor. Las actualizaciones de software se descargan principalmente a través de una conexión Wi-Fi, normalmente la conexión a Internet en la casa del conductor. Mediante este sistema, Tesla ha actualizado con éxito los reproductores de medios del coche, los mapas y el subsistema del piloto automático. La Figura 3 muestra el centro de actualización de un coche Tesla.

Figura 3. Una actualización disponible en el tablero central de un auto Tesla [1]

2.2. FORD MOTORS

El sistema SYNC de los modelos Ford permite al conductor utilizar comandos de voz para hacer una llamada, escuchar música, utilizar comandos de control de voz, y seleccionar aplicaciones, entre otras funcionalidades. La tecnología SYNC activada por voz también mejora la seguridad, al permitir mantener los ojos en la carretera y las manos en el volante.

Mediante actualizaciones de software, el sistema SYNC de Ford permite a los conductores disfrutar de muchas de las grandes características y capacidades mejoradas, a través de una pantalla inteligente organizada de manera muy intuitiva y natural.

El fabricante de autos Ford distribuye su software actualizado mediante un enlace Wi-Fi, mientras que el proceso de instalación se realiza normalmente cuando el vehículo se encuentra en la cochera y este se arranca de nuevo. Los planes de la compañía son cambiar a hacer actualizaciones vía satélite.

Mediante un sistema de envío de actualizaciones de software, el fabricante Ford planea reducir el número de versiones de su software. En lugar de crear variantes específicas para cada país, por ejemplo, Ford será capaz de cargar interfaces en diferentes idiomas mediante actualizaciones. La Figura 4 muestra la pantalla inteligente en el centro de mando de su sistema.

Figura 4. Sistema Ford SYNC y Touch [2]

2.3. GENERAL MOTORS

Mediante su sistema OnStar, General Motors entrega contenido y actualizaciones de software para sus modelos de autos. El fabricante soporta por medio de este servicio, capacidades para el manejo autónomo, que promete será más amplio en el futuro cercano. La entrega de software y contenido se realiza mediante un enlace inalámbrico 4G, pero no se descarta que en un futuro se realice mediante señal de satélite.

El fabricante General Motors considera que conforme los autos avances con más énfasis en el manejo autónomo e inteligente, la necesidad de actualizaciones de software de manera inalámbrica será cada vez más importante. La Figura 5 representa un mapa conceptual del sistema OnStar de GM.

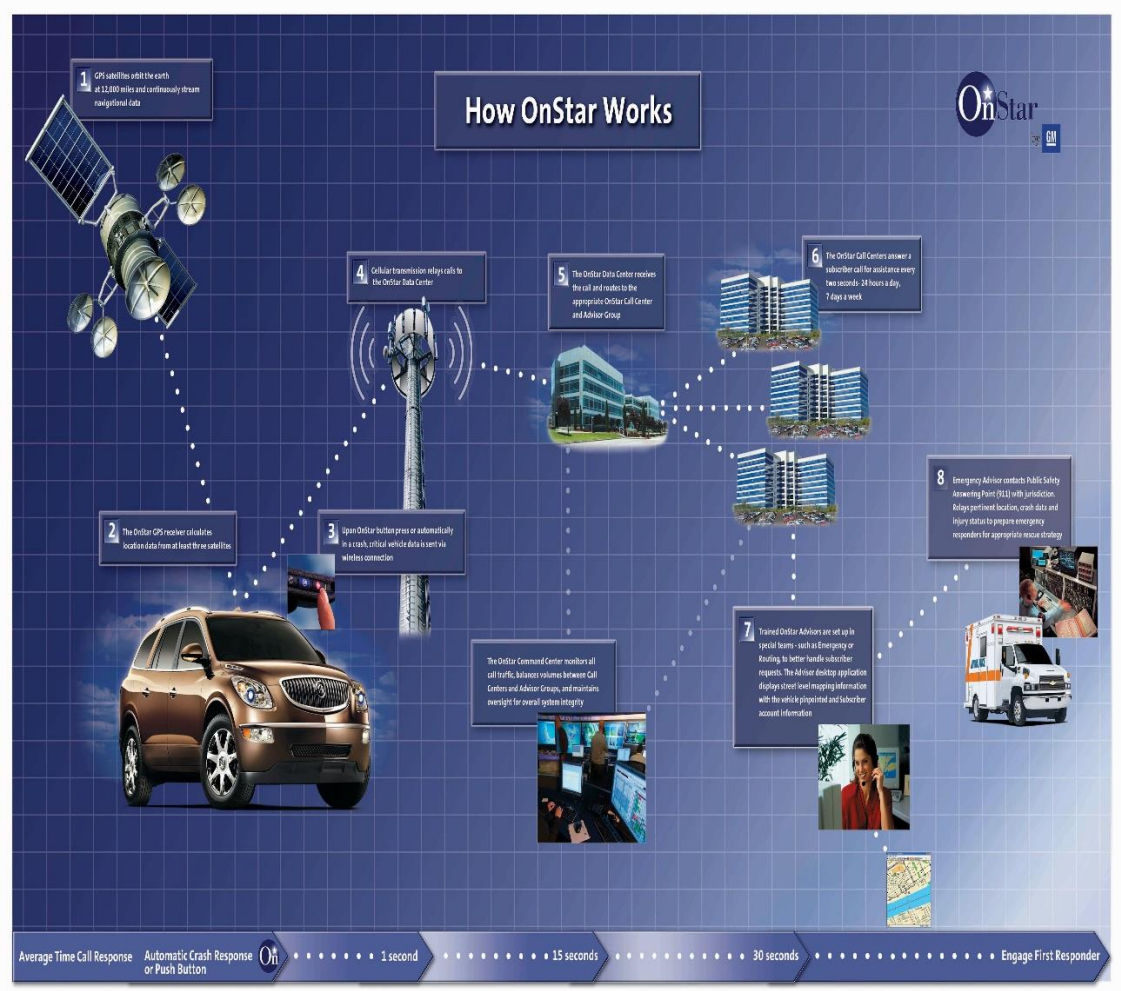


Figura 5. Modelo Conceptual del Sistema OnStar de GM [3]

Debido a todos los retos y oportunidades que plantea un carro actualizable, una consideración de negocios puede parecer crucial para los fabricantes de automóviles. En este sentido, la capacidad de mejorar continuamente las características y el rendimiento de los automóviles hará que la idea de comprar un coche nuevo sea más tentadora, de manera similar a lo que sucede en el mercado de los teléfonos inteligentes.

3. ARQUITECTURA DE LA SOLUCION PROPUESTA

Resumen: *En este capítulo se presenta la arquitectura de los diferentes bloques que conforman la solución propuesta. En particular, se revisa el protocolo de comunicación, el sistema de almacenamiento de archivos y el método de encriptación de la información.*

3.1. ACTUALIZACIONES DE SOFTWARE OTA

El Prototipo desarrollado en este trabajo fue diseñado para poder transmitir paquetes de datos de manera inalámbrica. El conjunto de paquetes conforman una o varias actualizaciones para diferentes ECU's dentro de los vehículos cuyo código embebido debe ser puesto al día.

Se supone que las posibles actualizaciones son enviadas usando algún mecanismo de comunicación inalámbrico unidireccional. Debido a esa razón el protocolo diseñado en este trabajo se planeó para soportar perdidas de comunicación y una fácil re sincronización. En un ejemplo de escenario real, la información es enviada por medio de una señal de radio satelital la cual es propensa a interrupciones y perturbaciones de información.

En esta fase del proyecto el objetivo principal es diseñar un protocolo de comunicación funcional para los propósitos antes mencionados. Igualmente es un objetivo desarrollar un prototipo de un dispositivo receptor capaz de mantener las actualizaciones en memoria no volátil y la capacidad de transmitirlas por medio del bus de CAN, que es uno de los buses estándares en la industria automotriz.

Actualmente el prototipo es capaz de recibir la actualización y guardarla en una memoria USB. Adicionalmente, el prototipo es capaz de mantener comunicación por el bus de CAN. Como etapas futuras de este proyecto, es necesario desarrollar el protocolo que tendrá que ser usado en el bus de CAN. Como prueba de concepto, actualmente la aplicación manda 3 mensajes diferentes con la finalidad de ejecutar la operación para encender las luces direccionales. Esto demuestra la capacidad de mantener la comunicación en una red de CAN.

Así mismo, la segunda fase de este proyecto debe considerar la integración del bus de CAN a un "bootloader" para poder así hacer la actualización usando el bus CAN mediante las capacidades desarrolladas en este trabajo. La Figura 6 muestra un diagrama conceptual sobre las fases de desarrollo de este proyecto.

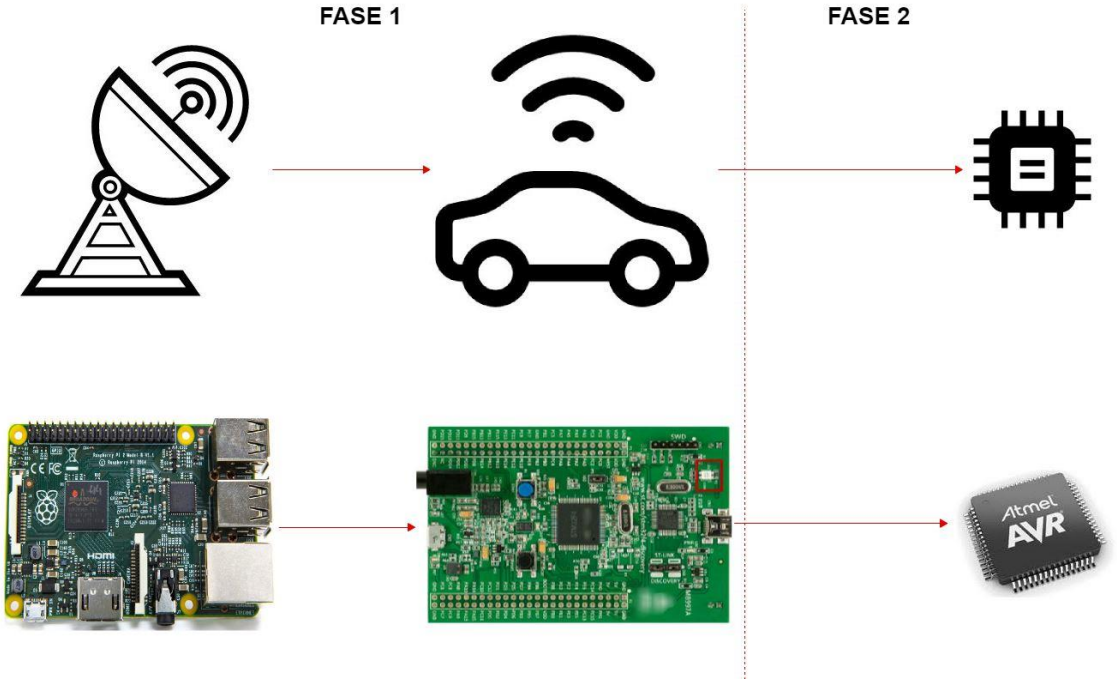


Figura 6. Fases en el desarrollo del proyecto

3.2. PROTOCOLO DE ACTUALIZACIÓN DE SOFTWARE

Para atender los retos que plantea este trabajo, es necesario un buen diseño de protocolo de comunicación que tome en cuenta fallos en la comunicación. Además, dicho protocolo deberá tener en cuenta actualizaciones de diferentes tamaños de manera eficiente, así como actualizaciones para más de una ECU dentro del auto.

3.2.1. Introducción al protocolo

El protocolo desarrollado consiste en 3 mensajes principales que pueden ser recibidos de manera secuencial o no. Esto significa que no es necesario recibir paquetes de la misma actualización secuencialmente para que se pueda guardar el archivo de manera íntegra en el sistema.

Con estas consideraciones en mente, se describen los mensajes usados por el protocolo de comunicaciones:

- **"New Broadcoast Message":** Este mensaje es usado para anunciar al dispositivo receptor que la transmisión de una nueva actualización será transmitida. El mensaje tiene contiene un ID único para diferenciar entre cada transmisión. Cada ID contiene una aplicación que será transmitida por un tiempo definido.

- **"Attribute Message":** Este mensaje es usado para poder informar al dispositivo receptor de toda la información relacionada a la actualización transmitida. Esta información es usada para identificar la ECU que tendrá que ser actualizada.

- **"Data Message":** Este mensaje contiene los datos que integran todo el paquete de actualización. Este mensaje contiene un ID para especificar el paquete transmitido y la transmisión a la que pertenece.

3.2.2. Descripcion de "New Broadcoast Message"

La estructura del mensaje "New Broadcoast Message" es descrita en la Tabla 1. Un total de seis campos integran el mensaje.

Tabla 1. Estructura del mensaje "New Broadcoast Message"

New Boradcoasat Message					
Sync	ID Message	Num of Attributes	Number of data Pkgs	SW Key	Checksum
4	1	1	4	1	1

La descripción de los campos del mensaje es como sigue:

- **“Sync”:** Este campos del mensaje es usado para la sincronización del mensaje y está representado por 4 bytes con los siguiente valores [0x00, 0x00, 0x55, 0x55].

- **“ID Message”:** Este campo del mensaje tiene un tamaño de 1 byte y es usado para identificar el tipo de mensaje transmitido. Para un mensaje de tipo “New Broadcoast Message” el ID tiene un valor de [0x05].

- **“Num of Attributes”:** Este campo del mensaje tiene un tamaño de 1 byte. Este valor representar el número de atributos que serán transmitidos para facilitar la identificación de la actualización transmitida.

- **“Number of data Pkgs”:** Este campo del mensaje tiene una longitud de 4 bytes. El byte de la izquierda es el más significativo, y representa el número de paquetes de datos que serán transmitidos para dicha actualización.

- **“SWKey”:** Este campo del mensaje tiene un tamaño de 1 byte. Este valor representa un identificador único el cual es usado para poder ligar los mensajes de atributo y paquete con una transmisión específica.

- **“Checksum”:** Este campo del mensaje tiene un tamaño de 1 byte. El valor “checksum” permite una operación basada en sumas “xor” para mantener la integridad de los datos. En este caso se suman los valores desde el número de atributos hasta el “SWKey”.

3.2.3. Descripcion de “Attribute Message”

La estructura del mensaje “Attribute Message” es descrita en la Tabla 2. Un total de siete campos integran el mensaje.

Tabla 2. Estructura del mensaje "Attribute Message"

Attribute Message						
Sync	ID Message	SWKey	Attr ID	Param Length	Parameter	Checksum
4	1	1	1	2	0-15	1

La descripción de los campos del mensaje es como sigue:

- **"Sync":** Este campos del mensaje es usado para la sincronización del mensaje y está representado por 4 bytes con los siguiente valores [0x00, 0x00, 0x55, 0x55].

- **"ID Message":** Este campo del mensaje tiene un tamaño de 1 byte. Este campo es usado para identificar el tipo de mensaje transmitido. Para un "Attribute Message" el ID tiene un valor de [0x06].

- **"SW Key":** Este campo del mensaje tiene un tamaño de 1 byte. Este valor representa un identificador único el cual es usado para poder ligar los mensajes de atributo y paquete con una transmisión específica.

- **"Attr ID":** Este campo del mensaje tiene un tamaño de 1 byte. Este campo es un identificador, usado para saber que atributo se está recibiendo en el mensaje. Este ID puede tener significados diferente en cada "automaker". Esta información es útil ya que contiene las propiedades del paquete recibido.

- **"Param Length":** Este campo del mensaje tiene un tamaño de 2 bytes. Este campo representa el número de bytes que se van a recibir como parámetro en el mensaje.

- **“Parameter”:** Este campo del mensaje puede tener un tamaño de 0 a 15 bytes. Los parámetros contienen una cadena de caracteres que representa a un atributo de la actualización transmitida.

- **“Checksum”:** Este campo del mensaje tiene un tamaño de 1 byte. El “checksum” permite operación basada en sumas “xor” para mantener la integridad de los datos. En este caso se suman los valores desde el número de atributos hasta el “SWKey”.

3.2.4. Descripción de “Data Message”

La estructura del mensaje “Data Message” es descrita en la Tabla 3. Un total de seis campos integran el mensaje.

Tabla 3. Estructura del mensaje “Data Message”

Data Message					
Sync	ID Message	SWKey	Pkg ID	Datas	Checksum
4	1	1	4	5	1

La descripción de los campos del mensaje es como sigue:

- **“Sync”:** Este campos del mensaje es usado para la sincronización del mensaje y está representado por 4 bytes con los siguiente valores [0x00,0x00,0x55,0x55].

- **“ID Message”:** Este campo del mensaje tiene un tamaño de 1 byte. Este campo es usado para identificar el tipo de mensaje transmitido. Para un “Data Message” el ID tiene un valor de [0x07].

- **“SW Key”:** Este campo del mensaje tiene un tamaño de 1 byte. Este valor representa un identificador único el cual es usado para ligar los mensajes de atributo y paquete con una transmisión específica.

- **“Package ID”:** Este campo del mensaje tiene un tamaño de 4 bytes. El “Package ID” es utilizado para diferenciar el paquete recibido. El valor de la izquierda es el más significativo.

- **“Datas”:** Este campo del mensaje tiene un tamaño de 4 bytes. Este campo representa los datos que forman parte de la actualización transmitida. Dicha información se transmite de manera encriptada.

- **“Checksum”:** Este campo del mensaje tiene un tamaño de 1 byte. El “checksum” permite operación basada en sumas “xor” la cual es usado para mantener la integridad de los datos. En este caso se suman los valores desde el número de atributos hasta el “SWKey”.

3.3. ARQUITECTURA DEL SOFTWARE OTA-RECEIVER

OTA-Receiver es el nombre de la aplicación desarrollada para el dispositivo receptor de la actualización la cual fue desarrollada en “Atollic TrueSTUDIO” para ARM V5.1.1 (Figura 7).

Figura 7. Entorno de Desarrollo Attollic TrueSTUDIO para ARM

La funcionalidad principal de la aplicación es recibir datos correspondientes a la actualización de otra ECU. Dichos datos son transmitidos usando el "Software Update Protocol" descrito en la sección anterior. Para una prueba de concepto, en este trabajo se reciben por medio de un enlace UART2. La Figura 8 muestra un diagrama de arquitectura de la aplicación "OTAReceiver Unit".

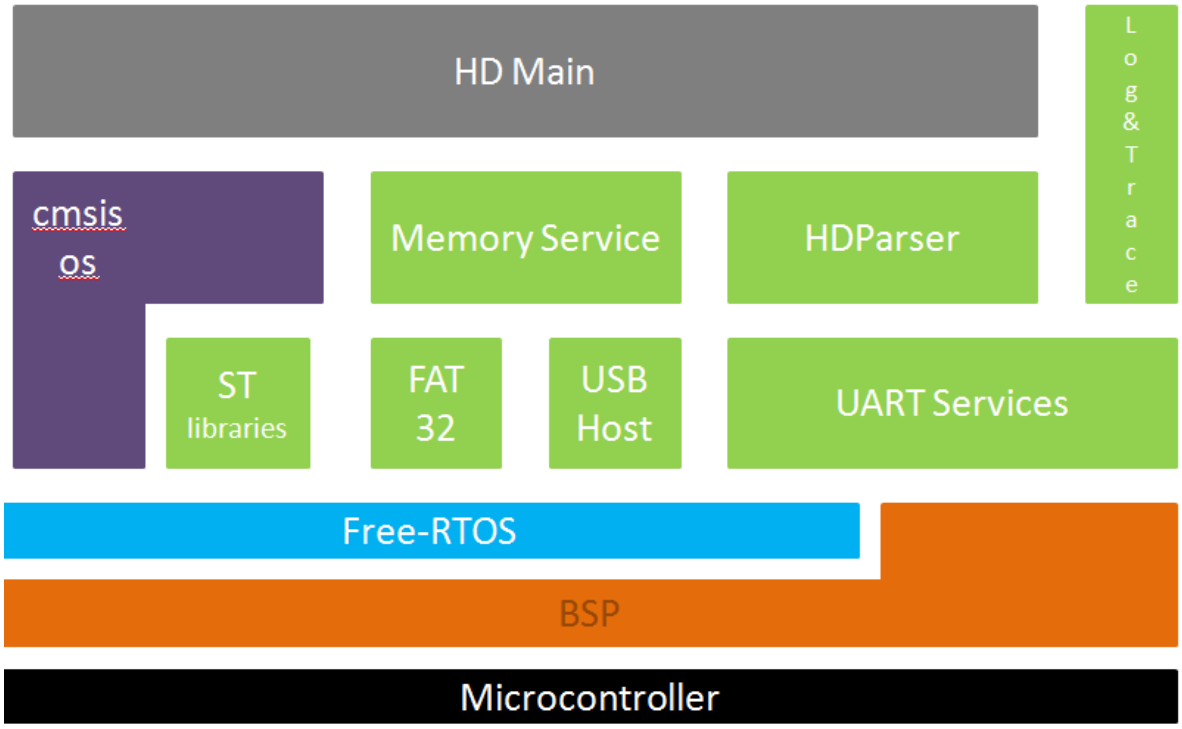


Figura 8. Arquitectura de la Aplicación "OTAReceiver Unit"

Los componentes de la aplicación se describen a continuación:

- **"HDMain":** Este módulo se encuentra en la capa más alta de la aplicación desarrollada y es responsable de atender los mensajes recibidos por "HDParser".

- **"HDParser":** Diseñado para identificar los mensajes recibidos y guardar la información de ellos en una lista, dichos mensajes serán consumidos por "HDMain".

- **"UART Services":** Se diseñaron dos servicios para los módulos que tienen la necesidad de usar el UART, un servicio es usado para el modulo "Log&Trace" y el otro para "HDParser".

- **“Memory Service”:** Contiene las funciones que “HDMain” necesita para poder almacenar la información en un formato que sea útil y de buena organización para las diferentes actualizaciones recibidas.

En la Figura 9 se puede observar un diagrama de flujo del comportamiento correspondiente a la aplicación desarrollada, básicamente la aplicación espera a tener completa una actualización para entonces enviarla a la ECU residente.

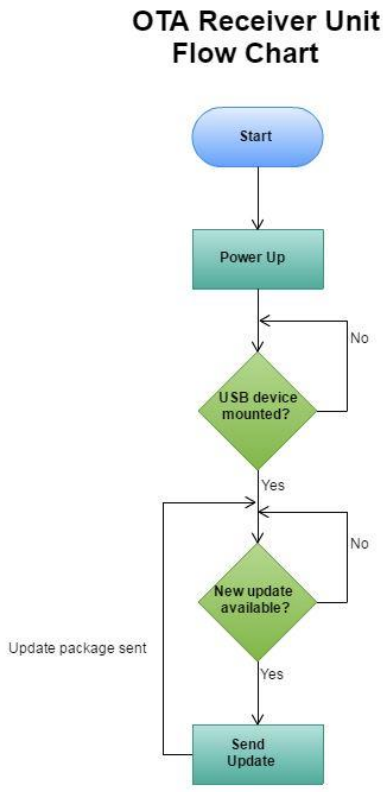


Figura 9. Diagrama de Flujo de la Aplicación “OTAReceiver Unit”

3.4. FUNCIONALIDAD DEL MODULO FAT

El modulo FAT fue creado como una alternativa para resolver los problemas de memoria que se presentaron en varias plataformas de hardware en las que se realizaron experimentos durante el

desarrollo de este proyecto, incluida la última tarjeta utilizada, que fue la plataforma de desarrollo DISCOVERY.

El módulo FAT tiene el propósito de manejar y gestionar la información de los paquetes de actualización recibida. Para las pruebas de concepto de este trabajo, dicha recepción de paquetes se realiza a través del puerto UART. Sin embargo, por diseño, dicho mecanismo de comunicación puede ser otro, como Wi-Fi o radio satelital. La información que administra el modulo FAT corresponde a un conjunto de datos o código que serán actualizado a una o varias ECU dentro del automóvil.

En escenarios de aplicación real, se espera que la cantidad de información recibida sea de tamaño variable y que esto presente retos debido al gran tamaño que pudiera llegar a tener. Muchas de las soluciones de la industria automotriz cuentan con una limitada cantidad de memoria. De este modo, la estrategia consiste en seccionar y posteriormente guardada en pequeñas unidades que les llamamos paquetes la información recibida para actualizar el código de una ECU.

La solución propuesta consiste en tratar dichos paquetes en un archivo binario que posteriormente es guardado en una unidad de memoria externa, como una tarjeta SD o una memoria USB. De esta forma se evita saturar la memoria del sistema con grandes cantidades de información.

Se probaron varios sistemas de almacenamiento, como una memoria flash externa o una tarjeta micro SD, ya que existen librerías de soporte previamente desarrolladas y disponibles para los desarrolladores. Finalmente, para probar los conceptos desarrollados en este trabajo se llegó a la conclusión que lo más práctico era implementar un driver propio para hacer uso de una memoria externa USB en la tarjeta de desarrollo DISCOVERY.

De la misma manera, conforme el proyecto se iba desarrollando y su complejidad incrementándose, se tuvo la necesidad de adaptar la funcionalidad de este módulo con el fin de satisfacer las nuevas necesidades que se incorporaron. Una de ellas tuvo que ver con tamaño y distribución de los paquetes que forman las actualizaciones. El modulo FAT fue inicialmente desarrollado para contener paquetes de 128 bytes.

En términos generales, el modulo FAT tienen dos grandes funcionalidades. Estas funcionalidades están separadas en sesiones de lectura y de escritura, ambas descritas con detalle en los capítulos siguientes.

Antes de comenzar a operar este módulo, es importante cumplir con ciertas precondiciones para un óptimo funcionamiento. Primero, es necesario considerar que el manejo del sistema de archivos impuesto por la tarjeta de desarrollo DISCOVERY es un poco diferente al manejo del utilizado en una PC normal de escritorio. En la tarjeta DISCOVER, es necesario montar el dispositivo donde los archivos serán gestionados, que pare pruebas de concepto en este proyecto, es una memoria USB externa. Una vez conectada la memoria y montado el modulo, éste estará listo para realizar la funcionalidad previamente descrita. La Figura 10 muestra un diagrama sobre la organización de los archivos binarios dentro del sistema FAT.

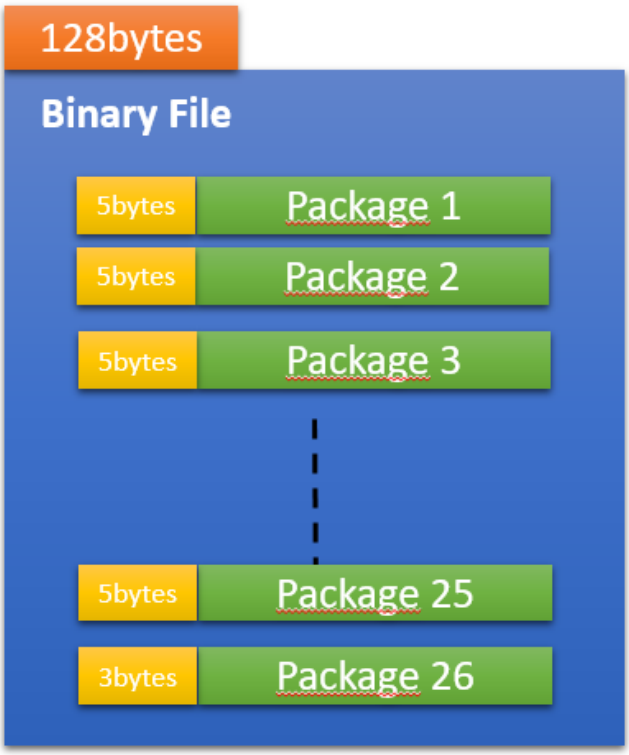


Figura 10. Diagrama de bloques de la organización de los archivos binarios en el sistema FAT

3.4.1. Sesión de Escritura Módulo FAT

La Figura 11 muestra un diagrama de estados para la sesión de escritura del módulo FAT. La figura ilustra los principales estados que se deben seguir para escribir paquetes en un archivo binario. Como se puede observar en la secuencia de inicio, antes de comenzar a abrir una sesión de escritura, el dispositivo donde se guardaran los archivos debe estar montado y listo para utilizarse. Esta verificación es necesaria para evitar múltiples problemas de sincronización que surgen al intentar usar un sistema de archivos FAT sin antes tener el dispositivo de soporte conectado previamente.

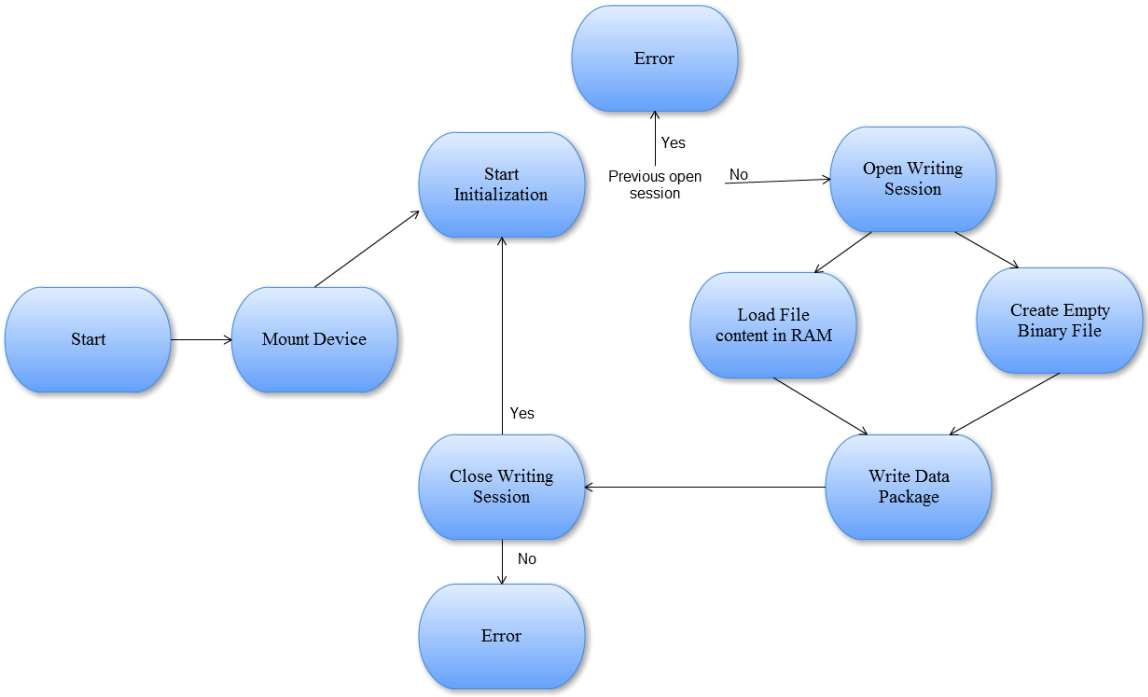


Figura 11. Diagrama de Estado Operación de Escritura Modulo FAT

El diagrama contempla un circuito que permite escribir varios paquetes en un solo archivo. Este ciclo puede crear un archivo nuevo si este no se encuentra previamente en el sistema. En caso de que el archivo ya exista en el sistema, éste se carga en la memoria RAM para su utilización.

Durante la inicialización hay una validación que permite a la aplicación conocer si existe una sesión previamente abierta. En el diseño actual del módulo solo se soporta una sesión a la vez. Esto permite evitar que los archivos modificados sean corrompidos y de esta manera asegurar que la memoria RAM se mantenga libre tanto como sea posible para otros fines. El propósito de tener diferentes sesiones para el manejo de archivos es evitar abrir y cerrar constantemente contextos para el mismo archivo, lo cual es computacionalmente costoso. Con el enfoque propuesto en este trabajo, el archivo es cerrado hasta que no se requiere más.

Cuando la aplicación comienza por primera vez una sesión de escritura, un archivo binario nuevo es creado. Sin embargo, si el archivo existía previamente, su contenido es cargado en memoria RAM para su operación. Debido a que el contenido del archivo binario en cuestión es cargado en memoria, es relativamente fácil editarlo, ya sea agregando nuevos paquetes al archivo, o remplazando algún paquete no recibido correctamente. De esta manera, la carga de trabajo es reducida de manera importante, y el diseño a implementar se simplifica considerablemente.

Por otro lado, el uso de sesiones también hace que el código sea más sencillo de administrar. Por ejemplo, cuando un archivo binario ya no se necesita editar más, o bien es necesario trabajar con un archivo diferente, la sesión activa debe ser cerrada, el archivo activo debe ser guardado y el contexto asociado al archivo es borrado. Finalmente la memoria RAM donde se respaldó la información es liberada y el ciclo se repite con los nuevos archivos binarios.

Si la sesión activa se cerró de manera satisfactoria, una bandera local cambiara su estado para indicar que una nueva sesión puede ser abierta. Como se puede apreciar en el diagrama de estados de la Figura 11, una vez que el dispositivo fue montado por primera vez, éste no será desmontado hasta que el sistema se vuelva a apagar o sea reiniciado. Se eligió este tipo de diseño debido a que montar un dispositivo requiere mucho tiempo del CPU y hacerlo durante cada sesión provocaría problemas de sincronía y una pérdida de desempeño del sistema global.

3.4.2. Sesión de Lectura Módulo FAT

De la misma manera que en la sesión de escritura, el punto de entrada para la operación de lectura es contar con un dispositivo USB previamente montado, y listo para su utilización. Es importante aclarar que aunque los diagramas de sesión de lectura y escritura muestran el montado del dispositivo, este proceso se realiza solo una vez durante la inicialización del sistema, a través de la etapa de Administración de Potencia (“Power Managment” en inglés).

Con este requisito satisfecho, el modulo verificara si existe una sesión activa abierta y si, no la hay se procede a levantar la sesión de lectura, se procede a buscar el archivo solicitado y si existe entonces su contenido es guardado en RAM específicamente en un arreglo que contendrá hasta 26 paquetes de 5 bytes cada uno para un total de 128 bytes por archivo binario.

La lectura en un archivo binario puede darse de dos maneras ya sea que se obtenga el contenido de determinado paquete o el contenido del binario completo. Estos dos tipos de lectura fueron pensados en que posiblemente existirían errores al leer algunos paquetes , de tal forma que en lugar de leer el contenido del binario completo solo se hiciera la lectura de aquel paquete que se necesitara volver a extraer ahorrando trabajo al procesador.

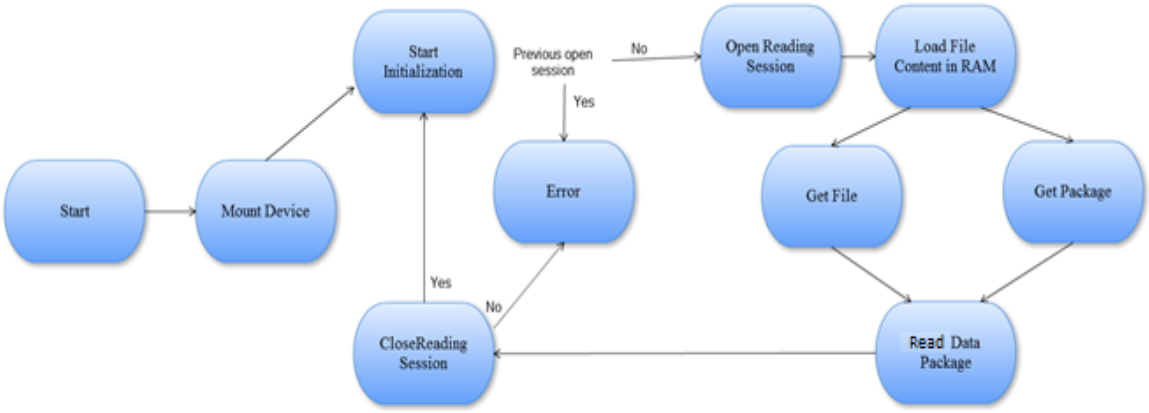


Figura 12. Diagrama de Estado Operación de Lectura Modulo FAT

Una vez que la sesión de lectura es cerrada, no es necesario respaldar el contenido de memoria RAM al archivo como en la sesión de escritura, simplemente el contenido de memoria es liberado y la bandera de sesión abierta es modificada para indicar que es posible abrir una nueva sesión. De tal manera que el contenido del archivo es dejado como en su estado original. En el siguiente diagrama de secuencia se ejemplificara la manera más detallada en la que una lectura por paquetes o por archivo trabajan.

La Figura 12 muestra un diagrama de estados para la operación de lectura de archivos binarios en el módulo FAT.

3.5. CIFRADO DE PAQUETES

La configuración del método de encriptación quedo definido de la siguiente manera:

- Algoritmo: “Rinjdael AES” en modo de bloque de tamaño de 128-bits y configurado en modo CBC (“Cipher Block Chaining”)
- Por el modo en que el algoritmo es configurado, también se determinaron las siguientes variables:
    - Valor de la llave: "0123456789abcdef" Este valor es conocido en ambos lados del proceso, tanto en el cifrado como en el descifrado. El algoritmo maneja que sea una llave de 16 caracteres.
- Valor de Primer bloque (IV): "ABCDEFGHIJKLMNOP" Por el modo que se seleccionó en el método. Es necesario definir este bloque para que sea agregado como valor aleatorio, para que siempre sea diferente. Por el momento se maneja como el valor mencionado previamente y también es de 16 caracteres porque debe de ser del tamaño del tamaño de bloque usado por el algoritmo.

El desarrollo se dividió en 2 partes, la parte de cifrado y la parte de descifrado. Como cada parte se iba a llevar en diferentes partes del sistema, por ser diferentes entornos es la razón por la que se hace la división.

3.5.1. Etapa de Cifrado

Para la encriptación se trabajaron en 2 plataformas, en primer lugar, se empezó a hacer investigación sobre que se podría utilizar para trabajar con este algoritmo de encriptación, para esta investigación se utilizó el Sistema operativo Slax 7.0.8 este es un sistema Linux el cual es cargado en una memoria USB, se eligió este sistema por el hecho de estar un poco limitado a diferencia de tener instalado un sistema Linux nativo en una máquina para obtener un sistema que fuera parecido en cuanto a tener limitaciones del sistema.

El entorno en este sistema operativo quedo de la siguiente manera:

- OS: Slax Linux 7.0.8 con Kernel 3.8.2
- GCC es el compilador que viene incluido en todas las distribuciones de Linux y es el compilador utilizado para compilar el código en este sistema operativo. La versión instalada fue la 4.7.2

Al empezar a investigar sobre posibles soluciones se encontró ya existe una librería de código abierto la cual permite el uso de un amplio rango de funciones de encriptación, sin tener que hacer cambios drásticos en el código fuente, además permite el manejo tanto de archivos o buffers de datos.

El nombre de esta librería es “mcrypt”, la cual viene acompañada de “libmcrypt” que la que contiene las funciones de encriptación. La librería “mcrypt” es la que provee los mecanismos para acceder a esas funciones. Para los experimentos desarrollados en este proyecto se instaló la versión 2.5.8 de “libmcrypt” en el sistema.

Cabe mencionar que, para esta distribución, para agregar los paquetes que hacían falta, se utilizó el administrador de paquetes de la distribución, que en este caso fue “slackpkg” el cual es bastante sencillo de utilizar porque buscaba los paquetes en repositorios de la distribución, los descargaba y los instalaba sin mayor problema de manera automática.

La Figura 13 muestra un diagrama de actividad para el cifrado de archivos en el sistema. Se inicia con la configuración de parámetros y el almacenamiento de información en un buffer y se finaliza con un archivo cifrado. El diagrama contempla la funcionalidad de los elementos “Encrypt”, “AES” y “mcrypt”.

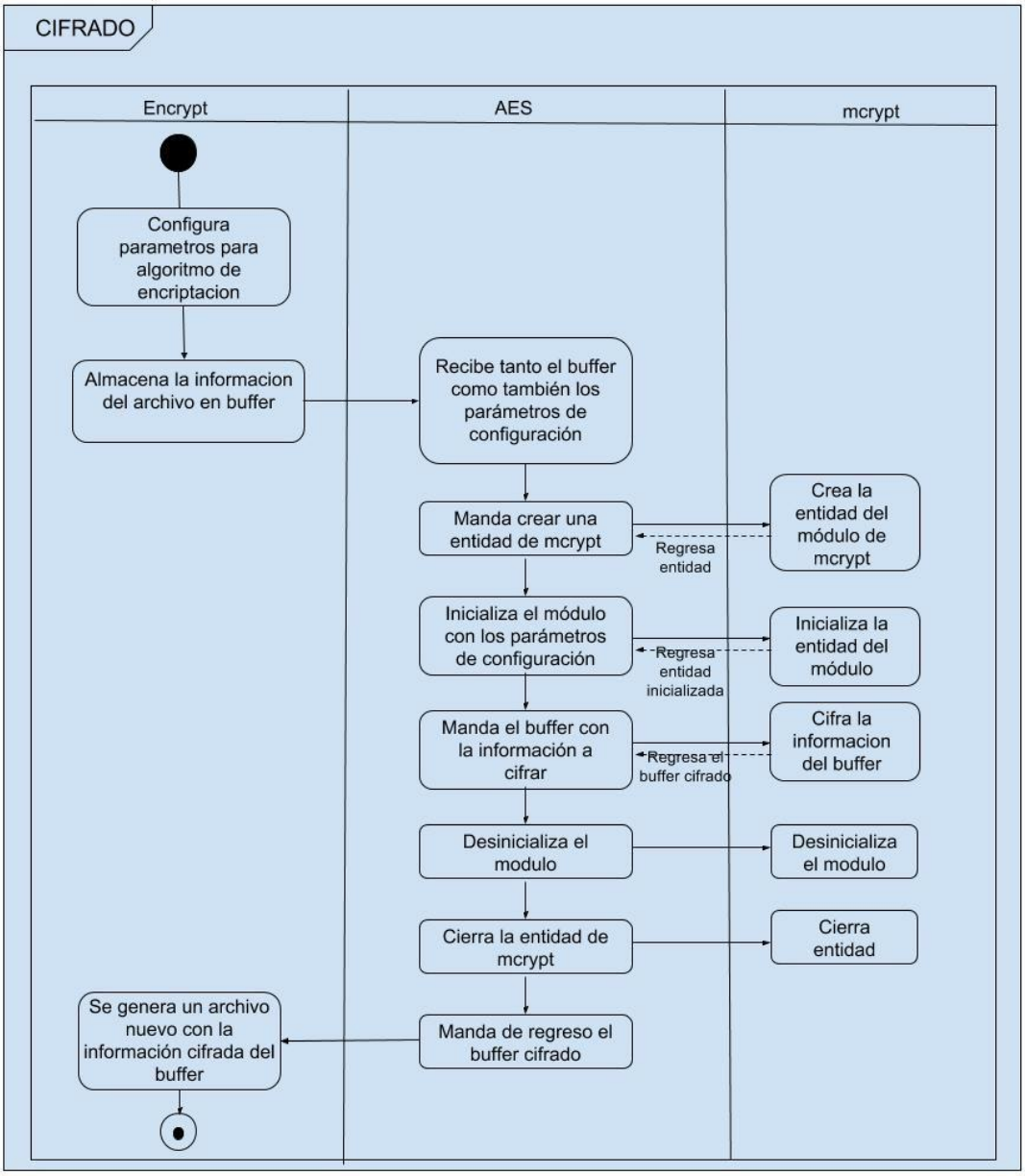


Figura 13. Diagrama de Actividad Cifrado de Paquetes

3.5.2. Etapa de Descifrado

La Figura 14 muestra el diagrama de actividad de la etapa de descifrado. En esta etapa, se consideran los elementos “Encrypt” y “AES-tiny”. El proceso inicia configurando parámetros del algoritmo de descifrado. En un paso anterior a la generación del archivo descifrado final, se llama los servicios de “AES-tiny”. Esta librería es una versión ligera que implementa el algoritmo AES para la plataforma de bajos recursos del fabricante ATMEL, de ahí el distintivo “tiny” (pequeño en español).

Como se ha mencionado en el diseño de la aplicación, el proceso de des encriptación se debe llevar a cabo solo en la ECU para la cual está destinado. De esta manera la cyber-seguridad se lleva hasta el extremo, al permitir solo que el dispositivo electrónico que debe recibir el paquete de actualización es el único, incluso dentro de la red del automóvil, que puede “conocer” su contenido. Este es un requisito muy importante para la seguridad del conductor y un tema vital para la industria automotriz.

Para este proyecto, una tarjeta de experimentación del fabricante ATMEL es la unidad objetivo para fines de prueba de concepto. La librería “AES-tiny” es desarrollada para este tipo de procesadores del fabricante ATMEL.

Para lograr este fin, es necesario realizar un proceso de compilación cruzada de los archivos fuente de la librería “AES-tiny” para que su apropiada ejecución dentro del procesador ATMEL. Debido a restricciones de tiempo, esta etapa de compilación cruzada se deja como trabajo futuro.

Otro elemento significativo fue en la configuración de parámetros. En este aspecto, se siguen usando los mismos valores, pero en el caso de la llave y del bloque IV, en lugar de mandar un arreglo de caracteres, se manda un arreglo con los mismos signos, pero con sus valores en hexadecimal.

Para las pruebas desarrolladas en este trabajo, se validó el proceso de des encriptación de paquetes con la librería “AES-tiny” compilada de manera cruzada para una distribución de Linux en computadora de escritorio.

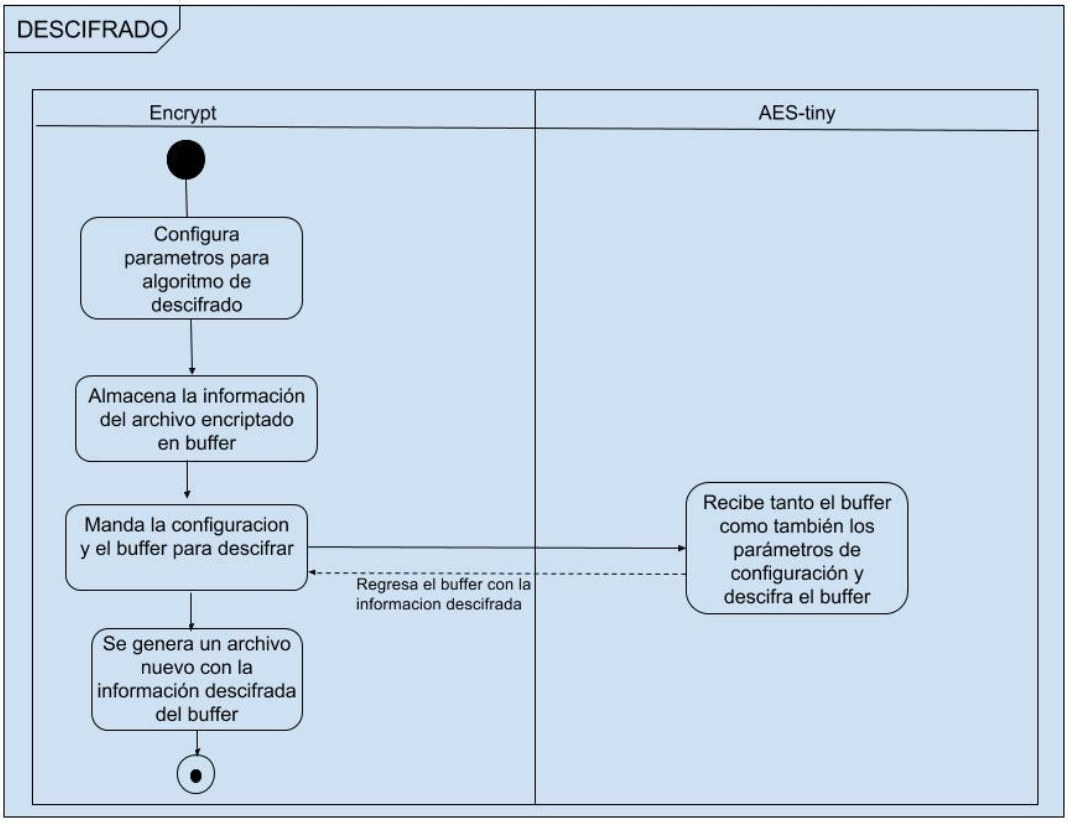


Figura 14. Diagrama de Actividad Descifrado de Paquetes

4. IMPLEMENTACIONES

Resumen: *En este capítulo se muestran las implementaciones de los principales bloques que forman la solución diseñada en el capítulo anterior.*

4.1. INTRODUCCIÓN

El proyecto general consiste en probar el concepto de un método para la actualización del código embebido (software embebido o firmware) de una ECU dentro de un automóvil mediante información recibida por un medio de transmisión, alámbrico o inalámbrico. En particular, se supone que la señal de Radio HD lleva información del software que debe ser actualizado en una ECU dentro del automóvil. Por cuestiones de seguridad informática (cyber-seguridad) la información debe ir encriptada en todo el proceso. De esta manera, solo el dispositivo ECU para el cual está dirigido cierto software de actualización deberá ser capaz de des encriptar la información, para luego grabarla en su memoria interna.

Para probar el concepto, se diseñó un escenario con una transmisión de una actualización usando comunicación UART y una "RASPBERRY B2" como transmisor. La Figura 15 muestra la tarjeta utilizada para este fin.

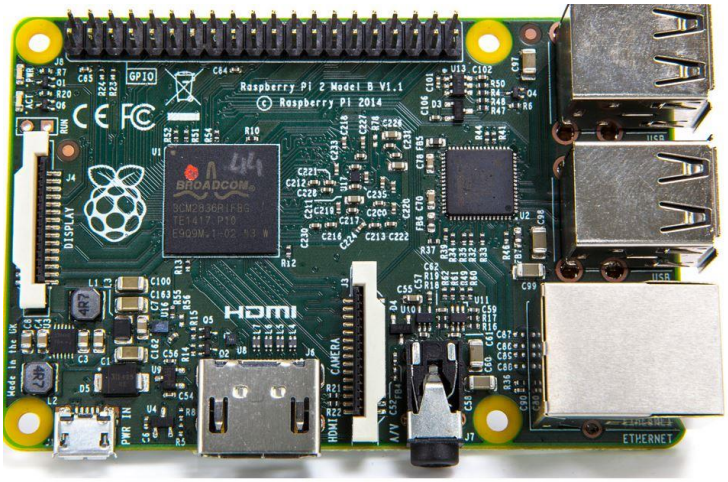


Figura 15. Tarjeta RaspBerry B2 para emular la transmisión de paquetes de actualización

Para el caso del receptor fue usado un microcontrolador CORTEX-M4 integrado en una tarjeta de desarrollo STM32F4 Discovery. La Figura 16 muestra la tarjeta de desarrollo para el procesador CORTEX que simula la recepción de paquetes de actualización.

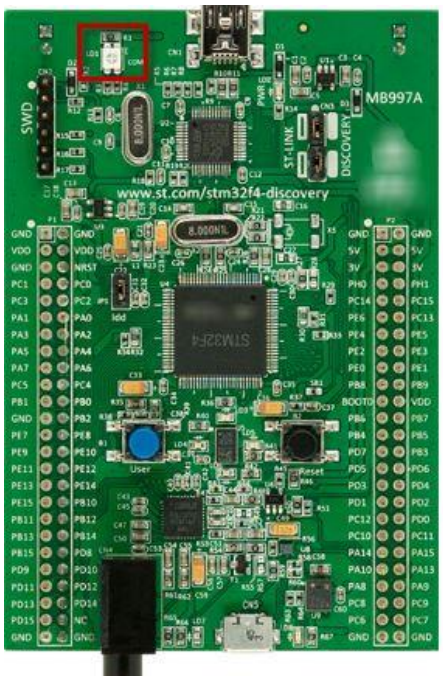


Figura 16. Tarjeta de Desarrollo para CORTEX-M4 para emular la recepción de paquetes de actualización

Finalmente, para fines demostrativos se interactuó con los paneles de experimentación de Laboratorio de Sistemas Embebidos Avanzados del ITESO para aplicar la actualización de software a una ECU de control de luces y modificar su comportamiento. La Figura 17 muestra el panel de experimentación de luces de un automóvil Audi.

Figura 17. Panel de Experimentación Automotriz [4]

4.1. UNIDAD RECEPTORA OTA

La unidad receptora de las actualizaciones transmitidas fue implementada en una tarjeta de desarrollo STM32F407VG desarrollada por STMicroelectronics. Esta tarjeta permite a los usuarios un fácil desarrollo de aplicaciones. Las principales características técnicas por las que se eligió esta tarjeta son las siguientes [5]:

- STM32F407VGT6 microcontroller featuring 32-bit ARM® CORTEX®-M4 with FPU core.1-Mbyte Flash memory, 192-Kbyte RAM in an LQFP100 pachage
- On-board ST-LINK/V2
- USB ST_LINK with re-enumeration capability.
- Board power supply: through USB bus or from an external 5V supply voltage
- External application power supply: 3V and 5V.
- Two push-buttons (user and reset)
- USB OTG FS with micro-AB connector
- CS43L22 audio DAC with integrated class D speaker driver

La distribución de pines de la tarjeta STM32F407VG usada para el proyecto está definido en la Tabla 1.

Tabla 4. Distribución de pines utilizada en el proyecto.

BOARD PINOUT			
STM32F4DISCOVERY	Pin Name	Pin Type	Alternate Function(s)
2	PE3	I/O	
6	VBAT	Power	
10	Vss	Power	
11	VDD	Power	
12	PH0-OSC IN	I/O	RCC OSC In
13	PH1-OSC OUT	I/O	RCC OSC Out
14	NRST	Reset	
15	PC0	I/O	
19	VDD	Power	
20	VSSA	Power	
21	VREF+	Power	
22	VDDA	Power	
23	PA0-WKUP	I/O	
25	PA2	I/O	USART2-TX
26	PA3	I/O	USART2-RX
27	VSS	Power	
28	VDD	Power	
37	PB2	I/O	
47	PB10	I/O	USART3-TX
48	PB11	I/O	USART3-TX
49	Vcap 1	Power	
50	VDD	Power	
59	PD12	I/O	
60	PD13	I/O	
61	PD14	I/O	
62	PD15	I/O	
68	PA9	I/O	USB OTG Fs Vbus
70	PA11	I/O	USB OTG Fs DM
71	PA12	I/O	USB OTG Fs DP
72	PA13	I/O	SYS JTMS SWDIO
73	Vcap 2	Power	

74	VSS	Power	
75	VDD	Power	
76	PA14	I/O	SYS JTCK SWCLK
81	PD0	I/O	CAN1 Rx
82	PD1	I/O	CAN1 Tx
86	PD5	I/O	
89	PB3	I/O	SYS JTDO SWO
94	BOOT 0	Boot	
98	PE1	I/O	
99	VSS	Power	
100	VDD	Power	

La distribución de pines del controlador STM32F405xx que corresponde a aquellos de la tarjeta de desarrollo STM32F405RG se muestra en la Figura 18.

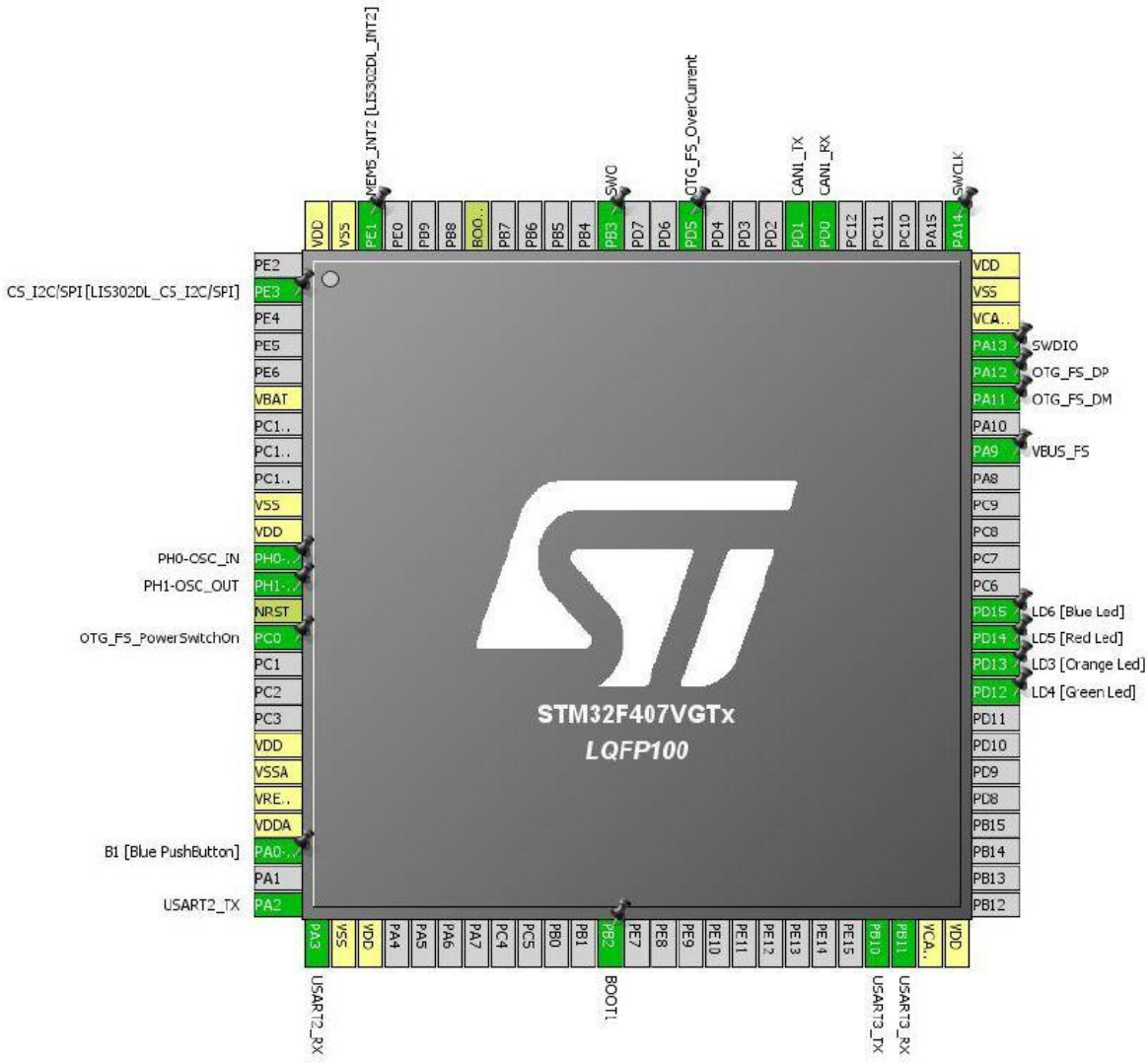


Figura 18. Distribución de Pines del Microcontrolador Utilizados en el Proyecto

Las características requeridas por el proyecto para poder llevar a cabo el desarrollo del prototipo y por cual la tarjeta de desarrollo fue seleccionada fueron las siguientes:

- 1 CAN Bus
- 2 UART Buses
- USB OTG FS

EL fabricante de semiconductores STMicroelectronics tiene un BSP configurable para la gama de microcontroladores SM32F4. Dicho BSP contiene driver que pueden ser fácilmente

configurados. Para la aplicación aquí desarrollada los drivers usados se configuraron de la siguiente manera:

- **CAN-1 Driver:** Este es usado para poder transmitir la actualización a otra ECU que está conectada al mismo bus. Actualmente solo se envía una secuencia de encendido y apagado de un dispositivo para así demostrar la capacidad de comunicación. En la segunda fase del proyecto se implementara la parte de actualización.

$$\text{BaudRate} = \frac{\text{Prescaler}}{\text{BitTime}}(1 + \text{TimSeg1} + \text{TimSeg2})$$

Tabla 5. Configuración del Driver de CAN-1

CAN-1	
Bit Timings Parameters	
Prescaler(For Time Quantum)	43
Time Quantum	1194.444ns
Time Quanta in Bit Segment1	5 Times
Time Quanta in Bit Segment2	2 Times
Time for One Bit	9555ns
ReSynchronization Jump Width	2 Times
Basic Parameters	
Time Triggered Communication Mode	Disable
Automatic Buss-Off Management	Enable
Automatic Wake-up Mode	Disable
No-Automatic Retransmission	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
Advanced Parameters	
Operating Mode	Normal

- **USART2 Driver:** Este driver es usado para mantener la comunicación con el transmisor de la actualización que será recibida.

Tabla 6. Configuración del Driver USART2

USART2	
Basic Parameters	
Baud Rate	115200

Word Length	8 bits(Including Parity)
Parity	None
Stop Bits	1
Mode	Asynchronous
Advanced Parameters	
Data Direction	Rx/Tx
Over Sampling	16 Samples

- **USART3 Driver:** Es usado para poder transmitir "logs" e identificar de manera sencilla los problemas que se puedan presentar.

Tabla 7. Configuración del Driver USART3.

USART3	
Basic Parameters	
Baud Rate	115200
Word Length	8 bits(Including Parity)
Parity	None
Stop Bits	1
Mode	Asynchronous
Advanced Parameters	
Data Direction	Rx/Tx
Over Sampling	16 Samples

- **FAT32 Library and USB Host driver:** El driver usado para USB actúa como un host debido a que es necesario almacenar diferentes actualizaciones

Tabla 8. Configuración de Librería FAT32 y USB Host driver (Parte 1).

USB OTG FS	
Basic Parameters	
Mode	Host Only
Speed	Host Full Speed 12MBit/s
Internal IP DMA	Disabled

Tabla 9. Configuración de Librería FAT32 y USB Host driver (Parte 2).

FATFS (Version R0.11)	
Function Parameters (Set defines)	
FS_TYNY	Disabled
FS_READONLY	Disabled
FS_MINIMIZE	Disabled
USE_STRFUNC	Enabled with LF -> CRLF conversion
USE_FIND	Disabled
USE_MKFS	Enabled
USE_FORWARD	Disabled
USE_LABEL	Disabled
USE_FASTSEEK	Enabled
USE_BUFF_WO_ALIGMENT	Enabled
Locale and Namespace Parameters(Set defines)	
CODE_PAGE	Latin 1(Windows)
USE_LFN	Disabled
MAX_LFN	255
LFN_UNICODE	ANSI/OEM
STRF_ENCODE	UTF-8
FS_RPATH	Disabled
Physical Drive Parameters	
VOLUMES	1
MAX_SS	512
MIN_SS	512
MULTI_PARTITION	Disabled
USE_TRIM	Disabled
FS_NOFSINFO	0
System Parameters	
FS_NORTC	Dynamic timestamp
NORTC_YEAR	2015
NORTC_MON	6
NORTC_MDAY	4
WORD_ACCESS	Byte access
FS_REENTRANT	Enabled
FS_TIMEOUT	1000
SYNC_t	osSemaphoreID
FS_LOCK	2

Tabla 10. Configuración de Librería FAT32 y USB Host driver (Parte 3).

USB HOST (Mass Storage Host Calss)	
Host Configuration	
USBH_MAX_NUM_ENPOINTS	2
USBH_MAX_NUM_INTERFACES	2
USBH_MAX_NUMSUPPORTED_CLASS	1
USBH_MAX_NUM_CONFIGURATION	1
USBH_KEEP_CFG_DESCRIPTOR	Enabled
USBH_MAX_SIZE_CONFIGURATION	256
USBH_MAX_DATA_BUFFER	512
USBH_DEBUG_LEVEL	0: No debug message
CMSIS-RTOS configuration	
USBH_USE_OS	Enabled
USBH_PROCESS_PRIO	Priority: normal(defualt)
USBH_PROCESS_STACK_SIZE	128

4.1.1. Configuración del Sistema Operativo de Tiempo Real

Además del BSP usado, un paquete para poder usar Free-RTOS fue incluido en el prototipo desarrollado. Dicho paquete incluye dos elementos Free-RTOS y CMSIS-OS. La descripción de estos elementos y su integración en el proyecto es como sigue:

- **FreeRTOS:** Soporte de tareas en tiempo real. Simplificación del diseño abstracto de las actividades en la unidad receptora OTA:
- **CMSIS-OS:** Esta es una capa que esta sobre "FreeRTOS" la cual es usada para adaptaciones necesarias para el correcto funcionamiento del sistema operativo en el microcontrolador.

El Sistema Operativo en Tiempo Real es una capa fundamental en el desarrollo de este proyecto. Esta capa simplifica la construcción de aplicaciones complejas con restricciones de tiempo real e impone reglas para su diseño. La Tabla 11 muestra la configuración usada para Free-RTOS en este proyecto.

Tabla 11. Configuración de Free-RTOS.

FREERTOS (CMSIS-RTOS version 1.02) (FreeRTOS version 8.2.3)	
Kernel Settings	
USE_PREEMPTION	Enabled
CPU_CLOCK_HZ	SustemCoreClock
TICK_RATE_HZ	1000
MAX_PRIORITIES	7
MINIMAL_STACK_SIZE	128
MAX_TASK_NAME_LEN	16
USE_16_BIT_TICKES	Disabled
IDLE_SHOULD_YIELD	Enabled
USE_MUTEXES	Enabled
USE_RECURSIVE_MUTEXES	Disabled
USE_COUNTING_SEMAPHORES	Disabled
QUEUE_REGISTRY_SIZE	8
USE_APPLICATION_TASK_TAG	Disabled
TOTAL_HEAP_SIZE	15360
Memory Management scheme	heap_4
USE_ALTERNATIVE_API	Disabled
ENABLE_BACKWARD_COMPATIBILITY	Enabled
USE_PORT_OPTIMISED_TASK_SELECTION	Disabled
USE_TICKLESS_IDLE	Disabled
USE_TASK_NOTIFICATIONS	Enabled
Hook Function Related Definitions	
USE_IDLE_HOOK	Disabled
USE_TICK_HOOK	Disabled
USE_MALLOC_FAILED_HOOK	Disabled
CHECK_FOR_STACK_OVERFLOW	Disabled
Run Time and Task Stats Gathering Related Definitions	
USE_TRACE_FACILITY	Enabled
GENERATE_RUN_TIME_STATS	Disabled
Co-routine Related Definitions	
USE_CO_ROUTINES	Disabled
MAX_CO_ROUTINE_PRIORITIES	2
Co-Routine Related Definitions	
USE_TIMERS	Disabled
TIMER_TASK_PRIORITY	2

TIMER_QUEUE_LENGTH	10
TIMER_TASK_STACK_DEPTH	256
Interrupt Nesting Behaviour Configuration	
LIBRARY_LOWEST_INTERRUPTION_PRIORITY	15
LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY	5
Include Definitions	
vTaskPrioritySet	Enabled
uxTaskPriorityGet	Enabled
vTaskDelete	Enabled
vTaskCleanUpResources	Disabled
vTaskSuspend	Enabled
vTaskDelayUnit	Disabled
vTaskDelay	Enabled
xTaskResumeFromISR	Enabled
xQueueGetMutexHolder	Enabled
xSemaphoreGetMutexHolder	Disabled
pcTaskGetTaskName	Disabled
uxTaskGetStackHighWaterMark	Disabled
xTaskGetCurrentTaskHandle	Disabled
eTaskGetState	Disabled
xEventGroupSetBitFromISR	Disabled
xTimerPendFunctionCall	Disabled

4.1.2. Esquema de conexión entre dispositivos

La interconexión entre los elementos que sirvieron como plataforma de pruebas se describe en la Figura 19. Los elementos y su funcionalidad son los siguientes:

- **Raspberry B2(A):** Encargada de encriptar la actualización que será transmitida así como también se encarga de la transmisión mediante UART.
- **FTDI Converter(B):**Este convertidor se encarga de mantener la comunicación entre Raspberry/Discover
- **FTDI Converter(C):** Encargado de mantener la comunicación entre la PC/Discovery, el propósito es poder visualizar archivos "logs" enviados por Discovery.
- **Discovery Board (D):** Esta tarjeta es encargada de la recepción y manejo de paquetes correspondientes a la nueva actualización recibida. Discovery tiene una conexión a un

dispositivo USB MSC el cual se usa para almacenar la información recibida, por último se puede observar con

- **MCP2551 (E):** "Tranceiver" usado para realizar la conexión a la red de CAN.
- **USB MSC Device (G):** Para poder llevar a cabo la conexión con un dispositivo USB MSC se usó un convertidor "on the go" (F) y la razón de usar esta conexión fue para poder almacenar la información reciba.
- **CAN Network (H):** La conexión a la red de CAN es un elemento que hasta ahora se tiene en fase solo de prueba y se planea realizar una futura implementación donde una secuencia de reprogramación de la memoria flash completa está integrada.

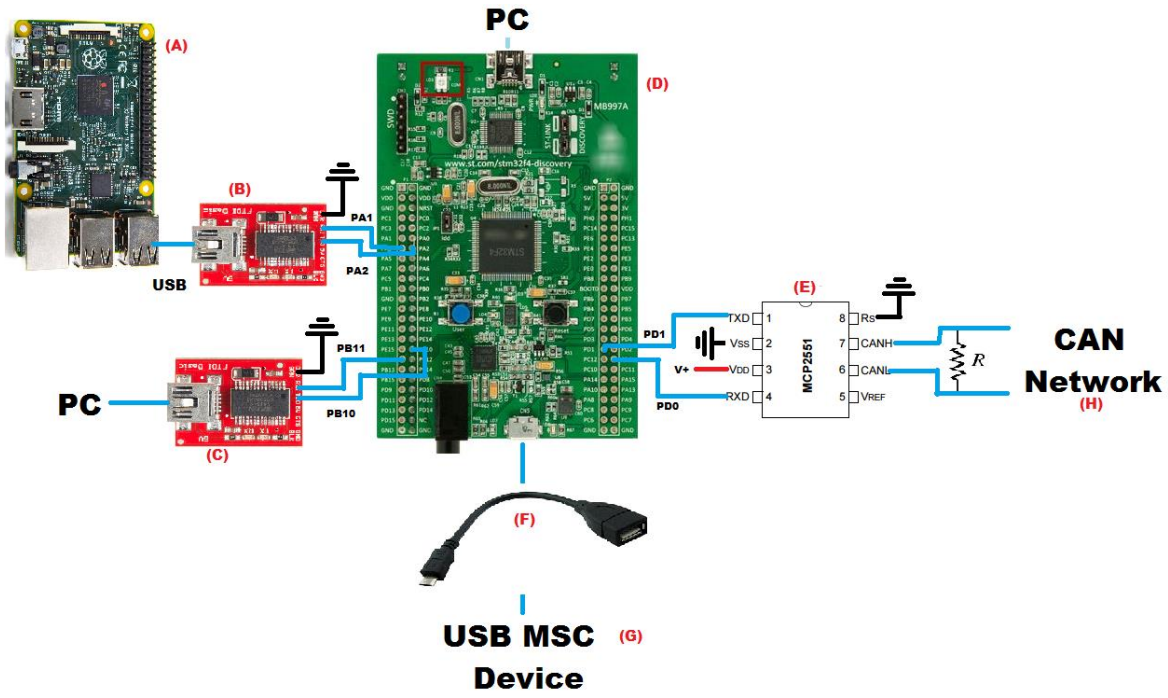


Figura 19. Esquema de Interconexión de los Elementos en la Plataforma de Pruebas

4.2. IMPLEMENTACION DE LA ACTUALIZACION DE SOFTWARE

4.2.1. Descripción del módulo “HDMain”

La estructura de archivos fue diseñada para poder encapsular el modulo encargado del protocolo, de este modo una aplicación externa puede manejar los servicios del módulo sin la necesidad de incluir múltiples cabeceras para el uso de este.

El archivo “HDApi.h” es el medio por el cual un módulo externo puede acceder a las funciones del protocolo. El desarrollo las interfaces para manejar el módulo de “HDRadio” está planeado para la siguiente fase del proyecto.

Los elementos que se pueden encontrar en dicha estructura se encuentran el manejo del dispositivo USB, así como del sistema de archivos “file system”, el manejador de UART2 y sus servicios. Finalmente, el modulo encargado del protocolo consta en dos elemento principales. Uno se encarga de analizar los datos recibidos para formar mensajes y el otro se encarga de atender dichos mensajes. La Figura 20 muestra un diagrama de la estructura de archivos del protocolo de comunicaciones.

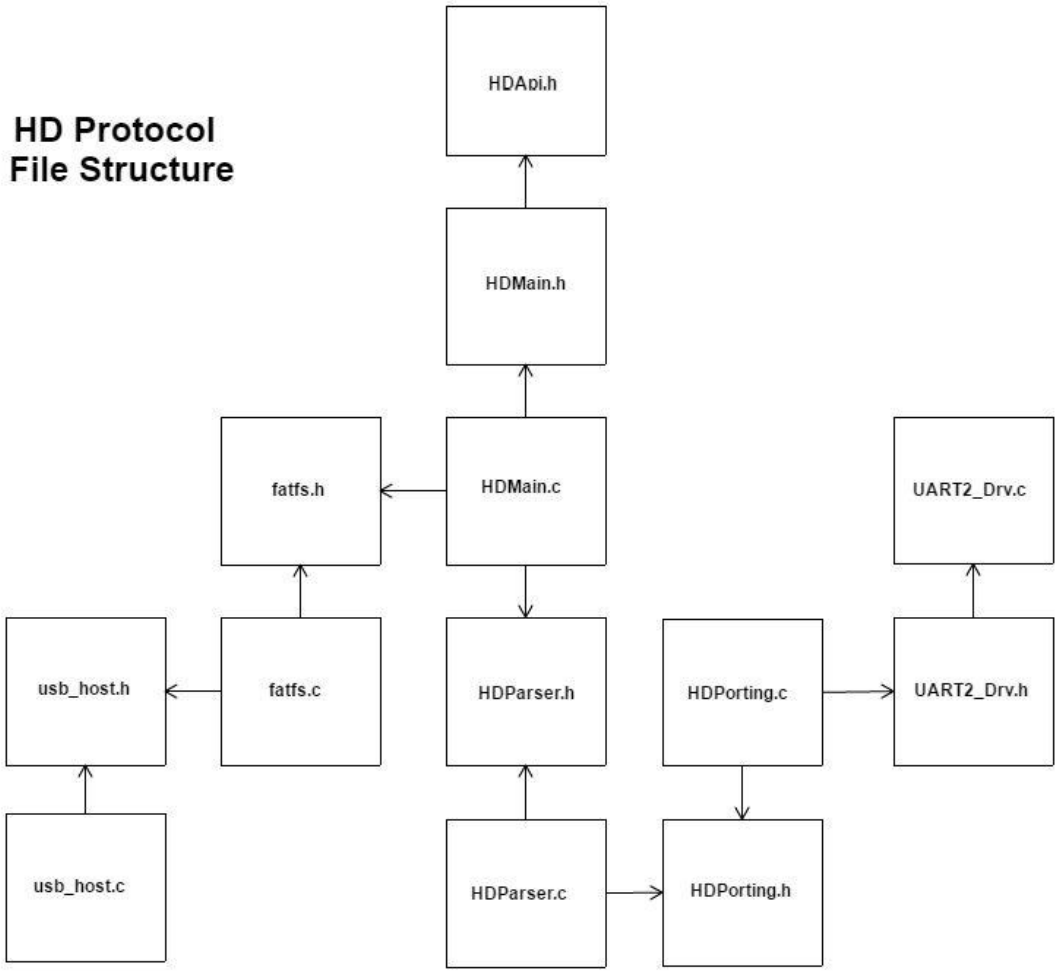


Figura 20. Estructura de Archivos del Protocolo de Comunicación

4.2.2. Descripción del módulo “HDMain”

El módulo “HDMain” es responsable crear las bases de datos para alojar las actualizaciones recibidas y hacer la búsqueda de actualizaciones completadas.

En el diseño propuesto en este trabajo, “HDMain” no tiene contacto con el módulo de CAN (el cual será el medio para enviar las actualizaciones recibidas) por lo que la secuencia de envió se

planea para la fase 2 del proyecto. En la versión actual, "HDMain" es capaz de consumir los mensajes recibidos y atenderlos según se requiera.

La máquina de estados para "HDMain" consta de 6 estados y 12 transiciones entre ellos. En la Figura 19 se puede observar la relación entre los diferentes estados. En futuras versiones, la máquina de estados mencionada necesita ser modificada para agregar los estados requeridos para realizar el procedimiento de envió de las actualizaciones cuando estas estén completas. En la versión actual, está diseñada para atender los mensajes recibidos y alojarlos en memoria no volátil.

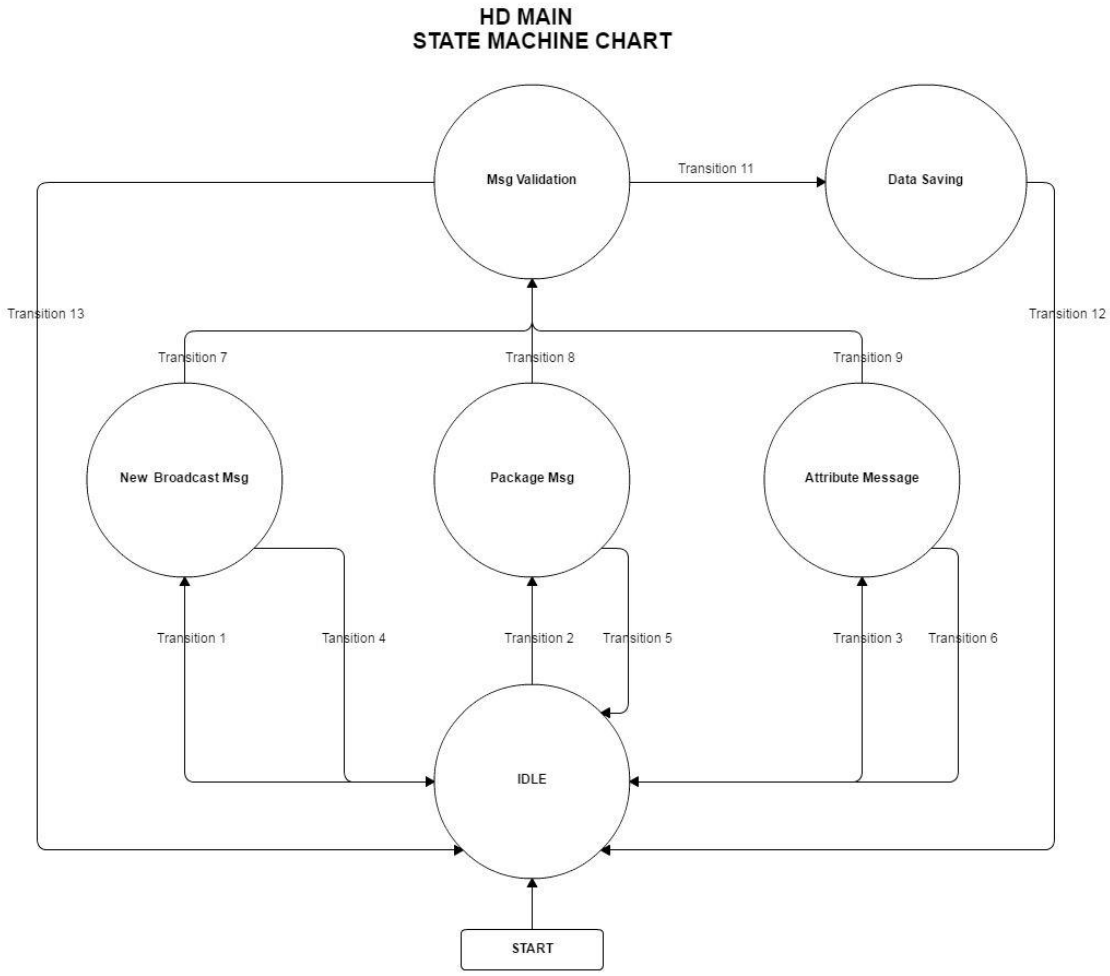


Figura 21. Máquina de Estados del Módulo HDMain

La descripción de los estados de la maquina en la Figura 19 es como sigue:

- **IDLE State:** En este estado se revisa si hay mensajes para ser atendidos, en caso de haber un mensaje en la cola de "HDParser" se revisa el ID del mensaje y dependiente de este se realiza la transición correspondiente al mensaje recibido.

- **New Broadcast Msg State:** Se interpretan los datos recibidos para llenar un mensaje local llamado HDM__stCurrentMsg el cual es usado para mantener la información antes de entrar al estado "Data Saving", los datos se almacenan en un formato conveniente para los "New Broadcast messges".

- **Package Msg State:** Se interpretan los datos recibidos y se almacenan en un formato conveniente para los "Package messages". También ocupa la estructura HDM__stCurrentMsg para mantener la información antes de ser almacenada.

- **Attribute Message State:** Se interpretan los datos recibidos y se almacenan en un formato conveniente para los "Attribute messages". También ocupa la estructura HDM__stCurrentMsg para mantener la información antes de ser almacenada.

- **Msg Validation State:** En este estado se revisa la integridad de los datos recibidos en el mensaje, la validación de los datos varía dependiendo el mensaje recibido. Actualmente el algoritmo usado para el CRC es muy sencillo solo consta de suma XOR entre los datos recibidos.

- **Data Saving State:** Una vez revisada la integración de los datos recibidos se almacena la información en memoria no volátil, este proceso es diferente para cada mensaje por atender. Una vez almacenada la información se activara el estado IDLE.
    - **Comportamiento para New Broadcast Msg:** Se ocupa el "SWKey" para revisar si existe una base de datos con el mismo "SWKey", en caso de encontrarse una el mensaje será omitido y de lo contrario se creara una nueva base de datos para alojar la actualización que será transmitida.

- **Comportamiento para Attribute Msg:** Se revisa en las base de datos existentes si hay una con el mismo "SWKey" que contiene el mensaje recibido, en caso de encontrarse dicha base de datos se agregara el nuevo atributo a la base de datos correspondiente, los atributos serán usados para identificar si la actualización que será recibida es correspondiente a alguna ECU disponible en el vehículo.
- **Comportamiento para Data Msg:** Se revisa en las base de datos existentes si hay una con el mismo "SWKey" que contiene el mensaje recibido, en caso de encontrarse dicha base de datos se agregara el nuevo paquete de datos recibido.

4.2.3. Descripción del Módulo "HDParser"

El módulo "HDParse" es responsable de analizar los datos recibidos por algún medio, alámbrico o inalámbrico, para interpretarlos como un mensaje y agregarlos a una lista que después deben ser consumidos por el módulo "HDMain". En las pruebas de concepto de este proyecto, el modulo recibe los paquetes por medio de un canal UART.

Este módulo también es responsable de la sincronización necesaria para el protocolo. La máquina de estados cuenta con 5 estados principales y a diferencia de HDMain se considera completada la máquina de estados para los fines necesarios, la fase dos se encargara de optimizar la implementación actual. La Figura 20 muestra la máquina de estados de este módulo.

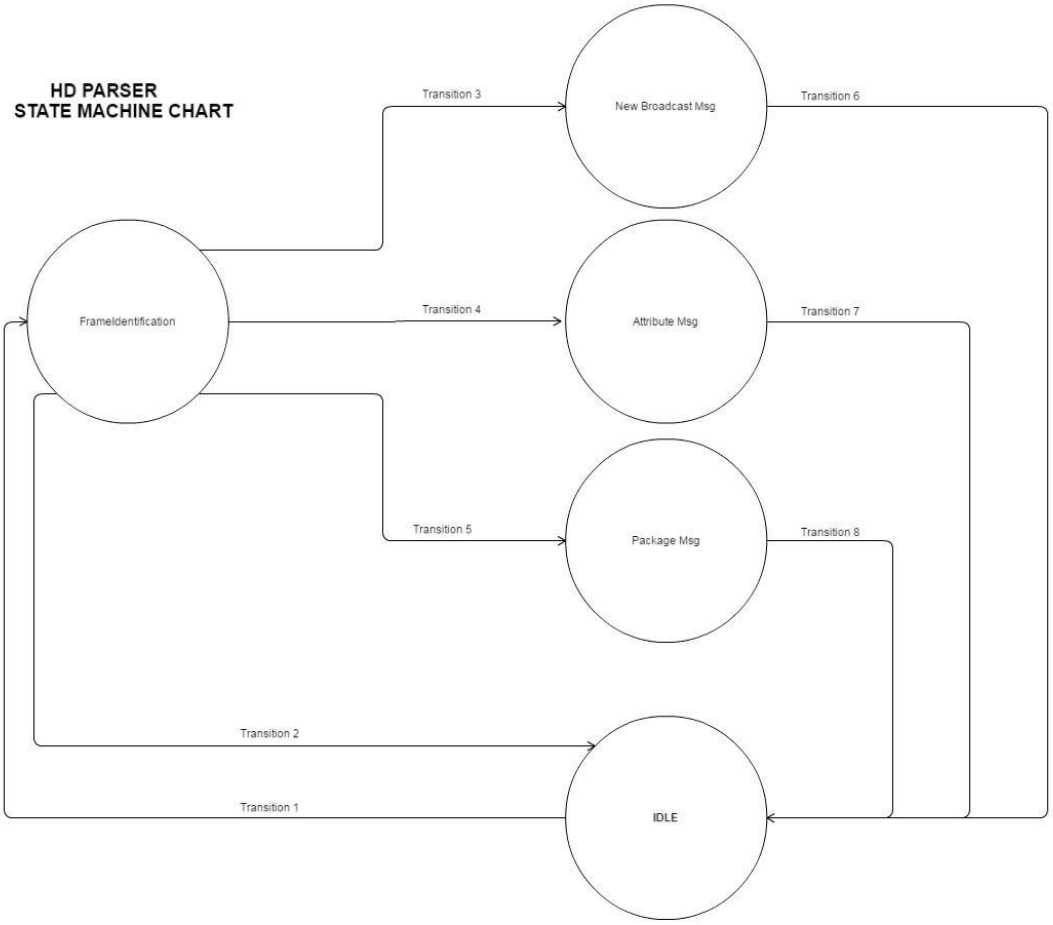


Figura 22. Máquina de Estados del Módulo HDParser

La descripción de los estados de la maquina en la Figura 20 es como sigue:

- **IDLE State:** Se revisa la posible recepción de un nuevo mensaje atreves de los 4 bytes usados para sincronización, en caso la comunicación se pierda este estado es capaz de volver a sincronizarse una vez que la comunicación se haya restablecido.

- **Frame Identification State:** Se revisa el byte correspondiente al ID del mensaje por recibir, dependiendo del ID se activa el siguiente estado.

- **New Broadcast Msg:** Una vez recibido el ID del mensaje, se comienza a almacenar el número de datos correspondientes al mensaje.

- **Attribute Msg:** Una vez recibido el ID del mensaje, ya que el mensaje de atributo varia su longitud en este estado se hace una revisión de los byte que hacen referencia al número de datos a ser recibidos y después se almacenan el número de datos correspondientes al mensaje.

- **Package Msg:** Una vez recibido el ID del mensaje, se comienza a almacenar el número de datos correspondientes al mensaje.

4.2.4. Procedimiento “Log and Trace”

El módulo “log and trace” es responsable de manejar los mensajes generados por el desarrollo de la unidad OTA Receiver. El conjunto de mensajes (“logs” en inglés) generados se transmiten mediante la interfaz UART3 antes descrita.

Actualmente el modulo cumple la funcionalidad requerida para el fin de informar los mensajes mínimos para una depuración básica. En la implementación actual existen tres tipos de “logs” los cuales ayudan al desarrollo para clasificar los mensajes y cuando se requiera, reducir el proceso o el uso del bus serial, desactivando el envío de mensajes. Los tipos de “logs” se describen a continuación:

- “Information Log”: mensajes usados para información simple. Este mensaje puede tener la prioridad más baja en otros subsistemas. Estos mensajes pueden ser desactivados por la definición global LNT_INF, la cual es una constante tipo booleano.

- “Debug Log”: mensajes usados para propósitos de depuración del código. Este tipo de mensajes se caracterizan por mostrar valores de variables que necesitan ser monitoreadas constantemente. Estos mensajes pueden ser desactivados por la definición global LNT_DEBUG, la cual es una constante tipo booleano.

- “Error Log”: mensajes usados para información simple. También puede tener la prioridad más baja en otros subsistemas. Estos mensajes pueden ser desactivados por la definición global LNT_ERR, la cual es una constante tipo booleano.

Este bloque de la implementación tiene interfaces diferentes para cada módulo que tiene que ser atendido. Dichas interfaces agregan información que es útil para que los mensajes sean útiles, así como el nombre de la función de donde se envió el mensaje de “log” y la línea donde está escrito el log en el archivo, entre otros datos útiles para la depuración de aplicaciones.

Las interfaces definidas en el módulo actual se implementan para soportar a los siguientes módulos:

- Módulo “Main”
    - [MAIN-INF]
    - [MAIN-DEBUG]
    - [MAIN-ERR]
- Módulo “UART2 driver”
    - [UART2-INF]
    - [UART2-DEBUG]
    - [UART2-ERR]
- Módulo “Power Manager”
    - [PWRM-INF]
    - [PWRM-DEBUG]
    - [PWRM-ERR]
- Módulo “HDParser”
    - [HDP-INF]
    - [HDP-DEBUG]
    - [HDP-ERR]

- Módulo “HDMain”
    - [HDM-INF]
    - [HDM-DEBUG]
    - [HDM-ERR]
- Módulo “FAT – Memory Service”
    - [FATSe-INF]
    - [FATSe-DEBUG]
    - [FATSe-ERR]
- Módulo “USB Driver”
    - [USB-INF]
    - [USB-DEBUG]
    - [USB-ERR]
- Módulo “CAN”
    - [CAN-INF]
    - [CAN-DEBUG]
    - [CAN-ERR]

4.2.5. Servicios del Módulo “Memory” (FAT)

4.2.5.1. Diagrama de Secuencias para Sesión de Escritura

La Figura 21 muestra un diagrama de secuencia donde se puede apreciar con detalle la forma en que interactúan las diferentes interfaces para escribir paquetes en un archivo binario. Durante el encendido, el “Power Manager” es el encargado de montar el dispositivo USB, para tenerlo listo en cuanto sea requerido. De este modo, si se hace una petición a FATFS antes de haber montado el dispositivo este nos arrojara un error en el módulo.

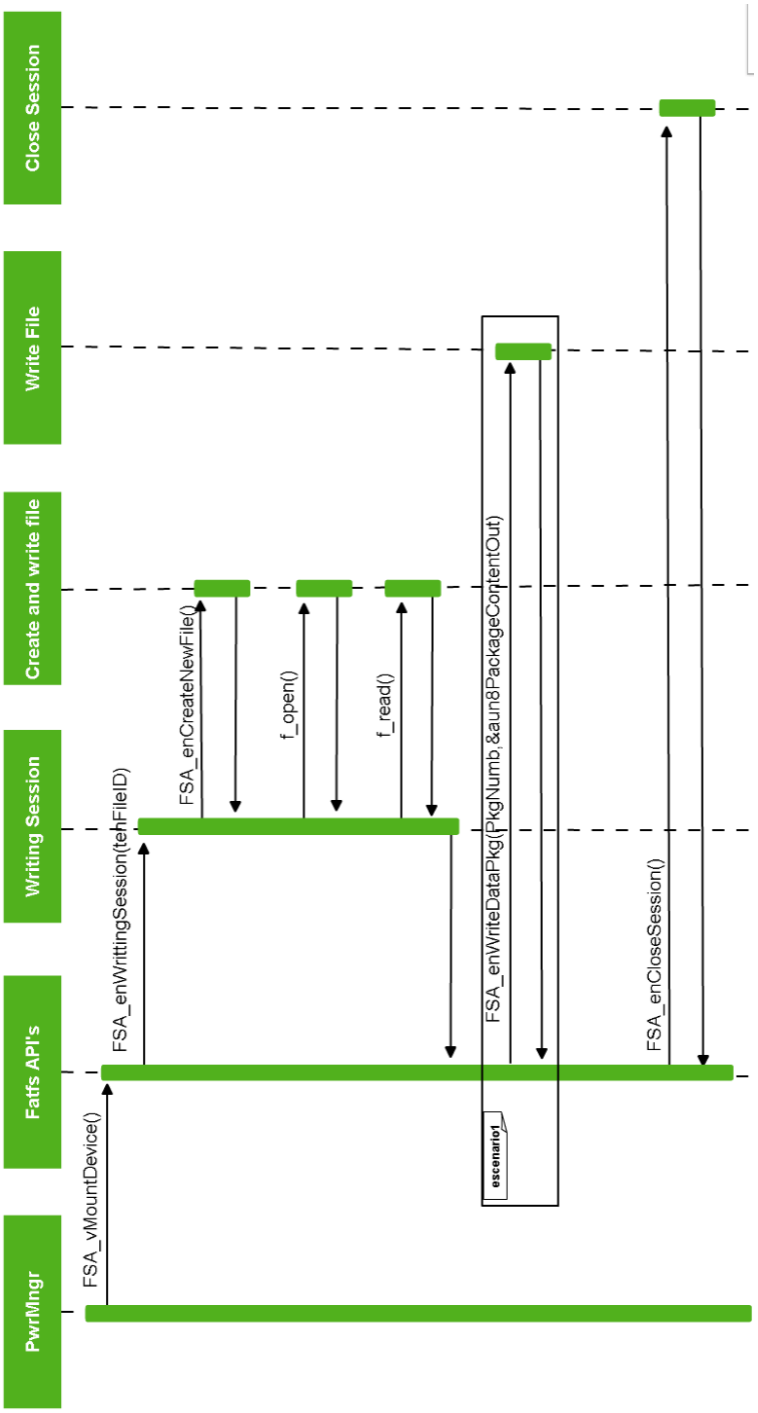


Figura 23. Diagrama de Secuencia de una Sesión de Escritura Modulo Memory

Cuando el dispositivo está listo, el modulo está preparado para atender peticiones, ya sea de lectura o escritura. Mediante la función "FSA_enWritingSession" una sesión de escritura es levantada. En este punto se especifica el nombre del archivo con el que se va a trabajar o a crear, en

su caso. La siguiente parte de la secuencia indica que un nuevo archivo es creado mediante la función "FSA_enCreateNewFile". En el caso de que su creación sea exitosa, se puede proseguir con la síguete parte de la secuencia. Indistintamente si el archivo fue creado o ya existía, su contenido es cargado a memoria RAM para ser manipulado desde ahí.

La escritura de paquetes se hace a través de la función "FSA_enWriteDataPkg". Esta función tiene dos parámetros de entrada. El primero hace referencia al número del paquete que se va a escribir. Este parámetro también define en qué posición del archivo se guardaran. El segundo parámetro es el contenido del paquete. Este último parámetro tiene un tamaño de hasta 5bytes.

Finalmente, cuando la edición del archivo se hizo con éxito se procede a hacer el cierre de sesión mediante la función "FSA_enCloseSession". Esta función guardara la información contenida en un arreglo de trabajo de 128bytes en el archivo binario que se especificó al abrir la sesión. De esta manera, aun cuando más de una modificación se hace en el arreglo de trabajo, solo se realiza una escritura en el archivo físico. Finalmente, la memoria es liberada y el modulo queda listo para la siguiente sesión.

4.2.5.2. Diagrama de Secuencia para Sesión de Lectura

Como se ha mencionado con anterioridad, el funcionamiento de este módulo comienza con el montaje de un dispositivo de almacenamiento. Este proceso es realizado por el "Power Managment". Una vez que el dispositivo está listo para usarse, una nueva sesión puede ser levantada. El diagrama de secuencias de la Figura 22 describe una sesión de escritura y las dos modalidades o enfoques para leer desde un archivo binario.

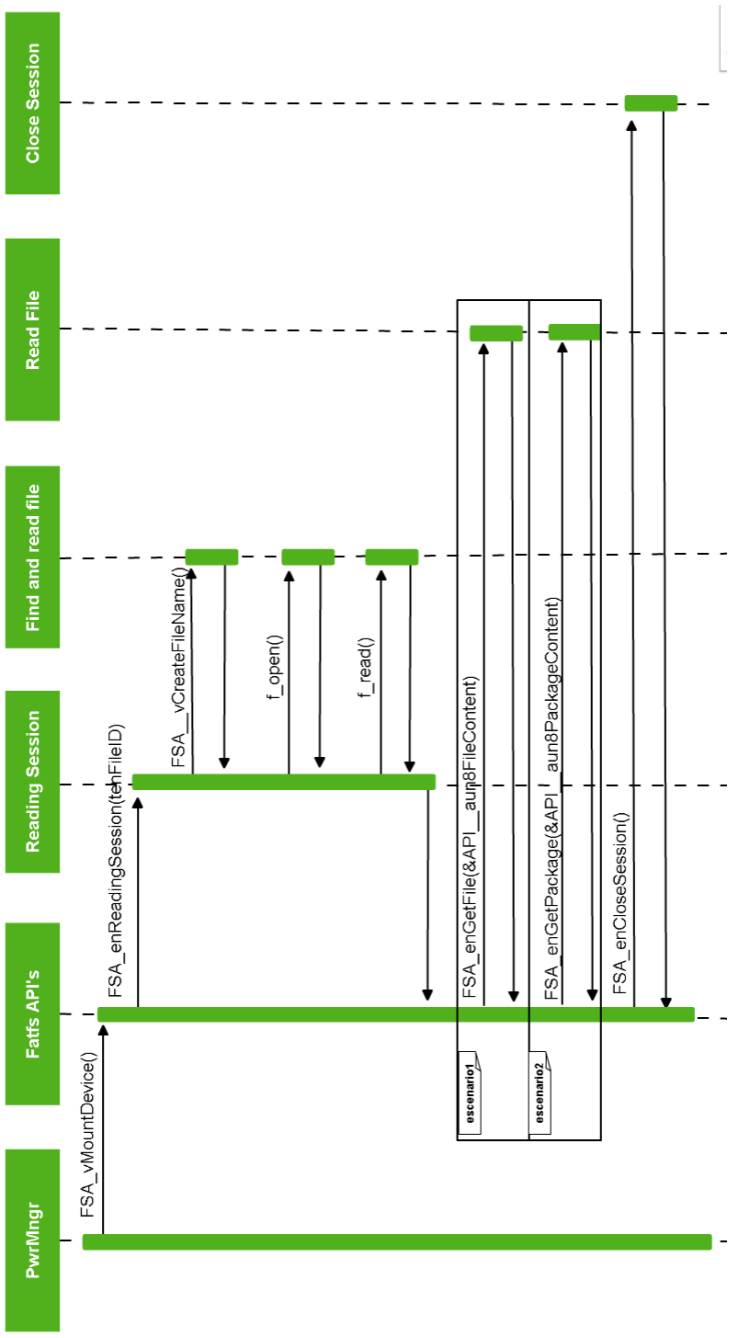


Figura 24. Diagrama de Secuencia de una Sesión de Lectura Módulo Memory (FAT)

Mediante la función “FSA_enReadingSession” se especifica el ID o nombre del archivo con el que se comenzara a trabajar. En este tipo de sesión, a diferencia de la sesión de escritura, el archivo debe haber sido forzosamente creado previamente. En otro caso, no tendría sentido hacer lecturas en un archivo en blanco. Cuando se supone que el archivo solicitado es encontrado, este es abierto, leído y después su contenido es cargado a nuestro arreglo de trabajo interno con un tamaño de 128bytes.

Este dato es importante, ya que si el archivo es más grande de lo especificado, esto ocasionará perdida de información al truncar el contenido del archivo al momento de ser cargado en memoria RAM. En versiones futuras de este trabajo, se puede contemplar un tamaño diferente de archivo.

Es posible hacer una lectura de paquetes o de archivo con la misma sesión de lectura que se abrió previamente. Para hacer una lectura de archivo bastara con llamar a la función "FSA_enGetFile" y proveer de un puntero a un arreglo bidimensional de 5x26 caracteres, lo que resulta en un tamaño total de 128bytes. De esta forma se garantiza que el contenido completo del binario puede ser transferido a otro modulo.

El segundo modo de lectura consiste en hacer un acceso a paquetes específicos mediante la función "FSA_enGetPackage". Para el uso de esta función se tendrá que especificar en sus parámetros de entrada, primero el ID del paquete que se desea obtener seguido de un arreglo de 5 bytes dónde se escribirá el contenido del paquete solicitado. Obtenida la información, y si ya no requiere ninguna operación adiciona con el archivo, se procede a hacer el cierre de la sesión.

El modulo tiene control e información de la sesión activa, así que no es necesario informar si la sesión a cerrar es de lectura o escritura. En el cierre de sesión se cerrara el contexto del archivo abierto y se liberara el arreglo de trabajo que mantenía el contenido del archivo que se leyó.

4.3. IMPLEMENTACION DE CAN Y CONFIGURACION DE CANALYZER

La comunicación entre la tarjeta de desarrollo DISCOVERY al panel de instrumentos Audi se hace utilizando el protocolo CAN. Por lo tanto, la configuración e implementación de la herramienta "CANalyzer" es esencial para este propósito. Dicha herramienta permite abrir una sesión de comunicación entre la tarjeta de desarrollo y el bus de comunicación del panel, de manera que es posible enviar mensajes a los nodos requeridos.

Es importante tomar en cuenta que para una comunicación adecuada entre la tarjeta y el panel Audi es necesaria la presencia del mensaje "Tester-Present". Dicho mensaje cuenta con un ID definido por el fabricante y que debe mandarse de manera periódica. La Tabla 12 indican, dependiendo de la ECU que deseemos controlar, los ID's de mensaje y el periodo con el que se deben enviar los comandos para mantener una comunicación funcional.

Tabla 12. ID's de Mensajes y Periodo.

ID de Mensaje	*DLC*	*Periodo*	*Elemento controlado*	*Acción*	*Bit modificado*
0x2C1	*3*	*50ms*	*Intermitente Izquierda*	*Encender/Apagar*	*b1=0x01(Prende) + b3=0x80(default)*
	3	*50ms*	*Intermitente derecha*	*Encender/Apagar*	*b1=0x02(Prende) + b3=0x80(default)*
	3	*50ms*	*Luces Altas*	*Encender/Apagar*	*b1=0x04(Prende) + b3=0x80(default)*
			Bocina	*Encender/Apagar*	*b1=0x80(Prende) + b3=0x80(default)*
0x621	*3*	*100ms*	*Cofre*	*Abierto/cerrado*	*b1=0x0A(abierto) + b1=0x02cerrado*
0x271	*1*	*100ms*	*Tester Present*	*Presencia*	*b1=0x07*

Como se puede notar en la Tabla 12, le mensaje "Tester Present" debe ser mandado al clúster cada 100ms, de lo contrario el protocolo comenzara a reportar errores y la comunicación se cerrará. Éste es el único mensaje del que no se pude prescindir durante la comunicación.

Como uno de los primeros objetivos específicos que se estableció durante el desarrollo de este proyecto, fue el de ser capaces de controlar dispositivos a través de nuestra tarjeta. Esto confirmaría que el protocolo implementado efectivamente estuviera funcional.

Una vez que esto quedo confirmado y que la comunicación con el protocolo CAN mostraba un funcionamiento conforme a lo esperado, el siguiente punto fue el envío de paquetes utilizando dicho protocolo. Estos paquetes de datos generados en archivos binarios pertenecen a un conjunto de instrucciones que servirán para reprogramar el ICU que se desee. La gestión y generación de estos paquetes se hace en el módulo FATFS el cual será explicado más adelante.

4.3.1. Pasos para Iniciar Comunicación Panel Audi

Considerando que la tarjeta de experimentación DISCOVERY será la responsable de mandar los mensajes al clúster (panel Audi), primero se tendrá que desconectar de la comunicación principal el módulo o el conjunto de instrumentos que interesa controlar. En el caso de este trabajo, debido a la facilidad, se procedió a desconectar las líneas que van a la red de "confort" para romper la comunicación.

Con base a los ID's de la Tabla 12, se puede controlar las direccionales derecha e izquierda, controlar las luces altas y bajas e incluso controlar la bocina del automóvil. Todas estas funcionalidades son controladas por la misma red en el auto. En las pruebas realizadas en este trabajo, para fines de prueba de concepto se manipularon las luces y direccionales.

En la Figura 23 se señala con un rectángulo en rojo los dos conectores que deben removerse, para desconectar el módulo de las luces de la red principal. Estos conectores corresponde a las líneas de "CAN High" y "CAN Low", respectivamente. Una vez más para comprobar que se desconectaron las luces de la red principal, se puede encender el clúster y comprobar que los controles del volante dejaron de tener control sobre las luces.

Esta última verificación es importante hacerla antes de conectar cualquier componente externo, en nuestro caso la tarjeta de experimentación. De otro modo, si el módulo de luces no se desconecta de la red principal y al mismo tiempo la tarjeta envía comandos a las luces, se pueden observar comportamientos erráticos en el funcionamiento, lo que puede provocar un daño en el hardware del panel y/o la tarjeta de desarrollo.

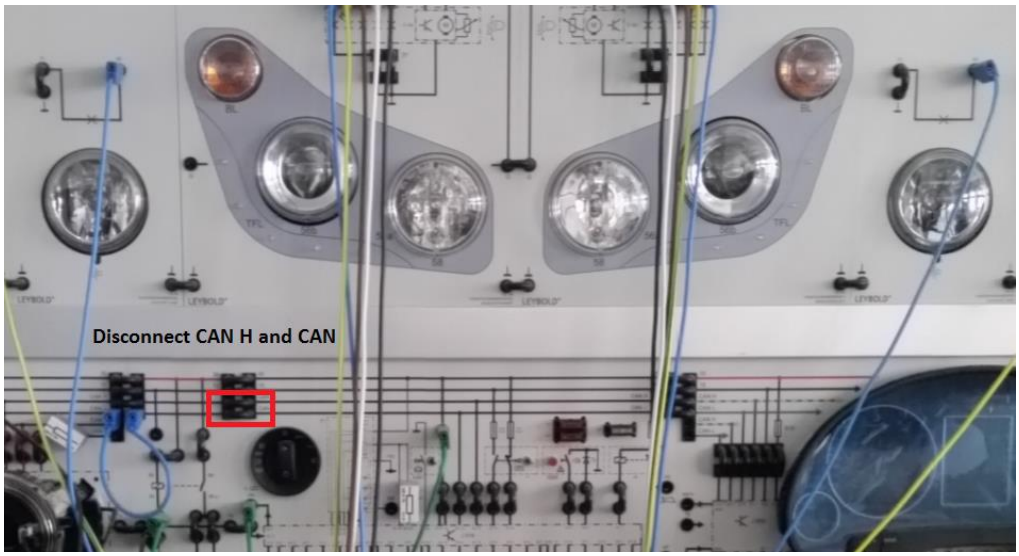


Figura 25. Conexiones Requeridas en el Panel Audi para Establecer una Comunicación

4.3.2. Abrir una Sesión con "CANalyzer"

La comunicación entre la tarjeta DISCOVERY al panel de instrumentos Audi se hace utilizando el protocolo CAN. Por lo tanto la configuración e implementación del "CANalyzer" es esencial. Dicho dispositivo permite abrir una sesión para iniciar la comunicación con el panel. Para comenzar primero es necesario definir al canal con el que se desea trabajar. Esta versión del "CANalyzer" permite la comunicación a través de dos canales.

Aunque es indistinto cual utilizar, en la Figura 24 se define como canal de comunicación a CAN1, que corresponde al primer canal de CANalyzer. Mediante experimentos previos, se determinó que la red de CAN del clúster Audi trabaja a una velocidad (Bau-rate) de 100 kbits/seg. Esto también se pude comprobar escaneando la velocidad de la red a través de la funcionalidad que proporciona el propio "CANalyzer", por medio del botón frontal "Scan".

Una vez con la velocidad configurada, finalmente se configuraron los temporizadores internos de la tarjeta de experimentación DISCOVERY y finalmente se prueba a mandar comandos a los módulos deseados.

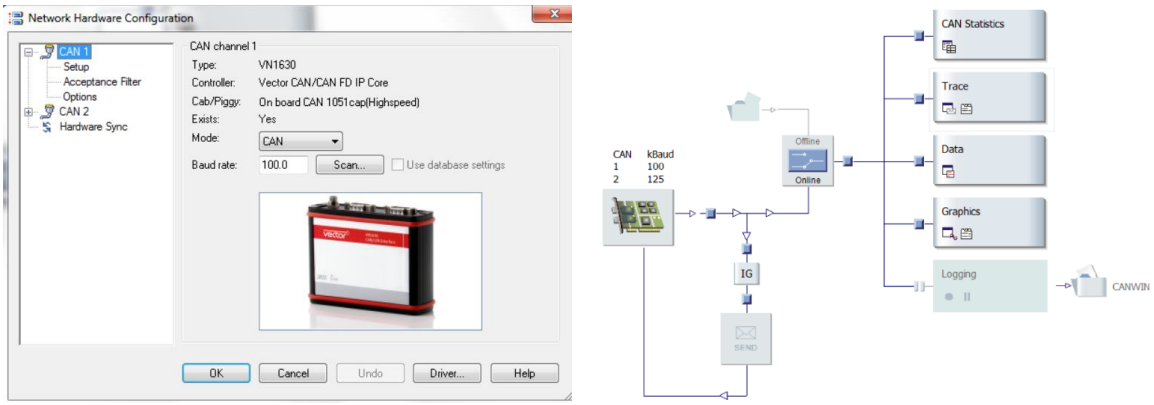


Figura 26. Apertura de una Sesión con CANalyzer

Una ventaja adicional de las herramientas proporcionadas por Vector, como el "CANalyzer", es que sus interfaces graficas son muy amigables. En la parte derecha de la Figura 24 puede verse la GUI desde donde se pueden monitorear en tiempo real la comunicación, verificar que mensajes, su frecuencia de llegada y si el protocolo detecto errores en la comunicación.

Adicionalmente, se pueden observar los ID's de los mensajes que se reciben. Se puede observar además, en el panel de trabajo, el canal que se está monitoreado y la velocidad de transmisión con la que se está trabajando.

4.3.3. Interpretación de Mensajes en el Panel de Trabajo

Desplegando la ventana "Trace" en la herramienta "CANalyzer", se podrá observar con detalle los mensajes que están siendo utilizados, tanto los enviados como los recibidos. Esta venta puede configurarse para ver los que resulte más interesante para el usuario. La Figura 25 muestra una vista de esta ventana.

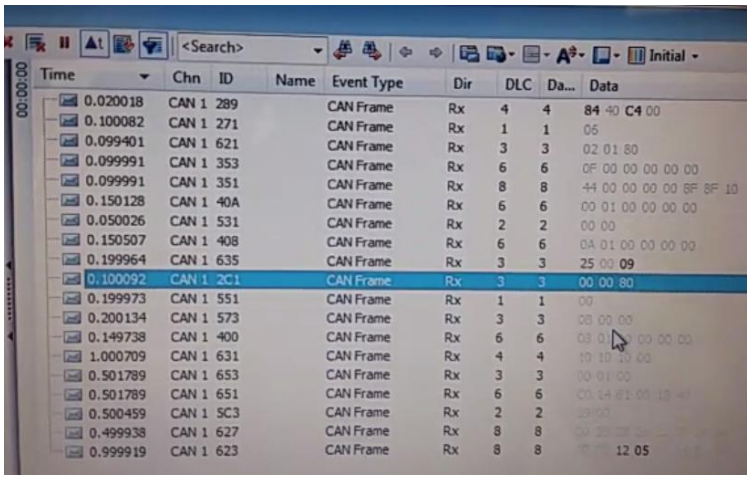


Figura 27. Vista "Trace" en la GUI de CANalyzer

Para el caso específico de este proyecto, y con el fin de conocer los paquetes se están enviando tal y como se requiere, las columnas más importantes son Time, ID, DLC y Data. La Tabla 13 muestra el significado de cada una de estas columnas.

Tabla 13. ID's de Mensajes y Periodo.

Nombre de la columna	*Definición*
Time	*Periodo en segundos en el que un mensaje es mandado o recibido*
Chn	*Canal por el cual el mensaje es enviado o recibido.*
ID	*Identificador del mensaje*
Event Type	*Tipo de evento recibido, también los errores son identificados en esta columna*
Dir	*Dirección de mensaje, Rx si la tarjeta o nodo principal lo manda o Tx si la tarjeta o nodo principal lo recibe.*
DLC	*Tamaño del mensaje en bytes.*
Data	*Datos de cada byte contenido en el mensaje.*

El uso e interpretación de esta lista de mensajes es esencial, ya que resulta útil para verificar posibles problemas en el diseño del protocolo debido a una pobre sincronización de los mensajes y la ausencia del mensaje "Tester Present". Adicionalmente, con el protocolo probado y funcionado, permite obtener información de los errores que en ocasiones se producían, tales como el "Error

Stuffing" o incluso algunas interferencias en el protocolo cunado se conecta una tarjeta sin desconectar el módulo de confort de la red principal, por ejemplo.

Otra funcionalidad que resulta muy útil, es la herramienta que se puede acceder mediante el símbolo "IG" en la interfaz de usuario. Mediante esta funcionalidad en la interface del proveedor Vector, se pueden programar mensajes para que se envíen de manera periódica al nodo que deseemos controlar. Para este proyecto, se utilizó esta funcionalidad para enviar tramas de datos directamente desde nuestra tarjeta. La Figura 26 muestra una vista de esta interfaz.

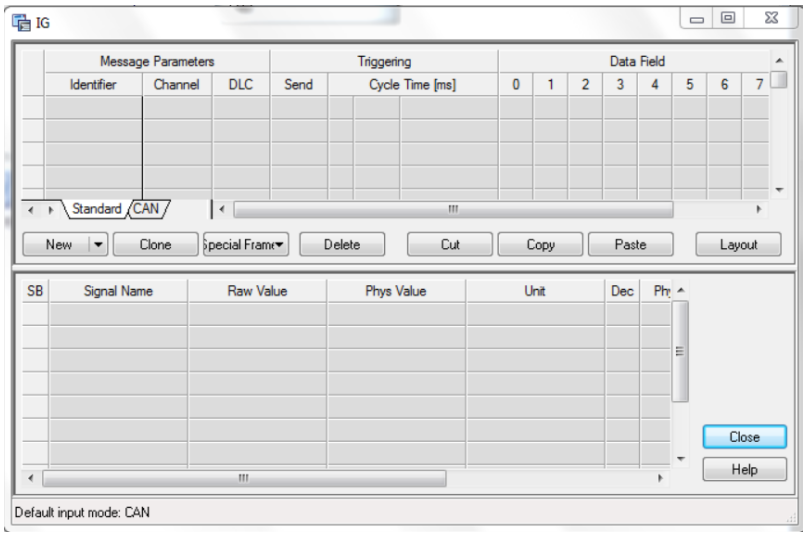


Figura 28. Ventana "IG" de la herramienta de Vector

4.4. IMPLEMENTACION DE CIFRADO

Para el procedimiento de encriptación se utilizaron dos plataformas. En primer lugar, se revisaron varias plataformas para soportar el algoritmo de encriptación. En esta etapa se utilizó el Sistema Operativo "Slax 7.0.8". Éste es un sistema operativo basado en Linux, el cual es posible ejecutarlo desde en una memoria USB. La decisión se basó en el hecho de su versatilidad para arrancar desde una memoria USB lo que lo hacía candidato a correo en plataformas embebidas. La configuración de este sistema operativo quedo de la siguiente manera:

- OS: Slax Linux 7.0.8 con Kernel 3.8.2

- GCC, el compilador por defecto en las distribuciones Linux, utilizado para generar código ejecutable en “Slax Linux”. La versión instalada fue la 4.7.2

En segundo lugar, se revisó las posibles soluciones para el algoritmo de encriptación. En esta etapa se encontró una librería de código abierto que permite el uso de un amplio rango de funciones de encriptación, sin tener que hacer cambios drásticos en el código fuente. Adicionalmente, permite el manejo tanto de archivos como de buffers de datos.

El nombre de esta librería es “mcrypt”. Esta librería de encriptación viene acompañada de “libmcrypt” que es la que contiene las funciones de encriptación. En su diseño, “mcrypt” provee los mecanismos para acceder a esas funciones. La versión instalada fue 2.5.8 de “libmcrypt” en el sistema.

Cabe mencionar que, para esta distribución “Slax Linux 7.0.8”, se utilizó el administrador de paquetes “slackpkg” para instalar los paquetes requeridos por “libmcrypt”. La descarga e instalación de paquetes mediante “slackpkg” es prácticamente automática.

Para verificar el funcionamiento de la librería, se generó código de prueba con un buffer de datos ficticio. Se realizaron los procesos de encriptación y des encriptación, para validar que el buffer de datos contenía la misma información antes y después de ser cifrados.

Una vez validado el funcionamiento de la librería, se procedió a trabajar con el firmware de proyectos previos a fin de generar un firmware objetivo para su programación en memoria flash. A partir de esta etapa, se hizo la separación de las aplicaciones. Por un lado, se desarrolló la parte de encriptación y por otro se hizo la parte de des encriptación.

Este enfoque se siguió con la finalidad de tener los códigos separados para trabajarlos independientemente, debido a la diferencia de los sistemas finales a los que iba dirigidos. Ambos códigos siguieron despendiendo de la librería “libmcrypt” para su adecuado funcionamiento.

4.4.1. Procedimiento de Encriptación

Una vez probado el funcionamiento de la librería de encriptación seleccionada, se procedió a hacer el desarrollo en la tarjeta de pruebas seleccionada, que para este proyecto fue una plataforma "Raspberry B2" con una distribución de "Raspbian" como sistema operativo.

El administrador de paquetes del sistema operativo Raspbian, el administrador "apt-get", también cuenta con los repositorios de las librerías de "mcrypt". Esto simplifico la adaptación del ambiente en la tarjeta para probar el código ya probado en PC. El ambiente de trabajo en el "Raspberry B2" quedo configurado de la siguiente manera:

- GCC: Version 4.6.3
- Libmcrypt: 2.5.8

Debido a que la versión de "libmcrypt" tanto en la PC como en la "Raspberry B2" es la misma, la compilación del código previamente probado en PC fue transparente. Con el código de encriptación trabajando, en la "Raspberry B2", se realizaron pruebas de envío de datos encriptados.

Como primera etapa, se dividió el archivo de firmware en secciones pequeñas de 128 bytes. Este tamaño se decidió debido a que el algoritmo maneja buffers que son múltiplos de 16. Esto planteo retos adicionales al proyecto. En primer lugar, se presentó la necesidad de dividir el archivo en tamaños pequeños, por lo que se llegaron a requerir alrededor de 2800 archivos pequeños en los experimentos. Por diseño, el sistema operativo tiene una variable de entorno para limitar el número de archivos que se pueden abrir por proceso. Por esto, fue necesario modificar dicha variable para permitir un mayor número de archivos abiertos por proceso, mediante el siguiente comando:

```
#ulimit -n 3000
```

Con esto fue posible generar la cantidad de archivos necesarios para el proceso de encriptación. La estructura final del módulo se detalla en la Figura 27.

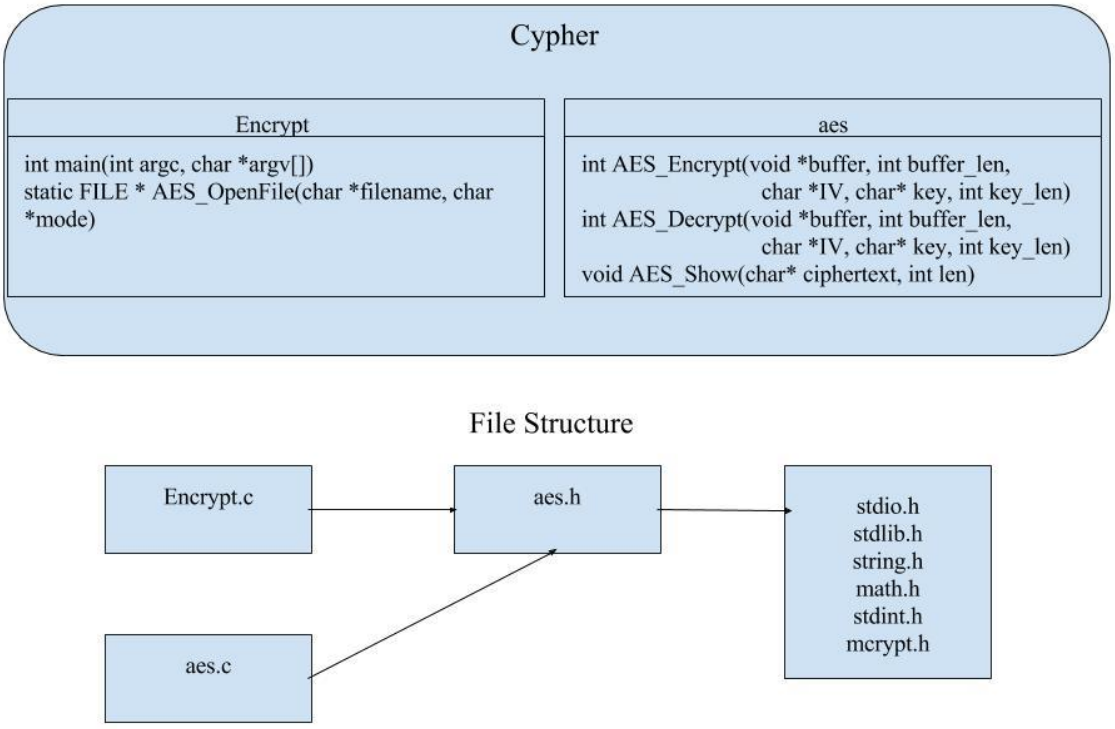


Figura 29. Estructura del Módulo "Cypher"

A fin de facilitar proceso de desarrollo, se optó por simplificar un par de tareas, las funciones de compilación y de ejecución de las aplicaciones por medio de un script. Estos se detallan a continuación.

4.4.1.1. Compilación

Siguiendo la línea de código abierto, se decidió usar la herramienta "make" para poder simplificar el proceso de compilación. Como la parte de la encriptación y del proceso "FrameSender" están en la misma plataforma, también se decidió que "make" se encargue de ambas en un solo paso.

Para poder ejecutar la compilación, simplemente ejecuta el comando “make” dentro de la carpeta “HDRadioProject” del proyecto. Esta carpeta que contiene el código de “Cypher” y de “FrameSender”, además del script requerido por “make”.

```
#make
```

La herramienta “make” se encarga de ingresar en cada una de las carpetas de los componentes, tanto de la parte de “Cypher” como de la parte de “SendFrame”, y hace la compilación de cada componente.

Un punto importante a destacar es que, en la compilación del componente de encriptación, se tiene que agregar la librería de “mcrypt” dentro de los parámetros de compilación. Si la compilación se hiciera en línea de comando, el comando a ejecutar quedaría de la siguiente manera:

```
#gcc aes.c Encrypt.c -c -lmcrypt -o Enc
```

4.4.1.2. Script de Compilación Automática

Para automatizar el proceso de compilación, se desarrolló un script que se encarga de la ejecución secuencial de la encriptación y del “framsender”. Esto con la finalidad de facilitar su ejecución y automatizar con el paso de parámetros para cada uno de los componentes. La sintaxis para la ejecución del script es la siguiente:

```
#./SendFrames.sh <filename_to_encrypt> <path_for_output>
```

En dónde:

- filename_to_encrypt debe de ser la ruta complete del archive a encriptar. Esta puede ser una ruta relativa, pero debe de tenerse en cuenta que es relativa a donde se está ejecutando el script;
- path_for_output también puede ser una ruta relativa o absoluta, pero aquí lo importante es que esta ruta sea hacia un directorio vacío, no debe de contener ningún otro archivo presente.

4.5. IMPLEMENTACIÓN DE LA DES ENCRIPTACIÓN

Coma se mencionó anteriormente, la parte del proceso de des encriptación descifrado queda como trabajo futuro de este proyecto. Sin embargo, a continuación se muestran algunos avances en su implementación básica.

4.5.1. Construcción de “libmcrypt” y código para des encriptación

4.5.1.1. Compilación Cruzada de “libmcrypt”

La primera etapa consiste en realizar una compilación cruzada de la librería “libmcrypt”. Para este propósito, es necesario descargar el código fuente de la misma. Adicionalmente, se requiere soporte para “autobuild”. Ésta es una herramienta de código abierto que permite hacer compilaciones configurando el ambiente de compilación [11].

Los retos principales se presentan en el uso de esta herramienta “autobuild”, ya que tiene dependencias de otras librerías en formato binario para realizar una compilación cruzada. Este problema tiene que ser resuelto instalando todas las dependencias requeridas por “autobuild” en la plataforma objetivo donde se pretenda compilar “libmcrypt” en versiones futuras de este proyecto.

Una alternativa para solventar los problemas de “autobuild” consiste en compilar manualmente el código fuente de “libmcrypt”. Para este procedimiento, se requiere utilizar “avr-gcc” considerando los parámetros requeridos por al usar la herramienta estándar “gcc”.

4.5.2. Código AES nativo para ATMEL

Una alternativa más estricta consiste en tratar directamente con el código nativo de ATMEL para soportar encriptación de 128-bits. De esta manera, se requiere soporte para el algoritmo AES de 128 bits. En este enfoque, existe la implementación "Tiny AES128 in C" disponible en código abierto [6]. Esta implementación está configurada con los mismos requerimientos del método de encriptación utilizado en la "Raspberry B2", que son:

- "Rinjdael AES"
- Modo de bloque de tamaño de 128-bits
- Configurado en modo CBC

Una ventaja de esta alternativa es que la implementación ya fue probada en arquitectura ATMEL con buenos resultados. En esta alternativa, lo primero que se requiere es probar la compilación cruzada del código fuente en la plataforma específica de este proyecto. En pruebas preliminares que se realizaron en este proyecto, el procedimiento de compilación cruzada usando "avr-gcc" no reporto ningún problema.

El siguiente poso debe ser integrar esta implementación en el código que ya se ha desarrollado para validar su funcionalidad. En pruebas preliminares en este trabajo, no se reportaron mayores problemas. Solo se requirieron algunos cambios mínimos en el código que hacer compatibles los datos de la nueva implementación.

4.5.1. Implementaciones y Pruebas Encriptación y Des Encriptación

Para probar los algoritmos de encriptación y des encriptación, se realizaron un conjunto de experimentos con las herramientas descritas en la sección anterior. En particular se realizaron pruebas tanto al código especifico de encriptación como el de des encriptación con la última versión del firmware implementado.

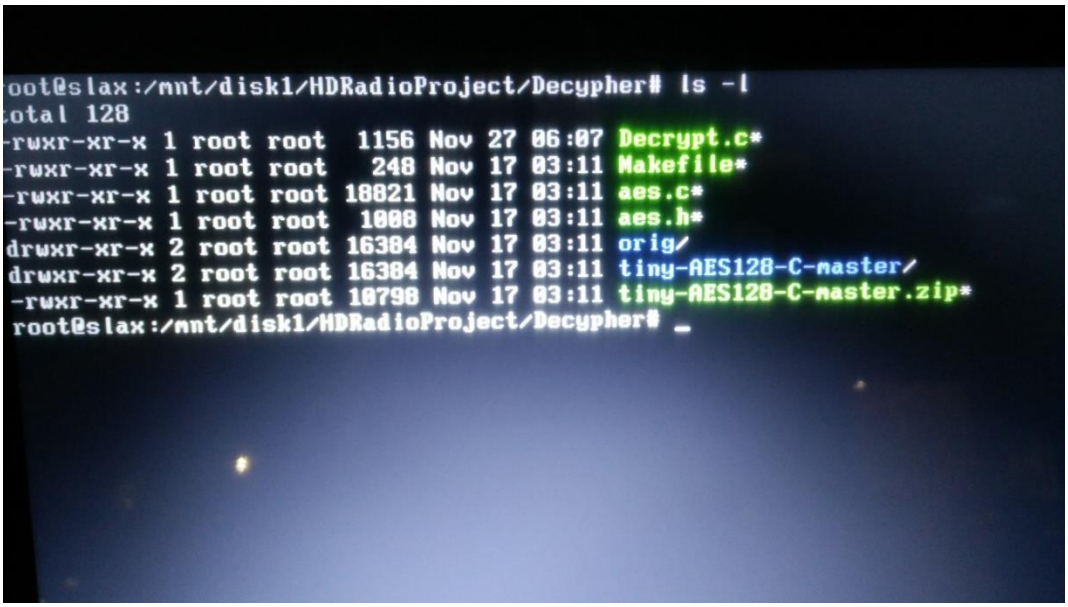


Figura 30. Conjunto de Archivos para Des Encriptación

La Figura 30 muestra la carpeta del procedimiento de des encriptación, antes de hacer la compilación del código. Esta compilación se realizó en el la computadora host de desarrollo.

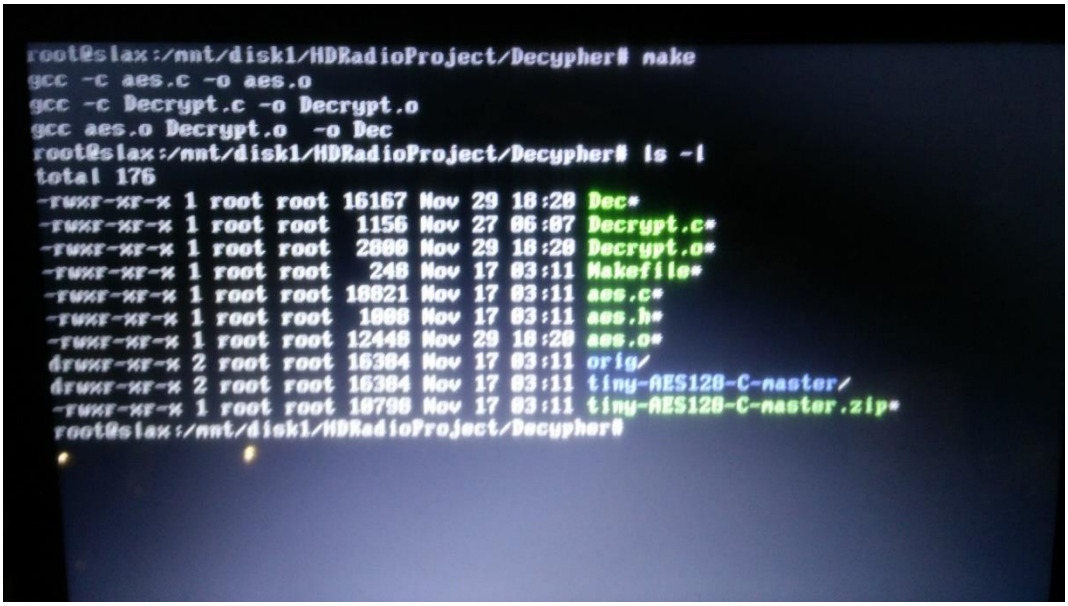


Figura 31. Conjunto de Archivos para Des Encriptación una vez Compilados

En la Figura 31 se muestra el proceso de compilación hecho mediante el procedimiento "make". En este caso, el procedimiento sigue utilizando GCC en esta etapa. Después del procedimiento, en la carpeta se observan los archivos de salida y el archivo ejecutable "Dec".

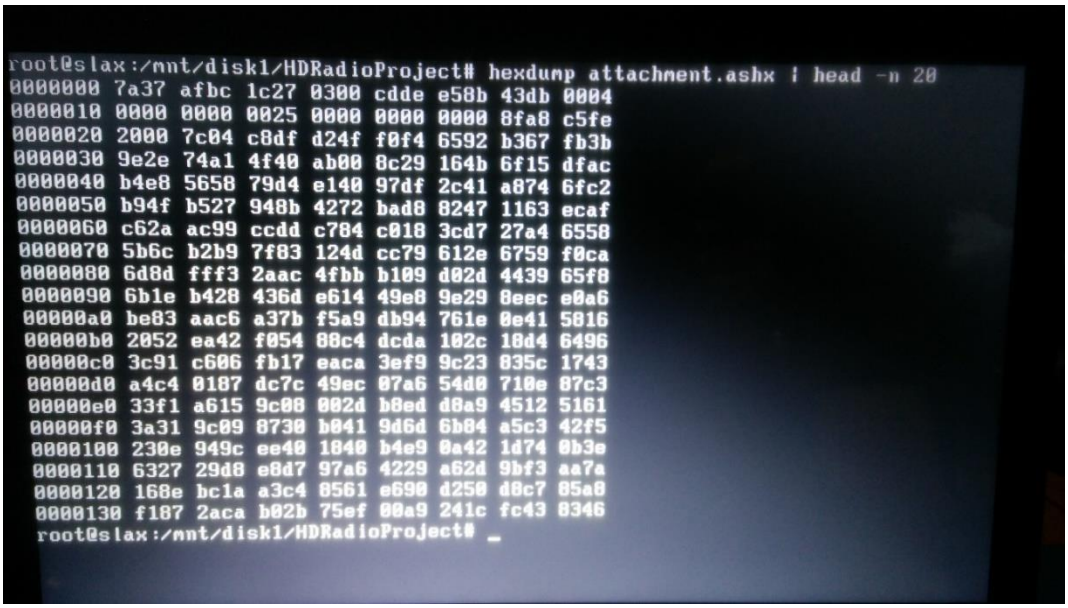


Figura 32. Contenido del archivo "attachment.ashx" para experimentación

En la Figura 32 se muestra el contenido inicial del archivo de experimentación "attachment.ashx" en su versión original. Se muestra solo una parte del archivo debido a su gran tamaño y con fines ilustrativos para comparación una vez encriptado con el código implementado.

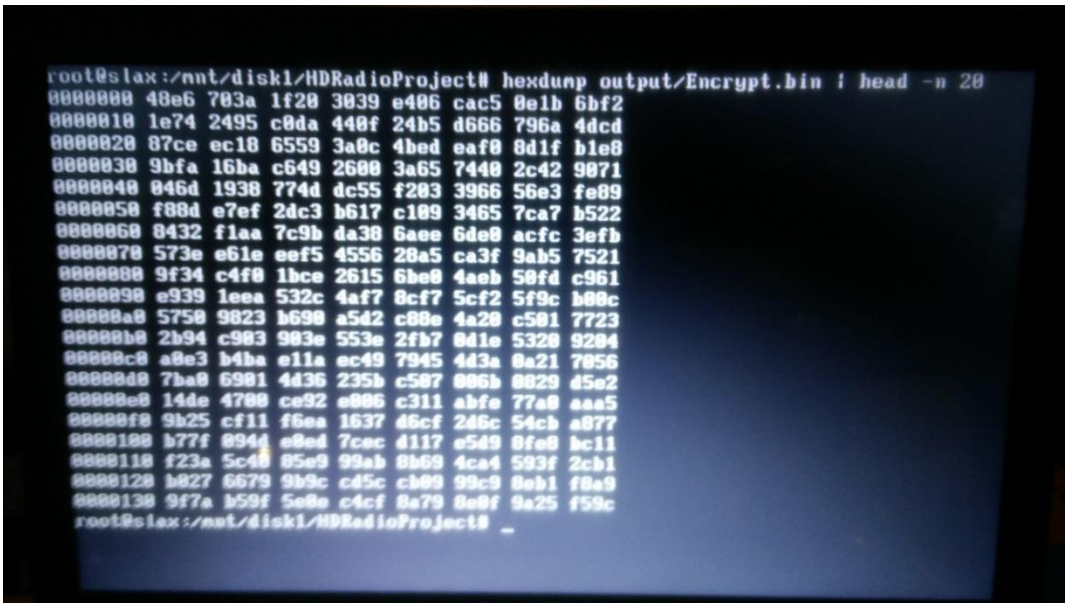


Figura 33. Contenido del archivo “attachment.ashx” encriptado

La Figura 33, muestra la parte inicial del mismo archivo “attachment.ashx” una vez encriptado con el código implementado en este trabajo. Como se puede observar, el contenido es diferente del archivo original (ver Figura 32). Esto indica que el método implementado trabajo de manera correcta. Adicionalmente, existen varias herramientas en línea que se pueden utilizar para validar que el procedimiento de encriptado trabajo como se espera [10].

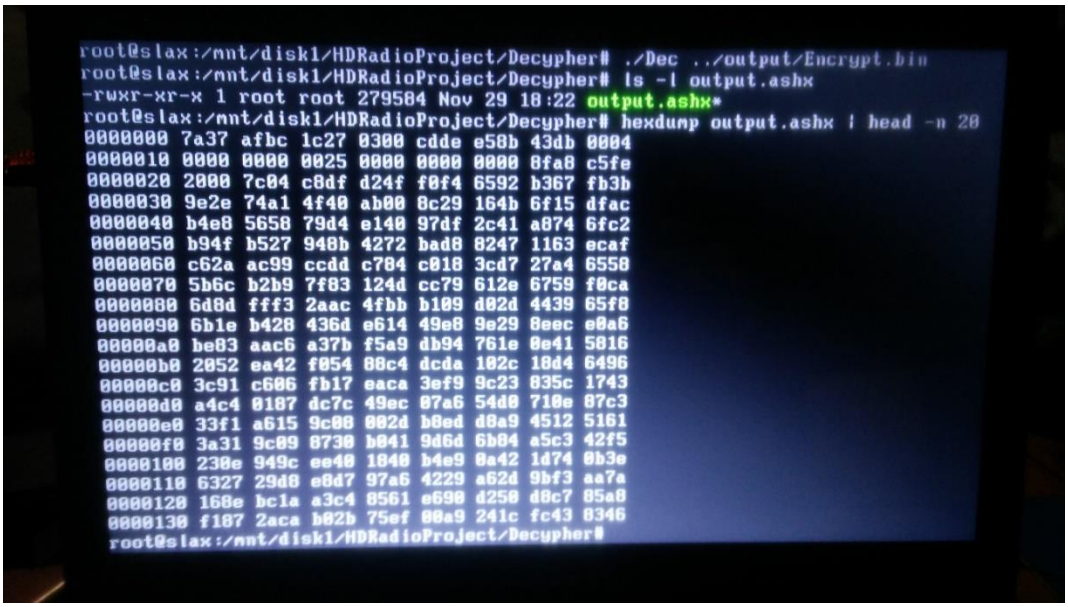


Figura 34. Contenido del archivo “attachment.ashx” des encriptado

En la Figura 34 se muestra el proceso de des encriptación del archivo resultado de la encriptación de “attachment.ashx”. Como es de esperarse, el archivo de salida de este último proceso de des encriptación es el mismo que el del archivo original (attachment.ashx).

5. CONCLUSIONES

Resumen: *En este capítulo se presentan las conclusiones de este proyecto y algunas pautas para el trabajo futuro de este desarrollo tecnológico.*

5.1. RESULTADOS

5.1.1. Módulo “OTA Receiver”

Durante el tiempo de desarrollo de la unidad receptora de la actualización (En este caso la tarjeta de desarrollo DISCOVERY) diferentes pruebas para validar los módulos fueron realizadas. Dichas pruebas fueron útiles para llevar a los resultados deseados los cuales estaban enfocados completamente en el desempeño del protocolo. Las pruebas realizadas sobre la unidad solo contemplan la conexión de la unidad a una PC.

En versiones anteriores se hicieron pruebas con la tarjeta “Raspberry B2” pero debido a que la actualización del protocolo solo se integró en la unidad receptora y no en la transmisora pruebas con dicho ambiente para la nueva versión del protocolo no fueron realizadas. Básicamente las pruebas realizadas consistieron en 5 experimentos descritos a continuación:

- Prueba sobre el driver de UART2 usado para la recepción de los datos
- Prueba sobre el analizador de mensajes
- Prueba sobre el modulo que atiende los mensajes recibidos
- Prueba de sincronización

Para probar la funcionalidad sobre el UART2 los experimentos realizados fueron usando el depurador que STMicroelectronics tiene incluido en la tarjeta “DISCOVERY”. Dicha prueba consistió en enviar un dato desde la PC mediante UART usando un programa llamado “Docklight” el cual fue usado también para las demás pruebas.

Como se menciona en el párrafo anterior, las pruebas para validación del servicio de UART2 fueron hechas mediante “debugger” y la razón de esto fue que el módulo de “log and trace” todavía no estaba implementado, al mismo tiempo una vez desarrollado esto se comenzó el desarrollo de “log and trace” y es así como los otros módulos fueron probados.

En la Figura 35 se observa la ventana de la herramienta “Docklight” con el conjunto de datos que se usaron para las diferentes pruebas y experimentos.

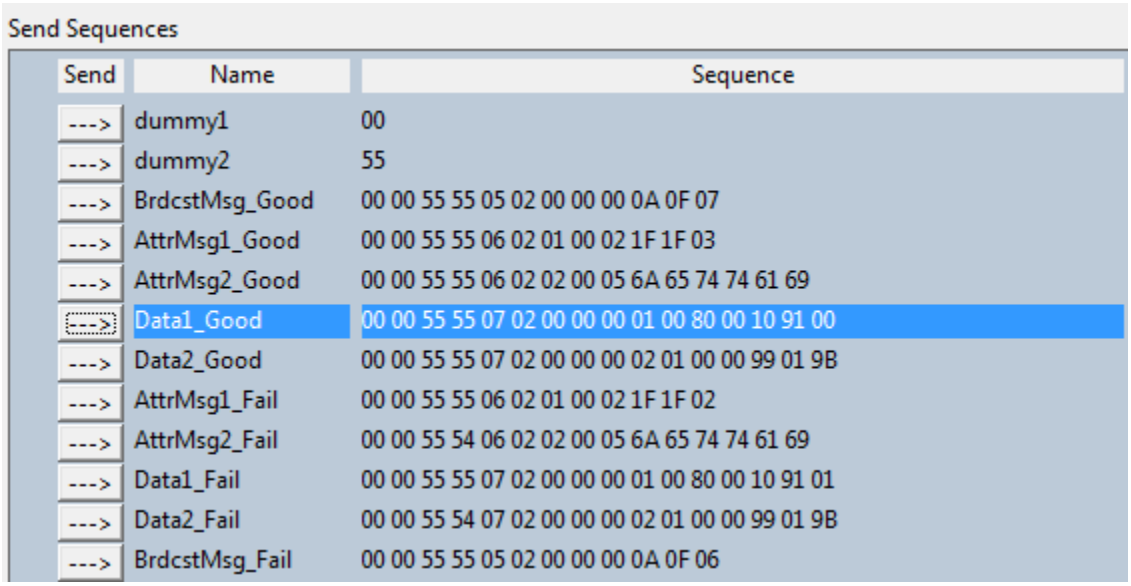


Figura 35. Herramienta Docklight para Depuración y Pruebas

Como se puede observar en la Figura 35, existen diferentes tramas de datos los cuales fueron usados para validar los diferentes aspectos de interés en el proyecto. Las respuestas o “logs” generados por la unidad receptora pueden ser visualizados mediante cualquier monitor serial con la configuración adecuada. Abajo se muestran las diferentes respuestas a las tramas recibidas:

- **“Dummy1”, “Dummy2”:** Son solo dos valores los cuales corresponden a la etapa de sincronización, estos se usaron para probar de manera más simple la validación de la sincronización. Se enviaban diferentes combinaciones de los datos como por ejemplo [dummy1,dummy2, dummy1,dummy2, dummy1,dummy2,] los cuales tenían que arrojar un error de sincronización por medio del módulo de “log and trace”

En el caso que la parte de sincronización sea enviara de manera correcta, es decir cuando se envía la siguiente secuencia [dummy1, dumm1, dummy2, dummy2] los rastros recibidos son los siguientes:

- **“BrdcstMsg_Good”, “AttrMsg1_Good”, “Data1_Good”:** Estos mensajes fueron usados para validar que en casos de éxito la unidad receptora estaba funcionando bien, dichos

mensajes están hechos siguiendo el formato definido por el protocolo. La Figura 36 muestra una parte del "log" para este caso.

```
11/30/2016 11:29:18 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:29:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x0] u8Byte_ctr[0]
11/30/2016 11:29:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:29:18 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:29:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[1]
11/30/2016 11:29:19 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:29:19 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:29:19 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x0] u8Byte_ctr[2]
11/30/2016 11:29:19 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:29:19 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:29:19 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[3]
11/30/2016 11:29:19 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:29:27 #2 [HDP-ERR]  HDP__boValiteSync(447) - Wrong Sync
11/30/2016 11:29:27 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:29:27 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x0] u8Byte_ctr[0]
11/30/2016 11:29:28 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:29:28 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:29:28 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[1]
```

Figura 36. Verificación de la Funcionalidad de la Unidad Receptora

Cada mensaje genera un comportamiento diferente, y archivos "logs" difieren en consecuencia en la unidad receptora. De este modo, un ejemplo de las respuestas deseadas para cada mensaje serian:

- Respuesta para "BrdcstMsg_Good".
- Respuesta para "AttrMsg1_Good"
- Respuesta para "Data1_Good". La Figura 37 muestra una sección del archivo "log" para este experimento.

```
11/30/2016 11:34:17 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:34:17 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x0] u8Byte_ctr[0]
11/30/2016 11:34:17 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:34:17 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:34:17 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x0] u8Byte_ctr[1]
11/30/2016 11:34:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:34:18 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:34:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[2]
11/30/2016 11:34:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:34:18 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:34:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[3]
11/30/2016 11:34:18 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:34:18 #2 [HDP-INF]  HDP__boValiteSync - ValidSync.
```

Figura 37. Respuesta para un "BrdcstMsg_Good"

```
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[2]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vIDLEStte_hdl - New Data[0x55] u8Byte_ctr[3]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__boValiteSync - ValidSync.
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vIDLEStte_hdl - Sync Validated
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - NewData[0x5]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - HDP_enNewBroadcstMsg
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vNewBrdcstStte_hdl - broad receibed
11/30/2016 11:40:08 #2 [HDP-INF]  HDP__vAddMsg - Msg Saved type[5]
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vIDLE_statHdl - MsgBuffler_len[1]
11/30/2016 11:40:08 #2 [HDP-INF]  HDP_vGetMsg - Msg LoadedID[5]
11/30/2016 11:40:08 #2 [HDM-INF]  HDM__vSetState - State -> Brdcst
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - ---BROADCOAST---------
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - enMsgID[0x5]
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - NumAttr[0x2]
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - NumPkg[0xa]
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - u8SwKey[0xf]
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - CRC[0x7]
11/30/2016 11:40:08 #2 [HDM-INF]  HDM__vSetState - State -> Vlidtion
11/30/2016 11:40:08 #2 [HDM-INF]  HDM__boValidateChksm -
11/30/2016 11:40:08 #2 Chksm validated[0x7] CRCVal[0x7]
11/30/2016 11:40:08 #2
11/30/2016 11:40:08 #2 [HDM-DEBUG]HDM__vMsgValition_statHdl - Valid Chcksm
11/30/2016 11:40:09 #2 [HDM-INF]  HDM__vSetState - State -> DtaSvng
11/30/2016 11:40:09 #2 [HDM-DEBUG]HDM__vDataSvng_statHdl - New Brdcst, Create Database...
11/30/2016 11:40:09 #2 [HDM-DEBUG]HDM__vDataSvng_statHdl - Database Created...
11/30/2016 11:40:09 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
```

Figura 38. Respuesta para un “BrdcstMsg_Good”

```
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__vIDLEStte_hdl - MsgTotal[0]
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__boValiteSync - ValidSync.
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__vIDLEStte_hdl - Sync Validated
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - NewData[0x6]
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - HDP_enAttrMsg
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-DEBUG]HDP__vAttrMsgStte_hdl - u16ParamMount[5]
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__vAttrMsgStte_hdl - Attri Received
11/30/2016 11:41:11 #2 [HDP-INF]  HDP__vAddMsg - Msg Saved type[6]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vIDLE_statHdl - MsgBuffler_len[1]
11/30/2016 11:41:11 #2 [HDP-INF]  HDP_vGetMsg - Msg LoadedID[6]
11/30/2016 11:41:11 #2 [HDM-INF]  HDM__vSetState - State -> Attr
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - Byte1[0x0] Byte2[0x5]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - ---ATTRIBUTE---------
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - enMsgID[0x6]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - u8SwKey[0x2]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - u8AttrID[0x2]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - u16NumParam[0x5]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - Params[jettai]
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - CRC[0x69]
11/30/2016 11:41:11 #2 [HDM-INF]  HDM__vSetState - State -> Vlidtion
11/30/2016 11:41:11 #2 [HDM-INF]  HDM__boValidateChksm -
11/30/2016 11:41:11 #2 Chksm validated[0x69] CRCVal[0x69]
11/30/2016 11:41:11 #2
11/30/2016 11:41:11 #2 [HDM-DEBUG]HDM__vMsgValition_statHdl - Valid Chcksm
11/30/2016 11:41:12 #2 [HDM-INF]  HDM__vSetState - State -> DtaSvng
11/30/2016 11:41:12 #2 [HDM-DEBUG]HDM__vDataSvng_statHdl - DB exist,Adding attribute...
11/30/2016 11:41:12 #2 [HDM-DEBUG]HDM__vDataSvng_statHdl - Attribute Added...
11/30/2016 11:41:12 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
```

Figura 39. Respuesta para un "BrdcstMsg_Good"

- **"BrdcstMsg_Fail", "AttrMsg1_Fail", "Data1_Fail":** Con este set de mensajes se probó la funcionalidad del "checksum", cada mensaje definido en "docklight" tiene un error de "check sum". Los otros mensajes etiquetados con "Fail" que no están incluidos en esta descripción tienen problemas con la sección de sincronización ya que se les modifico el valor del último byte de sincronización para validar esto así que la respuesta será similar al primer apartado de esta subsección. Respecto a los resultados recibidos por los mensajes de falla son:
    - Respuesta para "BrdcstMsg_Fail"
    - Respuesta para "AttrMsg1_Fail"
    - Respuesta para "Data1_Fail"

```
11/30/2016 11:41:40 #2 [HDP-INF]  HDP__vIDLEStte_hdl - Sync Validated
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - NewData[0x7]
11/30/2016 11:41:40 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - HDP_enPkgMsg
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:41:40 #2 [HDP-INF]  HDP__vPkgMsgStte_hdl - PkgMsg Received
11/30/2016 11:41:40 #2 [HDP-INF]  HDP__vAddMsg - Msg Saved type[7]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vIDLE_statHdl - MsgBuffler_len[1]
11/30/2016 11:41:40 #2 [HDP-INF]  HDP_vGetMsg - Msg LoadedID[7]
11/30/2016 11:41:40 #2 [HDM-INF]  HDM__vSetState - State -> Pkg
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - ---PACKAGE---------
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - enMsgID[0x7]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - u8SwKey[0x2]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - PkgID[0x1]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data1[0x0]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data2[0x80]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data3[0x0]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data4[0x10]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data5[0x91]
11/30/2016 11:41:40 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - CRC[0x0]
11/30/2016 11:41:41 #2 [HDM-INF]  HDM__vSetState - State -> Vlidtion
11/30/2016 11:41:41 #2 [HDM-INF]  HDM__boValidateChksm -
11/30/2016 11:41:41 #2 Chksm validated[0x0] CRCVal[0x0]
11/30/2016 11:41:41 #2
11/30/2016 11:41:41 #2 [HDM-DEBUG]HDM__vMsgValition_statHdl - Valid Chcksm
11/30/2016 11:41:41 #2 [HDM-INF]  HDM__vSetState - State -> DtaSvng
11/30/2016 11:41:41 #2 [HDM-DEBUG]HDM__vDataSvng_statHdl - DB exist,Adding package...
11/30/2016 11:41:41 #2 [HDM-DEBUG]HDM__vDataSvng_statHdl - Pkg Added...
11/30/2016 11:41:41 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
```

Figura 40. Vista archivo "log" de Respuesta para Experimentos (Parte 1)

```
11/30/2016 11:51:03 #2 [HDP-INF]  HDP__boValiteSync - ValidSync.
11/30/2016 11:51:03 #2 [HDP-INF]  HDP__vIDLEStte_hdl - Sync Validated
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - NewData[0x5]
11/30/2016 11:51:03 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - HDP_enNewBroadcstMsg
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:03 #2 [HDP-INF]  HDP__vNewBrdcstStte_hdl - broad receibed
11/30/2016 11:51:03 #2 [HDP-INF]  HDP__vAddMsg - Msg Saved type[5]
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vIDLE_statHdl - MsgBuffler_len[1]
11/30/2016 11:51:03 #2 [HDP-INF]  HDP_vGetMsg - Msg LoadedID[5]
11/30/2016 11:51:03 #2 [HDM-INF]  HDM__vSetState - State -> Brdcst
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - ---BROADCOAST---------
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - enMsgID[0x5]
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - NumAttr[0x2]
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - NumPkg[0xa]
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - u8SwKey[0xf]
11/30/2016 11:51:03 #2 [HDM-DEBUG]HDM__vBrdMsg_statHdl - CRC[0x6]
11/30/2016 11:51:03 #2 [HDM-INF]  HDM__vSetState - State -> Vlidtion
11/30/2016 11:51:03 #2 [HDM-ERR]  HDM__boValidateChksm(481) -
11/30/2016 11:51:03 #2 Invalid Chksm[0x7] CRCVal[0x6]
11/30/2016 11:51:03 #2
11/30/2016 11:51:04 #2 [HDM-ERR]  HDM__vMsgValition_statHdl(292) - Invalid Msg CheckSum
11/30/2016 11:51:04 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
```

Figura 41. Vista archivo "log" de Respuesta para Experimentos (Parte 2)

```
11/30/2016 11:51:33 #2 [HDP-INF]  HDP__vIDLEStte_hdl - Sync Validated
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - NewData[0x6]
11/30/2016 11:51:33 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - HDP_enAttrMsg
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-DEBUG]HDP__vAttrMsgStte_hdl - u16ParamMount[2]
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:33 #2 [HDP-INF]  HDP__vAttrMsgStte_hdl - Attri Received
11/30/2016 11:51:33 #2 [HDP-INF]  HDP__vAddMsg - Msg Saved type[6]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vIDLE_statHdl - MsgBuffler_len[1]
11/30/2016 11:51:33 #2 [HDP-INF]  HDP_vGetMsg - Msg LoadedID[6]
11/30/2016 11:51:33 #2 [HDM-INF]  HDM__vSetState - State -> Attr
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - Byte1[0x0] Byte2[0x2]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - ---ATTRIBUTE---------
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - enMsgID[0x6]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - u8SwKey[0x2]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - u8AttrID[0x1]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - u16NumParam[0x2]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - Params[□]
11/30/2016 11:51:33 #2 [HDM-DEBUG]HDM__vAttrMsg_statHdl - CRC[0x2]
11/30/2016 11:51:33 #2 [HDM-INF]  HDM__vSetState - State -> Vlidtion
11/30/2016 11:51:33 #2 [HDM-ERR]  HDM__boValidateChksm(481) -
11/30/2016 11:51:33 #2 Invalid Chksm[0x3] CRCVal[0x2]
11/30/2016 11:51:33 #2
11/30/2016 11:51:33 #2 [HDM-ERR]  HDM__vMsgValition_statHdl(308) - Invalid Msg CheckSum
11/30/2016 11:51:33 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
```

Figura 42. Vista archivo “log” de Respuesta para Experimentos (Parte 3)

```
11/30/2016 11:51:57 #2 [HDP-INF]  HDP__boValiteSync - ValidSync.
11/30/2016 11:51:57 #2 [HDP-INF]  HDP__vIDLEStte_hdl - Sync Validated
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - NewData[0x7]
11/30/2016 11:51:57 #2 [HDP-INF]  HDP__vFramIdentStte_hdl - HDP_enPkgMsg
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vNewData_nofit - New data notification
11/30/2016 11:51:57 #2 [HDP-INF]  HDP__vPkgMsgStte_hdl - PkgMsg Received
11/30/2016 11:51:57 #2 [HDP-INF]  HDP__vAddMsg - Msg Saved type[7]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vIDLE_statHdl - MsgBuffler_len[1]
11/30/2016 11:51:57 #2 [HDP-INF]  HDP_vGetMsg - Msg LoadedID[7]
11/30/2016 11:51:57 #2 [HDM-INF]  HDM__vSetState - State -> Pkg
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - ---PACKAGE---------
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - enMsgID[0x7]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - u8SwKey[0x2]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - PkgID[0x1]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data1[0x0]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data2[0x80]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data3[0x0]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data4[0x10]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - Data5[0x91]
11/30/2016 11:51:57 #2 [HDM-DEBUG]HDM__vPkgMsg_statHdl - CRC[0x1]
11/30/2016 11:51:58 #2 [HDM-INF]  HDM__vSetState - State -> Vlidtion
11/30/2016 11:51:58 #2 [HDM-ERR]  HDM__boValidateChksm(481) -
11/30/2016 11:51:58 #2 Invalid Chksm[0x0] CRCVal[0x1]
11/30/2016 11:51:58 #2
11/30/2016 11:51:58 #2 [HDM-ERR]  HDM__vMsgValition_statHdl(324) - Invalid Msg CheckSum
11/30/2016 11:51:58 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
```

Figura 43. Vista archivo "log" de Respuesta para Experimentos Adicionales

Una prueba extra realizada la cual estaba enfocada en el desempeño de la unidad receptora, se baso en hacer que los mensajes anteriores se enviaran constantemente, se hacia una desconexión del cable Tx proveniendo de la aplicación que enviaba los paquetes con esto la comunicación se tomaba como perdida y una vez que había pasado determinado tiempo (10 seg) volvíamos a realizar la conexión y de esta manera podíamos observar como la unidad receptora comenzaba a sincronizarse y atender los mensajes recibidos. La Figura 43 muestra una sección del archivo "log".

5.1.2. Obtención de Paquetes Receptor HD-Radio.

En una primera etapa del proyecto, fue necesario averiguar el funcionamiento del protocolo de HD-Radio, a fin de obtener datos sobre la manera en la que el radio envía los datos al receptor. En estas pruebas se encontró que la señal de HD-Radio era trasmitida mediante un generador y recibida por una radio capaz de captar esta señal en este formato. Sobre este último elemento, el receptor, se extraían archivos binarios los cuales contenían información referente a los datos que se habían transmitido por medio de la señal HD-Radio.

En este punto e crearon una serie de scripts en Pyton con el fin de encontrar patrones o secuencias en el contenido de estos archivos. Después de varios experimentos y su análisis, se pudo concluir que incluso aunque la trasmisión se hacía de manera repetitiva, las tramas dentro del archivo binario siempre cambiaban por lo que esto agregaba una especie de seguridad que dificultaba encontrar un patrón. Atribuimos este inconveniente a que los datos que llegaban a la radio receptora ya estaban encriptados desde la fuente.

5.1.3. Módulo FAT

En la Figura 44 se muestra una sección del "log" de una sesión de lectura y escritura en el módulo FAT. Como se mencionó en el apartado de implementación, antes de que el módulo FAT pueda ser utilizado, se necesitan una serie de precondiciones. La primer trama de la Figura 44 muestra un "Waiting for USB" indicando que la aplicación espera la entrada del dispositivo donde se almacenaran los archivos. Una vez que la USB es detectada se procede a montar la unidad. Como lo muestra la trama "Device mounted", esto indica el éxito en el montaje de la memoria. A partir de este momento, el modulo queda habilitado para recibir peticiones.

Esta prueba se hizo en una memoria totalmente en blanco, por lo tanto no había archivos que leer. Así, cuando se intentó abrir una sesión de lectura, se obtiene un mensaje de error al abrir el archivo "FILE10.BIN", lo podemos confirmar con la trama "FSA_enGetPackage (317) – Error

Opening File [FILE10.BIN]". Esto en consecuencia generada un error de lectura, ya que el archivo requerido no existe.

En un segundo caso de prueba, se validó el funcionamiento de la sesión y proceso de escritura. Para este fin, se creó un archivo binario nuevo, llamado "FILE11.BIN" y se procedio la apertura de la sesión de escritura "FSA_enWrittingSession – Writing Session Started". En seguida se realizó la escritura de un paquete en dicho archivo "PwrMngr_vmainTask – FSA_enWriteDataPkg". El procedimiento no reporta errores, por lo que se considera satisfactirio y se hace el cierre de la sesión de escritura. Posteriomente se guardar las versión final del archivo. En este punto el contendido que solo se ha modificado en RAM se procede a copiar al archivo binario correspondiente.

```
11/30/2016 13:23:52 #2 [PWRM-DEBUG]PwrMngr__vInitFsa - Waiting For USB...
11/30/2016 13:23:52 #2 [HDP-INF]  HDP_vInit - Module Init
11/30/2016 13:23:52 #2 [HDM-INF]  HDM__vSetState - State -> IDLE
11/30/2016 13:23:52 #2 [HDM-INF]  HDM__vSetState - Msg is clean
11/30/2016 13:23:52 #2 [CAN-INF]  CANDrv_vInit - CAN Init
11/30/2016 13:23:53 #2 [USB-ERR]  USBHost_vInit(59) - Notificiation Service not needed.
11/30/2016 13:23:53 #2 [FATSe-INF]  FSA_vInit - Module Init
11/30/2016 13:23:53 #2 [PWRM-INF]  PwrMngr__vInitFsa - Modules are Init.
11/30/2016 13:23:53 #2 [PWRM-DEBUG]PwrMngr__vInitFsa - Waiting For USB...
11/30/2016 13:23:53 #2 [FATSe-INF]  FSA_vMountDevice - Device mounted
11/30/2016 13:23:54 #2 [FATSe-INF]  FSA_enReadingSession - au8StringFileName[FILE10.BIN]
11/30/2016 13:23:54 #2 [FATSe-ERR]  FSA_enReadingSession(193) - Error Opening File[FILE10.BIN]
11/30/2016 13:23:54 #2 [PWRM-INF]  PwrMngr_vmainTask - Reading Session Opened [1]
11/30/2016 13:23:54 #2 [FATSe-ERR]  FSA_enGetFile(293) - Error Reading file[] ID[0]
11/30/2016 13:23:54 #2 [PWRM-INF]  PwrMngr_vmainTask - FSA_enGetFile[1]
11/30/2016 13:23:54 #2 [MAIN-DEBUG]PwrMngr_vmainTask - Result file[]
11/30/2016 13:23:54 #2 [FATSe-ERR]  FSA_enGetPackage(317) - Error Reading in file[] ID[0]
11/30/2016 13:23:54 #2 [PWRM-INF]  PwrMngr_vmainTask - FSA_enGetPackage[1]
11/30/2016 13:23:54 #2 [MAIN-DEBUG]PwrMngr_vmainTask - Result Package[]
11/30/2016 13:23:54 #2 [FATSe-ERR]  FSA_enCloseSession(354) - Error closing file[]
11/30/2016 13:23:54 #2 [FATSe-INF]  FSA_enCreateNewFile - File Created[FILE11.BIN]
11/30/2016 13:23:54 #2 [PWRM-INF]  PwrMngr_vmainTask - Creating File[0]
11/30/2016 13:23:54 #2 [FATSe-INF]  FSA_enWrittingSession - au8StringFileName[FILE11.BIN]
11/30/2016 13:23:54 #2 [FATSe-INF]  FSA_enWrittingSession - Writing Session Started
11/30/2016 13:23:54 #2 [FATSe-INF]  FSA_enWrittingSession - File loaded[FILE11.BIN]
11/30/2016 13:23:54 #2 [PWRM-INF]  PwrMngr_vmainTask - Writing Session Opened [0]
11/30/2016 13:23:54 #2 [PWRM-INF]  PwrMngr_vmainTask - FSA_enWriteDataPkg [0]
11/30/2016 13:32:02 #2 [FATSe-INF]  FSA_enCloseSession - File Saved
```

Figura 44. Vista archivo "log" de Sesiones de Lectura y Escritura del Modulo FAT

5.2. TRABAJO FUTURO

Se pretende que este proyecto continúe con su desarrollo mediante la implementación de funcionales adicionales y aquellas que quedaron pendientes o en un versión preliminar en este trabajo.

En particular, la versión final del módulo FAT que se presentó en este trabajo es totalmente funcional y es capaz de crear los archivos binarios, como originalmente estaban planeados. Esto archivos son de 128 bytes, dividido en paquetes de 5 bytes. Sin embargo debido a cambios hechos en el módulo de encriptación, se tendrán que realizar algunas adecuaciones a este módulo.

Actualmente, las funciones para este fin ya están declaradas pero no implementadas en el código del módulo. Dichas declaraciones se encuentran ubicadas al final del archivo "fatfs.c". El objetivo de estas nuevas funciones es gestionar la creación de folders para cada "software key", creación de archivo de atributos y la socialización de múltiples archivos binarios.

En la nueva implementación, se deberá ser capaz de crear múltiples folders, o bases de datos. Cada uno de estos folders corresponderá a un "software key" diferente. En primer lugar, cada folder creado contendrá invariablemente un archivo denominado "Attribuites.txt", el cual guardara información relacionada al los atributos del mensaje. Adicionalmente, contendrán uno o más archivos binarios dependiente de que tan extenso sea la información que se vaya a transmitir, de una manera similar a la versión actual, donde cada archivo binario deberá ser de 128 bytes. Este enfoque permitirá que gran cantidad del código actual sea reusable.

La idea esencial de esta nueva implementación es que cada archivo binario contenga cierto rango de paquetes únicos. La Figura 45 muestra una propuesta de arquitectura del módulo FAT propuesto como trabajo futuro. Como se puede observar, el archivo binario1 almacenara paquetes del 1 a 26, el archivo binario2 almacena paquetes del 27 al 52 y así sucesivamente. De tal manera que con solo una sencilla división aritmética sea posible ubicar de antemano en que archivo binario se encuentra el paquete de interés. Se cree que con este enfoque, se ahorrará tiempo y procesamiento en la localización de paquetes específicos.

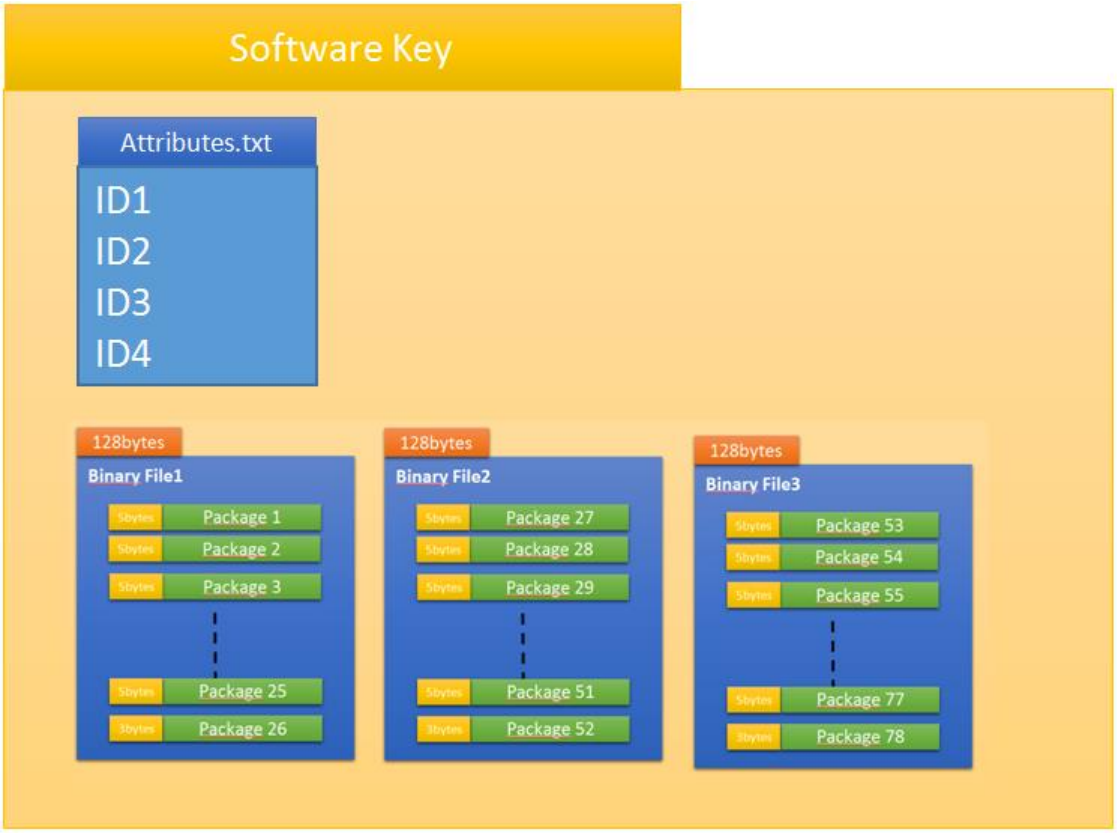


Figura 45. Arquitectura del Módulo FAT para Trabajo Futuro.

REFERENCIAS

[1] Tesla Software 8.0, https://www.tesla.com/software?redirect=no.

[2] Ford SYNC System, https://www.ford.com/technology/sync/.

[3] GM OnStar System, http://www.gm.com/index.html

[4] Training panel lighting NGLD739 5821, http://www.elwe-technology.com/technik/geraetekatalog-automotive/7x/training-panel-lighting-ng-7395821.html

[5] High-performance foundation line, ARM Cortex-M4 core, http://www.st.com/en/microcontrollers/stm32f407vg.html

[6] Tiny AES128 in C, https://github.com/kokke/tiny-AES128-C

[7] Over-the-Air Update for Connected Cars, http://es.slideshare.net/chheplo/overtheair-ota-updates-and-the-connected-car

[8] A. S.A. Quadri, B. Othman Sidek, *"An Introduction to Over-the-Air Programming in Wireless Sensor Network"*, International Journal of Computer Science and Network Solutions, Vol. 2, No. 2, ISSN 2345-3397, http://es.slideshare.net/reachquadri/what-is-over-the-air-programming

[9] Stefan Unterschutz, Volker Turau, *"Fail-Safe Over-The-Air Programming and Error Recovery in Wireless Networks"*, http://www.ti5.tu-harburg.de/publications/2012/WISES_OTAP.pdf

[10] AES – Symmetric Ciphers Online, http://aes.online-domain-tools.com/

[11] mcrypt: encryption/decryption library - Linux man page https://linux.die.net/man/3/mcrypt

[12] FreeRTOS official web page: http://www.freertos.org/

[13] STM32Cube Official web page: http://www.st.com/en/embedded-software/stm32cube-embedded-software.html?querycriteria=productId=LN1897

[14] UM1472 Manual Discovery kit with STM32F407VG MCU : http://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf

[15] HDRadio Mexico official web page: http://hdradio.com/mexico

APENDICE A. Convención de Nombres

They must
- Be as long as necessary,
- Be as short as possible,
- Be as self-explanatory as possible,
- Differ in the first 30 characters

Module prefix (0 or 2 to 8 characters)

Module prefix / Feature
▪ Module/Package name in upper case.
▪ No module prefix for function-local and system-relevant (global macros, global definitions) identifiers.
▪ The module prefix may not be used for individual elements of unions or structures, as it is already given in the name of the union/structure.
▪ Enumerated constants (for non-module-local lists) must use the module-prefix as they are treated like normal constants. Note: The module prefix in functions, variables and types is in upper case, in contrast to file names, where it is in lower case.

Scope (0 to 2 characters)

Scope	
0	function local, no module prefix
1	system-wide
2	module-global or file-local or function-static

Name (5 to 20 characters)

Name
▪ It must be in English!
▪ Always begin with a capital letter.
▪ Concatenated names are denoted by capitalizing each sub-word.
▪ Digits are allowed.
▪ Underscores are not allowed.
▪ Function names should be identified by a verb and an object-name
▪ Identifiers must be clearly distinguishable and not only differ in case.

Type identifier (1 to 7 characters)

Type Identifier	
Allocation identifier	
'n'	#define
't'	Typedef
'r'	Const
None	All other allocation areas
Data type identifier	
Prefix	Description
'p'	pointer to ...
'a'	Array of ...
'pa'	Pointer to array of ...
'ap'	Array of pointers to ...
'c'	char
's'	char[], Null terminated string
'u8'	uint8
'u16'	uint16
'u32'	uint32
'i8'	int8
'i16'	int16
'i32'	int32
'f'	float
'd'	double

Prefix	Description
'bo'	bool
'v'	void
'un'	union
'st'	structure
'en'	enum
'nen'	enum element
'bi8'	bit8
'bi16'	bit16
'bi32'	bit32
'bi'	Bit variable
'o'	C++ Class Object
'x'	Complex data types
'v'	volatile

Special Rules for Structs

Name (5 to 20 characters)

Type identifier (1 to 7 characters)

Example: typedef struct {
uint8 ui8SendRegister_Stat
uint32 ui32Data;

Suffix (optional)

Suffix	Meaning
_Alm	Alarm
_Cfg	Configuration constant etc
_Con	Confirmation
_Csr	Driver Cancel Service Routine
_Ctr	Counter
_Dbg	Debug support
_Flg	Flag
_Hdl	Handler
_Ind	Indication Routine e.g. for event handling.
_Iob	I/O block for drivers
_Isr	Driver Interrupt Service Routine
_Jmp	Jump (for debugging)
_Jsr	Job Service Routine for drivers
_Len	Length
_Mir	Mirror – Hardware register copy
_Msg	Message
_Msk	Mask
_Psr	Driver Power Fail Service Routine
_Req	Request Routine, e.g. a transmit request
_Ret	Function Return Label (for debugging)
_Stat	Status Flag
_Suf	Used for constants with suffix but without cast
_Tbl	Table
_Tim	Time
_Tmr	Timer
_Tsk	Task
_Tsr	Driver Timeout Service Routine

APENDICE B. Código Fuente

CANDrv.h

```
/*!
* \file     CANDrv.h
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
*****************/

#ifndef CAN_DRV_H_
#define CAN_DRV_H_

#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
CONSTANTS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
TYPEDEFS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
VARIABLES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
MACROS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
PROTOTYPES
*****************************************************************************************************
*****************/


/*****************************************************************************************************
* \brief      Function used to Initialize the CAN Service.
* \param      void
```

```
* \return     void
*******************************************************************************************/
void CANDrv_vInit(void);
/*******************************************************************************************
* \brief      Function used to test communication.
* \param      void
* \return     void
*******************************************************************************************/
void CANDrv_vTest(void);
/*******************************************************************************************
* \brief      Task used as message dispatcher.
* \param      Format used for tasks on freeRtos
* \return     void
*******************************************************************************************/
void CANDrv_vDispatcher_tsk(void const * argument);

#ifdef __cplusplus
}
#endif

#endif /* CAN_DRV_H_ */
/*******************************************************************************************
******************
END OF FILE
*******************************************************************************************
*****************/
```

CANDrv.c

```
/*!
* \file     CANDrv.c
* \brief
* \author   Isidro Santos
*/

/*******************************************************************************************
******************
INCLUDES
*******************************************************************************************
*****************/
#include "stm32f4xx_hal.h"
#include "Sys_LogTrace.h"
/*******************************************************************************************
******************
CONSTANTS - Constants used by this module
*******************************************************************************************
*****************/

/*******************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*******************************************************************************************
*****************/

/*******************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*******************************************************************************************
*****************/
CAN_HandleTypeDef  hcan1;
static uint8_t boState = 0;
/*******************************************************************************************
******************
MACROS - Macro definitions used by this module
*******************************************************************************************
*****************/
```

```
/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
******************/

/*****************************************************************************************************//**
* \brief      Unblock USB Reading Page
* \param[in]  pstMsg: pointer to message that it will be sent
* \return     >=0 - Success, <0 - Fail
*****************************************************************************************************/

/*****************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*****************************************************************************************************
******************/

void CANDrv_vInit(void)
{

    hcan1.Instance = CAN1;
    hcan1.Init.Prescaler = 44;
    hcan1.Init.Mode = CAN_MODE_NORMAL;
    hcan1.Init.SJW = CAN_SJW_2TQ;
    hcan1.Init.BS1 = CAN_BS1_5TQ;
    hcan1.Init.BS2 = CAN_BS2_2TQ;
    hcan1.Init.TTCM = DISABLE;
    hcan1.Init.ABOM = ENABLE;
    hcan1.Init.AWUM = DISABLE;
    hcan1.Init.NART = DISABLE;
    hcan1.Init.RFLM = DISABLE;
    hcan1.Init.TXFP = DISABLE;
    if (HAL_CAN_Init(&hcan1) != HAL_OK)
    {
        CAN_LOG_ERR("Init");
    }
    else
    {
        CAN_LOG_INF("CAN Init");
        boState=1;
    }

}


void CANDrv_vTest( void )
{
    CanTxMsgTypeDef Msg = {0};
    CanTxMsgTypeDef Msg1 = {0};
    unsigned int u8Ctr=0;


if (u8Ctr < 100)
{
    Msg.IDE = CAN_ID_STD;
    Msg.DLC = 0x03;
    Msg.StdId = 0x2c1; //
    Msg.RTR = CAN_RTR_DATA;
    Msg.Data[0] = 0x01;
    Msg.Data[1] = 0x00;
    Msg.Data[2] = 0x80;
    hcan1.pTxMsg = &Msg;
    HAL_CAN_Transmit(&hcan1,100);

}
else if ((u8Ctr>100) && (u8Ctr<200))
{
```

```
    Msg.IDE = CAN_ID_STD;
    Msg.DLC = 0x03;
    Msg.StdId = 0x2c1; //
    Msg.RTR = CAN_RTR_DATA;
    Msg.Data[0] = 0x04;
    Msg.Data[1] = 0x00;
    Msg.Data[2] = 0x80;
    hcan1.pTxMsg = &Msg;
    HAL_CAN_Transmit(&hcan1,100);
}
else
{
        u8Ctr = 0;
    Msg.IDE = CAN_ID_STD;
    Msg.DLC = 0x03;
    Msg.StdId = 0x2c1; //
    Msg.RTR = CAN_RTR_DATA;
    Msg.Data[0] = 0x00;
    Msg.Data[1] = 0x00;
    Msg.Data[2] = 0x80;
    hcan1.pTxMsg = &Msg;
    HAL_CAN_Transmit(&hcan1,100);
}
```

```
    Msg1.IDE = CAN_ID_STD;
    Msg1.DLC = 0x01;
    Msg1.StdId = 0x271; //Tester Present
    Msg1.RTR = CAN_RTR_DATA;
    Msg1.Data[0] = 0x07;
    hcan1.pTxMsg = &Msg1;
    HAL_CAN_Transmit(&hcan1,100);
}
```

```
void CANDrv_vDispatcher_tsk(void const * argument)
{
    while (1)
    {
        if(boState==1)
        {
                        CANDrv_vTest();
        }
                osDelay(100);
                osThreadYield();
    }
}
```

```
/*****************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
*****************************************************************************************************
*****************/
```

```
/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

GpioInterface.h

```
/*!
* \file     GpioInterface.h
```

```
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
*****************/

#ifndef GPIO_INTERFACE_H_
#define GPIO_INTERFACE_H_

#ifdef __cplusplus
extern "C" {
#endif


/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
CONSTANTS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
TYPEDEFS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
VARIABLES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
MACROS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
PROTOTYPES
*****************************************************************************************************
*****************/
/*****************************************************************************************************
* \brief      Gpio initialization
* \param      void
* \return     void
*****************************************************************************************************/
void Gpio_vInit(void);

#ifdef __cplusplus
}
#endif

#endif /* GPIO_INTERFACE_H_ */
/*****************************************************************************************************
******************
END OF FILE
```

```
*****************************************************************************************************
******************/
```

GpioInterface.c

```
/*!
* \file     GpioInterface.c
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
******************/
#include "GpioInterface.h"
#include "stm32f4xx_hal.h"
/*****************************************************************************************************
******************
CONSTANTS - Constants used by this module
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
MACROS - Macro definitions used by this module
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
******************/


/*****************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*****************************************************************************************************
******************/
void Gpio_vInit(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(CS_I2C_SPI_GPIO_Port, CS_I2C_SPI_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : CS_I2C_SPI_Pin */
  GPIO_InitStruct.Pin = CS_I2C_SPI_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(CS_I2C_SPI_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : OTG_FS_PowerSwitchOn_Pin */
  GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(OTG_FS_PowerSwitchOn_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : BOOT1_Pin */
  GPIO_InitStruct.Pin = BOOT1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(BOOT1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin */
  GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OTG_FS_OverCurrent_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : MEMS_INT2_Pin */
  GPIO_InitStruct.Pin = MEMS_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(MEMS_INT2_GPIO_Port, &GPIO_InitStruct);

}

/**************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
**************************************************************************************************
*****************/


/**************************************************************************************************
******************
END OF FILE
```

```
*****************************************************************************************************
******************/
```

UART2_Drv.h

```
/*!
* \file     UART2_Drv.h
* \brief    Module used as UART2 service for HDRadio protocol usage.
* \author   Isidro Santos
*/
/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
******************/
#ifndef UART_DRV_H_
#define UART_DRV_H_

#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
******************/
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"
/*****************************************************************************************************
******************
CONSTANTS
*****************************************************************************************************
******************/



/*****************************************************************************************************
******************
TYPEDEFS
*****************************************************************************************************
******************/

typedef void (*SCI2_tpvNotif)(void);

typedef enum SCI2_nenReturnCode
{
    SCI2_nenOK = 0x00     /*Code used for success            */
     ,SCI2_nenBuff_Full   /*Buffer of received datas is full */
     ,SCI2_nenError       /*General error                    */
} SCI2_tenReturnCode;

/*****************************************************************************************************
******************
VARIABLES
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
MACROS
*****************************************************************************************************
******************/
```

```
/*****************************************************************************************************
*******************
PROTOTYPES
*****************************************************************************************************
******************/

/*****************************************************************************************************
* \brief      Function used to initialize the UART 2.
* \param      void
* \return     void
*****************************************************************************************************/
void SCI2s_vInit( void );
/*****************************************************************************************************
* \brief      Function used to initialize the UART 2.
* \param[in]  pu8DataBuf:pointer to buffer with datas to be transmitted.
* \param[in]  u16Size:Size of data buffer to be transmitted.
* \return     void
*****************************************************************************************************/
void  SCI2s_vWrite(uint8_t *pu8DataBuf, uint16_t u16Size);
/*****************************************************************************************************
* \brief      Function used to initialize the UART 2.
* \param      void
* \return     uint8 - Number of datas in buffer.
*****************************************************************************************************/
uint8_t SCI2s_u8GetRXBuffLen(void);
/*****************************************************************************************************
* \brief      Function used to get the datas received.
* \param[out]  pu8InputVal: pointer to receive the data.
* \return     SCI2_tenReturnCode - return code.
*****************************************************************************************************/
SCI2_tenReturnCode SCI2s_vGetData(uint8_t* pu8InputVal);
/*****************************************************************************************************
* \brief      Function used get a notification when new data is received.
* \param[in]  pvNotif: function to be used as notification.
* \return     void
*****************************************************************************************************/
void SCI2_vSubscriber(SCI2_tpvNotif pvNotif);
/*****************************************************************************************************
* \brief      Main task of UART2 service
* \param      Format used for tasks on freeRtos
* \return     void
*****************************************************************************************************/
void SCI2_vReceiveTask(void const * argument);

#ifdef __cplusplus
}
#endif

#endif /*UART_DRV_H_*/
/*****************************************************************************************************
*******************
END OF FILE
*****************************************************************************************************
******************/
```

UART2_Drv.c

```
/*!
* \file     UART2_Drv.c
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
*******************
INCLUDES
```

```
*****************************************************************************************************
*****************/
#include "UART2_Drv.h"
#include "Sys_LogTrace.h"
/*****************************************************************************************************
******************
CONSTANTS - Constants used by this module
*****************************************************************************************************
*****************/
#define SCI2__nPORT_CFG         USART2
#define SCI2__nBAUDRATE_CFG     115200
#define SCI2__nWORLDLEN_CFG     UART_WORDLENGTH_8B
#define SCI2__nSTOPBIT_CFG      UART_STOPBITS_1
#define SCI2__nPARITY_CFG       UART_PARITY_NONE
#define SCI2__nMODE_CFG         UART_MODE_TX_RX
#define SCI2__nHWFLOW_CTL_CFG   UART_HWCONTROL_NONE
#define SCI2__nOVER_SAMP_CFG    UART_OVERSAMPLING_16

/*****************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*****************************************************************************************************
*****************/
#define SCI2_nMAX_BUFFER_LEN  255

/*****************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*****************************************************************************************************
*****************/
static UART_HandleTypeDef     SCI2__stHndl               = {0};
static osThreadDef_t          SCI2__stTaksHndl           = {0};
static osThreadId             SCI2__tTaskID              = {0};
static uint8_t                SCI2__u8DataBuff[255]      = {0};
static uint8_t                SCI2__u8DataBuff_EndIndx   = 0;
static uint8_t                SCI2__u8DataBuff_StartIndx = 0;
/*****************************************************************************************************
******************
MACROS - Macro definitions used by this module
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
*****************/

/*****************************************************************************************************//**
* \brief      Unblock USB Reading Page
* \param[in]  pstMsg: pointer to message that it will be sent
* \return     >=0 - Success, <0 - Fail
*****************************************************************************************************/

static SCI2_tpvNotif SCI2__pvNotification_Cbk = NULL;

static void SCI2_vSetNotif(void);
/*****************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*****************************************************************************************************
*****************/


void SCI2s_vInit( void )
{
    SCI2__stHndl.Instance          = SCI2__nPORT_CFG;
    SCI2__stHndl.Init.BaudRate     = SCI2__nBAUDRATE_CFG;
    SCI2__stHndl.Init.WordLength   = SCI2__nWORLDLEN_CFG;
```

```
    SCI2__stHndl.Init.StopBits     = SCI2__nSTOPBIT_CFG;
    SCI2__stHndl.Init.Parity       = SCI2__nPARITY_CFG;
    SCI2__stHndl.Init.Mode         = SCI2__nMODE_CFG;
    SCI2__stHndl.Init.HwFlowCtl    = SCI2__nHWFLOW_CTL_CFG;
    SCI2__stHndl.Init.OverSampling = SCI2__nOVER_SAMP_CFG;

    HAL_UART_Init(&SCI2__stHndl);

}


void SCI2_vSubscriber(SCI2_tpvNotif pvNotif)
{
    if (NULL != pvNotif)
    {
        SCI2__pvNotification_Cbk=pvNotif;
    }
}



void SCI2s_vWrite(uint8_t *pu8DataBuf, uint16_t u16Size )
{
    HAL_UART_StateTypeDef enReturnVal;
    enReturnVal = HAL_UART_GetState(&SCI2__stHndl);

    if (enReturnVal == HAL_UART_STATE_READY)
    {
        HAL_UART_Transmit(&SCI2__stHndl,(uint8_t *)pu8DataBuf,(uint16_t)u16Size, (uint32_t)10);
    }
}

SCI2_tenReturnCode SCI2s_vGetData(uint8_t* pu8InputVal)
{
    SCI2_tenReturnCode enReturnVal = SCI2_nenError;

    if (SCI2__u8DataBuff_EndIndx != SCI2__u8DataBuff_StartIndx)
    {
        *pu8InputVal = SCI2__u8DataBuff[SCI2__u8DataBuff_StartIndx];
        enReturnVal = SCI2_nenOK;
        SCI2__u8DataBuff_StartIndx++;
        if (SCI2__u8DataBuff_StartIndx > SCI2_nMAX_BUFFER_LEN)
        {
            SCI2__u8DataBuff_StartIndx = 0;
        }
    }
    else
    {
        *pu8InputVal = 0;
    }
    return enReturnVal;
}

uint8_t SCI2s_u8GetRXBuffLen(void)
{
    uint8_t u8ReturnVal = 0;
    u8ReturnVal = SCI2__u8DataBuff_EndIndx - SCI2__u8DataBuff_StartIndx;
    return u8ReturnVal;
}

void SCI2_vReceiveTask(void const * argument)
{
    HAL_StatusTypeDef enResultVal = HAL_ERROR;

    while(1)
    {
```

```
        enResultVal =
HAL_UART_Receive(&SCI2__stHndl,&SCI2__u8DataBuff[SCI2__u8DataBuff_EndIndx],(uint16_t)1,
(uint32_t)10);
        if (enResultVal == HAL_OK)
        {
            if (SCI2__u8DataBuff_EndIndx < SCI2_nMAX_BUFFER_LEN)
            {
                SCI2_vSetNotif();
                SCI2__u8DataBuff_EndIndx++;
            }
            else
            {
                SCI2__u8DataBuff_EndIndx=0;
            }
        }

        osThreadYield();
    }
}

/**************************************************************************************************
*******************
PRIVATE FUNCTIONS - Functions accessed only by this module
**************************************************************************************************
*****************/

static void SCI2_vSetNotif(void)
{
    if (NULL != SCI2__pvNotification_Cbk)
    {
        SCI2__pvNotification_Cbk();
    }
}
```

```
/**************************************************************************************************
*******************
END OF FILE
**************************************************************************************************
*****************/
```

HDParser.h

```
/**************************************************************************************
*********************************
PREVENT REDUNDANT INCLUSION
**************************************************************************************************
*****************/
#ifndef HD_PROTOCOL_H
#define HD_PROTOCOL_H


#ifdef __cplusplus
extern "C" {
#endif

/**************************************************************************************************
*******************
INCLUDES
```

```
*****************************************************************************************************
*****************/
#include "HDTypes.h"
/*****************************************************************************************************
*******************
CONSTANTS
*****************************************************************************************************
*****************/

#define HDP_nDAT_BUFF_LEN             (20U) /*Buffer len value*/
/*****************************************************************************************************
*******************
TYPEDEFS
*****************************************************************************************************
*****************/

typedef enum HDP_nenMsgID
{
    HDP_enNewBroadcstMsg = 0x05 /* Broadcoasting Msg      */
    ,HDP_enAttrMsg              /* Attribute Msg          */
    ,HDP_enPkgMsg               /* Package Msg            */
    ,HDP_enMsgID_NB             /* Total of enum elements */
}HDP_tenMsgIDType;

typedef struct
{
    HDP_tenMsgIDType  enMsgID;                  /* Element used to know which kind of MSG is
allocated */
    uint8             u8SwKey;                  /* Element used to identify the broadcasting
session   */
    uint8             u8DataBuf[HDP_nDAT_BUFF_LEN]; /* Element contains the Msg data buffer
*/
}HDP_tstDataFramType;
/*****************************************************************************************************
*******************
VARIABLES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
*******************
MACROS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
*******************
PROTOTYPES
*****************************************************************************************************
*****************/


/*****************************************************************************************************
* \brief      Main task of UART2 service
* \param      void
* \return     void
*****************************************************************************************************/
void  HDP_vInit(void);
/*****************************************************************************************************
* \brief      Function used to deinitialize the parser.
* \param      void
* \return     void
*****************************************************************************************************/
void  HDP_vDeInit(void);
/*****************************************************************************************************
* \brief      Function used to get notifications.
* \param      void
* \return     void
*****************************************************************************************************/
```

```
void  HDP_vNewData_nofit(void);
/*****************************************************************************************
* \brief      Main task
* \param      Format used for tasks on freeRtos
* \return     void
*****************************************************************************************/
void  HDP_vMsg_tsk(void const * argument);
/*****************************************************************************************
* \brief      Function used to get the message received.
* \param[out] pstNewMsg: pointer used to get the message.
* \return     void
*****************************************************************************************/
void  HDP_vGetMsg(HDP_tstDataFramType *pstNewMsg);
/*****************************************************************************************
* \brief      Function used to get the number of messages on buffer.
* \param      Format used for tasks on freeRtos
* \return     uint8
*****************************************************************************************/
uint8 HDP_u8GetMsgBuffLen(void);

#ifdef __cplusplus
}
#endif

#endif
/*****************************************************************************************
********************
END OF FILE
*****************************************************************************************
******************/
```

HDParser.c

```
/*****************************************************************************************
********************
INCLUDES
*****************************************************************************************
******************/
#include "Sys_LogTrace.h"
#include "HDPorting.h"
#include "HDParser.h"
#include "string.h"

/*****************************************************************************************
********************
CONSTANTS - Constants used by this module
*****************************************************************************************
******************/
#define HDP__nu8NEW_DATA_EVNT          (0x01U)


#define HDP__nMAX_SYNC_LEN             (4U)
#define HDP__nMSG_BUF_LEN              (10U)

const uint8 HDP__nFrameSync[HDP__nMAX_SYNC_LEN] = {0x00,0x00,0x55,0x55};
/*****************************************************************************************
********************
TYPEDEFS - Local type definitions used by this module
*****************************************************************************************
******************/
typedef void(*pvAction_Cbk)(uint8 u8Action);


typedef enum HDP__nenMainStates
{
    HDP__nenIDLEState = 0x00
    ,HDP__nenFramIdent
```

```
    ,HDP__nenNewBroad
    ,HDP__nenAttrMsg
    ,HDP__nenPkgMsg
    ,HDP__nenMainStts_NB
}HDP__tenMainStates;


typedef struct
{
    HDP__tenMainStates    enMainState;  /*!< Element used to know current state  */
    pvAction_Cbk          pvSateHndl;    /*!< Handler used for current state  */
}HDP__tstMainStateFlow;


typedef enum HDP__nenProtcolStatus
{
    HDP__nenNotInit = 0x00
    ,HDP__nenInitialized
}HDP__tenProtcolStatus;


/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
*****************/

/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************/
static void HDP__vIDLEStte_hdl(uint8 u8Evnt);
/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************/
static void HDP__vFramIdentStte_hdl(uint8 u8Evnt);
/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************/
static void HDP__vNewBrdcstStte_hdl(uint8 u8Evnt);
/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************/
static void HDP__vAttrMsgStte_hdl(uint8 u8Evnt);
/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************/
static void HDP__vPkgMsgStte_hdl(uint8 u8Evnt);
/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************/
static uint8 HDP__boValiteSync(uint8 *pu8Sync);
/*****************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
```

```
* \return
*******************************************************************************************************/
static void HDP__vSetNewState(HDP__tenMainStates enNewState);
/*******************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*******************************************************************************************************/
static void HDP__vAddMsg(HDP_tstDataFramType *NewMsg);
/*******************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*******************************************************************************************************/
static uint16 HDP__u16CalcAttrLen(uint8 u8Byte1, uint8 u8Byte2);

/*******************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*******************************************************************************************************
******************/
const HDP__tstMainStateFlow HDP__astMainStateFlow[] =
{
    /*   enMainState          |       pvSateHndl        */
    { HDP__nenIDLEState      , &HDP__vIDLEStte_hdl    }
    ,{ HDP__nenFramIdent     , &HDP__vFramIdentStte_hdl}
    ,{ HDP__nenNewBroad      , &HDP__vNewBrdcstStte_hdl}
    ,{ HDP__nenAttrMsg       , &HDP__vAttrMsgStte_hdl  }
    ,{ HDP__nenPkgMsg        , &HDP__vPkgMsgStte_hdl   }
};

static HDP__tstMainStateFlow HDP__stMachineContext = { 0 };
static HDP__tenProtcolStatus HDP__enProtclStatus   = HDP__nenNotInit;
static HDP_tstDataFramType   HDP__astMsgBuffer[HDP__nMSG_BUF_LEN] ={0};
static uint8              HDP__u8CurrentIndx = 0;
static uint8              HDP__u8StartIndx = 0;
static HDP_tstDataFramType   HDP__stCurrentMsg_tmp     = {0};
/*******************************************************************************************************
******************
MACROS - Macro definitions used by this module
*******************************************************************************************************
******************/
#define HDP__au8MainStates_NB      (uint8)(sizeof(HDP__astMainStateFlow) /
sizeof(HDP__astMainStateFlow[0]))  /*!< COMMENT  */


/*******************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*******************************************************************************************************
******************/

void HDP_vInit(void)
{
    if (HDP__nenNotInit == HDP__enProtclStatus)
    {
        HDP__enProtclStatus = HDP__nenInitialized;
        HDP_LOG_INF("Module Init");
        HDP__stMachineContext.enMainState     =
HDP__astMainStateFlow[HDP__nenIDLEState].enMainState;
        HDP__stMachineContext.pvSateHndl      = HDP__astMainStateFlow[HDP__nenIDLEState].pvSateHndl;
    }
    else
    {
        HDP_LOG_ERR("Module already Init");
    }

}
void HDP_vDeInit(void)
```

```
{

}

void HDP_vNewData_nofit(void)
{
    HDP_LOG_INF("New data notification");
    if (NULL != HDP__stMachineContext.pvSateHndl)
    {
        HDP__stMachineContext.pvSateHndl(HDP__nu8NEW_DATA_EVNT);
    }
}


void HDP_vMsg_tsk(void const * argument)
{
    while(True)
    {
        if (HDPortSCI_u8CheckRx() > 0)
        {
            HDP_vNewData_nofit();
        }
        osDelay(10);
        osThreadYield();
    }
}


void HDP_vGetMsg(HDP_tstDataFramType *pstNewMsg)
{
    if (NULL != pstNewMsg)
    {
        memmove((const void *)pstNewMsg, (void *)&HDP__astMsgBuffer[HDP__u8StartIndx],
sizeof(HDP_tstDataFramType));
        HDP__u8StartIndx++;
        if (HDP__u8StartIndx >= HDP__nMSG_BUF_LEN)
        {
            HDP__u8StartIndx = 0;
        }
        HDP_LOG_INF("Msg LoadedID[%d]", pstNewMsg->enMsgID);
    }
}


uint8 HDP_u8GetMsgBuffLen(void)
{
    uint8 u8ReturnVal = 0;

    u8ReturnVal = HDP__u8CurrentIndx - HDP__u8StartIndx;

    return u8ReturnVal;
}

/*****************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
*****************************************************************************************************
*****************/


static void HDP__vIDLEStte_hdl(uint8 u8Evnt)
{
    static  uint8 au8Sync_tmp[HDP__nMAX_SYNC_LEN] = {0};
    static  uint8 u8Byte_ctr = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
```

```
            if (HDPortSCI_u8CheckRx() > 0)
            {
                au8Sync_tmp[u8Byte_ctr]= HDPortSCI_u8ReadRx();
                HDP_LOG_INF("New Data[0x%x] u8Byte_ctr[%d]", au8Sync_tmp[u8Byte_ctr], u8Byte_ctr);
                HDP_LOG_INF("MsgTotal[%d]", HDP_u8GetMsgBuffLen());
                u8Byte_ctr++;

                if (u8Byte_ctr >= HDP__nMAX_SYNC_LEN)
                {
                    if (True == HDP__boValiteSync((uint8 *)&au8Sync_tmp[0]))
                    {
                        HDP__vSetNewState(HDP__nenFramIdent);
                    }
                    u8Byte_ctr = 0;

                }
                else if ((1 == u8Byte_ctr) && ( 0x00 != au8Sync_tmp[0]))
                {
                    u8Byte_ctr = 0;
                    HDP_LOG_ERR("Wrong Sync");
                }
            }
            else
            {
                HDP_LOG_ERR("No data available");
            }

        }break;
        default:
        {
        }break;
    }
}



static void HDP__vFramIdentStte_hdl(uint8 u8Evnt)
{
    uint8 u8Data = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                u8Data = HDPortSCI_u8ReadRx();
                HDP_LOG_INF("NewData[0x%x]", u8Data);
                switch(u8Data)
                {
                    case HDP_enNewBroadcstMsg:
                    {
                        HDP_LOG_INF("HDP_enNewBroadcstMsg");
                        HDP__stCurrentMsg_tmp.enMsgID = HDP_enNewBroadcstMsg;
                        HDP__vSetNewState(HDP__nenNewBroad);
                    }break;
                    case HDP_enAttrMsg:
                    {
                        HDP_LOG_INF("HDP_enAttrMsg");
                        HDP__stCurrentMsg_tmp.enMsgID = HDP_enAttrMsg;
                        HDP__vSetNewState(HDP__nenAttrMsg);

                    }break;
                    case HDP_enPkgMsg:
                    {
                        HDP_LOG_INF("HDP_enPkgMsg");
                        HDP__stCurrentMsg_tmp.enMsgID = HDP_enPkgMsg;
                        HDP__vSetNewState(HDP__nenPkgMsg);
```

```
                    }break;
                    default:
                    {
                        HDP_LOG_ERR("MsgID was not identified");
                        HDP__vSetNewState(HDP__nenIDLEState);
                    }
                }
            }
        }break;
        default:
        {
        }break;
    }
}

static void HDP__vNewBrdcstStte_hdl(uint8 u8Evnt)
{
    static uint8 u8Data_ctr = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = HDPortSCI_u8ReadRx();
                u8Data_ctr++;
                if (u8Data_ctr >= 7)
                {
                    HDP_LOG_INF("broad receibed");
                    HDP__vAddMsg(&HDP__stCurrentMsg_tmp);
                    HDP__vSetNewState(HDP__nenIDLEState);
                    u8Data_ctr = 0;
                }
            }
        }break;
        default:
        {
        }break;
    }
}

static void HDP__vPkgMsgStte_hdl(uint8 u8Evnt)
{
    static uint8 u8Data_ctr = 0;
    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                if (0 == u8Data_ctr)
                {
                    HDP__stCurrentMsg_tmp.u8SwKey = HDPortSCI_u8ReadRx();
                }
                else
                {
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr-1] = HDPortSCI_u8ReadRx();
                }
                u8Data_ctr++;
                if (u8Data_ctr >= 11)
                {
                    HDP_LOG_INF("PkgMsg Received");
                    u8Data_ctr = 0;
                    HDP__vAddMsg(&HDP__stCurrentMsg_tmp);
                    HDP__vSetNewState(HDP__nenIDLEState);
                }
```

```
            }
        }break;
        default:
        {
        }break;
    }

}

static void HDP__vAttrMsgStte_hdl(uint8 u8Evnt)
{
    static uint8 u8Byte_ctr = 0;
    static uint8 u8Data_ctr = 0;
    static uint8 u8LengthBtyte1 = 0;
    static uint8 u8LengthBtyte2 = 0;
    static uint16 u16ParamMount = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                if (0 == u8Byte_ctr)
                {
                    HDP__stCurrentMsg_tmp.u8SwKey = HDPortSCI_u8ReadRx();
                    u8Byte_ctr++;
                }
                else if (2 == u8Byte_ctr)
                {
                    u8LengthBtyte1 = HDPortSCI_u8ReadRx();
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = u8LengthBtyte1;
                    u8Byte_ctr++;
                    u8Data_ctr++;
                }
                else if (3 == u8Byte_ctr)
                {
                    u8LengthBtyte2 = HDPortSCI_u8ReadRx();
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = u8LengthBtyte2;
                    u16ParamMount = HDP__u16CalcAttrLen(u8LengthBtyte1, u8LengthBtyte2);
                    HDP_LOG_DEBUG("u16ParamMount[%d]", u16ParamMount);
                    u8Byte_ctr++;
                    u8Data_ctr++;
                }
                else
                {
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = HDPortSCI_u8ReadRx();
                    u8Data_ctr++;
                    u8Byte_ctr++;
                    if (u8Data_ctr >= (u16ParamMount + 4))
                    {

                        HDP_LOG_INF("Attri Received");
                        HDP__vAddMsg(&HDP__stCurrentMsg_tmp);
                        HDP__vSetNewState(HDP__nenIDLEState);
                        u8Data_ctr = 0;
                        u8Byte_ctr = 0;
                    }

                }

            }
        }break;
        default:
        {
        }break;
    }

}
```

```
static uint8 HDP__boValiteSync(uint8 *pu8Sync)
{
    uint8 boReturnVal = False;
    uint8 u8Data_ctr = 0;
    uint8 *pu8TempPtr = pu8Sync;
    uint8 u8ValidData = 0;

    for (u8Data_ctr = 0; u8Data_ctr < 4;u8Data_ctr ++)
    {
        if (HDP__nFrameSync[u8Data_ctr] == *pu8TempPtr)
        {
            u8ValidData++;
            pu8TempPtr++;
        }
        else
        {
            HDP_LOG_ERR("Wrong Sync");
            break;
        }
    }
    if (u8ValidData == 4)
    {
        HDP_LOG_INF("ValidSync.");
        boReturnVal = True;
    }
    return boReturnVal;
}

static void HDP__vSetNewState(HDP__tenMainStates enNewState)
{
    if (enNewState < HDP__nenMainStts_NB)
    {
        HDP__stMachineContext.enMainState     = HDP__astMainStateFlow[enNewState].enMainState;
        HDP__stMachineContext.pvSateHndl      = HDP__astMainStateFlow[enNewState].pvSateHndl;
        if (HDP__nenIDLEState == enNewState)
        {
            memset(&HDP__stCurrentMsg_tmp,0x00,sizeof(HDP__stCurrentMsg_tmp)); /*Used to reset
temporal Msg*/
        }
    }
    else
    {
        HDP_LOG_ERR("State not allowed.");
    }
}

static void HDP__vAddMsg(HDP_tstDataFramType *NewMsg)
{
    if (NULL != NewMsg)
    {
        memmove((void *)&HDP__astMsgBuffer[HDP__u8CurrentIndx], (const void *)NewMsg,
sizeof(HDP_tstDataFramType));
        HDP__u8CurrentIndx++;
        if (HDP__u8CurrentIndx >= HDP__nMSG_BUF_LEN)
        {
            HDP__u8CurrentIndx = 0;
        }
        HDP_LOG_INF("Msg Saved type[%d]",NewMsg->enMsgID);
    }
    else
    {
        HDP_LOG_ERR("Null mesagge.");
    }
}

static uint16 HDP__u16CalcAttrLen(uint8 u8Byte1, uint8 u8Byte2)
```

```
{
    uint16 u16ResultVal = 0;

    u16ResultVal = u8Byte1 << 8;
    u16ResultVal = u16ResultVal | u8Byte2;

    return u16ResultVal;
}

/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDPorting.h

```
/*!
* \file      HDPorting.h
* \brief
* \author    Isidro Santos
*/

/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
*****************/

#ifndef HD_PORTING_H
#define HD_PORTING_H


#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/

#include "HDTypes.h"


/*****************************************************************************************************
******************
CONSTANTS
*****************************************************************************************************
*****************/

#undef  HD_nMC9S12_MCU
#define HD_nSTM32F407

/*****************************************************************************************************
******************
TYPEDEFS
*****************************************************************************************************
*****************/
```

```
typedef void (*HD_tpvNotif)(void);


/*****************************************************************************************************
******************
VARIABLES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
MACROS
*****************************************************************************************************
*****************/

#define SCPIPORT_TMOUT 500 /**/


/*****************************************************************************************************
******************
PUBLIC FUNCTIONS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
* \brief      Used to initialize the Serial port
* \param[in]  pvNotif: notification function
* \return     void
*****************************************************************************************************/
void  HDPortSCI_vInit( HD_tpvNotif pvNotif );
/*****************************************************************************************************
* \brief      function used to get the number of received datas in buffer.
* \param      void
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8CheckRx( void );
/*****************************************************************************************************
* \brief      function to read the buffer.
* \param      void
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8ReadRx( void );

#ifdef HD_nMC9S12_MCU
/*****************************************************************************************************
* \brief      Function used to send a data by UART
* \param[in]  u8SCIPort: port to be used
* \param[in]  pu8DataBuffer: pointer to buffer
* \param[in]  u8Size: size of buffer to be transmitted
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8WriteTx(uint8 u8SCIPort ,uint8 *pu8DataBuffer ,uint8 u8Size);
#endif
#ifdef HD_nSTM32F407
/*****************************************************************************************************
* \brief      function to read the buffer.
* \param[in]  pu8DataBuffer: pointer to buffer
* \param[in]  u16Size: size of buffer to be transmitted
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8WriteTx(uint8 *pu8DataBuffer ,uint16 u16Size);
#endif

#ifdef __cplusplus
}
```

```
#endif

#endif  /* HD_PORTING_H */
/**************************************************************************************************
******************
END OF FILE
**************************************************************************************************
*****************/
```

HDPorting.c

```
/**************************************************************************************************
******************
INCLUDES
**************************************************************************************************
*****************/
#include "Sys_LogTrace.h"
#include "HDPorting.h"
#include "HDParser.h"
#include "string.h"

/**************************************************************************************************
******************
CONSTANTS - Constants used by this module
**************************************************************************************************
*****************/
#define HDP__nu8NEW_DATA_EVNT          (0x01U)

#define HDP__nMAX_SYNC_LEN             (4U)
#define HDP__nMSG_BUF_LEN              (10U)

const uint8 HDP__nFrameSync[HDP__nMAX_SYNC_LEN] = {0x00,0x00,0x55,0x55};
/**************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
**************************************************************************************************
*****************/
typedef void(*pvAction_Cbk)(uint8 u8Action);

typedef enum HDP__nenMainStates
{
    HDP__nenIDLEState = 0x00
    ,HDP__nenFramIdent
    ,HDP__nenNewBroad
    ,HDP__nenAttrMsg
    ,HDP__nenPkgMsg
    ,HDP__nenMainStts_NB
}HDP__tenMainStates;


typedef struct
{
    HDP__tenMainStates    enMainState;  /*!< Element used to know current state  */
    pvAction_Cbk          pvSateHnd1;    /*!< Handler used for current state  */
}HDP__tstMainStateFlow;


typedef enum HDP__nenProtcolStatus
{
    HDP__nenNotInit = 0x00
    ,HDP__nenInitialized
}HDP__tenProtcolStatus;
```

```
/*****************************************************************************************************
*******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
*****************/

/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vIDLEStte_hdl(uint8 u8Evnt);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vFramIdentStte_hdl(uint8 u8Evnt);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vNewBrdcstStte_hdl(uint8 u8Evnt);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vAttrMsgStte_hdl(uint8 u8Evnt);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vPkgMsgStte_hdl(uint8 u8Evnt);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static uint8 HDP__boValiteSync(uint8 *pu8Sync);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vSetNewState(HDP__tenMainStates enNewState);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static void HDP__vAddMsg(HDP_tstDataFramType *NewMsg);
/*****************************************************************************************************//**
* \brief      Handler for menu options
* \param[in]  u8Action: Action to be performed
* \return
*****************************************************************************************************/
static uint16 HDP__u16CalcAttrLen(uint8 u8Byte1, uint8 u8Byte2);

/*****************************************************************************************************
*******************
LOCAL VARIABLES - Module global variables
*****************************************************************************************************
*****************/
const HDP__tstMainStateFlow HDP__astMainStateFlow[] =
```

```
{
    /*   enMainState          |       pvSateHndl        */
    { HDP__nenIDLEState      , &HDP__vIDLESttе_hdl    }
    ,{ HDP__nenFramIdent     , &HDP__vFramIdentStte_hdl}
    ,{ HDP__nenNewBroad      , &HDP__vNewBrdcstStte_hdl}
    ,{ HDP__nenAttrMsg       , &HDP__vAttrMsgStte_hdl  }
    ,{ HDP__nenPkgMsg        , &HDP__vPkgMsgStte_hdl   }
};

static HDP__tstMainStateFlow HDP__stMachineContext = { 0 };
static HDP__tenProtcolStatus HDP__enProtclStatus   = HDP__nenNotInit;
static HDP_tstDataFramType   HDP__astMsgBuffer[HDP__nMSG_BUF_LEN] ={0};
static uint8               HDP__u8CurrentIndx = 0;
static uint8               HDP__u8StartIndx = 0;
static HDP_tstDataFramType   HDP__stCurrentMsg_tmp     = {0};
/**********************************************************************************************************
******************
MACROS - Macro definitions used by this module
**********************************************************************************************************
*****************/
#define HDP__au8MainStates_NB      (uint8)(sizeof(HDP__astMainStateFlow) /
sizeof(HDP__astMainStateFlow[0]))  /*!< COMMENT  */


/**********************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
**********************************************************************************************************
*****************/

void HDP_vInit(void)
{
    if (HDP__nenNotInit == HDP__enProtclStatus)
    {
        HDP__enProtclStatus = HDP__nenInitialized;
        HDP_LOG_INF("Module Init");
        HDP__stMachineContext.enMainState      =
HDP__astMainStateFlow[HDP__nenIDLEState].enMainState;
        HDP__stMachineContext.pvSateHndl       = HDP__astMainStateFlow[HDP__nenIDLEState].pvSateHndl;
    }
    else
    {
        HDP_LOG_ERR("Module already Init");
    }

}
void HDP_vDeInit(void)
{

}

void HDP_vNewData_nofit(void)
{
    HDP_LOG_INF("New data notification");
    if (NULL != HDP__stMachineContext.pvSateHndl)
    {
        HDP__stMachineContext.pvSateHndl(HDP__nu8NEW_DATA_EVNT);
    }
}


void HDP_vMsg_tsk(void const * argument)
{
    while(True)
    {
        if (HDPortSCI_u8CheckRx() > 0)
        {
            HDP_vNewData_nofit();
        }
```

```
        osDelay(10);
        osThreadYield();
    }
}

void HDP_vGetMsg(HDP_tstDataFramType *pstNewMsg)
{
    if (NULL != pstNewMsg)
    {
        memmove((const void *)pstNewMsg, (void *)&HDP__astMsgBuffer[HDP__u8StartIndx],
sizeof(HDP_tstDataFramType));
        HDP__u8StartIndx++;
        if (HDP__u8StartIndx >= HDP__nMSG_BUF_LEN)
        {
            HDP__u8StartIndx = 0;
        }
        HDP_LOG_INF("Msg LoadedID[%d]", pstNewMsg->enMsgID);
    }
}

uint8 HDP_u8GetMsgBuffLen(void)
{
    uint8 u8ReturnVal = 0;

    u8ReturnVal = HDP__u8CurrentIndx - HDP__u8StartIndx;

    return u8ReturnVal;
}

/**********************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
**********************************************************************************************************
*****************/

static void HDP__vIDLEStte_hdl(uint8 u8Evnt)
{
    static  uint8 au8Sync_tmp[HDP__nMAX_SYNC_LEN] = {0};
    static  uint8 u8Byte_ctr = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                au8Sync_tmp[u8Byte_ctr]= HDPortSCI_u8ReadRx();
                HDP_LOG_INF("New Data[0x%x] u8Byte_ctr[%d]", au8Sync_tmp[u8Byte_ctr], u8Byte_ctr);
                HDP_LOG_INF("MsgTotal[%d]", HDP_u8GetMsgBuffLen());
                u8Byte_ctr++;

                if (u8Byte_ctr >= HDP__nMAX_SYNC_LEN)
                {
                    if (True == HDP__boValiteSync((uint8 *)&au8Sync_tmp[0]))
                    {
                        HDP__vSetNewState(HDP__nenFramIdent);
                    }
                    u8Byte_ctr = 0;

                }
                else if ((1 == u8Byte_ctr) && ( 0x00 != au8Sync_tmp[0]))
                {
                    u8Byte_ctr = 0;
                    HDP_LOG_ERR("Wrong Sync");
                }
            }
```

```
            else
            {
                HDP_LOG_ERR("No data available");
            }

        }break;
        default:
        {
        }break;
    }
}


static void HDP__vFramIdentStte_hdl(uint8 u8Evnt)
{
    uint8 u8Data = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                u8Data = HDPortSCI_u8ReadRx();
                HDP_LOG_INF("NewData[0x%x]", u8Data);
                switch(u8Data)
                {
                    case HDP_enNewBroadcstMsg:
                    {
                        HDP_LOG_INF("HDP_enNewBroadcstMsg");
                        HDP__stCurrentMsg_tmp.enMsgID = HDP_enNewBroadcstMsg;
                        HDP__vSetNewState(HDP__nenNewBroad);
                    }break;
                    case HDP_enAttrMsg:
                    {
                        HDP_LOG_INF("HDP_enAttrMsg");
                        HDP__stCurrentMsg_tmp.enMsgID = HDP_enAttrMsg;
                        HDP__vSetNewState(HDP__nenAttrMsg);

                    }break;
                    case HDP_enPkgMsg:
                    {
                        HDP_LOG_INF("HDP_enPkgMsg");
                        HDP__stCurrentMsg_tmp.enMsgID = HDP_enPkgMsg;
                        HDP__vSetNewState(HDP__nenPkgMsg);
                    }break;
                    default:
                    {
                        HDP_LOG_ERR("MsgID was not identified");
                        HDP__vSetNewState(HDP__nenIDLEState);
                    }
                }
            }
        }break;
        default:
        {
        }break;
    }
}

static void HDP__vNewBrdcstStte_hdl(uint8 u8Evnt)
{
    static uint8 u8Data_ctr = 0;

    switch (u8Evnt)
    {
```

```
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = HDPortSCI_u8ReadRx();
                u8Data_ctr++;
                if (u8Data_ctr >= 7)
                {
                    HDP_LOG_INF("broad receibed");
                    HDP__vAddMsg(&HDP__stCurrentMsg_tmp);
                    HDP__vSetNewState(HDP__nenIDLEState);
                    u8Data_ctr = 0;
                }
            }
        }break;
        default:
        {
        }break;
    }
}

static void HDP__vPkgMsgStte_hdl(uint8 u8Evnt)
{
    static uint8 u8Data_ctr = 0;
    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
            {
                if (0 == u8Data_ctr)
                {
                    HDP__stCurrentMsg_tmp.u8SwKey = HDPortSCI_u8ReadRx();
                }
                else
                {
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr-1] = HDPortSCI_u8ReadRx();
                }
                u8Data_ctr++;
                if (u8Data_ctr >= 11)
                {
                    HDP_LOG_INF("PkgMsg Received");
                    u8Data_ctr = 0;
                    HDP__vAddMsg(&HDP__stCurrentMsg_tmp);
                    HDP__vSetNewState(HDP__nenIDLEState);
                }
            }
        }break;
        default:
        {
        }break;
    }

}

static void HDP__vAttrMsgStte_hdl(uint8 u8Evnt)
{
    static uint8 u8Byte_ctr = 0;
    static uint8 u8Data_ctr = 0;
    static uint8 u8LengthBtyte1 = 0;
    static uint8 u8LengthBtyte2 = 0;
    static uint16 u16ParamMount = 0;

    switch (u8Evnt)
    {
        case HDP__nu8NEW_DATA_EVNT:
        {
            if (HDPortSCI_u8CheckRx() > 0)
```

```
            {
                if (0 == u8Byte_ctr)
                {
                    HDP__stCurrentMsg_tmp.u8SwKey = HDPortSCI_u8ReadRx();
                    u8Byte_ctr++;
                }
                else if (2 == u8Byte_ctr)
                {
                    u8LengthBtyte1 = HDPortSCI_u8ReadRx();
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = u8LengthBtyte1;
                    u8Byte_ctr++;
                    u8Data_ctr++;
                }
                else if (3 == u8Byte_ctr)
                {
                    u8LengthBtyte2 = HDPortSCI_u8ReadRx();
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = u8LengthBtyte2;
                    u16ParamMount = HDP__u16CalcAttrLen(u8LengthBtyte1, u8LengthBtyte2);
                    HDP_LOG_DEBUG("u16ParamMount[%d]", u16ParamMount);
                    u8Byte_ctr++;
                    u8Data_ctr++;
                }
                else
                {
                    HDP__stCurrentMsg_tmp.u8DataBuf[u8Data_ctr] = HDPortSCI_u8ReadRx();
                    u8Data_ctr++;
                    u8Byte_ctr++;
                    if (u8Data_ctr >= (u16ParamMount + 4))
                    {

                        HDP_LOG_INF("Attri Received");
                        HDP__vAddMsg(&HDP__stCurrentMsg_tmp);
                        HDP__vSetNewState(HDP__nenIDLEState);
                        u8Data_ctr = 0;
                        u8Byte_ctr = 0;
                    }

                }

            }
        }break;
        default:
        {
        }break;
    }

}

static uint8 HDP__boValiteSync(uint8 *pu8Sync)
{
    uint8 boReturnVal = False;
    uint8 u8Data_ctr = 0;
    uint8 *pu8TempPtr = pu8Sync;
    uint8 u8ValidData = 0;

    for (u8Data_ctr = 0; u8Data_ctr < 4;u8Data_ctr ++)
    {
        if (HDP__nFrameSync[u8Data_ctr] == *pu8TempPtr)
        {
            u8ValidData++;
            pu8TempPtr++;
        }
        else
        {
            HDP_LOG_ERR("Wrong Sync");
            break;
        }
    }
    if (u8ValidData == 4)
```

```
    {
        HDP_LOG_INF("ValidSync.");
        boReturnVal = True;
    }
    return boReturnVal;
}

static void HDP__vSetNewState(HDP__tenMainStates enNewState)
{
    if (enNewState < HDP__nenMainStts_NB)
    {
        HDP__stMachineContext.enMainState     = HDP__astMainStateFlow[enNewState].enMainState;
        HDP__stMachineContext.pvSateHndl      = HDP__astMainStateFlow[enNewState].pvSateHndl;
        if (HDP__nenIDLEState == enNewState)
        {
            memset(&HDP__stCurrentMsg_tmp,0x00,sizeof(HDP__stCurrentMsg_tmp)); /*Used to reset
temporal Msg*/
        }
    }
    else
    {
        HDP_LOG_ERR("State not allowed.");
    }
}

static void HDP__vAddMsg(HDP_tstDataFramType *NewMsg)
{
    if (NULL != NewMsg)
    {
        memmove((void *)&HDP__astMsgBuffer[HDP__u8CurrentIndx], (const void *)NewMsg,
sizeof(HDP_tstDataFramType));
        HDP__u8CurrentIndx++;
        if (HDP__u8CurrentIndx >= HDP__nMSG_BUF_LEN)
        {
            HDP__u8CurrentIndx = 0;
        }
        HDP_LOG_INF("Msg Saved type[%d]",NewMsg->enMsgID);
    }
    else
    {
        HDP_LOG_ERR("Null mesagge.");
    }
}

static uint16 HDP__u16CalcAttrLen(uint8 u8Byte1, uint8 u8Byte2)
{
    uint16 u16ResultVal = 0;

    u16ResultVal = u8Byte1 << 8;
    u16ResultVal = u16ResultVal | u8Byte2;

    return u16ResultVal;
}

/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDPorting.h

```
/*!
* \file      HDPorting.h
* \brief
```

```
* \author   Isidro Santos
*/

/*************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*************************************************************************************************
*****************/

#ifndef HD_PORTING_H
#define HD_PORTING_H


#ifdef __cplusplus
extern "C" {
#endif

/*************************************************************************************************
******************
INCLUDES
*************************************************************************************************
*****************/

#include "HDTypes.h"


/*************************************************************************************************
******************
CONSTANTS
*************************************************************************************************
*****************/

#undef  HD_nMC9S12_MCU
#define HD_nSTM32F407

/*************************************************************************************************
******************
TYPEDEFS
*************************************************************************************************
*****************/

typedef void (*HD_tpvNotif)(void);


/*************************************************************************************************
******************
VARIABLES
*************************************************************************************************
*****************/

/*************************************************************************************************
******************
MACROS
*************************************************************************************************
*****************/

#define SCPIPORT_TMOUT 500 /**/


/*************************************************************************************************
******************
PUBLIC FUNCTIONS
```

```
*****************************************************************************************************
*****************/

/*****************************************************************************************************
* \brief      Used to initialize the Serial port
* \param[in]  pvNotif: notification function
* \return     void
*****************************************************************************************************/
void  HDPortSCI_vInit( HD_tpvNotif pvNotif );
/*****************************************************************************************************
* \brief      function used to get the number of received datas in buffer.
* \param      void
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8CheckRx( void );
/*****************************************************************************************************
* \brief      function to read the buffer.
* \param      void
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8ReadRx( void );

#ifdef HD_nMC9S12_MCU
/*****************************************************************************************************
* \brief      Function used to send a data by UART
* \param[in]  u8SCIPort: port to be used
* \param[in]  pu8DataBuffer: pointer to buffer
* \param[in]  u8Size: size of buffer to be transmitted
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8WriteTx(uint8 u8SCIPort ,uint8 *pu8DataBuffer ,uint8 u8Size);
#endif
#ifdef HD_nSTM32F407
/*****************************************************************************************************
* \brief      function to read the buffer.
* \param[in]  pu8DataBuffer: pointer to buffer
* \param[in]  u16Size: size of buffer to be transmitted
* \return     uint8
*****************************************************************************************************/
uint8 HDPortSCI_u8WriteTx(uint8 *pu8DataBuffer ,uint16 u16Size);
#endif

#ifdef __cplusplus
}
#endif

#endif  /* HD_PORTING_H */
/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDPorting.c

```
/*!
* \file     HDPorting.c
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/
```

```c
#include "HDPorting.h"

#ifdef HD_nMC9S12_MCU
#include "serial.h"
#endif
#ifdef HD_nSTM32F407
#include "UART2_Drv.h"
#endif

/*****************************************************************************************************
******************
CONSTANTS - Constants used by this module
*****************************************************************************************************
*****************/

#ifdef HD_nMC9S12_MCU
#define HDR_nSCI_CHANNEL (0x00U)
#endif
/*****************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
MACROS - Macro definitions used by this module
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
PUBLIC FUNCTIONS
*****************************************************************************************************
*****************/
void  HDPortSCI_vInit( HD_tpvNotif pvNotif )
{
#ifdef HD_nMC9S12_MCU
    vfnSCI_Init(&SCI_config[HDR_nSCI_CHANNEL]);
#endif
    if (NULL != pvNotif)
    {
        SCI2_vSubscriber((SCI2_tpvNotif)pvNotif);
    }
}

uint8 HDPortSCI_u8CheckRx( void )
{
    uint8 u8ReturnVal = 0x00;
#ifdef HD_nMC9S12_MCU
    u8ReturnVal = u8SCI_CheckRx(HDR_nSCI_CHANNEL);
#endif
#ifdef HD_nSTM32F407
```

```
    u8ReturnVal = SCI2s_u8GetRXBuffLen();
#endif
    return u8ReturnVal;
}

uint8 HDPortSCI_u8ReadRx( void )
{
    uint8 u8ReturnVal = 0x00;
#ifdef HD_nMC9S12_MCU
    u8ReturnVal = u8SCI_ReadRx(HDR_nSCI_CHANNEL);
#endif
#ifdef HD_nSTM32F407
    SCI2s_vGetData(&u8ReturnVal);
#endif
    return u8ReturnVal;
}


#ifdef HD_nMC9S12_MCU
uint8 HDPortSCI_u8WriteTx(uint8 u8SCIPort ,uint8 *pu8DataBuffer ,uint8 u8Size)
#endif
#ifdef HD_nSTM32F407
uint8 HDPortSCI_u8WriteTx(uint8 *pu8DataBuffer ,uint16 u16Size)
#endif
{
    uint8 u8ReturnVal = 0x00;
#ifdef HD_nMC9S12_MCU
    vfnSCI_WriteBufferTx(u8SCIPort, pu8DataBuffer, u8Size);
#endif
#ifdef HD_nSTM32F407
    SCI2s_vWrite((uint8_t *)pu8DataBuffer, (uint16_t)u16Size);
#endif

    return u8ReturnVal;
}


/*****************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
*****************************************************************************************************
*****************/


/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDApi.h

```
/*!
* \file      HDApi.h
* \brief
* \author    Isidro Santos
*/

/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
*****************/
```

```
#ifndef HD_API_H
#define HD_API_H

#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
CONSTANTS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
TYPEDEFS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
VARIABLES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
MACROS
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
PROTOTYPES
*****************************************************************************************************
*****************/

#ifdef __cplusplus
}
#endif

#endif  /* HD_API_H */
/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDMain.h

```
/*!
* \file     HDMain.h
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
```

```c
/*PREVENT REDUNDANT INCLUSION
*****************************************************************************************
*****************/
#ifndef HD_MAIN_H
#define HD_MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************
*******************
INCLUDES
*****************************************************************************************
*****************/
#include "HDTypes.h"
#include "HDApi.h"

/*****************************************************************************************
*******************
CONSTANTS
*****************************************************************************************
*****************/

/*****************************************************************************************
*******************
TYPEDEFS
*****************************************************************************************
*****************/

/*****************************************************************************************
*******************
VARIABLES
*****************************************************************************************
*****************/

/*****************************************************************************************
*******************
MACROS
*****************************************************************************************
*****************/

/*****************************************************************************************
*******************
PROTOTYPES
*****************************************************************************************
*****************/
/*****************************************************************************************
* \brief      function used to initialize the HDRadio protocol
* \param      void
* \return     void
*****************************************************************************************/
void HDM_vInit(void);
/*****************************************************************************************
* \brief      function to read the buffer.
* \param      Format used for tasks on freeRtos
* \return     void
*****************************************************************************************/
void HDP_vStateHdl_tsk(void const * argument);

#ifdef __cplusplus
}
#endif

#endif  /* HD_MAIN_H */
```

```
/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDMain.c

```
/*!
* \file     HDMain.c
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/

#include "Sys_LogTrace.h"
#include "HDMain.h"
#include "cmsis_os.h"
#include "HDParser.h"
#include "fatfs.h"

/*****************************************************************************************************
******************
MACROS - Macro definitions used by this module
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*****************************************************************************************************
*****************/


typedef void(*pvState_Hdl)(void);

typedef enum HDP__nenMainStates
{
    HDM__nenIDLEState = 0x00
    ,HDM__nenBrdcstMsg
    ,HDM__nenAttrMsg
    ,HDM__nenPkgMsg
    ,HDM__nenMsgValidtn
    ,HDM__nenDataSavng
    ,HDM__nenMainStts_NB
}HDM__tenMainStates;

typedef struct
{
    HDM__tenMainStates    enMainState;  /*!< Element used to know current state  */
    pvState_Hdl           pvSateHndl;    /*!< Handler used for current state  */
}HDM__tstStateItems;

typedef enum HDP__nenProtcolStatus
{
    HDM__nenNotInit = 0x00
    ,HDM__nenInitialized
}HDM__tenHDRecvrStatus;
```

```
typedef struct HDM__tstBrdcstMsg
{
    uint8  u8SwKey;
    uint8  u8NumOfAttr;
    uint32 u32NumOfDataPkgs;
    Bool   boValidate;
}HDM__stBrdcstMsg;

typedef struct HDM__tstAttrMsg
{
    uint8  u8SwKey;
    uint8 u8AttrID;
    uint16 u16NumParam;
    char   *pu8AttrStr;
    Bool   boValidate;
}HDM__stAttrMsg;

typedef struct HDM__tstPkgMsg
{
    uint8  u8SwKey;
    uint32 u32PkgId;
    uint8 *pu8DataPkg;
    Bool   boValidate;
}HDM__stPkgMsg;

typedef union
{
    HDM__stBrdcstMsg stBrdcstMsg;
    HDM__stAttrMsg   stAttrMsg;
    HDM__stPkgMsg    stPkgMsg;
}HDM__stMsg;


/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
*****************/
static void   HDM__vIDLE_statHdl(void);
static void   HDM__vBrdMsg_statHdl(void);
static void   HDM__vAttrMsg_statHdl(void);
static void   HDM__vPkgMsg_statHdl(void);
static void   HDM__vMsgValition_statHdl(void);
static void   HDM__vDataSvng_statHdl(void);
static void   HDM__vSetState(HDM__tenMainStates enNewState);
static uint32 HDM__u32IDByteParsr(uint8 u8Byte1, uint8 u8Byte2, uint8 u8Byte3, uint8 u8Byte4);
static uint16 HDM__u16ParamByteParsr(uint8 u8Byte1, uint8 u8Byte2);
static Bool   HDM__boValidateChksm(uint8 *pu8Datas, uint8 u8Size);

/*****************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*****************************************************************************************************
*****************/
#define HDM__au8MainStates_NB      (uint8)(sizeof(HDP__astStates) / sizeof(HDP__astStates[0]))  /*!<
COMMENT  */

static HDM__tstStateItems    HDM__stMachineContext = { 0 };
static HDM__tenHDRecvrStatus HDM__enReceiverStatus = HDM__nenNotInit;
static HDM__stMsg           HDM__stCurrentMsg    = { 0 };
static HDP_tstDataFramType   HDM__stReceivedMsg    = { 0 };

/*****************************************************************************************************
******************
CONSTANTS - Constants used by this module
*****************************************************************************************************
*****************/
```

```
const char acStatesStr[HDM__nenMainStts_NB][10] =
{
    "IDLE"
    ,"Brdcst"
    ,"Attr"
    ,"Pkg"
    ,"Vlidtion"
    ,"DtaSvng"
};

const HDM__tstStateItems HDP__astStates[HDM__nenMainStts_NB] =
{
    /*   enMainState      |       pvSateHndl          */
    { HDM__nenIDLEState   , &HDM__vIDLE_statHdl       }
    ,{ HDM__nenBrdcstMsg , &HDM__vBrdMsg_statHdl     }
    ,{ HDM__nenAttrMsg   , &HDM__vAttrMsg_statHdl    }
    ,{ HDM__nenPkgMsg    , &HDM__vPkgMsg_statHdl     }
    ,{ HDM__nenMsgValidtn, &HDM__vMsgValition_statHdl}
    ,{ HDM__nenDataSavng , &HDM__vDataSvng_statHdl   }
};

/*****************************************************************************************************
******************
PUBLIC FUNCTIONS
*****************************************************************************************************
******************/

void HDM_vInit(void)
{
    if (HDM__nenInitialized != HDM__enReceiverStatus)
    {
        HDM__vSetState(HDM__nenIDLEState);
        HDM__enReceiverStatus = HDM__nenInitialized;
    }
    else
    {
        HDM_LOG_ERR("Module already Init");
    }
}

void HDP_vStateHdl_tsk(void const * argument)
{
    while (True)
    {
        HDM__stMachineContext.pvSateHndl();
        osDelay(100);
        osThreadYield();
    }
}

/*****************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
*****************************************************************************************************
******************/

static void HDM__vIDLE_statHdl(void)
{
    uint8 u8Buffer_len = 0;

    u8Buffer_len = HDP_u8GetMsgBuffLen();
    if (HDP_u8GetMsgBuffLen() > 0)
    {
        HDM_LOG_DEBUG("MsgBuffler_len[%d]", u8Buffer_len);
        HDP_vGetMsg(&HDM__stReceivedMsg);
```

```
        switch (HDM__stReceivedMsg.enMsgID)
        {
            case HDP_enNewBroadcstMsg:
            {
                HDM__vSetState(HDM__nenBrdcstMsg);
            }break;
            case HDP_enAttrMsg:
            {
                HDM__vSetState(HDM__nenAttrMsg);
            }break;
            case HDP_enPkgMsg:
            {
                HDM__vSetState(HDM__nenPkgMsg);
            }break;
            default:
            {
                HDM_LOG_ERR("MsgNotSupported[%d]", HDM__stReceivedMsg.enMsgID);
                memset((void *)&HDM__stReceivedMsg, 0x00, sizeof(HDM__stReceivedMsg));
            }break;
        }
    }
}

static void HDM__vBrdMsg_statHdl(void)
{
    HDM__stBrdcstMsg *stTempMsg = (HDM__stBrdcstMsg *)&HDM__stCurrentMsg;

        stTempMsg->u8NumOfAttr = HDM__stReceivedMsg.u8DataBuf[0];
        stTempMsg->u32NumOfDataPkgs = HDM__u32IDByteParsr(HDM__stReceivedMsg.u8DataBuf[1],
                                                          HDM__stReceivedMsg.u8DataBuf[2],
                                                          HDM__stReceivedMsg.u8DataBuf[3],
                                                          HDM__stReceivedMsg.u8DataBuf[4]);
        stTempMsg->u8SwKey = HDM__stReceivedMsg.u8DataBuf[5];
        HDM_LOG_DEBUG("---BROADCOAST---------");
        HDM_LOG_DEBUG("enMsgID[0x%x]", HDM__stReceivedMsg.enMsgID);
        HDM_LOG_DEBUG("NumAttr[0x%x]", stTempMsg->u8NumOfAttr);
        HDM_LOG_DEBUG("NumPkg[0x%x]", stTempMsg->u32NumOfDataPkgs);
        HDM_LOG_DEBUG("u8SwKey[0x%x]", stTempMsg->u8SwKey);
        HDM_LOG_DEBUG("CRC[0x%x]", HDM__stReceivedMsg.u8DataBuf[6]);
        HDM__vSetState(HDM__nenMsgValidtn);
}

static void HDM__vAttrMsg_statHdl(void)
{
    HDM__stAttrMsg *stTempMsg = (HDM__stAttrMsg *)&HDM__stCurrentMsg;

    stTempMsg->u8SwKey      = HDM__stReceivedMsg.u8SwKey;
    stTempMsg->u8AttrID     = HDM__stReceivedMsg.u8DataBuf[0];
    HDM_LOG_DEBUG("Byte1[0x%x] Byte2[0x%x]", HDM__stReceivedMsg.u8DataBuf[1],
HDM__stReceivedMsg.u8DataBuf[2]);
    stTempMsg->u16NumParam  = HDM__u16ParamByteParsr(HDM__stReceivedMsg.u8DataBuf[1],
HDM__stReceivedMsg.u8DataBuf[2]);
    stTempMsg->pu8AttrStr   = &HDM__stReceivedMsg.u8DataBuf[3];


    HDM_LOG_DEBUG("---ATTRIBUTE---------");
    HDM_LOG_DEBUG("enMsgID[0x%x]", HDM__stReceivedMsg.enMsgID);
    HDM_LOG_DEBUG("u8SwKey[0x%x]", stTempMsg->u8SwKey);
    HDM_LOG_DEBUG("u8AttrID[0x%x]", stTempMsg->u8AttrID);
    HDM_LOG_DEBUG("u16NumParam[0x%x]", stTempMsg->u16NumParam);
    HDM_LOG_DEBUG("Params[%s]", &HDM__stReceivedMsg.u8DataBuf[3]);
    HDM_LOG_DEBUG("CRC[0x%x]", HDM__stReceivedMsg.u8DataBuf[(3+stTempMsg->u16NumParam)]);
    HDM__vSetState(HDM__nenMsgValidtn);
}

static void HDM__vPkgMsg_statHdl(void)
{
```

```
    HDM__stPkgMsg *stTempMsg = (HDM__stPkgMsg *)&HDM__stCurrentMsg;

  stTempMsg->u8SwKey = HDM__stReceivedMsg.u8SwKey;
  stTempMsg->u32PkgId = HDM__u32IDByteParsr(HDM__stReceivedMsg.u8DataBuf[0],
            HDM__stReceivedMsg.u8DataBuf[1],
            HDM__stReceivedMsg.u8DataBuf[2],
            HDM__stReceivedMsg.u8DataBuf[3]);
   stTempMsg->pu8DataPkg = (uint8 *)&HDM__stReceivedMsg.u8DataBuf[4];
   HDM_LOG_DEBUG("---PACKAGE---------");
   HDM_LOG_DEBUG("enMsgID[0x%x]", HDM__stReceivedMsg.enMsgID);
   HDM_LOG_DEBUG("u8SwKey[0x%x]", HDM__stReceivedMsg.u8SwKey);
   HDM_LOG_DEBUG("PkgID[0x%x]", stTempMsg->u32PkgId);
   HDM_LOG_DEBUG("Data1[0x%x]", stTempMsg->pu8DataPkg[0]);
   HDM_LOG_DEBUG("Data2[0x%x]", stTempMsg->pu8DataPkg[1]);
   HDM_LOG_DEBUG("Data3[0x%x]", stTempMsg->pu8DataPkg[2]);
   HDM_LOG_DEBUG("Data4[0x%x]", stTempMsg->pu8DataPkg[3]);
   HDM_LOG_DEBUG("Data5[0x%x]", stTempMsg->pu8DataPkg[4]);
   HDM_LOG_DEBUG("CRC[0x%x]", HDM__stReceivedMsg.u8DataBuf[9]);

    HDM__vSetState(HDM__nenMsgValidtn);

}


static void HDM__vMsgValition_statHdl(void)
{


    switch (HDM__stReceivedMsg.enMsgID)
    {
        case HDP_enNewBroadcstMsg:
        {
            HDM__stBrdcstMsg *stTempMsg = (HDM__stBrdcstMsg *)&HDM__stCurrentMsg;
            if (HDM__boValidateChksm(&HDM__stReceivedMsg.u8DataBuf[0], 6))
            {
                stTempMsg->boValidate = True;
                HDM_LOG_DEBUG("Valid Chcksm");
                HDM__vSetState(HDM__nenDataSavng);
            }
            else
            {
                stTempMsg->boValidate = False;
                HDM_LOG_ERR("Invalid Msg CheckSum");
                HDM__vSetState(HDM__nenIDLEState);
            }
        }break;
        case HDP_enAttrMsg:
        {
            HDM__stAttrMsg *stTempMsg = (HDM__stAttrMsg *)&HDM__stCurrentMsg;
            if (HDM__boValidateChksm(&HDM__stReceivedMsg.u8DataBuf[0], 3+stTempMsg->u16NumParam))
            {
                stTempMsg->boValidate = True;
                HDM_LOG_DEBUG("Valid Chcksm");
                HDM__vSetState(HDM__nenDataSavng);
            }
            else
            {
                stTempMsg->boValidate = False;
                HDM_LOG_ERR("Invalid Msg CheckSum");
                HDM__vSetState(HDM__nenIDLEState);
            }
        }break;
        case HDP_enPkgMsg:
        {
            HDM__stPkgMsg *stTempMsg = (HDM__stPkgMsg *)&HDM__stCurrentMsg;
            if (HDM__boValidateChksm(&HDM__stReceivedMsg.u8DataBuf[0], 9))
            {
                stTempMsg->boValidate = True;
```

```
                HDM_LOG_DEBUG("Valid Chcksm");
                HDM__vSetState(HDM__nenDataSavng);
            }
            else
            {
                stTempMsg->boValidate = False;
                HDM_LOG_ERR("Invalid Msg CheckSum");
                HDM__vSetState(HDM__nenIDLEState);
            }
        }break;
        default:
        {
        }break;
    }

}

static void HDM__vDataSvng_statHdl(void)
{

    switch (HDM__stReceivedMsg.enMsgID)
    {
        case HDP_enNewBroadcstMsg:
        {
            HDM__stBrdcstMsg *stTempMsg = (HDM__stBrdcstMsg *)&HDM__stCurrentMsg;
            if (True == stTempMsg->boValidate)
            {
                if (FSA_enAlrdyExist == FSA_enChechBrdcstSwKey((FSA_stBrdcstMsg *)stTempMsg))
                {
                    HDM_LOG_DEBUG("Already exist Do nothing...");
                    if (FSA_enSuccess == FSA_enCreateNewDB((FSA_stBrdcstMsg *)stTempMsg))
                    {
                        HDM_LOG_DEBUG("Database Created...");
                    }
                    else
                    {
                        HDM_LOG_ERR("Error creating Database...");
                    }
                }
                else
                {
                    HDM_LOG_DEBUG("New Brdcst, Create Database...");
                }
            }
        }break;
        case HDP_enAttrMsg:
        {
            HDM__stAttrMsg *stTempMsg = (HDM__stAttrMsg *)&HDM__stCurrentMsg;
            if (True == stTempMsg->boValidate)
            {
                if (FSA_enAlrdyExist == FSA_enCheckAttrSwKey((FSA_stAttrMsg *)stTempMsg))
                {
                    HDM_LOG_DEBUG("DB exist,Adding attribute...");
                    if (FSA_enSuccess == FSA_enAddAttribute((FSA_stAttrMsg *)stTempMsg))
                    {
                        HDM_LOG_DEBUG("Attribute Added...");
                    }
                    else
                    {
                        HDM_LOG_DEBUG("Attribute exist or error...");
                    }
                }
                else
                {
                    HDM_LOG_DEBUG("Brdcst DB no exist,Do nothing...");
                }
            }
        }break;
```

```
        case HDP_enPkgMsg:
        {
            HDM__stPkgMsg *stTempMsg = (HDM__stPkgMsg *)&HDM__stCurrentMsg;
            if (True == stTempMsg->boValidate)
            {
                if (FSA_enAlrdyExist == FSA_enCheckDataPkgSwKey((FSA_stPkgMsg *)stTempMsg))
                {
                    HDM_LOG_DEBUG("DB exist,Adding package...");
                    if (FSA_enSuccess == FSA_enAddDataPkg((FSA_stPkgMsg *)stTempMsg))
                    {
                        HDM_LOG_DEBUG("Pkg Added...");
                    }
                    else
                    {
                        HDM_LOG_ERR("DataPkg exist or error...");
                    }
                }
                else
                {
                    HDM_LOG_DEBUG("Brdcst DB no exist,Do nothing...");
                }

            }
        }break;
        default:
        {
        }break;
    }
    HDM__vSetState(HDM__nenIDLEState);

}

static void HDM__vSetState(HDM__tenMainStates enNewState)
{
    if (enNewState < HDM__nenMainStts_NB)
    {
        HDM__stMachineContext.pvSateHndl  = HDP__astStates[enNewState].pvSateHndl;
        HDM__stMachineContext.enMainState = HDP__astStates[enNewState].enMainState;
        HDM_LOG_INF("State -> %s", acStatesStr[enNewState]);
        if (HDM__nenIDLEState == enNewState)
        {
            HDM_LOG_INF("Msg is clean");
            memset((void *)&HDM__stReceivedMsg, 0x00, sizeof(HDM__stReceivedMsg));
            memset((void *)&HDM__stCurrentMsg, 0x00, sizeof(HDM__stCurrentMsg));
        }
    }
    else
    {
        HDM_LOG_ERR("State not allowed.");
    }
}

static uint32 HDM__u32IDByteParsr(uint8 u8Byte1, uint8 u8Byte2, uint8 u8Byte3, uint8 u8Byte4)
{
    uint32 u32ReturnVal = 0;

    u32ReturnVal |= (u8Byte1 << 24) & 0xFFFFFFFF;
    u32ReturnVal |= (u8Byte2 << 16) & 0xFFFFFFFF;
    u32ReturnVal |= (u8Byte3 << 8) & 0xFFFFFFFF;
    u32ReturnVal |= u8Byte4 & 0xFFFFFFFF;

    return u32ReturnVal;
}
static uint16 HDM__u16ParamByteParsr(uint8 u8Byte1, uint8 u8Byte2)
{
    uint16 u16ReturnVal = 0;

    u16ReturnVal = u8Byte1 << 8;
    u16ReturnVal = u16ReturnVal | u8Byte2;
```

```
    return u16ReturnVal;
}

static Bool HDM__boValidateChksm(uint8 *pu8Datas, uint8 u8Size)
{
    Bool boReturnVal = 0;
    uint8 u8Chksm = 0x00;
    uint8 u8Data_iter = 0;


    for (u8Data_iter = 0; u8Data_iter <= u8Size - 1; u8Data_iter++)
    {
        u8Chksm ^= pu8Datas[u8Data_iter];
    }

    if (u8Chksm == pu8Datas[u8Size])
    {
        boReturnVal = True;
        HDM_LOG_INF("\nChksm validated[0x%x] CRCVal[0x%x]\n", u8Chksm, pu8Datas[u8Size]);
    }
    else
    {
        boReturnVal = False;
        HDM_LOG_ERR("\nInvalid Chksm[0x%x] CRCVal[0x%x]\n", u8Chksm, pu8Datas[u8Size]);
    }

    return boReturnVal;
}

/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

HDTypes.h

```
/*!
* \file      HDTypes.h
* \brief
* \author    Isidro Santos
*/

/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
*****************/

#ifndef HD_TYPES_H
#define HD_TYPES_H


#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
```

```
CONSTANTS
*******************************************************************************************************
*****************/

#ifndef True
    #define True 1
#endif

#ifndef False
    #define False 0
#endif

/*******************************************************************************************************
******************
TYPEDEFS
*******************************************************************************************************
*****************/

typedef signed char         int8;
typedef signed short        int16;
typedef int                 int32;
typedef unsigned char       Bool;
typedef unsigned char       uint8;
typedef unsigned short      uint16;
typedef unsigned int        uint32;
typedef unsigned long long  uint64;

/*******************************************************************************************************
******************
VARIABLES
*******************************************************************************************************
*****************/

/*******************************************************************************************************
******************
MACROS
*******************************************************************************************************
*****************/

#ifdef __cplusplus
}
#endif

#endif  /* HD_TYPES_H */
/*******************************************************************************************************
******************
END OF FILE
*******************************************************************************************************
*****************/
```

Sys_LogTrace.h

```
/*!
* \file     UART2_Drv.h
```

```
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
******************/

#ifndef LOG_N_TRACE_H_
#define LOG_N_TRACE_H_

#ifdef __cplusplus
extern "C" {
#endif

/*****************************************************************************************************
******************
INCLUDES
*****************************************************************************************************
******************/
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"
/*****************************************************************************************************
******************
CONSTANTS
*****************************************************************************************************
******************/

#define LNT_INF            1     /*Used to enable information logs */
#define LNT_DEBUG          1     /*Used to enable debug logs       */
#define LNT_ERR            1     /*Used to enable error logs       */
#define LNT_nMAX_STR_LEN   255   /*Max len of string               */

/*****************************************************************************************************
******************
TYPEDEFS
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
VARIABLES
*****************************************************************************************************
******************/

/*****************************************************************************************************
******************
MACROS
*****************************************************************************************************
******************/


#if LNT_INF
#define LNT_LOG_INF(arg, ...)      LNT_Print("\n[INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define MAIN_LOG_INF(arg, ...)     LNT_Print("\n[MAIN-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define UART2_LOG_INF(arg, ...)    LNT_Print("\n[UART2-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define PWRM_LOG_INF(arg, ...)     LNT_Print("\n[PWRM-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
```

```
#define HDP_LOG_INF(arg, ...)       LNT_Print("\n[HDP-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define HDM_LOG_INF(arg, ...)       LNT_Print("\n[HDM-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define FATSe_LOG_INF(arg, ...)     LNT_Print("\n[FATSe-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define USB_LOG_INF(arg, ...)       LNT_Print("\n[USB-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#define CAN_LOG_INF(arg, ...)       LNT_Print("\n[CAN-INF]  %s - "      arg, __FUNCTION__
,##__VA_ARGS__)
#else
#define LNT_LOG_INF(arg, ...)
#define MAIN_LOG_INF(arg, ...)
#define UART2_LOG_INF(arg, ...)
#define PWRM_LOG_INF(arg, ...)
#define HDP_LOG_INF(arg, ...)
#define HDM_LOG_INF(arg, ...)
#define FATSe_LOG_INF(arg, ...)
#define USB_LOG_INF(arg, ...)
#define CAN_LOG_INF(arg, ...)
#endif

#if LNT_DEBUG
#define LNT_LOG_DEBUG(arg, ...)     LNT_Print("\n[DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define PWRM_LOG_DEBUG(arg, ...)    LNT_Print("\n[PWRM-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define CAN_LOG_DEBUG(arg, ...)     LNT_Print("\n[CAN-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define USB_LOG_DEBUG(arg, ...)     LNT_Print("\n[USB-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define FATSe_LOG_DEBUG(arg, ...)   LNT_Print("\n[FATSe-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define HDM_LOG_DEBUG(arg, ...)     LNT_Print("\n[HDM-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define HDP_LOG_DEBUG(arg, ...)     LNT_Print("\n[HDP-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define UART2_LOG_DEBUG(arg, ...)   LNT_Print("\n[UART2-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#define MAIN_LOG_DEBUG(arg, ...)    LNT_Print("\n[MAIN-DEBUG]%s - "      arg , __FUNCTION__
,##__VA_ARGS__)
#else
#define LNT_LOG_DEBUG(arg,...)
#define PWRM_LOG_DEBUG(arg,...)
#define CAN_LOG_DEBUG(arg, ...)
#define USB_LOG_DEBUG(arg, ...)
#define FATSe_LOG_DEBUG(arg, ...)
#define HDM_LOG_DEBUG(arg, ...)
#define HDP_LOG_DEBUG(arg, ...)
#define UART2_LOG_DEBUG(arg, ...)
#define MAIN_LOG_DEBUG(arg, ...)
#endif


#if LNT_ERR
#define LNT_LOG_ERR(arg, ...)       LNT_Print("\n[ERR]  %s(%d) - "  arg , __FUNCTION__ , __LINE__
,##__VA_ARGS__)
#define MAIN_LOG_ERR(arg, ...)      LNT_Print("\n[MAIN-ERR]  %s(%d) - "  arg , __FUNCTION__ ,
__LINE__     ,##__VA_ARGS__)
#define UART2_LOG_ERR(arg, ...)     LNT_Print("\n[UART2-ERR]  %s(%d) - "  arg , __FUNCTION__ ,
__LINE__     ,##__VA_ARGS__)
#define PWRM_LOG_ERR(arg, ...)      LNT_Print("\n[PWRM-ERR]  %s(%d) - "  arg , __FUNCTION__ ,
__LINE__     ,##__VA_ARGS__)
#define HDP_LOG_ERR(arg, ...)       LNT_Print("\n[HDP-ERR]  %s(%d) - "  arg , __FUNCTION__ , __LINE__
,##__VA_ARGS__)
#define HDM_LOG_ERR(arg, ...)       LNT_Print("\n[HDM-ERR]  %s(%d) - "  arg , __FUNCTION__ , __LINE__
,##__VA_ARGS__)
```

```
#define FATSe_LOG_ERR(arg, ...)     LNT_Print("\n[FATSe-ERR]  %s(%d) - "  arg , __FUNCTION__ ,
__LINE__     ,##__VA_ARGS__)
#define USB_LOG_ERR(arg, ...)       LNT_Print("\n[USB-ERR]  %s(%d) - "  arg , __FUNCTION__ , __LINE__
,##__VA_ARGS__)
#define CAN_LOG_ERR(arg, ...)       LNT_Print("\n[CAN-ERR]  %s(%d) - "  arg , __FUNCTION__ , __LINE__
,##__VA_ARGS__)
#else
#define LNT_LOG_ERR(arg, ...)
#define MAIN_LOG_ERR(arg, ...)
#define UART2_LOG_ERR(arg, ...)
#define PWRM_LOG_ERR(arg, ...)
#define HDP_LOG_ERR(arg, ...)
#define HDM_LOG_ERR(arg, ...)
#define FATSe_LOG_ERR(arg, ...)
#define USB_LOG_ERR(arg, ...)
#define CAN_LOG_ERR(arg, ...)
#endif

/*****************************************************************************************************
******************
PROTOTYPES
*****************************************************************************************************
*****************/

/*****************************************************************************************************
* \brief      Log and Trace initialization
* \param      void
* \return     void
*****************************************************************************************************/
void LNT_vInit(void);
/*****************************************************************************************************
* \brief      function to read the buffer.
* \param[in]  pu8DataBuf:pointer to buffer with datas to be transmitted.
* \param[in]  u16Size:Size of data buffer to be transmitted.
* \return     void
*****************************************************************************************************/
void LNT_vWrite(uint8_t *pu8DataBuf, uint16_t u16Size );

/*****************************************************************************************************
* \brief     Function used to print the logs.
* \param     Same format as standard library
* \return    int
*****************************************************************************************************/
int LNT_Print(char *fmt, ...);

#ifdef __cplusplus
}
#endif

#endif /*LOG_N_TRACE_H_*/
/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

Sys_LogTrace.c

```
/*!
* \file     UART2_Drv.c
* \brief
* \author   Isidro Santos
*/

/*****************************************************************************************************
******************
```

```
INCLUDES
*******************************************************************************************************
******************/

#include "Sys_LogTrace.h"
#include "stdio.h"
#include "stdarg.h"
#include "cmsis_os.h"
/*******************************************************************************************************
******************
CONSTANTS - Constants used by this module
*******************************************************************************************************
******************/
#define LNT__nPORT_CFG         USART3
#define LNT__nBAUDRATE_CFG     115200
#define LNT__nWORLDLEN_CFG     UART_WORDLENGTH_8B
#define LNT__nSTOPBIT_CFG      UART_STOPBITS_1
#define LNT__nPARITY_CFG       UART_PARITY_NONE
#define LNT__nMODE_CFG         UART_MODE_TX
#define LNT__nHWFLOW_CTL_CFG   UART_HWCONTROL_NONE
#define LNT__nOVER_SAMP_CFG    UART_OVERSAMPLING_16

/*******************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*******************************************************************************************************
******************/

/*******************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*******************************************************************************************************
******************/
static UART_HandleTypeDef     LNT__stUARTfHndl          = {0};
static osMutexId  LNT__tMutexID = NULL;

/*******************************************************************************************************
******************
MACROS - Macro definitions used by this module
*******************************************************************************************************
******************/

/*******************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*******************************************************************************************************
******************/

/*******************************************************************************************//**
* \brief      Unblock USB Reading Page
* \param[in]  pstMsg: pointer to message that it will be sent
* \return     >=0 - Success, <0 - Fail
*******************************************************************************************/

/*******************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*******************************************************************************************************
******************/

void LNT_vInit(void)
{

    LNT__stUARTfHndl.Instance          = LNT__nPORT_CFG;
    LNT__stUARTfHndl.Init.BaudRate     = LNT__nBAUDRATE_CFG;
    LNT__stUARTfHndl.Init.WordLength   = LNT__nWORLDLEN_CFG;
    LNT__stUARTfHndl.Init.StopBits     = LNT__nSTOPBIT_CFG;
    LNT__stUARTfHndl.Init.Parity       = LNT__nPARITY_CFG;
    LNT__stUARTfHndl.Init.Mode         = LNT__nMODE_CFG;
```

```
    LNT__stUARTfHndl.Init.HwFlowCtl    = LNT__nHWFLOW_CTL_CFG;
    LNT__stUARTfHndl.Init.OverSampling = LNT__nOVER_SAMP_CFG;

    HAL_UART_Init(&LNT__stUARTfHndl);

    LNT__tMutexID = osMutexCreate(NULL);

}

void LNT_vWrite(uint8_t *pu8DataBuf, uint16_t u16Size )
{
    HAL_UART_StateTypeDef enReturnVal;
    enReturnVal = HAL_UART_GetState(&LNT__stUARTfHndl);

    //osMutexWait(LNT__tMutexID, 2);
    if (enReturnVal == HAL_UART_STATE_READY)
    {
        HAL_UART_Transmit(&LNT__stUARTfHndl,(uint8_t *)pu8DataBuf,(uint16_t)u16Size, (uint32_t)10);
    }
    //osMutexRelease(LNT__tMutexID);
}

int LNT_Print(char *fmt, ...)
{
    int i32ReturnVal = -1;
    char buf[100];     // this should really be sized appropriately
                       // possibly in response to a call to vsnprintf()
    va_list vl;
    va_start(vl, fmt);

    i32ReturnVal = vsnprintf( buf, sizeof( buf), fmt, vl);

    va_end( vl);


    LNT_vWrite(buf,strlen(buf));

    return i32ReturnVal;

}


/*****************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
*****************************************************************************************************
*****************/

/*****************************************************************************************************
******************
END OF FILE
*****************************************************************************************************
*****************/
```

PwrMngr.h

```
/*****************************************************************************************************
******************
PREVENT REDUNDANT INCLUSION
*****************************************************************************************************
*****************/
#ifndef PWR_MNGR_H
#define PWR_MNGR_H


#ifdef __cplusplus
extern "C" {
```

```
#endif

/**************************************************************************************************
*******************
INCLUDES
**************************************************************************************************
*****************/
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"

/**************************************************************************************************
*******************
CONSTANTS
**************************************************************************************************
*****************/

/**************************************************************************************************
*******************
TYPEDEFS
**************************************************************************************************
*****************/

/**************************************************************************************************
*******************
VARIABLES
**************************************************************************************************
*****************/

/**************************************************************************************************
*******************
MACROS
**************************************************************************************************
*****************/

/**************************************************************************************************
*******************
PROTOTYPES
**************************************************************************************************
*****************/

/**************************************************************************************************
* \brief      Power manager main task
* \param      Format used for tasks on freeRtos
* \return     void
**************************************************************************************************/
void PwrMngr_vmainTask(void const * argument);

#ifdef __cplusplus
}
#endif

#endif /* PWR_MNGR_H */

/**************************************************************************************************
*******************
END OF FILE
**************************************************************************************************
*****************/
```

PwrMngr.c

```
/**************************************************************************************************
*******************
INCLUDES
**************************************************************************************************
*****************/
```

```c
#include "PwrMngr.h"
#include "UART2_Drv.h"
#include "usb_host.h"
#include "Sys_LogTrace.h"
#include "fatfs.h"
#include "HDParser.h"
#include "HDMain.h"
#include "CANDrv.h"

#define USB_TEST

/*****************************************************************************************************
******************
MACROS - Macro definitions used by this module
*****************************************************************************************************
******************/
#define PWR_SCI2
#define PWR_HDMSG
#define PWR_HDREC
#define PWR_CAN


/*****************************************************************************************************
******************
TYPEDEFS - Local type definitions used by this module
*****************************************************************************************************
******************/

typedef void(*PwrMngr_pvInit)(void);

typedef struct
{
    os_pthread     pvTask;
    PwrMngr_pvInit pvInitFunc;
    char           acTaskName[5];
    uint32_t       u32StackSize;
    osPriority     enPrior;
}PwrMngr__tstTaksHdl;

/*****************************************************************************************************
******************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************************************
******************/


static void PwrMngr__vInitFsa(void);

/*****************************************************************************************************
******************
LOCAL VARIABLES - Module global variables
*****************************************************************************************************
******************/

const PwrMngr__tstTaksHdl PwrMngr__astTasksTable[] =
{
    {&PwrMngr_vmainTask       ,NULL          ,"1    ", 1024,osPriorityAboveNormal}
#ifdef PWR_SCI2
    ,{&SCI2_vReceiveTask      ,&SCI2s_vInit ,"SCI2 ", 512 ,osPriorityNormal     }
#endif
#ifdef PWR_HDMSG
    ,{&HDP_vMsg_tsk           ,&HDP_vInit   ,"HDMsg", 512 ,osPriorityNormal     }
#endif
#ifdef PWR_HDREC
    ,{&HDP_vStateHdl_tsk      ,&HDM_vInit   ,"HDRec", 512 ,osPriorityNormal     }
#endif
#ifdef PWR_CAN
    ,{&CANDrv_vDispatcher_tsk ,&CANDrv_vInit,"CAN  ", 512 ,osPriorityNormal     }
#endif
```

```
};


/*************************************************************************************************
******************
CONSTANTS - Constants used by this module
*************************************************************************************************
******************/

#define PwrMngr_nNUM_TASK  sizeof(PwrMngr__astTasksTable)/sizeof(PwrMngr__tstTaksHdl)


/*************************************************************************************************
******************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*************************************************************************************************
******************/

void PwrMngr_vPwrUp(void)
{

    uint8_t u8Task_Iter = 0;
    PwrMngr_pvInit pvTempFunc = NULL;
    osThreadDef_t  stTempTaksHdl= {0};
    LNT_vInit();

    for (u8Task_Iter = 0; u8Task_Iter<PwrMngr_nNUM_TASK;u8Task_Iter++)
    {
        pvTempFunc = PwrMngr__astTasksTable[u8Task_Iter].pvInitFunc;
        if (NULL != pvTempFunc)
        {
            pvTempFunc();
        }
    }


    for (u8Task_Iter = 0; u8Task_Iter<PwrMngr_nNUM_TASK;u8Task_Iter++)
    {
        if ( '1' == PwrMngr__astTasksTable[u8Task_Iter].acTaskName[0])
        {
            osThreadDef(defaultTask, PwrMngr__astTasksTable[u8Task_Iter].pvTask,
PwrMngr__astTasksTable[u8Task_Iter].enPrior, 0, PwrMngr__astTasksTable[u8Task_Iter].u32StackSize);
            osThreadCreate(osThread(defaultTask), NULL);
        }
        else
        {
            stTempTaksHdl.name      =(char *) &PwrMngr__astTasksTable[u8Task_Iter].acTaskName[0];
            stTempTaksHdl.pthread   =  PwrMngr__astTasksTable[u8Task_Iter].pvTask;
            stTempTaksHdl.tpriority = PwrMngr__astTasksTable[u8Task_Iter].enPrior;
            stTempTaksHdl.instances = 0;
            stTempTaksHdl.stacksize = PwrMngr__astTasksTable[u8Task_Iter].u32StackSize;
            osThreadCreate (&stTempTaksHdl, NULL);
        }
    }
    osKernelStart();
}

void PwrMngr_vmainTask(void const * argument)
{
    /*Temporal variables for test */
#ifdef USB_TEST
    ID    API__tenFileID;
    uint8_t API__aun8FileContent[26][5] = {0};
    uint8_t API__aun8PackageContentOut[5] ="Pack2";
    uint8_t API__aun8PackageContent[5] = {0};
    FSA_tenReturnCode enReturnValue = FSA_enError;
    uint8_t boFlag = FALSE;
```

```
#endif
    PwrMngr__vInitFsa();

    for(;;)
    {
#ifdef USB_TEST
        if ((USBHost_enReady == USBHost_enGetDevStatus()) && (FALSE == boFlag))
        {
            boFlag = TRUE;
            osDelay(100);

            /*Steps to start a Reading Session*/
            API__tenFileID =10;
            enReturnValue = FSA_enReadingSession(API__tenFileID);
            PWRM_LOG_INF(""
            "Reading Session Opened [%d]",enReturnValue);
            enReturnValue = FSA_enGetFile(0,&API__aun8FileContent);
            PWRM_LOG_INF("FSA_enGetFile[%d]",enReturnValue);
            MAIN_LOG_DEBUG("Result file[%s]",&API__aun8FileContent[0][0]);
            enReturnValue = FSA_enGetPackage(0,0,&API__aun8PackageContent[0]);
            PWRM_LOG_INF("FSA_enGetPackage[%d]",enReturnValue);
            MAIN_LOG_DEBUG("Result Package[%s]",API__aun8PackageContent);
            FSA_enCloseSession();

            osDelay(10);

            /*Steps to start a Writing Session*/
            API__tenFileID =11;
            enReturnValue = FSA_enCreateNewFile(&API__tenFileID);
            PWRM_LOG_INF("Creating File[%d]",enReturnValue);
            enReturnValue = FSA_enWrittingSession(API__tenFileID);
            PWRM_LOG_INF("Writing Session Opened [%d]",enReturnValue);
            /*param1:Not used
            param2:Number of package to write
            param3:Pointer to string with package content
            */
            enReturnValue = FSA_enWriteDataPkg(0,1,&API__aun8PackageContentOut[0]);
            PWRM_LOG_INF("FSA_enWriteDataPkg [%d]",enReturnValue);
            FSA_enCloseSession();
        }
        osDelay(100);
        osThreadYield();
#endif
    }

}
/**************************************************************************************************
******************
PRIVATE FUNCTIONS - Functions accessed only by this module
**************************************************************************************************
******************/

static void PwrMngr__vInitFsa(void)
{
    USBHost_vInit(NULL);
    FSA_vInit();
    PWRM_LOG_INF("Modules are Init.");
    while(USBHost_enReady !=USBHost_enGetDevStatus())
    {
        PWRM_LOG_DEBUG("Waiting For USB...");
        osDelay(500);
    }
    FSA_vMountDevice();
    osDelay(200);
}


/**************************************************************************************************
******************
```

```
END OF FILE
*************************************************************************************************
*****************/
```

Fatfs.c

```
/*!
* \file     fatfs.c
* \brief    This module is in charge to manage the information received from the
UART port.
                    The information received in this module is saving in a little
chunks called “Packages”.
                    These packages are contained in binary file for finally will
store inside a USB memory.
* \author   Eduardo Luquin
*/

/*************************************************************************************
***********************************
INCLUDES
*************************************************************************************
***********************************/
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"
#include <string.h>
#include "fatfs.h"
#include "Sys_LogTrace.h"
#include "usb_host.h"
/*************************************************************************************
***********************************
CONSTANTS - Constants used by this module
*************************************************************************************
***********************************/

#define FSA__u8FileSizeBytes        (128u)  /*Maximus size in byte of each file*/
#define FSA__u8PackageSizeBytes     (5u)    /*Maximus size in byte of each
package*/
/*maximum number of packages by file*/
#define FSA__u8PackagesByFile           (26u)

/*************************************************************************************
***********************************
TYPEDEFS - Local type definitions used by this module
*************************************************************************************
***********************************/
typedef enum FSA_tenFileState
{
    FSA_enRedingMode = 0x00,
    FSA_enWrittingMode,
    FSA_enInvalidMode,
}FSA_tenFileState;

/*************************************************************************************
***********************************
LOCAL VARIABLES - Module global variables
*************************************************************************************
***********************************/
/* Return value for USBH */
```

```
uint8_t retUSBH;
/* USBH logical drive path */
char USBH_Path[4];
/*File information content at RAM*/
uint8 FSA__aun8FileContent[26][5];
/*Relative path where packages will be created */
uint8 FSA__au8StringPathName[5] = "FILE";
/*The file name active ready to write or read*/
uint8 FSA__au8CurrentFileName[15];
/*Define if the current file is reding or writing mode*/
FSA_tenFileState FSA__enCurrentFileState = FSA_enInvalidMode;
/*Define the current FILE in use*/
static FIL   WorkFile;
/*Define the module initialization*/
static FSA_tenFatState FSA__enFatState = FSA_enNotInit;

static FATFS WorkFatFs;
/* File system object for User logical drive */
FATFS FSA__stUSBDeviceHdl;

/*****************************************************************************
***********************************
MACROS - Macro definitions used by this module
*****************************************************************************
***********************************/

/*****************************************************************************
***********************************
LOCAL FUNCTION PROTOTYPES - Function prototypes used by module
*****************************************************************************
***********************************/

/*****************************************************************************
***********//**
* \brief      Unblock USB Reading Page
* \param[in]  pstMsg: pointer to message that it will be sent
* \return     >=0 - Success, <0 - Fail
*****************************************************************************
**************/
static void FSA__vCreateFileName(ID enFileID,uint8 *pu8String);
static void FSA__vUSBNotif_ckb(USBHost_tenDevStatus enStatus);
```

```
/*****************************************************************************
***********************************
* \brief      function used to initialize the FAT module
* \param      void
* \return     void
*****************************************************************************
***********************************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*****************************************************************************
***********************************/
void FSA_vInit(void)
{
    if (FSA_enNotInit == FSA__enFatState)
    {
```

```
        retUSBH = FATFS_LinkDriver(&USBH_Driver, USBH_Path);
        //USBHost_vInit(&FSA__vUSBNotif_ckb);
        FATSe_LOG_INF("Module Init");
        FSA__enFatState = FSA_enIdleNoDev;
    }
    else
    {
        FATSe_LOG_ERR("Module Already Init");
    }
}

/*************************************************************************************
***********************************
* \brief      Timer for the FAT module
* \param      void
* \return     DWOR: Timer duration
/*************************************************************************************
***********************************
PUBLIC FUNCTIONS - Functions accessed by other OTAReceiver SW Modules
*************************************************************************************
***********************************/
DWORD get_fattime(void)
{
  return 0;
}

/*************************************************************************************
***********
* \brief      function to create new binariry file
* \param      ID: Id for the new file created
* \return     FSA_tenReturnCode: Error or success return
/*************************************************************************************
************
FSA_enCreateNewFile
*************************************************************************************
***********/
FSA_tenReturnCode FSA_enCreateNewFile(ID *ptenFileID)
{
    FSA_tenReturnCode enReturnCode = FSA_enError;
    FIL               MyFile;                          /* File object */
    uint8                    au8StringFileName[15];
    ID                       tenFileNameID = *ptenFileID;


    if (FSA_enInit == FSA__enFatState)
    {
       FSA__vCreateFileName(tenFileNameID,au8StringFileName);

             if(f_open(&MyFile, (TCHAR const*)au8StringFileName, FA_CREATE_ALWAYS
| FA_WRITE) == FR_OK)
             {
                 FATSe_LOG_INF("File Created[%s]",au8StringFileName);
                    f_close(&MyFile);
                    enReturnCode = FSA_enSuccess;
             }
             else
             {
                    FATSe_LOG_ERR( "Error Creating file[%s]",au8StringFileName);
```

```
            }
    }
    else
    {
       FATSe_LOG_ERR("Module Not Init");
    }

    return enReturnCode;
}

/*************************************************************************
************
* \brief      function to deleted a binariry file
* \param      ID: Id for the file to be deleted
* \return     FSA_tenReturnCode: Error or success return
*************************************************************************
************
FSA_enDeleteFile
*************************************************************************
***********/
FSA_tenReturnCode FSA_enDeleteFile(ID *ptenFileID)
{
    FSA_tenReturnCode enReturnCode = FSA_enError;
    uint8                     au8StringFileName[15];
    ID                        tenFileNameID = *ptenFileID;

    if (FSA_enInit == FSA__enFatState)
     {
       FSA__vCreateFileName(tenFileNameID,au8StringFileName);

             if(f_unlink (au8StringFileName) == FR_OK)
             {
                    FATSe_LOG_INF( "File Deleted[%s]",au8StringFileName);
                    enReturnCode = FSA_enSuccess;
             }
             else
             {
                    FATSe_LOG_ERR( "Error deleting file[%s]",au8StringFileName);
             }
     }
    else
    {
       FATSe_LOG_ERR("Module Not Init");
    }

    return enReturnCode;
}

/*************************************************************************
***************
* \brief      function used to start a raeding session, load content file in
FSA__aun8FileContent
* \param      ID: file id for opening the session
* \return     FSA_tenReturnCode: Error or success return
*************************************************************************
************//**
FSA_enReadingSession
```

```
*****************************************************************************
**************/
FSA_tenReturnCode FSA_enReadingSession(ID tenFileID)
{
    FSA_tenReturnCode enReturnCode = FSA_enError;
       uint32_t              wbytes;
       uint8                 au8StringFileName[15];

       if (FSA_enInit == FSA__enFatState)
       {
              if (FSA_enInvalidMode == FSA__enCurrentFileState)
              {
                     FSA__vCreateFileName(tenFileID,au8StringFileName);
                     FATSe_LOG_INF("au8StringFileName[%s]",au8StringFileName);

                     if(f_open(&WorkFile, (TCHAR const*)au8StringFileName,FA_READ )
== FR_OK)
                     {
                            FATSe_LOG_INF("Reading Session Started");

                            if(f_read(&WorkFile, FSA__aun8FileContent,
sizeof(FSA__aun8FileContent), (void *)&wbytes) == FR_OK)
                            {
                                   FATSe_LOG_INF("File
loaded[%s]",au8StringFileName);

       memcpy(FSA__au8CurrentFileName,au8StringFileName,strlen((char*)au8StringFil
eName));
                                   FSA__enCurrentFileState = FSA_enRedingMode;
                                   enReturnCode = FSA_enSuccess;
                            }
                            else
                            {
                                   FATSe_LOG_ERR( "Error loading
File[%s]",au8StringFileName);
                            }
                     }
                     else
                     {
                            FATSe_LOG_ERR( "Error Opening
File[%s]",au8StringFileName);
                     }
              }
              else
              {
                     FATSe_LOG_ERR( "Error! Currently there is an open session");
              }
       }
       else
       {
              FATSe_LOG_ERR("Module Not Init");
       }

    return enReturnCode;
}

/*****************************************************************************
***************
```

```
* \brief      function used to start a Writing session, load content file in
FSA__aun8FileContent
* \param      ID: file id for opening the session
* \return     FSA_tenReturnCode: Error or success return
/*************************************************************************
***********//**
FSA_enWrittingSession
*************************************************************************
**************/
FSA_tenReturnCode FSA_enWrittingSession(ID tenFileID)
{
    FSA_tenReturnCode enReturnCode = FSA_enError;
       uint32_t              wbytes;
       uint8                 au8StringFileName[15];

       if (FSA_enInit == FSA__enFatState)
       {
              if (FSA_enInvalidMode == FSA__enCurrentFileState)
              {
                     FSA__vCreateFileName(tenFileID,au8StringFileName);
                     FATSe_LOG_INF("au8StringFileName[%s]",au8StringFileName);

                     if(f_open(&WorkFile, (TCHAR
const*)au8StringFileName,FA_READ|FA_WRITE ) == FR_OK)
                     {
                            FATSe_LOG_INF("Writing Session Started");

                            if(f_read(&WorkFile, FSA__aun8FileContent,
sizeof(FSA__aun8FileContent), (void *)&wbytes) == FR_OK)
                            {
                                   FATSe_LOG_INF("File
loaded[%s]",au8StringFileName);

       memcpy(FSA__au8CurrentFileName,au8StringFileName,strlen((char*)au8StringFil
eName));
                                   FSA__enCurrentFileState = FSA_enWrittingMode;
                                   enReturnCode = FSA_enSuccess;
                            }
                            else
                            {
                                   FATSe_LOG_ERR( "Error loading
File[%s]",au8StringFileName);
                            }
                     }
                     else
                     {
                            FATSe_LOG_ERR( "Error Opening
File[%s]",au8StringFileName);
                     }
              }
              else
              {
                     FATSe_LOG_ERR( "Error! Currently there is an open session");
              }
       }
       else
       {
              FATSe_LOG_ERR("Module Not Init");
```

```
       }

    return enReturnCode;
}

/*************************************************************************
************
* \brief      function used to write a data package in the file previously opened
* \param      ID:Not used  uint32:Package number to write   uint8:Buffer with the
content to write
* \return     FSA_tenReturnCode: Error or success return
/*************************************************************************
************
FSA_enWriteDataPkg
*************************************************************************
***********/
FSA_tenReturnCode FSA_enWriteDataPkg(ID tenFileID, uint32 u32PkgIndx,uint8
*pu8DataBuff)
{
    FSA_tenReturnCode enReturnCode = FSA_enError;
    uint8             *au8String = &FSA__aun8FileContent[u32PkgIndx][0];

    if( (FSA_enWrittingMode == FSA__enCurrentFileState) && (u32PkgIndx <
FSA__u8PackagesByFile) )
    {
        memcpy ((void*)au8String, (const void*)pu8DataBuff,
(size_t)FSA__u8PackageSizeBytes);
        enReturnCode = FSA_enSuccess;
    }
    else
    {
       FATSe_LOG_ERR( "Error writting in file[%s]
ID[%d]",FSA__au8CurrentFileName,tenFileID);
    }
    return enReturnCode;
}

/*************************************************************************
************
* \brief      function used to read all content file, of current file opened
* \param      ID:Not used   uint8:Buffer with the content read
* \return     FSA_tenReturnCode: Error or success return
/*************************************************************************
************
FSA_enGetFile
*************************************************************************
***********/
FSA_tenReturnCode FSA_enGetFile(ID tenFileID,uint8 *puDataBuf)
{
       FSA_tenReturnCode enReturnCode = FSA_enError;

       if(FSA_enRedingMode == FSA__enCurrentFileState)
       {
    memcpy(puDataBuf,FSA__aun8FileContent,sizeof(FSA__aun8FileContent));
       enReturnCode = FSA_enSuccess;
       }
       else
       {
```

```
            FATSe_LOG_ERR( "Error Reading file[%s]
ID[%d]",FSA__au8CurrentFileName,tenFileID);
    }
    return enReturnCode;
}

/**************************************************************************
***********
* \brief      function used to read a data package in the file previously opened
* \param      ID:Not used  uint32:Package number      uint8:Buffer with the
content readed
* \return     FSA_tenReturnCode: Error or success return
/**************************************************************************
***********
FSA_enGetPackage
**************************************************************************
**********/
FSA_tenReturnCode FSA_enGetPackage(ID tenFileID, uint32 u32PkgIndx,uint8
*pu8DataBuf)
{
    FSA_tenReturnCode enReturnCode = FSA_enError;

    if( (FSA_enRedingMode == FSA__enCurrentFileState) && (u32PkgIndx <
FSA__u8PackagesByFile) )
    {
       memset(pu8DataBuf,0,FSA__u8PackageSizeBytes);
            memcpy((void*)pu8DataBuf,(const
void*)&FSA__aun8FileContent[u32PkgIndx][0],(size_t)FSA__u8PackageSizeBytes);
            memcpy((void*)pu8DataBuf+5,(const void*)'\n',(size_t)1);//solo
prueba,meto caracter nulo
            FATSe_LOG_INF("pu8DataBuf: %5s", pu8DataBuf);
            FATSe_LOG_INF("Content: %s", &FSA__aun8FileContent[u32PkgIndx][0]);

        enReturnCode = FSA_enSuccess;
    }
    else
    {
       FATSe_LOG_ERR( "Error Reading in file[%s]
ID[%d]",FSA__au8CurrentFileName,tenFileID);
    }
    return enReturnCode;

}

/**************************************************************************
***********
* \brief      function used to close a writing or reading session
* \param      void
* \return     FSA_tenReturnCode: Error or success return
/**************************************************************************
***********
FSA_enCloseSession
**************************************************************************
**********/
FSA_tenReturnCode FSA_enCloseSession()
{
    FSA_tenReturnCode enReturnCode = FSA_enError;
    uint32_t          wbytes;
```

```
    if (FSA_enInit == FSA__enFatState)
    {
       if(FSA_enWrittingMode == FSA__enCurrentFileState)
           {
               /*Write new changes to file*/
               if(f_write(&WorkFile, FSA__aun8FileContent,
sizeof(FSA__aun8FileContent), (void *)&wbytes) == FR_OK);
                {
                    FATSe_LOG_INF("File Saved");
                    f_close (&WorkFile);
                    enReturnCode = FSA_enSuccess;
                }
           }
           if(FSA_enRedingMode == FSA__enCurrentFileState)
           {
              if(f_write(&WorkFile, FSA__aun8FileContent,
sizeof(FSA__aun8FileContent), (void *)&wbytes) == FR_OK);
               {
                     FATSe_LOG_INF("File Saved");
                  f_close (&WorkFile);
                  enReturnCode = FSA_enSuccess;
               }
           }
           if((FSA_enRedingMode != FSA__enCurrentFileState) && (FSA_enWrittingMode
!= FSA__enCurrentFileState))
           {
              FATSe_LOG_ERR( "Error closing file[%s]",FSA__au8CurrentFileName);
           }

           /*Clear volatile memory and flags*/
           FSA__enCurrentFileState = FSA_enInvalidMode;
           memset(FSA__aun8FileContent,0,sizeof(FSA__aun8FileContent));
           memset(FSA__au8CurrentFileName,0,sizeof(FSA__au8CurrentFileName));

    }
    else
    {
       FATSe_LOG_ERR("Module Not Init");
    }
    return enReturnCode;
}

/**************************************************************************
************
* \brief      function used to test the FAT module(Not use in a real
implementation)
* \param      void
* \return     void
**************************************************************************
************
FSA_vTest
**************************************************************************
***********/
void FSA_vTest( void )
{
    FATFS mynewdiskFatFs;                                  /* File system object
for User logical drive */
```

```
    FIL MyFile;                                             /* File object */
    uint32_t wbytes;                                        /* File write counts */
    uint8_t wtext[] = "File";                                                   /*
File write buffer */

        if(f_mount(&mynewdiskFatFs, (TCHAR const*)USBH_Path, 0) == FR_OK)
        {
               FATSe_LOG_INF("Device mounted");
                if(f_open(&MyFile, "Test1.bin", FA_CREATE_ALWAYS | FA_WRITE) ==
FR_OK)
                {
                    FATSe_LOG_INF("File open");
                       if(f_write(&MyFile, wtext, sizeof(wtext), (void *)&wbytes)
== FR_OK);
                       {
                              FATSe_LOG_INF("File created.");
                              f_close(&MyFile);
                       }
                }
        }
}

/*************************************************************************
************
* \brief      function used to Mount a FATFS system
* \param      void
* \return     void
/*************************************************************************
************
FSA_vMountDevice
*************************************************************************
***********/
void FSA_vMountDevice(void)
{
       if(f_mount(&WorkFatFs, (TCHAR const*)USBH_Path, 1) == FR_OK)
       {
              FATSe_LOG_INF("Device mounted");
              FSA__enFatState = FSA_enInit;
       }
       else
       {
              FATSe_LOG_ERR( "Error Mounting Device");
       }
}

/*************************************************************************
************
* \brief      function used to retunr the current FAT module state
* \param      void
* \return     FSA_tenFatState: Current state.
/*************************************************************************
************
FSA_enGetState
*************************************************************************
***********/
FSA_tenFatState FSA_enGetState(void)
{
       return FSA__enFatState;
```

```
}

/*****************************************************************************
************************************
PRIVATE FUNCTIONS - Functions accessed only by this module
*****************************************************************************
************************************/


/*****************************************************************************
************
* \brief      function used to mount a device when the host is ready
* \param      USBHost_tenDevStatus: Host status
* \return     void
/*****************************************************************************
************
FSA__vUSBNotif_ckb
*****************************************************************************
**********/

static void FSA__vUSBNotif_ckb(USBHost_tenDevStatus enStatus)
{

    switch(enStatus)
    {
#if 1
        case USBHost_enReady:
        {
            if (f_mount(&FSA__stUSBDeviceHdl, (TCHAR const*)USBH_Path, 1) ==
FR_OK)
            {
                FSA__enFatState = FSA_enInit;
                FATSe_LOG_INF("USB Device mounted.");

            }
            else
            {
                FSA__enFatState = FSA_enIdleNoDev;
                FATSe_LOG_ERR("Mounting...");
            }
        }break;
#endif
        default:
        {
            FATSe_LOG_INF( "USB notification not needed");
        }break;
    }

}

/*****************************************************************************
************
* \brief      function used to generate the file names to delete, create, read or
write.
* \param      ID: File id uint8:String with the name generated
* \return     void
/*****************************************************************************
************
```

```
/*****************************************************************************
***********
FSA__vCreateFileName
*****************************************************************************
**********/
void FSA__vCreateFileName(ID enFileID,uint8 *pu8String)
{
    uint8 au8StringFileName[15] = {0};
    sprintf((char*)au8StringFileName,"%s%d.BIN",FSA__au8StringPathName,enFileID);
    puts((char*)au8StringFileName);
    memcpy (pu8String, au8StringFileName, strlen((char*)au8StringFileName)+1 );
}

/*****************************************************************************
***********
FSA_enChechBrdcstSwKey
*****************************************************************************
**********/
FSA_tenReturnCode FSA_enChechBrdcstSwKey(FSA_stBrdcstMsg *pstInputMsg)
{
    FSA_tenReturnCode enReturnVal = FSA_enAlrdyExist;
    return enReturnVal;
}
/*****************************************************************************
***********
FSA_enCreateNewDB
*****************************************************************************
**********/
FSA_tenReturnCode FSA_enCreateNewDB(FSA_stBrdcstMsg *pstInputMsg)
{
    FSA_tenReturnCode enReturnVal = FSA_enSuccess;
    return enReturnVal;
}
/*****************************************************************************
***********
FSA_enCheckAttrSwKey
*****************************************************************************
**********/
FSA_tenReturnCode FSA_enCheckAttrSwKey(FSA_stAttrMsg *pstInputMsg)
{
    FSA_tenReturnCode enReturnVal = FSA_enAlrdyExist;
    return enReturnVal;
}
/*****************************************************************************
***********
FSA_enAddAttribute
*****************************************************************************
**********/
FSA_tenReturnCode FSA_enAddAttribute(FSA_stAttrMsg *pstInputMsg)
{
    FSA_tenReturnCode enReturnVal = FSA_enSuccess;
    return enReturnVal;
}
/*****************************************************************************
***********
FSA_enCheckDataPkgSwKey
```

```
**********************************************************************************
**********/
FSA_tenReturnCode FSA_enCheckDataPkgSwKey(FSA_stPkgMsg *pstInputMsg)
{
    FSA_tenReturnCode enReturnVal = FSA_enAlrdyExist;
    return enReturnVal;
}
/**********************************************************************************
***********
FSA_enAddDataPkg
**********************************************************************************
**********/
FSA_tenReturnCode FSA_enAddDataPkg(FSA_stPkgMsg *pstInputMsg)
{
    FSA_tenReturnCode enReturnVal = FSA_enSuccess;
    return enReturnVal;
}


/**********************************************************************************
***********************************
END OF FILE
**********************************************************************************
***********************************/
```

Encrypt.c

```
#include "aes.h"


static FILE *AES_OpenFile(char *filename, char *mode)

{

      FILE *fp = fopen(filename, mode);


      if (!fp) {

            printf("ERROR: Can't open %s\n", filename);

            exit (EXIT_FAILURE);

      }


      return fp;
```

```
}

int main (int argc, char *argv[])
{
	FILE *fpin, *fpout;
	char* IV = "ABCDEFGHIJKLMNOP";
	char *key = "0123456789abcdef";
	unsigned long fileLen;
	int keysize = 16;
	char* buffer;
	int buffer_len = 128;
	char fileoutname[50];
	long fcount = 1;
	int align = 0;

	fpin = AES_OpenFile(argv[1], "rb");



	fseek(fpin, 0, SEEK_END);
	fileLen = ftell(fpin);
	fseek(fpin, 0, SEEK_SET);

	printf("File size: %lu\n", fileLen);

	fileLen+=(fileLen%16);
	printf("Size fixed: %lu\n", fileLen);
	//Pass file stream to buffer
	buffer = (char *) malloc(fileLen);
```

```
	if(!buffer)
		exit(EXIT_FAILURE);

	fread(buffer, fileLen, 1, fpin);
	sprintf(fileoutname, "%s/Encrypt.bin",argv[2]);
	fpout = AES_OpenFile(fileoutname, "wb");

	AES_Encrypt(buffer, fileLen, IV, key, keysize);

		fwrite(buffer, fileLen, 1 , fpout);

		close (fpout);

	fclose(fpin);


	return 0;

}
```

Aes.h

```
/*!
* \file     aes.h
* \brief
* \author   José Luis Díaz
*/
```

```
/*****************************************************************************
*****************************************

PREVENT REDUNDANT INCLUSION

*****************************************************************************
*****************************************/

#ifndef AES_H

#define AES_H



#ifdef __cplusplus

extern "C" {

#endif



/*****************************************************************************
*****************************************

INCLUDES

*****************************************************************************
*****************************************/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#include <mcrypt.h>


#include <math.h>

#include <stdint.h>

/*****************************************************************************
*****************************************

CONSTANTS
```

```
**********************************************************************
****************************************/

/**********************************************************************
****************************************
TYPEDEFS
**********************************************************************
****************************************/


/**********************************************************************
****************************************
VARIABLES
**********************************************************************
****************************************/

/**********************************************************************
****************************************
MACROS
**********************************************************************
****************************************/

/**********************************************************************
****************************************
PROTOTYPES
**********************************************************************
****************************************/
/**********************************************************************
********************
* \brief      Function used to send all the needed parameters in order to
Encrypt a data buffer
```

```
* \param      void* buffer:          This is the data buffer that will be
encrypt
*                   int buffer_len:  Size of buffer
*                   char* IV:              First block with random data
*                   char* key:       key used by algorithm to cyrypt
*                   int key_len:           size of key
* \return     int:                   Returns 0 if there is not errors in
crypt process,
**************************************************************************
*******************/
int AES_Encrypt (void* buffer,
                        int buffer_len,
                        char* IV,
                        char* key,
                        int key_len);


/**************************************************************************
*******************
* \brief      Function used to send all the needed parameters in order to
Decrypt a data buffer
* \param      void* buffer:          This is the data buffer that will be
encrypt
*                   int buffer_len:  Size of buffer
*                   char* IV:              First block with random data
*                   char* key:       key used by algorithm to cyrypt
*                   int key_len:           size of key
* \return     int:                   Returns 0 if there is not errors in
crypt process,
**************************************************************************
*******************/
int AES_Decrypt (void* buffer,
```

```
                        int buffer_len,
                        char* IV,
                        char* key,
                        int key_len);

/*****************************************************************************
*********************
* \brief      Function used to print the content of data cypher
* \param      char* ciphertext:      char pointer to the data to show
* \return     void
*****************************************************************************
*********************/
void AES_Show(char* ciphertext, int len);

#ifdef __cplusplus
}
#endif

#endif  /* AES_H */
/*****************************************************************************
****************************************
END OF FILE
*****************************************************************************
****************************************/
```

Aes.c

```
#include "aes.h"

int AES_Encrypt (void* buffer,
```

```
                        int buffer_len,
                        char* IV,
                        char* key,
                        int key_len)
{
	MCRYPT td = mcrypt_module_open("rijndael-128", NULL, "cbc", NULL);
	int blocksize = mcrypt_enc_get_block_size(td);
	int ret = 0;

	if (buffer_len % blocksize != 0)
		ret = 1;
	else
		printf("Buffer size: %d\n", buffer_len);

	if (ret == 0) {
		ret = mcrypt_generic_init(td, key, key_len, IV);
		ret = mcrypt_generic(td, buffer, buffer_len);
		ret = mcrypt_generic_deinit(td);
	}
	mcrypt_module_close(td);

	return ret;
}

int AES_Decrypt (void* buffer,
                        int buffer_len,
                        char* IV,
                        char* key,
```

```
                        int key_len)
{
	MCRYPT td = mcrypt_module_open("rijndael-128", NULL, "cbc", NULL);
	int blocksize = mcrypt_enc_get_block_size(td);
	int ret = 0;

	if (buffer_len % blocksize != 0)
		ret = 1;
	if (ret == 0) {
	ret = mcrypt_generic_init(td, key, key_len, IV);
	ret = mdecrypt_generic(td, buffer, buffer_len);
	ret = mcrypt_generic_deinit(td);
	}
	mcrypt_module_close(td);
	return ret;
}

void AES_Show(char* ciphertext, int len)
{
	int i;

	for (i = 0; i < len; i++)
		printf("%d", ciphertext[i]);

	printf("\n");
}
```

Decrypt.c

Decrypt.c

```
#include <stdio.h>
#include <stdlib.h>
#include "aes.h"
static FILE *AES_OpenFile(char *filename, char *mode)
{
	FILE *fp = fopen(filename, mode);

	if (!fp) {
		printf("ERROR: Can't open %s\n", filename);
		exit (1);
	}

	return fp;
}

int main (int argc, char *argv[])
{
	FILE *fpin, *fpout;

	uint8_t key[] = { 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
0x38, 0x39, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66};
	uint8_t IV[] = { 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49,
0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50};
	unsigned long fileLen;
	int keysize = 16;
	uint8_t * buffer_in;
	uint8_t * buffer_out;
```

```
int buffer_len = 128;

fpin = AES_OpenFile(argv[1], "rb");



fseek(fpin, 0, SEEK_END);
fileLen = ftell(fpin);
fseek(fpin, 0, SEEK_SET);

//Pass file stream to buffer
buffer_in = (uint8_t *) malloc(fileLen);
buffer_out = (uint8_t *) malloc(fileLen);
if(!buffer_in)
      exit(1);

fread(buffer_in, fileLen, 1, fpin);

fclose(fpin);

AES128_CBC_decrypt_buffer(buffer_out, buffer_in, fileLen, key, IV);

fpout = AES_OpenFile("output.ashx", "wb");

fwrite(buffer_out, fileLen, 1 , fpout);

close (fpout);

return 0;
```

```
}
```

Makefile principal

```
all:
	cd Cypher; make
	cd Sender; make

clean:
	cd Cypher; make clean
	cd Sender; make clean
	rm -rf output/*
```

Makefile para cifrado

```
CC=gcc
CRYPTLIB=-lmcrypt
CFLAGS=-c $(CRYPTLIB)
SOURCES=aes.c Encrypt.c
OBJECTS=$(SOURCES:.c=.o)
EXECUTABLE=Enc

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	$(CC) $(OBJECTS) $(CRYPTLIB) -o $@

.c.o:
	$(CC) $(CFLAGS) $< -o $@
```

```
clean:
	rm -v $(OBJECTS) $(EXECUTABLE)
```

Makefile para Sender

```
CC=gcc
INC=-I UART_Driver/
CFLAGS=-c $(INC)
SOURCES=FrameSender.c UART_Driver/UARTDvr.c
OBJECTS=$(SOURCES:.c=.o)
EXECUTABLE=Sender

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	$(CC) $(OBJECTS) $(INC) -o $@

.c.o:
	$(CC) $(CFLAGS) $< -o $@

clean:
	rm -v $(OBJECTS) $(EXECUTABLE)
```

Makefile para descrifrado (Es solo para usar en la maquina HOST, no para cross compilacion)

```
CC=gcc
CFLAGS=-c
```

```
SOURCES=aes.c Decrypt.c

OBJECTS=$(SOURCES:.c=.o)

EXECUTABLE=Dec


all: $(SOURCES) $(EXECUTABLE)


$(EXECUTABLE): $(OBJECTS)

	$(CC) $(OBJECTS) $(CRYPTLIB) -o $@


.c.o:

	$(CC) $(CFLAGS) $< -o $@


clean:

	rm -v $(OBJECTS) $(EXECUTABLE)
```

SendFrames.sh

```
#!/bin/bash

#Script to automate the crypher and Send frames on Rasperry

#	It got two parameters:

#	path for the file to encrypt

#	path where output will be stored


echo "Setting system to allow to work with more files for process"

ulimit -n 3000

$PWD/Cypher/Enc $1 $2

COUNT=`ls $2 | wc -w`

echo "Total files to send: $COUNT"
```

```
$PWD/Sender/Sender $COUNT $2
```