



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica
Superior d'Enginyeria
Informàtica



etsinf

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Generic Car Controller

Una unidad controladora auxiliar para
vehículos automóvil

TRABAJO FINAL DE GRADO

Grado en ingeniería informática

Autor	Álvaro Graciá Gil
Tutor	Antonio Martí Campoy
Curso	2018 – 2019



Esta obra está bajo una licencia de Creative Commons Reconocimiento – No Comercial – Compartir Igual 4.0 Internacional.

Dedicado a mi familia. Gracias a todos por darme la posibilidad de completar mis estudios y ayudarme durante todos estos años de formación y crecimiento personal.

RESUMEN

Este documento trata sobre el diseño y la implementación de una controladora para vehículos automóvil.

Está formada por un componente *hardware* basado en un sistema microcontrolador, el cual se integra a través del bus de datos CAN en los sistemas existentes en la mayoría de los vehículos actuales. Y, por otra parte, del *software* específico incluido en el propio microcontrolador, el cual gestiona las comunicaciones con el automóvil y le dota de características adicionales y personalizadas.

Algunos ejemplos de las funcionalidades implementadas son:

- El control de una radio no original a través de los botones incorporados en el volante
- Un sistema de diagnóstico que permite en tiempo real la obtención de múltiples parámetros del automóvil que normalmente son inaccesibles.
- Un sistema de geolocalización 24/7.
- El uso de la pantalla original del vehículo para mostrar notificaciones y textos personalizados.
- Control y configuración del sistema a través de una aplicación móvil para *Android*.

Palabras clave: Controladora para vehículos automóvil, *CAN BUS*, Geolocalización, *Arduino*, *Android*, *GPS*, *GSM*, *Traccar*, Potenciómetro digital, Sistemas Empotrados

RESUM

Aquest document tracta sobre el disseny i la implementació d'una controladora per a vehicles automòbil.

Està formada per un component *hardware* basat en un sistema microcontrolador, el qual s'integra a través el bus de dades CAN en els sistemes existents en la majoria dels vehicles actuals. I, d'altra banda, del *software* específic inclòs en el propi microcontrolador, el qual gestiona les comunicacions amb l'automòbil i el dota de característiques addicionals i personalitzades.

Alguns exemples de les funcionalitats implementades són:

- El control d'una ràdio no original a través dels botons incorporats al volant.
- Un sistema de diagnòstic que permet en temps real l'obtenció de múltiples paràmetres de l'automòbil que normalment són inacessibles.
- Un sistema de geolocalització 24/7.
- L'ús de la pantalla original del vehicle per mostrar notificacions i textos personalitzats.
- Control i configuració del sistema mitjançant una aplicació mòbil per a *Android*.

Paraules clau: Controladora per a vehicles automòbil, *CAN BUS*, Geolocalització, *Arduino*, *Android*, *GPS*, *GSM*, *Traccar*, Potenciòmetre digital, Sistemes Encastats.

ABSTRACT

This document deals with the design and the implementation of a custom controller for an automotive vehicle.

It consists of a hardware component based on a microcontroller system, which uses a CAN data bus to integrate with the existing systems available in the majority of current vehicles. And, on the other hand, the specific software included in the microcontroller itself, which manages the communications with the vehicle and gives it additional and personalized functions.

Some examples of the implemented functions are:

- Control of a non-original stereo system through the built-in buttons on the steering wheel.
- A diagnostic system that allows in real time to fetch multiple internal parameters those normally are inaccessible.
- A 24/7 geolocation platform.
- A notification manager which can show personalized texts on the original screen of the vehicle.
- Control and configuration of the system using a mobile application for *Android*.

Keywords: Vehicle controller, Automotive controller, *CAN BUS*, Geolocation, *Arduino*, *Android*, *GPS*, *GSM*, *Traccar*, Digital potentiometer. Embedded Systems.

ÍNDICE

ÍNDICE.....	9
ÍNDICE DE TABLAS.....	11
ÍNDICE DE ILUSTRACIONES.....	12
GLOSARIO DE TÉRMINOS.....	13
1 – INTRODUCCIÓN Y OBJETIVOS.....	17
1.1 – OBJETIVOS	17
2 – REQUISITOS.....	19
2.1 – REQUISITOS HARDWARE.....	19
2.2 – REQUISITOS SOFTWARE.....	19
2.3 – REQUISITOS FUNCIONALES.....	20
3 – DISEÑO E IMPLEMENTACIÓN HARDWARE.....	23
3.1 – ESQUEMA BÁSICO.....	23
3.2 – ELECCIÓN DE PLATAFORMA.....	23
3.3 – ALIMENTACIÓN	24
3.4 – COMPONENTES	25
3.5 – ESQUEMÁTICO	28
3.6 – PLACA BASE	31
3.7 – MECANIZADO	33
4 – DISEÑO E IMPLEMENTACIÓN SOFTWARE	35
4.1 – SOFTWARE DEL MICROCONTROLADOR.....	35
4.1.1 – Enfoque sobre el desarrollo	35
4.1.2 – Estructura interna	35
4.1.3 – Módulos software y funcionalidades.....	36
4.1.3.1 – Núcleo – Gestión del sistema.....	36
4.1.3.2 – ROM – Configuración en EEPROM	37
4.1.3.3 – CAN – Comunicaciones por bus	37
4.1.3.4 – RAD – Control de radio	38
4.1.3.5 – GPS – Sistema de posicionamiento global	39
4.1.3.6 – GSM – Comunicaciones por red móvil.....	39
4.1.3.7 – TRC – Traccar – Plataforma de seguimiento	40
4.1.4 – Bibliotecas adicionales.....	40
4.1.5 – Herramientas de desarrollo	41
4.1.6 – Protocolo de comunicaciones interno.....	41
4.1.6.1 – Acciones implementadas	42
4.2 – SOFTWARE DE CONTROL – APLICACIÓN MÓVIL / TABLET.....	43
4.2.1 – Herramientas de desarrollo	44
4.2.2 – Imágenes de la aplicación	44
4.3 – SOFTWARE DE SEGUIMIENTO – PLATAFORMA TRACCAR.....	45
4.3.1 – Configuración.....	46
4.3.2 – STunnel	47
4.3.3 – Protocolo OsmAnd.....	47
5 – ESPECIFICACIÓN Y ADAPTACIÓN DE VEHÍCULOS	49
5.1 – ESQUEMA BÁSICO PARA DEFINIR UN VEHÍCULO	49
5.2 – CASO ESPECÍFICO – OPEL ASTRA H.....	50

5.2.1 – Breve resumen sobre el vehículo	50
5.2.2 – Cableado	50
5.2.3 – Bus de datos.....	51
5.2.4 – Componentes del vehículo	52
5.2.4.1 – DIS –pantalla original	52
5.2.4.1.1 – Envío de texto a la pantalla	53
5.2.4.1.2 – Diagnóstico del vehículo – “Modo Test” o “Menú secreto”	54
5.2.4.2 – EHU – Entertainment Unit	55
5.2.4.3 – CIM – Column Integrated Module / Steering Column Module	56
5.2.5 – Acciones implementadas	57
7 – PRESUPUESTO	59
7.1 – HERRAMIENTAS.....	59
7.2 – COMPONENTES	59
7.3 – CONSUMIBLES.....	60
7.4 – DESARROLLO	60
7.5 – COSTES PERIÓDICOS	60
7.6 – TOTAL	61
8 – CONCLUSIÓN	63
8.1 – IMPACTO ESPERADO	63
8.2 – POSIBLES MEJORAS	63
9 – BIBLIOGRAFÍA	65

ÍNDICE DE TABLAS

TABLA 1 – PARÁMETROS DEL NÚCLEO EN TIEMPO DE COMPILACIÓN	36
TABLA 2 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO CAN	38
TABLA 3 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO POT	38
TABLA 4 – RELACIÓN DE VALORES RESISTIVOS Y ACCIONES WIRED REMOTE	38
TABLA 5 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO GPS.....	39
TABLA 6 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO GSM	39
TABLA 7 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO TRC.....	40
TABLA 8 – BIBLIOTECAS EMPLEADAS EN EL DESARROLLO.....	40
TABLA 9 – SINTAXIS BNF DE LOS COMANDOS DE CONFIGURACIÓN	41
TABLA 10 – ACCIONES – FUNCIONES DEL NÚCLEO – SYS.....	42
TABLA 11 – ACCIONES – FUNCIONES DEL NÚCLEO – OPT.....	42
TABLA 12 – ACCIONES – FUNCIONES DEL NÚCLEO – LOG	42
TABLA 13 – ACCIONES – FUNCIONES DEL NÚCLEO – POW.....	42
TABLA 14 – ACCIONES – FUNCIONES DEL MÓDULO – ROM.....	43
TABLA 15 – ACCIONES – FUNCIONES DEL MÓDULO – RAD	43
TABLA 16 – ACCIONES – FUNCIONES DEL MÓDULO – CAN	43
TABLA 17 – TRACCAR – CONFIGURACIÓN	46
TABLA 18 – TRACCAR – CREACIÓN DE CERTIFICADOS MEDIANTE LA UTILIDAD OPENSSL	47
TABLA 19 – TRACCAR – FICHERO DE CONFIGURACIÓN DE STUNNEL	47
TABLA 20 – TRACCAR – PARÁMETROS DEL PROTOCOLO OSMAND	48
TABLA 21 – ESPECIFICACIÓN DE VEHÍCULOS – EVENTOS.....	49
TABLA 22 – ESPECIFICACIÓN DE VEHÍCULOS – FICHERO DE CONFIGURACIÓN	49
TABLA 23 – OPEL ASTRA H – TIPOS DE PANTALLA	52
TABLA 24 – OPEL ASTRA H – FORMATO DEL MENSAJE DIS_TEXT_REQUEST.....	53
TABLA 25 – OPEL ASTRA H – FORMATO DEL MENSAJE DIS_REQUEST_ACK.....	54
TABLA 26 – OPEL ASTRA H – FORMATO DE LA SECUENCIA DE MENSAJES DIS_SEND_TEXT	54
TABLA 27 – OPEL ASTRA H – FORMATO DEL MENSAJE DE LA PULSACIÓN O LIBERACIÓN DE LOS BOTONES DEL VOLANTE.....	56
TABLA 28 – OPEL ASTRA H – EJEMPLO DE MENSAJES AL MANTENER UN BOTÓN DURANTE 350MS.....	57
TABLA 29 – OPEL ASTRA H – FORMATO DEL MENSAJE AL MOVER LAS RUEDAS DEL VOLANTE.....	57
TABLA 30 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – OPT	57
TABLA 31 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – LOG	57
TABLA 32 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – BC.....	58
TABLA 33 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – DIS.....	58
TABLA 34 – PRESUPUESTO – HERRAMIENTAS	59
TABLA 35 – PRESUPUESTO – COMPONENTES.....	59
TABLA 36 – PRESUPUESTO – CONSUMIBLES	60
TABLA 37 – PRESUPUESTO – DESARROLLO.....	60
TABLA 38 – PRESUPUESTO – COSTES PERIÓDICOS.....	60
TABLA 39 – PRESUPUESTO – TOTAL.....	61

ÍNDICE DE ILUSTRACIONES

ILUSTRACIÓN 1 – ESQUEMA BÁSICO	23
ILUSTRACIÓN 2 – ARDUINO NANO V3	25
ILUSTRACIÓN 3 – PLACA COMERCIAL MCP2515	25
ILUSTRACIÓN 4 – MANDO WIRED REMOTE	26
ILUSTRACIÓN 5 – CONECTOR JACK	26
ILUSTRACIÓN 6 – MCP4131	27
ILUSTRACIÓN 7 – PINOUT - MCP4131	27
ILUSTRACIÓN 8 – PC817X.....	27
ILUSTRACIÓN 9 – ESQUEMÁTICO DEL PC817X.....	27
ILUSTRACIÓN 10 – GY-NEO6MV2.....	27
ILUSTRACIÓN 11 – SIM800L	28
ILUSTRACIÓN 12 – HC-05 / HC-06	28
ILUSTRACIÓN 13 – DSN5000	28
ILUSTRACIÓN 14 – ESQUEMÁTICO V4.3 (CAN BUS)	29
ILUSTRACIÓN 15 – ESQUEMÁTICO V4.3 (ARDUINO)	29
ILUSTRACIÓN 16 – ESQUEMÁTICO V4.3 (UART, GPIO Y ALIMENTACIÓN)	29
ILUSTRACIÓN 17 – ESQUEMÁTICO V4.3 (BLUETOOTH)	30
ILUSTRACIÓN 18 – ESQUEMÁTICO V4.3 (GPS Y GSM)	30
ILUSTRACIÓN 19 – ESQUEMÁTICO V4.3 (CONTROL DE RADIO).....	30
ILUSTRACIÓN 20 – DISTRIBUCIÓN DE COMPONENTES (PLACA V4.3).....	31
ILUSTRACIÓN 21 – PLACA V4.3 (FRONTAL).....	31
ILUSTRACIÓN 22 – PLACA V4.3 (TRASERA)	31
ILUSTRACIÓN 23 – PROTOTIPOS DE PLACA (DE V1.0 A V4.3)	32
ILUSTRACIÓN 24 – VISTA DEL MECANIZADO (1).....	33
ILUSTRACIÓN 25 – VISTA DEL MECANIZADO (2).....	33
ILUSTRACIÓN 26 – VISTA DEL ENSAMBLADO COMPLETO	34
ILUSTRACIÓN 27 – ESQUEMA INTERNO DEL SOFTWARE DE LA CONTROLADORA	36
ILUSTRACIÓN 28 – ICONO OFICIAL DE ANDROID.....	43
ILUSTRACIÓN 29 – ICONO OFICIAL DE ANDROID STUDIO.....	44
ILUSTRACIÓN 30 – APLICACIÓN ANDROID – ACCIONES	44
ILUSTRACIÓN 31 – APLICACIÓN ANDROID – CONEXIÓN	44
ILUSTRACIÓN 32 – APLICACIÓN ANDROID – CONFIGURACIÓN	45
ILUSTRACIÓN 33 – APLICACIÓN ANDROID – REGISTRO	45
ILUSTRACIÓN 34 – ICONO OFICIAL DE TRACCAR	45
ILUSTRACIÓN 35 – TRACCAR – INTERFAZ WEB.....	45
ILUSTRACIÓN 36 – TRACCAR – ESQUEMA DE CONEXIONES DE RED CON STUNNEL	47
ILUSTRACIÓN 37 – OPEL ASTRA H	50
ILUSTRACIÓN 38 – ADAPTADOR PARA LA RADIO ORIGINAL	51
ILUSTRACIÓN 39 – PINES DE LA RADIO ORIGINAL (CD30).....	51
ILUSTRACIÓN 40 – ESQUEMA MS-CAN DEL OPEL ASTRA H	51
ILUSTRACIÓN 41 – MODELO DE PANTALLA TID	52
ILUSTRACIÓN 42 – MODELO DE PANTALLA BID	52
ILUSTRACIÓN 43 – MODELO DE PANTALLA GID.....	53
ILUSTRACIÓN 44 – MODELO DE PANTALLA CID	53
ILUSTRACIÓN 45 – “MODO TEST” O “MENÚ SECRETO” (1)	55
ILUSTRACIÓN 46 – “MODO TEST” O “MENÚ SECRETO” (2).....	55
ILUSTRACIÓN 47 – SISTEMA DE SONIDO CD30 MP3	55
ILUSTRACIÓN 48 – VOLANTE VAUXHALL H	56

GLOSARIO DE TÉRMINOS

Android: Es un sistema operativo basado en el núcleo de Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, aunque posteriormente se añadió soporte para relojes inteligentes, televisores y automóviles.

APN: De sus siglas "*Access Point Name*" o "Nombre del Punto de Acceso". Es un nombre de host que se emplea normalmente en telefonía y que sirve para obtener la dirección IP del proveedor del servicio de acceso a una red de datos de comunicación inalámbrica externa.

Arduino: Es una plataforma para crear prototipos electrónicos, la cual está basada en hardware y software libre. Cuenta con un gran abanico de modelos distintos según las necesidades que se requieran. Se creó en el año 2005 por Massimo Banzi para cubrir la necesidad académica del resto de estudiantes de su mismo centro, aunque posteriormente creció y acabó convirtiéndose en una de las plataformas más extendidas y reconocidas en su ámbito.

Bluetooth: Es un estándar abierto para transmitir información a través de redes Inalámbricas de área personal (WPAN). Permite la transmisión de voz y datos entre dispositivos utilizando un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz.

BNF: Notación de *Backus-Naur*. Es una técnica de notaciones usada para describir la sintaxis de lenguajes formales como lenguajes de programación, documentos, conjuntos de instrucciones o protocolos de comunicaciones.

CAD: De sus siglas "*Computer-Aided Design*" o "Diseño Asistido por Ordenador". Es el proceso de crear la representación de un objeto tridimensional a través de un software especializado.

CAN: De sus siglas "*Controller Area Network*". Es un protocolo de comunicaciones desarrollado por la firma alemana Bosch para el intercambio de información entre unidades de control empleando una topología en bus.

Controlador: Es un componente electrónico o un software que se integra entre varios sistemas y realiza funciones de gestión y control en el entorno que se integra.

EEPROM: De sus siglas "*Electrically Erasable Programmable Read-Only Memory*" o "Memoria de Sólo Lectura Borrable Electrónicamente". Es un tipo de memoria ROM no volátil que puede ser programada y borrada utilizando una corriente eléctrica.

ESP: Es el nombre que recibe el sistema de control de estabilidad en automóviles. Se trata de un dispositivo de seguridad activa que actúa frenando individualmente cada rueda en situaciones de riesgo para evitar que patine o derrape.

Geocerca: Es un perímetro virtual en un mapa, el cual permite establecer los límites de una ubicación física para realizar tareas de monitorización o seguimiento.

Gerber: Es un formato de archivo estandarizado en la industria, el cual contiene toda la información necesaria para la fabricación de un circuito impreso.

GPIO: De sus siglas "*General Purpose Input/Output*" o "Entrada/Salida de Propósito General". Es un terminal de un integrado que puede ser utilizado como una entrada para obtener el valor binario correspondiente a la señal que se encuentra en ese terminal, o también puede ser una salida y establecer el nivel lógico al que está el terminal.

GPS: De sus siglas "*Global Positioning System*" o "Sistema de Posicionamiento Global". Es un sistema de navegación que, mediante el uso de satélites, permite determinar la posición de cualquier objeto

alrededor del planeta con una precisión teórica de centímetros, aunque en la práctica son unos pocos metros.

GSM: De sus siglas *“Global System for Mobile Communication”* o *“Sistema global para las comunicaciones móviles”*. Es un estándar de telefonía móvil digital que define un sistema para la transmisión de voz, mensajes de texto y datos entre dispositivos móviles empleando antenas de telefonía distribuidas por el proveedor.

I2C: De sus siglas *“Inter-Integrated Circuit”* o *“Circuito Inter-Integrado”*. Es un tipo de bus serie de datos. Se utiliza principalmente para la comunicación entre diferentes integrados de un mismo circuito.

IDE: De sus siglas *“Integrated Development Environment”* o *“Entorno de Desarrollo Integrado”*. Es un conjunto de aplicaciones que proporcionan al programador herramientas y servicios con la intención de facilitar el desarrollo de software.

Interrupción: Es una señal recibida por el procesador para indicarle que debe atender una situación prioritaria. Esto detendrá el flujo de ejecución actual y pasará a ejecutar un código específico asociado a dicha señal. Tras esta ejecución, el flujo de ejecución se restaurará y continuará en el punto donde había sido interrumpido.

IoT: De sus siglas *“Internet Of Things”* o *“Internet De las Cosas”*. Es un concepto que se refiere a la interconexión digital total de objetos cotidianos y dispositivos a través de internet.

LED: De sus siglas *“Light-Emitting Diode”* o *“Diodo Emisor de Luz”*. Se trata de una fuente de luz constituida por un diodo de unión p-n que emite luz cuando está activo.

MOSFET: Es un dispositivo semiconductor utilizado para amplificar o conmutar señales electrónicas. Es el transistor más utilizado actualmente en la industria electrónica.

PCB: De sus siglas *“Printed Circuit Board”* o *“Placa de Circuito Impreso”*. Es una superficie que cuenta con caminos, pistas y buses de material conductor sobre una base no conductora. El circuito impreso se utiliza para conectar eléctricamente, y sostener mecánicamente distintos componentes electrónicos.

Polling: En castellano *“Sondeo”*. Se refiere a una consulta periódica, generalmente hacia un dispositivo de hardware.

Potenciómetro: Es un resistor variable, el cual varía su valor resistivo entre sus terminales en función de la posición indicada por el usuario.

Raspberry Pi: Es un ordenador de placa reducida de diseño libre. Cuenta con gran cantidad de software adaptado y ofrece altas prestaciones a bajo coste.

SIM: Es una tarjeta inteligente desmontable usada en teléfonos móviles y módems GSM. Las tarjetas SIM almacenan de forma segura en su interior la clave necesaria para identificarse ante la red del proveedor.

SPI: De sus siglas *“Serial Peripheral Interface”* o *“Interfaz Serie para Periféricos”*. Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados.

UART: De las siglas *“Universal Asynchronous Receiver-Transmitter”* o *“Transmisor-Receptor Asíncrono Universal”*. Es un dispositivo que controla los puertos y dispositivos serie. Normalmente se encuentra integrado en el microcontrolador o en una tarjeta adicional del dispositivo.

UTC: Es el principal estándar de medición de tiempo por el cual se regulan y sincronizan los relojes a nivel mundial.

Watchdog: Conocido como “Perro guardián”. Es un mecanismo de seguridad que provoca un reinicio del sistema en caso de que éste se haya bloqueado.

Wired/Remote: Protocolo utilizado para controlar equipos de sonido mediante un mando a distancia cableado. Lo incorporan los sistemas de las marcas *Sony*, *Kenwood* y *Pioneer*.

1 – INTRODUCCIÓN Y OBJETIVOS

Conforme los automóviles se han ido modernizando, los sistemas internos han dejado de ser únicamente mecánicos y se han convertido en sistemas inteligentes, controlados mediante microcontroladores repartidos por todo el vehículo. Actualmente los automóviles cuentan con decenas de microcontroladores y dispositivos interconectados entre sí¹.

Para ello principalmente utilizan una tecnología de comunicación en bus llamada *CAN BUS*. Esta tecnología fue estandarizada (*ISO 11898 [14]*) a finales de la década de 1990 y define un protocolo de comunicaciones que es ampliamente utilizado en automoción y en la industria. Por ejemplo, algunos ascensores o robots se comunican utilizando *CAN BUS*.

Este documento presenta el desarrollo de una controladora genérica que se conecta al bus de datos de un automóvil y le permite al usuario acceder a nuevas funciones. Dichas funciones no están limitadas a un único modelo de vehículo determinado, ya que el proyecto se estructura de forma genérica para que resulte sencillo implementar nuevas funcionalidades y a su vez sean abstraídas de forma que múltiples modelos y marcas se beneficien de las mismas prestaciones con poco esfuerzo en la implementación.

Resulta muy común que los vehículos dispongan de funcionalidades *extra* que se encuentran bloqueadas de fábrica en algunos modelos o versiones de gama media/baja. También es frecuente que los sistemas internos se desarrollen para que únicamente funcionen sin limitaciones con dispositivos originales o autorizados. Un ejemplo representativo son los sistemas de sonido, donde es posible que al cambiar la radio original por otra no oficial dejen de funcionar ciertas características, como los controles integrados en el volante o incluso el sistema de “manos libres”.

También ocurre que ciertas opciones se encuentran deshabilitadas para el mercado general porque no resultan útiles para la mayoría de los clientes, aunque sí lo son para otros. Un ejemplo práctico es la posibilidad de obtener en tiempo real algunos datos del vehículo para analizarlos y detectar valores anómalos o posibles averías.

En esos casos y más tiene cabida una controladora adicional personalizada, la cual permite aumentar las prestaciones que ofrece el sistema de fábrica e interceptar información útil directamente desde los sensores y otras unidades del vehículo.

1.1 – OBJETIVOS

El objetivo principal de este trabajo es crear un dispositivo el cual se conecte e integre en un vehículo automóvil con la finalidad de incrementar sus funcionalidades.

Tanto el diseño del apartado hardware como el apartado software deben ser altamente genéricos y parametrizables de forma que se adapten al amplio abanico de posibles modelos de automóvil existentes en el mercado.

¹ Artículo sobre la evolución de las unidades electrónicas en automóviles <https://www.embitel.com/blog/embedded-blog/automotive-control-units-development-innovations-mechanical-to-electronics>

Debido al entorno donde debe funcionar, el ensamblado completo debe ser de un tamaño reducido para facilitar su colocación en el interior del vehículo, además de tener un bajo consumo energético.

Por otra parte, las funcionalidades adicionales deben ser completamente configurables y adaptables, de forma que con simples ajustes funcionen con cualquier modelo de automóvil.

Algunas de estas funcionalidades deben ser:

- Acceder a la información interna del automóvil a través de la pantalla integrada en él.
- Integrar los controles del volante y el salpicadero con equipos de audio multimarca.
- Geolocalizar el vehículo en tiempo real.
- Control remoto del sistema de forma inalámbrica.

2 – REQUISITOS

A continuación, se presentan los requisitos que debe cumplir la controladora. Estos requisitos se desglosan en requisitos hardware, software y funcionales.

2.1 – REQUISITOS HARDWARE

- Debe ser compatible a nivel eléctrico con el vehículo.
- Debe comunicarse con el automóvil a través de un bus de datos *CAN*. Por razones de seguridad, este bus de datos debe estar diferenciado del empleado en el bloque motor del vehículo.
- El computador utilizado debe disponer al menos de las siguientes características:
 - Al menos 2KB de memoria RAM y 32KB para el código.
 - Una interfaz UART.
 - Conectividad SPI e I2C.
 - Conectores *GPIO* adicionales que puedan ser empleados para manejar periféricos sencillos como por ejemplo LEDs o módulos de radio frecuencia.
- Debe incluir una controladora *CAN BUS* conectada a la CPU.
- Toda la controladora y sus componentes deben estar ensamblados en una única placa de pequeñas dimensiones, la cual se colocará dentro de un mecanizado para evitar problemas como falsos contactos y derivaciones eléctricas.

2.2 – REQUISITOS SOFTWARE

- El desarrollo debe realizarse manteniendo un equilibrio entre eficiencia, consumo reducido de memoria RAM y un tamaño del ejecutable aceptable.
- El código debe seguir una estructura que facilite su fiabilidad y seguridad.
- El sistema debe arrancar y ser completamente funcional en pocos segundos.
- Debe tener una estructura que permita añadir o quitar funcionalidades de forma sencilla. Estas funcionalidades se añadirán en forma de **módulos software**, los cuales serán partes de código enfocadas a dotar a la controladora y/o al vehículo de nuevas características.
- La forma de implementar las nuevas funcionalidades por parte de los módulos software debe ser lo más **genérica** posible, de forma que sean válidas para cualquier modelo de vehículo cambiando parámetros en la configuración o añadiendo pequeños códigos únicos de cada modelo de automóvil.
- Tanto el software principal como los módulos software expondrán al exterior una serie de **acciones** que serán registradas en el arranque. Estas acciones serán las empleadas por el usuario y por el resto de módulos para utilizar las funciones que aporten.
- El software debe contar con un fichero de **configuración** donde se indiquen los **parámetros y módulos a emplear en tiempo de compilación**.
 - Los parámetros deben estar en forma de sentencias del pre-procesador o expresiones constantes.
 - Se debe poder indicar el vehículo y los módulos que se van a utilizar para que en tiempo de compilación se enlace el código y se carguen las configuraciones específicas para ese automóvil.

- Se debe poder indicar la velocidad de funcionamiento del *CAN BUS* y la frecuencia del oscilador conectado al módulo *CAN*.
- Deben poder existir **opciones** modificables en **tiempo de ejecución**. Esta serie de opciones deben contar con algún sistema para almacenarlos de forma persistente y que no se pierdan tras el apagado.
- Debe incluir un sistema de **registro de información**, de forma que sea posible visualizar el funcionamiento de la controladora en tiempo real a través de la interfaz serie.
- Debe haber un **sistema de comunicaciones** bidireccional que emplee la interfaz serie para recibir **comandos**. Este sistema de comunicaciones debe poder:
 - Ver, modificar o eliminar las **opciones** en tiempo real.
 - Gestionar el sistema de **registro** para detallar qué debe ser mostrado.
 - Invocar distintas **acciones** dependiendo del **comando** recibido.
 - Los comandos deben estar en formato texto para facilitar su escritura al usuario.
 - Incluir las siguientes acciones de forma predeterminada:
 - Obtener la memoria RAM libre del sistema.
 - Configurar el *watchdog* y las interrupciones del sistema.
 - Reiniciar el sistema.
 - La misma **acción** debe poder ser registrada varias veces desde distintos módulos software para poder ser manejada por todos.

2.3 – REQUISITOS FUNCIONALES

Estos requisitos centran en las funcionalidades adicionales para controladora en sí y para el automóvil. Es importante destacar que estos requisitos dependen directamente de las funciones adicionales que se vayan a emplear.

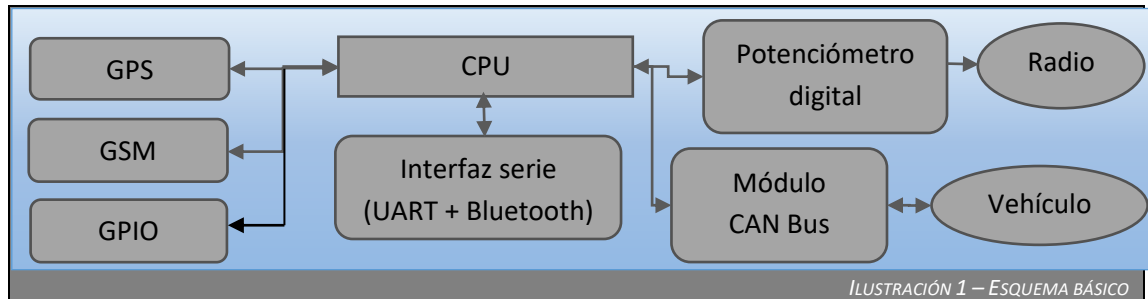
- Interceptar los mensajes CAN asociados a las pulsaciones de los botones físicos que se sitúan en el salpicadero, en el volante o cualquier otro lugar accesible para el usuario y asociarlas de forma sencilla a la ejecución de acciones personalizadas.
- Controlar equipos de sonido que cumplen el protocolo Wired/Remote, los cuales están diseñados especialmente para automóviles.
 - Para ello se debe conectar un potenciómetro digital que soporte un rango de resistencias entre 0Ω y 62.5Ω .
 - También se debe conectar algún sistema para conmutar señales de forma digital, ya sea por GPIO, SPI o I2C.
- Geolocalizar el automóvil y obtener información sobre él en tiempo real a través de internet garantizando la privacidad y la seguridad de las conexiones.
 - A nivel de hardware se debe conectar por UART, SPI o I2C:
 - Un modem GSM.
 - Un receptor GPS que soporte el estándar NMEA 0183 [8] [15].
 - Requiere de una plataforma de seguimiento accesible a través de internet.
- A través de una aplicación móvil, acceder a las acciones, a las opciones y a los registros de forma inalámbrica.
 - Requiere de un módulo Bluetooth conectado al UART principal.

- Requiere de un terminal móvil o tablet que disponga de conectividad Bluetooth.
- A través de mensajes CAN, utilizar la pantalla integrada del automóvil para mostrar textos con caracteres alfanuméricos, ya que va a ser el medio principal para visualizar datos directamente desde la controladora.
- En caso de que el texto a mostrar tenga una longitud demasiado larga y no quepa en las dimensiones de la pantalla, se debe realizar un efecto de marquesina deslizante para que se visualice el texto progresivamente.
- El sistema debe ser capaz de acceder a la configuración básica del vehículo para leer o modificarla. Por motivos de seguridad, dichos parámetros no pueden estar relacionados directamente con la conducción. Ejemplo de configuración sí accesible: la hora y fecha del automóvil.
- El software debe incluir un modo de simulación de vehículo. A rasgos generales, debe ser capaz de invertir las funciones de la controladora para que esta actúe como un vehículo en sí y se puedan realizar pruebas fuera del automóvil simulando un entorno completo.

3 – DISEÑO E IMPLEMENTACIÓN HARDWARE

3.1 – ESQUEMA BÁSICO

El apartado de hardware sigue el esquema de la ilustración 1:



- La CPU gestiona todos los componentes y decide qué acciones llevar a cabo según los datos de entrada que reciba. Principalmente la información se recibe desde el *CAN BUS*, aunque también puede llegar de la mano del módulo GSM o de la interfaz serie.
- Los módulos GPS y GSM en conjunto permiten geolocalizar el automóvil y comunicar su posición a un servidor, además de notificar distintos cambios en el vehículo como el estado del contacto, voltaje de la batería, excesos de velocidad, etc...
- El GPIO sirve para extender la funcionalidad de la controladora mediante dispositivos externos sencillos como LEDs o módulos de radiofrecuencia.
- El potenciómetro se emplea para enviar órdenes a equipos de sonido.
- El módulo CAN (o módulos CAN, no está limitado a un único bus) es uno de los componentes más importantes que forman la controladora del vehículo. Gestiona todas las comunicaciones con el automóvil y por ello se vuelve un componente crítico e imprescindible.
- La interfaz serie permite interactuar con el software de la controladora a través de comandos y gestionar la controladora en tiempo real.

3.2 – ELECCIÓN DE PLATAFORMA

Durante la fase de diseño se valoraron distintas plataformas que tomar como base para construir el sistema. Principalmente se plantearon Raspberry Pi [10] y distintos modelos de Arduino [11].

Se llevaron a cabo distintas pruebas con la Raspberry Pi. La velocidad de respuesta era muy rápida, pero tenía una serie de problemas:

- La Raspberry Pi necesita un sistema operativo para funcionar, lo que implica un arranque mucho más lento en comparación a un microcontrolador sin sistema operativo. Se probó una distribución de Linux basada en Debian además de Windows IoT [3]. Ambos sistemas eran muy lentos al arrancar, ya que comenzaban a funcionar completamente pasadas unas decenas de segundos o incluso minutos.
- Aunque el consumo eléctrico de una Raspberry Pi es bajo en términos absolutos, es demasiado elevado si pensamos en dejar el sistema siempre encendido alimentándose de la batería del automóvil. La Raspberry Pi consume alrededor de 200mA/h mientras

que, por ejemplo, un Arduino Nano con todos los componentes adicionales empleados (GPS, GSM, CAN, etc...) en funcionamiento ronda los 80mA/h.

- Por la forma que tiene resulta complicado situar componentes uno al lado del otro, obligando a colocarlos encima a modo de *shield*. Esto limita la libertad de diseño del hardware y dificulta la creación del circuito con dimensiones reducidas.
- Durante su funcionamiento, la Raspberry Pi puede aumentar mucho su temperatura, lo cual obligaría a utilizar algún sistema de refrigeración activo como un ventilador. Esto dificultaría la integración en el vehículo.
- El coste económico es mayor en comparación a otras posibles soluciones.

Tras esto se decidió tomar Arduino como la plataforma a emplear, ya que su diversidad de modelos permitió elegir el componente idóneo cumpliendo con los objetivos.

3.3 - ALIMENTACIÓN

La mayoría de los automóviles trabajan a tensiones de 12V o raramente a 24V [16], aunque carecen de estabilidad en la señal. Por ejemplo, durante el arranque, la tensión de un automóvil que funciona a 12V puede bajar hasta 9V y un par de segundos después, cuando el alternador comienza a generar electricidad, se producen picos de alrededor de 14.5V.

Para conseguir regular la tensión de entrada a los 5V de funcionamiento que requiere el Arduino, este cuenta con un regulador de voltaje lineal integrado que admite tensiones entre 6V y 20V. Para el caso de algunos vehículos este rango puede no ser lo suficientemente amplio. Y aunque así sea, es muy susceptible a quemarse tras múltiples encendidos debido a la inestabilidad de la tensión.

Por ello se decide utilizar una fuente conmutada a la entrada que estabilice la tensión a 5V. Estas fuentes rondan el 98% de eficiencia y no aumentan tanto su temperatura en funcionamiento en comparación a los reguladores lineales.

Para alimentar el Arduino Nano V3 con una fuente externa, en lugar de usar el pin *VIN* se utiliza el pin *+5V*, puenteando así el regulador interno y evitando que entre en funcionamiento. Hay que tener en cuenta que en algunos modelos de Arduino esto deshabilita también el pin de *+3.3V*.

En el caso de utilizar el modem GSM SIM800L debemos tener en cuenta que este se alimenta a 3.7V (aunque algunos fabricantes indican que funciona correctamente a 5V) y que en las fases de conexión a la red requiere de unos picos de corriente muy elevados (rondando el amperio), por lo que es recomendable utilizar otra fuente sólo para este módulo.

Al ser un dispositivo que potencialmente va a estar encendido siempre y alimentándose de una batería es de vital importancia tener en cuenta el consumo total del sistema para no agotar completamente la batería del automóvil y mermar el arranque del vehículo.

Se recomienda utilizar una batería adicional para esta controladora o tomar la alimentación directamente del contacto para que sólo funcione con el automóvil encendido y generando corriente mediante el alternador.

3.4 – COMPONENTES

A continuación, se van a citar los componentes electrónicos con los que se ha realizado la implementación del proyecto. Estos componentes permiten alcanzar los objetivos y requisitos de este proyecto además de mantener un buen equilibrio entre rendimiento y coste económico.

Es importante destacar que el proyecto permite el uso de hardware diferente al mostrado tras este párrafo. Dependiendo del vehículo en que se vaya a emplear y de las características del mismo es posible que se tengan que cambiar ciertos componentes o que simplemente no sean todos necesarios.

CPU – ARDUINO NANO V3 (ATMEGA 328P)

Por su sencillez, facilidad de compra, precio reducido y documentación disponible, emplear Arduino es una opción muy atractiva. En el caso del Nano V3 su reducido tamaño es una ventaja añadida a la hora de integrarlo en un circuito que en circunstancias normales se colocará en pequeños huecos (guantero, bajo alguna tapa, detrás de la radio, etc...)



ILUSTRACIÓN 2 –
ARDUINO NANO V3

En la ilustración 2 se puede ver el Arduino Nano V3.

Incorpora el procesador ATmega 328P, el cual dispone de 32KB de memoria Flash para el código a ejecutar y 2KB de memoria RAM. Estas características son suficientes para almacenar el ejecutable y los datos que requiere la controladora.

Este modelo de Arduino incorpora una interfaz UART hardware y 20 pines GPIO, los cuales son más que suficientes para este proyecto. Existen herramientas que permiten utilizar cualquier par de pines GPIO como una interfaz UART emulada por software. Aunque su rendimiento es menor, está comprobado que es suficiente y funciona correctamente con los componentes descritos a continuación.



ILUSTRACIÓN 3 –
PLACA COMERCIAL MCP2515

CAN BUS – MCP2515

Microchip dispone de un integrado llamado MCP2515, el cual gestiona las comunicaciones CAN siguiendo el estándar **ISO 11898**, el cual define un protocolo de comunicaciones en bus y es el que principalmente emplean los automóviles.

En la ilustración 3 se puede ver una placa comercial que ensambla el integrado MCP2515 y el transceptor TJA1050.

Las comunicaciones con el integrado se realizan mediante el protocolo *SPI*. Este integrado dispone de unos buffers que almacenan un máximo de dos mensajes en recepción y hasta tres para el envío, lo que resulta más que suficiente para las funcionalidades implementadas y las velocidades de bus típicas en los automóviles (generalmente un máximo de 500Kbit/s)

Para reducir la cantidad de mensajes recibidos a procesar, el integrado dispone de seis filtros que permiten establecer los identificadores válidos y descartar los mensajes indeseados. De

esta forma se previenen problemas de desbordamiento en los buffers de recepción y transferencias innecesarias entre el integrado y la CPU.

Es muy importante elegir bien y tener presente el oscilador que acompaña al MCP2515, pues este decidirá los parámetros temporales que se deben utilizar a la hora de configurar la velocidad del bus de datos. Para ello resulta conveniente calcularlos previamente ² y ver si genera alguna desviación temporal que pueda comprometer las comunicaciones.

POTENCIÓMETRO—MCP4131

No existe un estándar oficial que permita manejar los equipos de sonido, aunque afortunadamente existe un protocolo acordado entre varias marcas líderes en el sector.

Esta tecnología se conoce como **“Wired Remote (W/R)”** y está presente en equipos de sonido de las marcas *Sony*, *Pioneer* y *Kenwood*. Estas marcas copan el mercado de las autoradios de gama media y alta.

Esta tecnología emplea un conector Jack tripolar hembra de 3.5mm situado en la parte trasera de la radio para conectar un mando a distancia fabricado y suministrado por la marca. Se puede ver el mando en la ilustración 4 y el conector Jack macho en la ilustración 5.



ILUSTRACIÓN 4 – MANDO WIRED
REMOTE

Este mando simplemente cuenta en su interior con un conjunto de resistencias conectadas entre los botones y los terminales del conector en el extremo del cable. Cada acción enviada a la radio se corresponde con el valor específico de una resistencia entre los terminales **“Tip”** y **“Sleeve”** del conector Jack.

De esta forma, el equipo de sonido mide la resistencia entre los terminales del conector Jack y procesa la señal correspondiente.

Las señales enviadas al equipo de sonido deben mantenerse activas durante un tiempo mínimo de 50 milisegundos para ser procesadas correctamente y hasta un máximo de 110 milisegundos antes de que se asuma como una acción nueva y se procese repetidas veces.

El Jack cuenta con un terminal más llamado **“Ring”**, el cual si se conecta el terminal **“Sleeve”** se puede variar la acción enviada, duplicando así la cantidad de señales posibles.

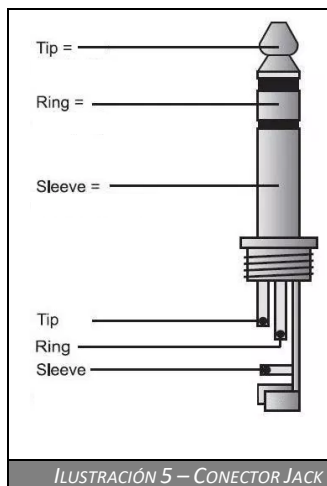


ILUSTRACIÓN 5 – CONECTOR JACK

Por razones de seguridad, se recomienda incorporar un diodo entre estos terminales para evitar corrientes de retorno hacia el equipo de sonido.

La forma más sencilla para variar la resistencia a voluntad a través de código software es mediante el uso de un potenciómetro digital.

² CAN Bus Bit Timing Calculator - <https://www.kvaser.com/support/calculators/bit-timing-calculator/>



El **MCP4131** es el integrado idóneo, pues su resistencia máxima es de 100K Ω con una precisión de 129 pasos (770 Ω por paso), lo que es suficiente para los valores que se indican en los requisitos. Se puede ver el encapsulado en la ilustración 6.

ILUSTRACIÓN 6 –
MCP4131

La ilustración 7 muestra las conexiones de que dispone este potenciómetro. Su interfaz de comunicaciones es SPI al igual que la controladora CAN, lo cual facilita el desarrollo y reduce el uso de pines GPIO necesarios a sólo un pin adicional. (Pin 1 \rightarrow CS / Chip Select)

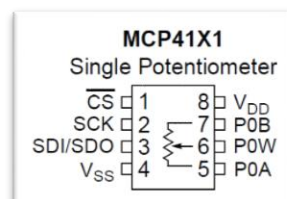


ILUSTRACIÓN 7 –
PINOUT - MCP4131

OPTOACOPLADOR – PC817X



ILUSTRACIÓN 8 –
PC817X

Para enviar señales a los equipos de sonido, además del potenciómetro, para el control de la radio es necesario incorporar un mecanismo para interconectar el terminal “Ring” con el terminal “Sleeve” del conector Jack.

Para ello existe un abanico de opciones como transistores, MOSFETs o incluso relés [5].

El optoacoplador **PC817X** (se puede ver en la ilustración 8) es un componente adecuado ya que cumple con los requisitos y tiene un coste muy bajo. Este optoacoplador está formado por un encapsulado que incluye en su interior un diodo LED y un transistor que se activa mediante luz. Estos componentes están separados entre sí, pero se posicionan de forma que al aplicar tensión al LED este se ilumina y activa a su vez la circulación de corriente a través del transistor (ver ilustración 9).

Las principales ventajas de este sistema son:

- La conmutación se realiza sin partes mecánicas, lo que aumenta la vida útil del componente al evitar desgaste.
- No crean grandes campos magnéticos que puedan generar ruido eléctrico en la señal.
- Tienen un consumo eléctrico casi despreciable.
- Son de un tamaño muy reducido.
- Tienen un coste económico muy bajo.

Debido a la tensión de funcionamiento del sistema y la tensión requerida por el diodo LED incorporado en el PC817X, es necesario colocar una resistencia a la entrada del componente para cumplir sus requisitos eléctricos y evitar daños. En este caso, con una resistencia de 220 Ω es suficiente [13].



ILUSTRACIÓN 9 –
ESQUEMÁTICO DEL PC817X

GPS – GY-NEO6MV2



ILUSTRACIÓN 10 –
GY-NEO6MV2

El receptor GPS elegido es el **GY-NEO6MV2**. Este receptor cumple con los requisitos, ya que utiliza el estándar **NMEA 0183** [8] [15]. Se puede ver el receptor GPS en la ilustración 10.

La tensión a la que funciona este componente es la misma que el Arduino, por lo que la conexión se realiza directamente y sin complicaciones.

Incluye una antena cerámica de reducidas dimensiones, la cual puede ser fácilmente adherida en su parte trasera.

MODEM GSM – SIM800L



ILUSTRACIÓN 11 –
SIM800L

El modem GSM SIM800L (se puede ver en la ilustración 11) es un componente que dota de la opción de conectarse a la red móvil y comunicarse por internet.

Gracias a esto es posible conectarse con la controladora de forma remota o enviar al servidor ciertos parámetros del vehículo en tiempo real.

Dispone de una interfaz UART para establecer las configuraciones y enviar las solicitudes por la red mediante comandos AT, los cuales son ampliamente utilizados y conocidos en este tipo de dispositivos.

En el reverso cuenta con un zócalo para conectar una tarjeta *Micro SIM*, la cual se empleará para identificarse en la red GSM del proveedor correspondiente.

BLUETOOTH – HC-05 / HC-06



ILUSTRACIÓN 12 –
HC-05 / HC-06

Para poder acceder a la controladora de forma inalámbrica se ha escogido el protocolo *Bluetooth*, ya que está ampliamente extendido en dispositivos móviles. El módulo HC-05/HC-06, el cual se puede ver en la ilustración 12, resulta adecuado para los requisitos del proyecto.

Debido a que el Arduino emplea una tensión de +5V para representar el nivel alto lógico en sus salidas, es necesario disminuir el voltaje hasta +3.3V al pin RX del módulo bluetooth. La forma más sencilla es mediante un divisor resistivo (es suficiente con resistencias de 2.2k Ω y 1k Ω), pero esto provoca un descenso en la velocidad máxima de transferencia y como máximo funciona correctamente a 115.200 baudios/s

FUENTE DE ALIMENTACIÓN – DSN5000

En esta implementación se ha empleado la fuente de alimentación conmutada **DSN5000** para suministrar a todo el hardware la electricidad en el voltaje correcto según los requisitos. Se puede ver el componente en la ilustración 13.



ILUSTRACIÓN 13 –
DSN5000

3.5 – ESQUEMÁTICO

A continuación, se detalla todo el esquemático de conexiones empleado, el cual corresponde a la versión 4.3 de la placa.

En la ilustración 14 se muestra el esquemático referente a la controladora *CAN BUS*. El conector J1 corresponde a la interfaz de comunicaciones con el MCP2515, mientras que los conectores J2 y J3 corresponden al conector de la placa base y al conector del propio bus del CAN.

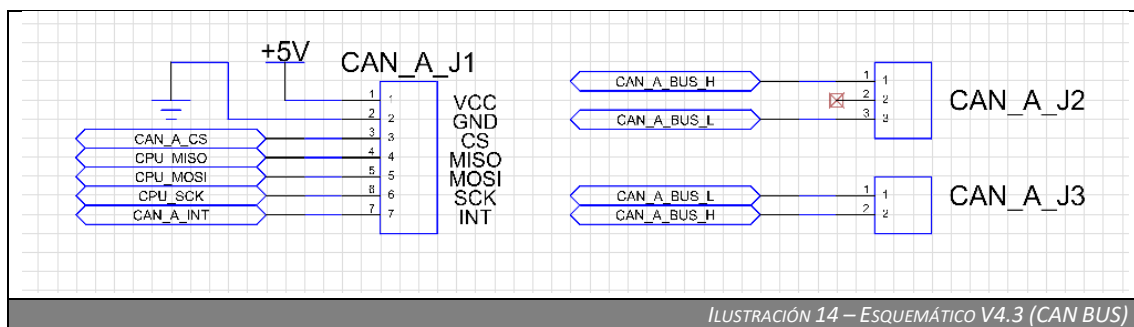


ILUSTRACIÓN 14 – ESQUEMÁTICO V4.3 (CAN BUS)

La ilustración 15 muestra todo el conexionado con el Arduino Nano V3.

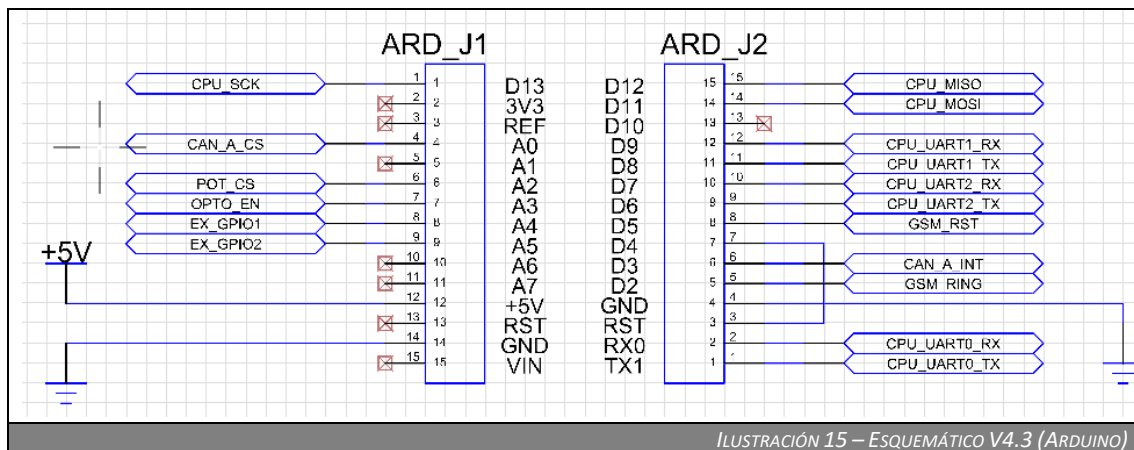


ILUSTRACIÓN 15 – ESQUEMÁTICO V4.3 (ARDUINO)

En la ilustración 16 se muestra el conector de alimentación, el conector externo GPIO y la interfaz serie externa. Esta interfaz serie cuenta con un conmutador para elegir si se conecta al puerto UART hardware del Arduino (el cual se utiliza para programarlo) o a un puerto UART alternativo.

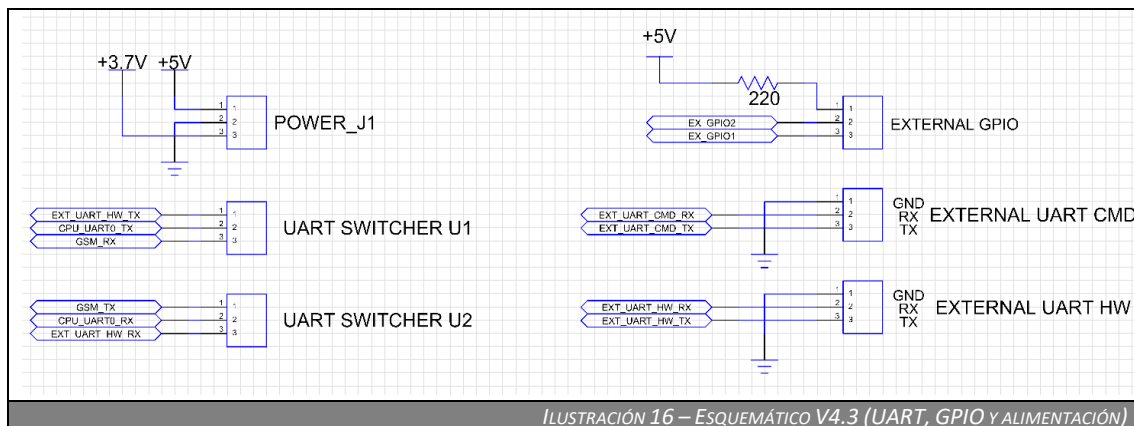
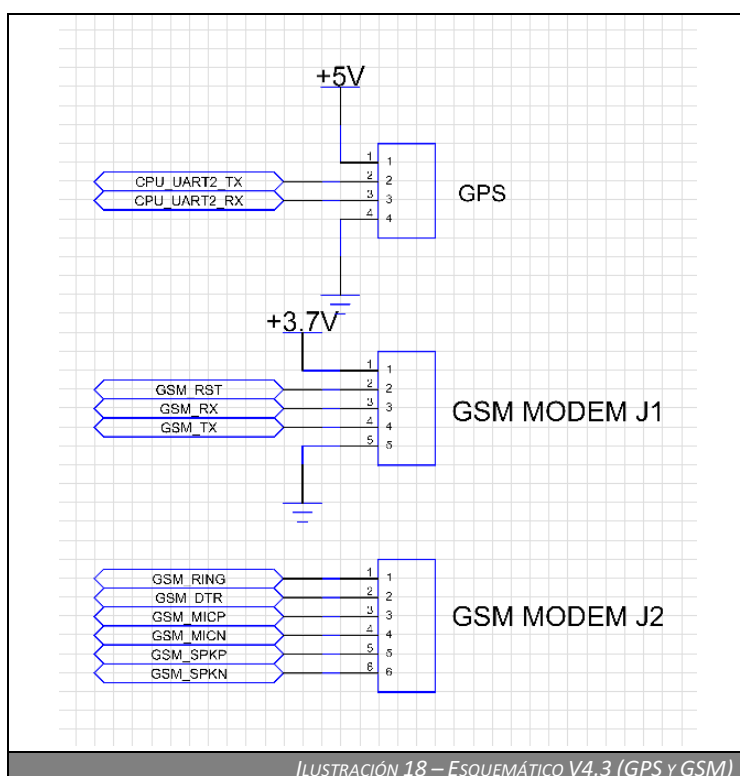
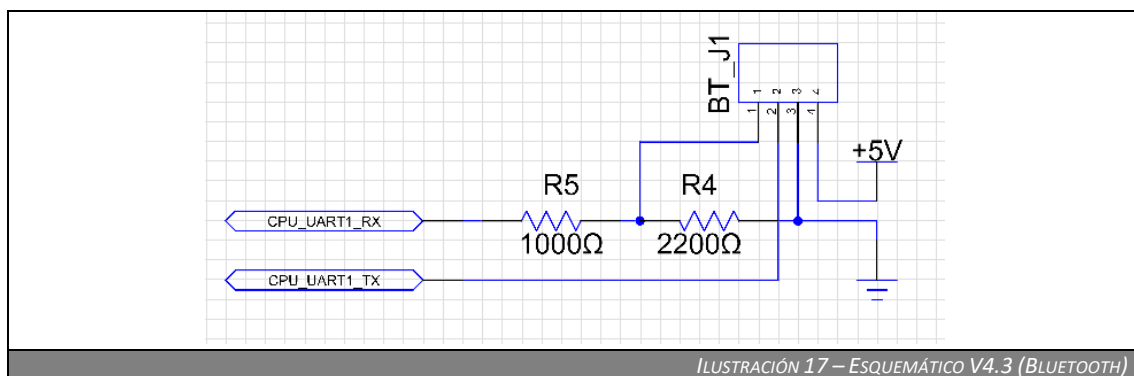


ILUSTRACIÓN 16 – ESQUEMÁTICO V4.3 (UART, GPIO Y ALIMENTACIÓN)

En la ilustración 17 se muestra el esquema de conexionado para el módulo hardware Bluetooth. En él se puede ver un divisor resistivo que adapta la tensión de las señales y le permite conectarse con el Arduino.

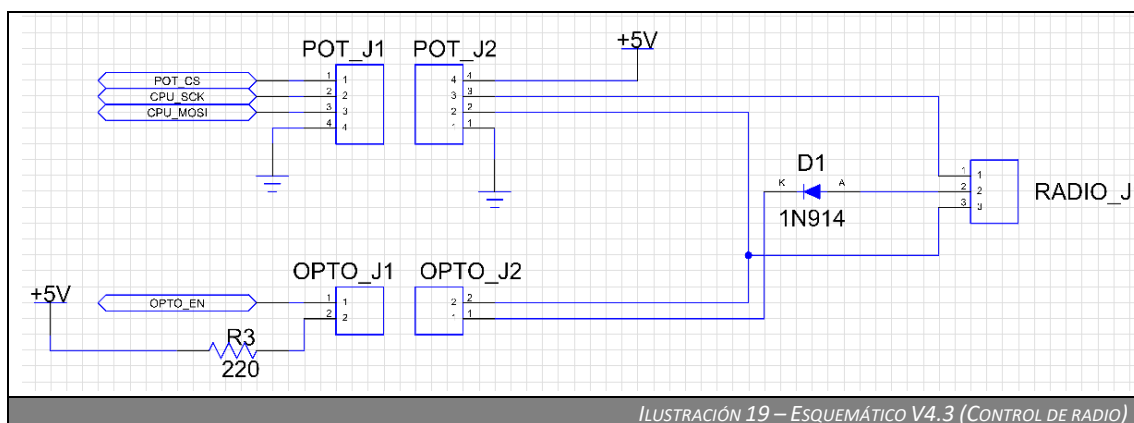


La ilustración 18 muestra el esquema de conexionado del receptor GPS y el modem GSM.

La conexión de la antena GPS no se incluye en este esquemático, ya que va directamente conectada al receptor GPS.

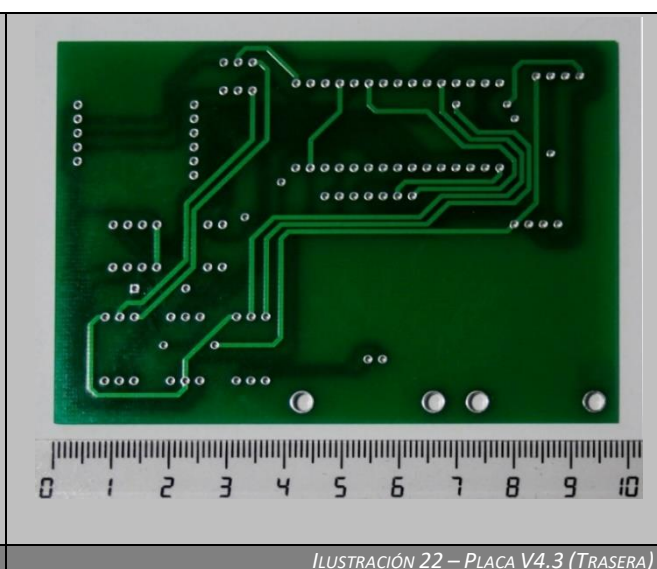
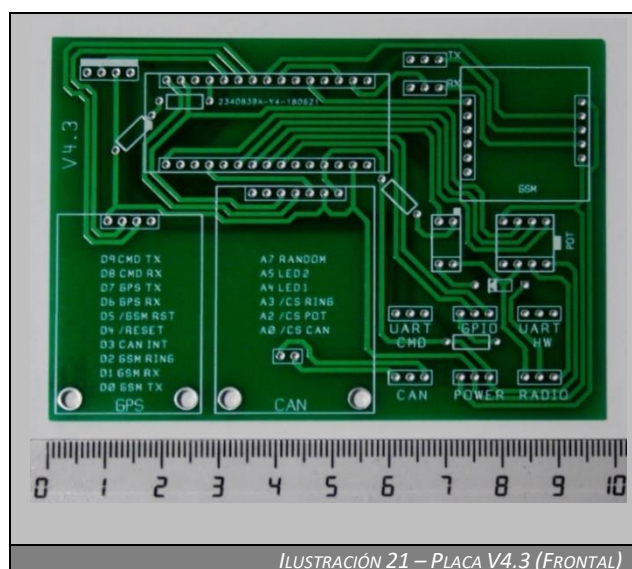
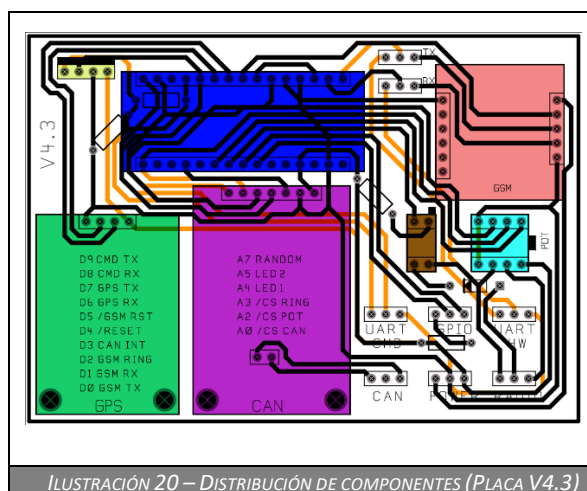
El conector J1 del modem GSM corresponde a las conexiones de un lado, mientras que el conector J2 corresponde al lado opuesto.

La ilustración 19 muestra el esquema de conexionado del sistema de control de la radio. Este utiliza el potenciómetro MCP4131 y un optoacoplador PC817X. Además, incluye un diodo para evitar corrientes de retorno hacia el equipo de sonido.



3.6 – PLACA BASE

Para garantizar la correcta interconexión de todos los componentes se ha diseñado una placa de circuito impreso (PCB). El diseño y la placa se pueden ver en las ilustraciones 20, 21 y 22. Debido al número de componentes y a su conexionado, la placa requiere pistas por ambas caras, lo que da lugar a un proceso de construcción un poco complejo y dificulta la construcción manual.



El diseño de la placa se ha realizado mediante la utilidad PCB Web ³ debido a su reducida curva de aprendizaje. Resulta muy sencilla de utilizar y ofrece buenos resultados. Permite exportar el diseño en formato *Gerber* y de esta forma enviarlo a un fabricante.

En este caso se han fabricado con la compañía JLCPCB ⁴ debido a su relación entre precio, calidad del producto y velocidad de envío.

³ PCBWeb - <http://www.pcbweb.com/>

⁴ JLCPCB - <https://jlcpcb.com/>

A continuación, en la ilustración 23 se muestra una comparativa entre los distintos prototipos y la versión final.

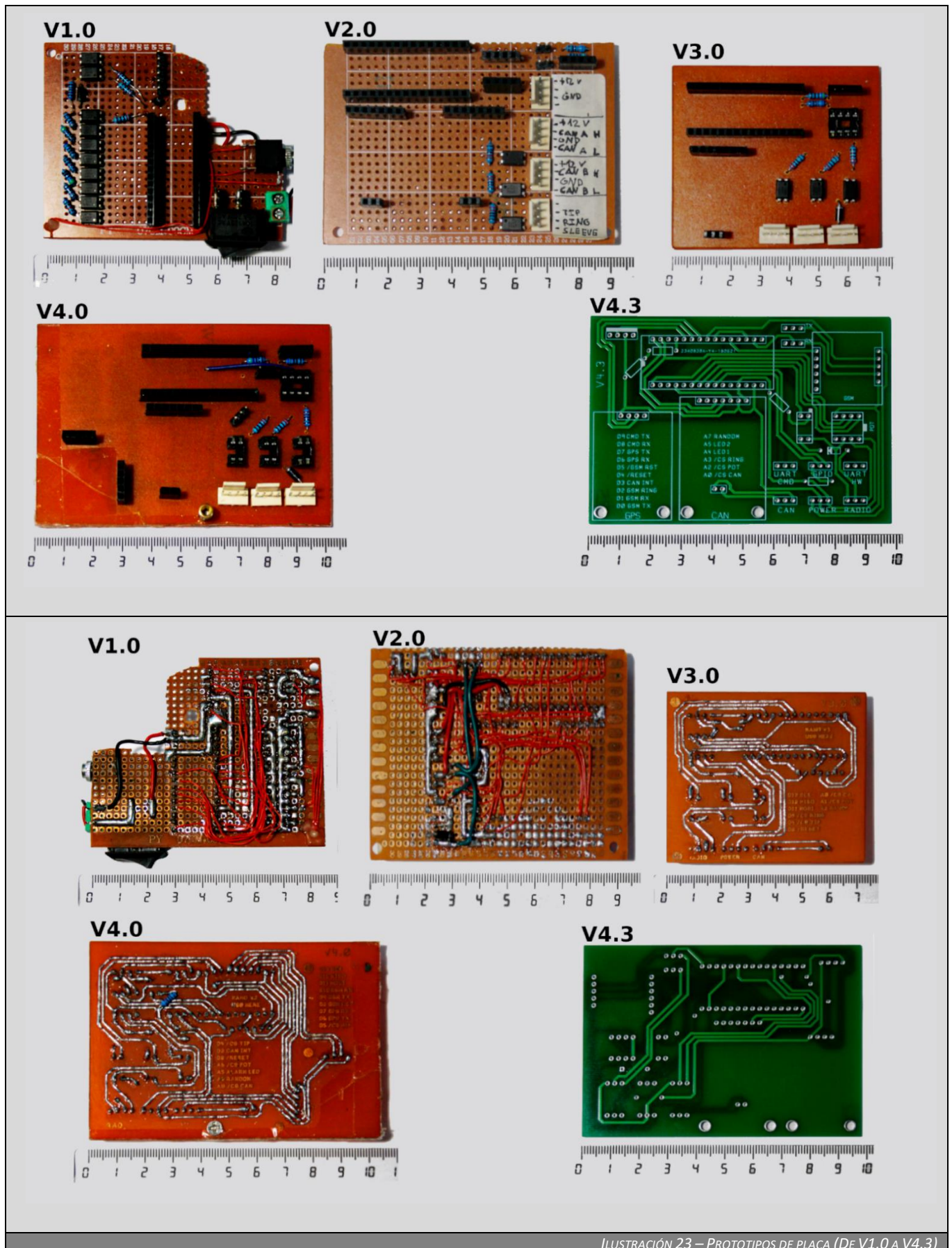


ILUSTRACIÓN 23 – PROTOTIPOS DE PLACA (DE V1.0 A V4.3)

3.7 – MECANIZADO

Una vez está todo el circuito ensamblado es necesario encapsularlo de forma que sus contactos se queden aislados para poder instalarlo de forma segura en el automóvil y evitar posibles cortocircuitos, filtraciones de humedad, etc...

Se ha creado un modelo 3D de la carcasa con las medidas necesarias y los huecos de los conectores en sus posiciones exactas. Este modelo puede ser impreso mediante una impresora 3D o enviado a producir a un fabricante.

Existen multitud de herramientas de diseño CAD, tanto gratuitas, como de pago y en versiones online y offline.

Este modelo se ha llevado a cabo usando la herramienta online Tinkercad ⁵. Esta herramienta es muy sencilla y cómoda de utilizar, pues al ser online no es necesario instalar ninguna dependencia adicional y funciona desde cualquier equipo con un navegador web moderno y una conexión a internet.

Las ilustraciones 24 y 25 muestran el modelo del mecanizado (tanto la carcasa como la tapa).

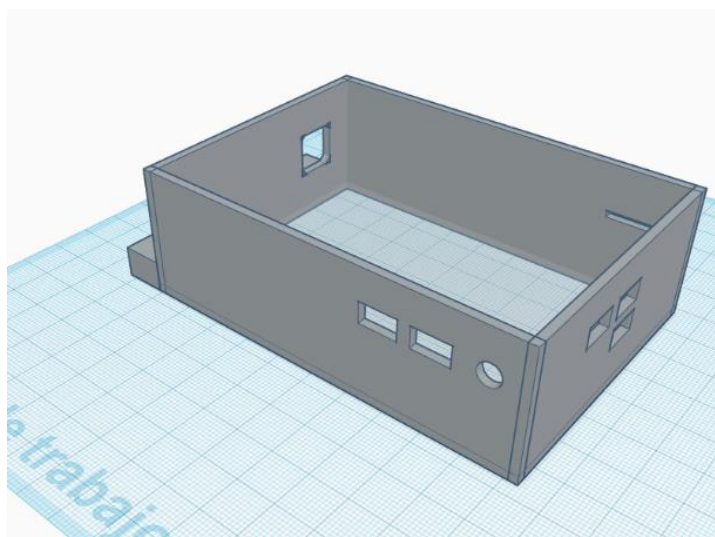


ILUSTRACIÓN 24 – VISTA DEL MECANIZADO (1)

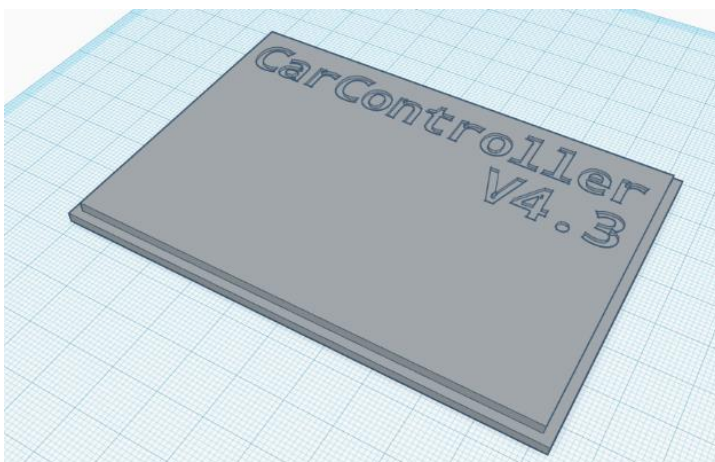


ILUSTRACIÓN 25 – VISTA DEL MECANIZADO (2)

⁵ Tinkercad - <https://www.tinkercad.com/>

La ilustración 26 muestra la controladora completamente ensamblada e introducida en el mecanizado.



ILUSTRACIÓN 26 – VISTA DEL ENSAMBLADO COMPLETO

4 – DISEÑO E IMPLEMENTACIÓN SOFTWARE

En este apartado se presenta en detalle el diseño y la implementación del software. Este software se divide en tres partes, software del microcontrolador, software de control móvil y software de la plataforma de seguimiento.

Tanto el software del microcontrolador como el software de control móvil han sido desarrollados siguiendo un ciclo de vida en cascada y una metodología tradicional.

4.1 – SOFTWARE DEL MICROCONTROLADOR

Esta parte incluye el código específico para manejar el hardware citado en puntos anteriores y proporciona e implementa la mayor parte de las funcionalidades especificadas en los requisitos.

Este código se compila en base a una serie de parámetros directamente ligados con las funciones habilitadas y con el modelo del vehículo, dando lugar a un ejecutable binario que reside en la memoria del microprocesador y que será ejecutado una vez el sistema entre en funcionamiento.

4.1.1 – ENFOQUE SOBRE EL DESARROLLO

Para proveer al código fuente de portabilidad, seguridad y fiabilidad, se han cumplido las recomendaciones MISRA C (*Motor Industry Software Reliability Association*) [4] [7].

Como se ha indicado en los requisitos y también por las decisiones tomadas en el diseño de hardware, especialmente al elegir el Arduino Nano, el código debe mostrar un equilibrio entre eficiencia, consumo reducido de memoria RAM y un tamaño del ejecutable aceptable.

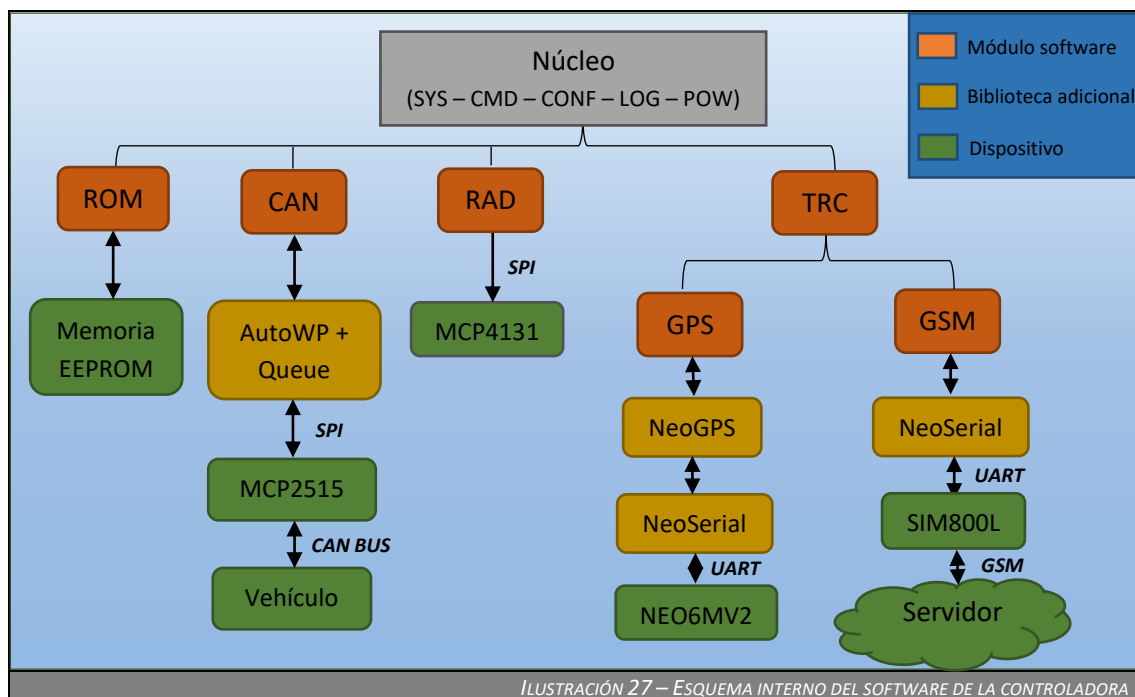
Como ejemplo, debido a que normalmente la memoria RAM de los sistemas microcontrolador es de un tamaño reducido, las cadenas de texto y los datos a los que no se accede frecuentemente se almacenan en la memoria *Flash* en lugar de la *SRAM*. En Arduino, a esta utilidad se le llama *PROGMEM* [11].

Esta opción resulta muy útil cuando se requiere de gran cantidad de datos incluidos en el programa sin que esto reduzca la memoria RAM disponible. No debemos olvidar que un acceso a memoria *SRAM* es casi instantáneo a lo que ciclos de ejecución se refiere y una lectura desde la memoria *Flash* puede llegar a ser cientos de veces más lenta, por lo que es importante reservar la memoria RAM para los datos que se usan con mayor frecuencia.

Por otra parte, la modularidad en el desarrollo resulta crucial. La única forma de conseguir compatibilidad entre distintos vehículos es partiendo de una base mayoritariamente genérica e incluir pequeñas partes específicas para cada marca y modelo.

4.1.2 – ESTRUCTURA INTERNA

En la ilustración 27 se muestra un esquema de la estructura interna de los módulos software, las bibliotecas empleadas y los protocolos de comunicaciones empleados con los periféricos.



4.1.3 – MÓDULOS SOFTWARE Y FUNCIONALIDADES

En este punto se detallan las distintas partes y módulos software implementados.

4.1.3.1 – NÚCLEO – GESTIÓN DEL SISTEMA

Este apartado describe el módulo principal del sistema, que es el responsable de dotar de unas funcionalidades básicas y jerarquizar el sistema empleando un sistema modular. Se puede dividir conceptualmente en distintas partes, aunque forman parte de un todo indivisible.

- **SYS** - Sistema general y manejo de módulos.
- **CMD** - Sistema de manejo de eventos y acciones.
- **CONF** - Configuración de módulos en tiempo de ejecución.
- **LOG** - Sistema de registro de eventos y depuración.
- **POW** - Sistema de alimentación y control de energía.

En la tabla 1 se detallan los parámetros obligatorios en el fichero de configuración:

Parámetro	Descripción
CAR_MODEL	Indica el modelo del vehículo al que se va a conectar.
BOARD_VERSION	Indica la revisión de la placa base que se va a emplear. Es la forma de indicar la configuración de pines correcta.
DEFAULT_BAUDRATE	Tasa de baudios que emplea la interfaz serie principal.
SERIAL_CONTROL_MODE	Indica los manejadores de comandos que deben ser incluidos en el sistema en tiempo de compilación. <ul style="list-style-type: none"> - SC_FULL - Todos los módulos - SC_MID - Módulos básicos y utilidades. - SC_MIN - Módulos mínimos.
USE_STATIC_MEM	Indica si los módulos deben utilizar memoria estática o dinámica.
POW_ENABLE_WATCHDOG	Habilita el <i>watchdog</i> integrado en el sistema.
POW_WATCHDOG_TIMEOUT	Indica el tiempo que debe esperar el <i>watchdog</i> para actuar.

TABLA 1 – PARÁMETROS DEL NÚCLEO EN TIEMPO DE COMPILACIÓN

4.1.3.2 – ROM – CONFIGURACIÓN EN EEPROM

Resulta esencial en muchas circunstancias disponer de un soporte no volátil para almacenar información. En este caso se emplea una EEPROM donde guardar algunas configuraciones que pueden variarse en tiempo de ejecución.

Además de configuración, también incorpora un sencillo sistema de entradas indexadas donde cada módulo puede guardar los datos que considere sin que haya riesgo de solaparse con otros.

4.1.3.3 – CAN – COMUNICACIONES POR BUS

Este módulo es el encargado de gestionar todas las comunicaciones con el automóvil a través de la red *CAN BUS*. Es especialmente crítico ya que debe ser robusto y fiable, además de cumplir con unas restricciones temporales muy marcadas o en su defecto comenzaría a perder información y a causar errores en las comunicaciones.

Incluye un sistema de colas tanto para envío como para recepción, un sistema de interrupciones, listas blanca y negra y un sistema de envíos periódicos entre otros.

Para evitar confusiones, la unidad de información que se envía por *CAN BUS* se va a denominar “Mensaje” en adelante.

La lista blanca permite establecer desde qué identificadores CAN queremos recibir mensajes, de forma que el resto de mensajes se ignorarán. Por el contrario, la lista negra permite establecer qué identificadores queremos ignorar.

En la tabla 2 se detallan los parámetros referentes al módulo CAN.

Parámetro	Descripción
COMPILE_CAN	Habilita / Deshabilita el módulo CAN. Generalmente sólo se deshabilita por tareas de depuración.
COMPILE_CAN_PIN_CMD_AVAILABLE	Indica si la controladora CAN dispone de un pin conectado al GPIO de la CPU que indique si hay mensajes por recibir.
COMPILE_CAN_INTERRUPT	Habilita el sistema de interrupciones. En caso contrario utiliza un sistema de <i>Polling</i> .
COMPILE_CAN_WHITELIST	Habilita el sistema de “lista blanca”. Normalmente gestionado por la propia controladora.
COMPILE_CAN_BLACKLIST	Habilita el sistema de “lista negra”. En este caso es gestionado directamente por el módulo software.
COMPILE_CAN_PERIODICAL	Habilita el sistema de comandos periódicos. Permite establecer una serie de comandos que se enviarán periódicamente. Resulta útil en mensajes de difusión de estado y similares.
CAN_CLOCKSET	La frecuencia del oscilador conectado a la controladora. Frecuencias válidas: MCP_8MHZ, MCP_16MHZ, MCP_20MHZ
CAN_ENABLE_TX_QUEUE	Habilita la cola en los mensajes de salida.
CAN_RX_QUEUE_LENGTH	Indica el tamaño máximo de mensajes que puede haber en las colas de entrada o salida. En caso de excederse el sistema registra el error y descarta los mensajes antiguos.
CAN_TX_QUEUE_LENGTH	
CAN_MIN_COMMAND_INTERVAL (milisegundos)	El tiempo mínimo que debe transcurrir entre el envío de mensajes. Esto es necesario para evitar una posible saturación en el bus de datos.

CAN_INTERRUPT_MODE	Indica el modo en que funciona el sistema de interrupciones. Los modos disponibles son: <ul style="list-style-type: none"> - CAN_INTERRUPT_MODE_FETCH: La rutina de interrupción obtiene el mensaje directamente desde la controladora en el momento que llega. - CAN_INTERRUPT_MODE_FLAG: La rutina de interrupción habilita un <i>flag</i> para que se obtenga el mensaje fuera de la propia rutina.
CAN_TX_RETRY_TIMEOUT (milisegundos)	El tiempo que se espera para reintentar un envío fallido.

TABLA 2 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO CAN

4.1.3.4 – RAD – CONTROL DE RADIO

Este módulo software se emplea para controlar equipos de sonido siguiendo el protocolo *Wired/Remote*.

Con la intención de facilitar el desarrollo, por cuestiones de diseño y de seguridad, este módulo enmascara las acciones de la radio de forma que el resto de módulos sólo pueden activar estas acciones y no deben indicar el valor específico de la resistencia para cada caso.

La tabla 3 muestra los parámetros de configuración del módulo software RAD.

Parámetro	Descripción
POT_SHOW_KOHMS	Habilita la visualización de los valores de la resistencia. Utilizado para tareas de depuración y desarrollo.
RADIO_COMMAND_MIN_INTERVAL	Definen el intervalo de tiempo (en milisegundos) que debe mantenerse un único comando de la radio.
RADIO_COMMAND_MAX_INTERVAL	
RADIO_USE_TIP_OPTOCOUPERS	Indica si se utilizan optoacopladores externos para desconectar los terminales del potenciómetro o por el contrario se utiliza el mecanismo integrado en el propio potenciómetro.

TABLA 3 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO POT

En la tabla 4 se muestran las distintas acciones y los valores de la resistencia asociados que se pueden emplear para controlar un equipo de sonido mediante el protocolo *Wired/Remote*. En caso de que el valor de la resistencia se encuentre fuera de estos rangos, el equipo de sonido simplemente ignorará la acción.

Acción	Valor de la resistencia (kΩ)			Puentea “Ring”
	Mínimo	Nominal	Máximo	
Origen / Apagar	0.4	1.2	2	No
Silenciar (ATT)	2.5	3.5	4.5	No
Display / Pantalla	5	5.75	6.5	No
Siguiente canción	7	8	9	No
Canción anterior	9.5	11.25	13	No
Subir volumen	13.5	16	18.5	No
Bajar volumen	19	24	29	No
Band / Escape	37.5	62.75	88	No
Subir carpeta	7	8	9	Sí
Bajar carpeta	9.5	11.25	13	Sí
Control por voz	37.5	62.75	88	Sí

TABLA 4 – RELACIÓN DE VALORES RESISTIVOS Y ACCIONES WIRED REMOTE

4.1.3.5 – GPS – SISTEMA DE POSICIONAMIENTO GLOBAL

La funcionalidad de geolocalización viene gestionada por este módulo, el cual encapsula toda la complejidad interna del sistema GPS y facilita su uso al resto del sistema. Su principal función es analizar los mensajes recibidos por el receptor GPS en el formato **NMEA 0183** y convertirlos a estructuras de datos más manejables para los otros módulos.

Generalmente se utiliza en conjunción a los módulos GSM y TRC que se detallan a continuación, aunque puede utilizarse de forma independiente para, por ejemplo, obtener la hora UTC a través de los satélites GPS.

En la tabla 5 se pueden ver algunos parámetros de configuración de este módulo software.

Parámetro	Descripción
GPS_BAUDRATE	La tasa de baudios que emplea el receptor GPS. Sólo se utiliza en caso de UART.
GPS_SERIAL_ALWAYS_ENABLED	Indica si el receptor GPS siempre debe estar activado o por el contrario se activa por solicitud de algún módulos software.

TABLA 5 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO GPS

4.1.3.6 – GSM – COMUNICACIONES POR RED MÓVIL

Este módulo gestiona toda la conectividad a internet del sistema. Es el encargado de comunicarse con el módulo GSM [1] (a través de comandos AT ⁶), configurarlo, y manejar todas las solicitudes y respuestas mediante un sistema de eventos interno.

También permite establecer un servicio a la escucha para recibir comandos a través de internet, aunque esta característica no se recomienda por cuestiones de seguridad y está deshabilitada por defecto.

En la tabla 6 se detallan algunos parámetros en la configuración del módulo GSM.

Parámetro	Descripción
GSM_BAUDRATE	La tasa de baudios que emplea el módulo GSM. Sólo se utiliza en caso de UART.
GSM_GET_MAX_TIMEOUT	El tiempo máximo de espera para recibir la respuesta a una solicitud antes de darse por fallida.
GSM_SETUP_MAX_TIMEOUT	El tiempo máximo de espera al iniciar la controladora GSM y conectarse a la red móvil antes de un reinicio forzado.
GSM_DISABLE_LEDS	Deshabilita los LEDs incorporados en la placa del modem GSM.
GSM_USE_FLASH_STRINGS	Especifica si los comandos AT se almacenan en la memoria RAM o en la sección de código del programa (PROGMEM).
GSM_COMMAND_QUEUE_LENGTH	Indica el tamaño máximo de la cola donde se almacenan los comandos AT pendientes de procesar.
GSM_APN	Indica el parámetro APN que se utiliza en la conexión.
GSM_PIN	Indica el código secreto de la tarjeta SIM.

TABLA 6 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO GSM

⁶ Comandos AT - https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes

4.1.3.7 – TRC – TRACCAR – PLATAFORMA DE SEGUIMIENTO

Este módulo software requiere a su vez de los módulos GPS y GSM para su funcionamiento. Se centra en tomar las posiciones obtenidas por el módulo GPS y en base a ellas, formar las solicitudes con el formato correcto del servicio *Traccar* para enviarlas a través de la red empleando el módulo *GSM*.

Además de la posición también puede enviar mediciones sobre el estado del vehículo en tiempo real como la posición del contacto, el voltaje de la batería o la velocidad a la que está circulando.

A continuación, en la tabla 7, se muestran algunos parámetros de configuración.

Parámetro	Descripción
TRACCAR_NOTIFICATION_INTERVAL_ON	Indica el intervalo de notificaciones. Se pueden especificar tiempos diferentes dependiendo de si el automóvil está encendido o no.
TRACCAR_NOTIFICATION_INTERVAL_OFF	
TRACCAR_SEND_TIMESTAMP	Indica si se envía la fecha y la hora con cada notificación.
TRACCAR_MAX_REQUEST_RETRIES	Especifica el número máximo de reintentos al reenviar una notificación fallida antes de descartarla.
TRACCAR_NOTIFICATION_QUEUE_LENGTH	Indica el tamaño máximo de la cola donde se almacenan las notificaciones.
TRACCAR_HOST	El nombre de host donde se encuentra el servicio <i>Traccar</i> en funcionamiento.
TRACCAR_PORT	El puerto donde está el servicio <i>Traccar</i> a la escucha.
TRACCAR_DEVICE_ID	El identificador único del dispositivo. Este valor se configura en el servidor y generalmente es un texto.

TABLA 7 – PARÁMETROS EN TIEMPO DE COMPILACIÓN DEL MÓDULO TRC

4.1.4 – BIBLIOTECAS ADICIONALES

En la tabla 8 se pueden ver varias bibliotecas que se han empleado durante el desarrollo.

Nombre	Autor	Página oficial
Arduino MCP2515 CAN interface library	AutoWP	https://github.com/autowp/arduino-mcp2515
	Biblioteca para el componente MCP2515.	
NeoGPS	Slash Devin	https://github.com/SlashDevin/NeoGPS
	Biblioteca eficiente para la gestión de sentencias NMEA 0183.	
FastGPIO	Pololu	https://github.com/pololu/fastgpio-arduino
	Gestión rápida de pines de entrada y salida.	
Queue	SMFSW	https://github.com/SMFSW/Queue
	Estructuras de colas FIFO y LIFO.	
NeoHWSerial	Slash Devin	https://github.com/SlashDevin/NeoHWSerial
	Ampliación de la clase “Serial” con más funcionalidades.	
NeoSWSerial	Slash Devin	https://github.com/SlashDevin/NeoSWSerial
	Reemplazo eficiente de la clase “SoftwareSerial”	
NeoICSerial	Slash Devin	https://github.com/SlashDevin/NeoICSerial
	Comunicaciones UART usando los pines “Input Capture”.	

TABLA 8 – BIBLIOTECAS EMPLEADAS EN EL DESARROLLO

4.1.5 – HERRAMIENTAS DE DESARROLLO

Para llevar a cabo el desarrollo del software del microcontrolador se han empleado distintas herramientas.

Para compilar código compatible con la amplia gama de modelos de *Arduino* es necesario tener instalado el entorno oficial del *Arduino*. Este *kit* incluye un editor muy sencillo para crear los programas (conocidos como *sketches*), pero a medida que el proyecto crece y aumenta de tamaño se vuelve tedioso y poco práctico.

Por ello se ha empleado principalmente la utilidad *Visual Studio 2017* de *Microsoft*, la cual proporciona un entorno de desarrollo muy completo para gran cantidad de plataformas como *Windows*, *Windows IoT*, *Web (ASP.Net, MVC)*, e incluso móvil con *Xamarin*.

Aunque no soporta la plataforma *Arduino* de forma oficial, existe un *plugin* llamado *Visual Micro*⁷ que añade el soporte y todos los recursos necesarios para llevar a cabo un desarrollo completo.

Para la gestión del código se ha utilizado un repositorio *subversión*, el cual ha sido gestionado por otro *plugin* llamado *AnkhSVN*⁸. Aunque el uso de un repositorio de código para un desarrollo individual pueda parecer excesivo, resultan de gran ayuda las características que ofrece este servicio. Características como el histórico de cambios, la posibilidad de crear distintas “ramas” o tener el código centralizado para, por ejemplo, crear copias de seguridad fácilmente, lo hacen una herramienta indispensable.

4.1.6 – PROTOCOLO DE COMUNICACIONES INTERNO

El núcleo procesa los **comandos** recibidos por el puerto serie. Estos comandos permiten gestionar las opciones en tiempo de ejecución o pueden estar asociados a **acciones** del propio núcleo o de los módulos software. La especificación de estos comandos es crucial, ya que son la interfaz que va a permitir al usuario interactuar con el sistema y al resto de módulos entre sí.

Los comandos se crean en formato texto y cuentan con una sintaxis muy similar a una llamada a función en lenguaje C, aunque con ligeras variaciones.

Cada comando está compuesto por una parte que indica un prefijo, una función y una serie de parámetros variables separados por el carácter ‘;’.

En la tabla 9 se muestra la sintaxis de los comandos.

Sintaxis BNF (Backus-Naur)	
$\langle \text{comando} \rangle ::= \langle \text{prefijo} \rangle : \langle \text{función} \rangle ([\{ \langle \text{Parámetro} \rangle \} ;]);$	
Ejemplos:	
<ul style="list-style-type: none">• SYS:free();• CAN:send(0x203;0x01;0x20;0x94);	<ul style="list-style-type: none">• LOG:set(CAN;1);• POW:on();

TABLA 9 – SINTAXIS BNF DE LOS COMANDOS DE CONFIGURACIÓN

⁷ Visual Micro - <https://www.visualmicro.com/>

⁸ AnkhSVN - <https://ankhsvn.open.collab.net/>

Los manejadores de los comandos pueden ser sobrescritos para ampliar su funcionalidad desde otro módulo. Es decir, un comando puede ser ampliado con más funciones desde distintos módulos software.

4.1.6.1 – ACCIONES IMPLEMENTADAS

En las tablas 10 a 16 se detallan las acciones incluidas en la base del proyecto. En la implementación del vehículo es posible añadir más acciones, pero estas deben ser específicas para el modelo del automóvil.

Núcleo – Prefijo SYS		
Función	Parámetros	Descripción
free	-	Devuelve la cantidad de memoria RAM libre (en bytes)
reset	-	Reinicia el sistema.

TABLA 10 – ACCIONES – FUNCIONES DEL NÚCLEO – SYS

Núcleo – Prefijo OPT		
Función	Parámetros	Descripción
get	1. Nombre del módulo	Obtiene o establece si se debe activar un módulo específico. <u>Módulos válidos:</u> <ul style="list-style-type: none"> • TRC – Traccar
set	1. Nombre del módulo 2. Valor	

TABLA 11 – ACCIONES – FUNCIONES DEL NÚCLEO – OPT

Núcleo – Prefijo LOG		
Función	Parámetros	Descripción
get	1. Nombre del registro	Obtiene o establece si se debe mostrar un registro específico. <u>Registros válidos:</u> <ul style="list-style-type: none"> • CAN – CAN BUS • POT – Potenciómetro • STA – Estado • TRC – Traccar • GSM – Acceso a internet • GPS – Geolocalización • DBG – Depuración / Desarrollo
set	1. Nombre del registro 2. Valor	

TABLA 12 – ACCIONES – FUNCIONES DEL NÚCLEO – LOG

Núcleo – Prefijo POW		
Función	Parámetros	Descripción
on	-	Cambia al modo de vehículo encendido.
off	-	Cambia al modo de vehículo apagado.

TABLA 13 – ACCIONES – FUNCIONES DEL NÚCLEO – POW

Módulo – Prefijo ROM		
Función	Parámetros	Descripción
save	-	Guarda la configuración actual en la memoria.
load	-	Carga la configuración desde la memoria.
clear	-	Borra toda la configuración de la memoria.
remove	1. Índice de la entrada	Borra la entrada especificada de la memoria.
show	-	Muestra todas las entradas de la memoria.

TABLA 14 – ACCIONES – FUNCIONES DEL MÓDULO – ROM

Módulo – Prefijo RAD		
Función	Parámetros	Descripción
vol+	-	Subir volumen.
vol-	-	Bajar volumen.
next	-	Siguiente canción
prev	-	Canción anterior
dis	-	Alternar pantalla entre ecualizador, pista actual y reloj.
mute	-	Silenciar. (Modo ATT)
src	-	Alternar la fuente entre USB, Bluetooth, Radio o Auxiliar.
band	-	Función band.
f_up	-	Subir carpeta en el árbol de directorios.
f_down	-	Bajar carpeta en el árbol de directorios.
voice	-	Activar el control por voz.

TABLA 15 – ACCIONES – FUNCIONES DEL MÓDULO – RAD

Módulo – Prefijo CAN		
Función	Parámetros	Descripción
send	1. ID del mensaje 2. Contenido del mensaje (hasta 8 bytes)	Envía un mensaje por el bus de datos CAN.

TABLA 16 – ACCIONES – FUNCIONES DEL MÓDULO – CAN

4.2 – SOFTWARE DE CONTROL – APLICACIÓN MÓVIL / TABLET

Para gestionar la controladora es posible conectarse a ella utilizando el protocolo UART con un ordenador y mandar los comandos pertinentes. Claramente esto resulta muy farragoso además de poco intuitivo si no se tiene costumbre con la sintaxis y el nombre de las opciones.

Con la intención de facilitar las tareas de uso, configuración y depuración se ha creado un aplicativo para teléfonos móviles y *tablets*, que permite conectarse a la controladora y gestionarla de forma inalámbrica usando la conectividad *Bluetooth*. Este aplicativo funciona bajo el sistema operativo *Android* [2] (se puede ver el icono oficial en la ilustración 28) con la versión 4.4 o superior.

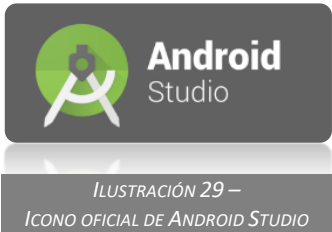


ILUSTRACIÓN 28 –
ICONO OFICIAL DE
ANDROID

Implementa la llamada a todas las acciones registradas en la controladora. Tanto las que se han comentado anteriormente como las específicas del automóvil en el que se han realizado las pruebas.

El diseño de la interfaz gráfica se ha centrado en crear una aplicación sencilla, práctica y que permita un acceso rápido a todas sus funciones con pocas pulsaciones.

4.2.1 – HERRAMIENTAS DE DESARROLLO



Se ha desarrollado con la herramienta oficial *Android Studio* (se puede ver el icono en la ilustración 29), creada de la mano de *Google Inc.* Es un IDE muy completo y especialmente diseñado para el desarrollo de aplicaciones nativas en *Android*.

Se integra completamente con todo el *kit* de desarrollo de *Android*, lo que permite, entre otras cosas, la creación de máquinas virtuales para probar las aplicaciones en múltiples versiones del sistema operativo o con distintos tamaños y relaciones de aspecto de la pantalla.

4.2.2 – IMÁGENES DE LA APLICACIÓN

En las ilustraciones 30, 31, 32 y 33 se puede ver el menú principal de la aplicación donde se encuentran múltiples controles como botones y selectores, los cuales se emplean para enviar acciones y modificar distintas configuraciones del sistema.

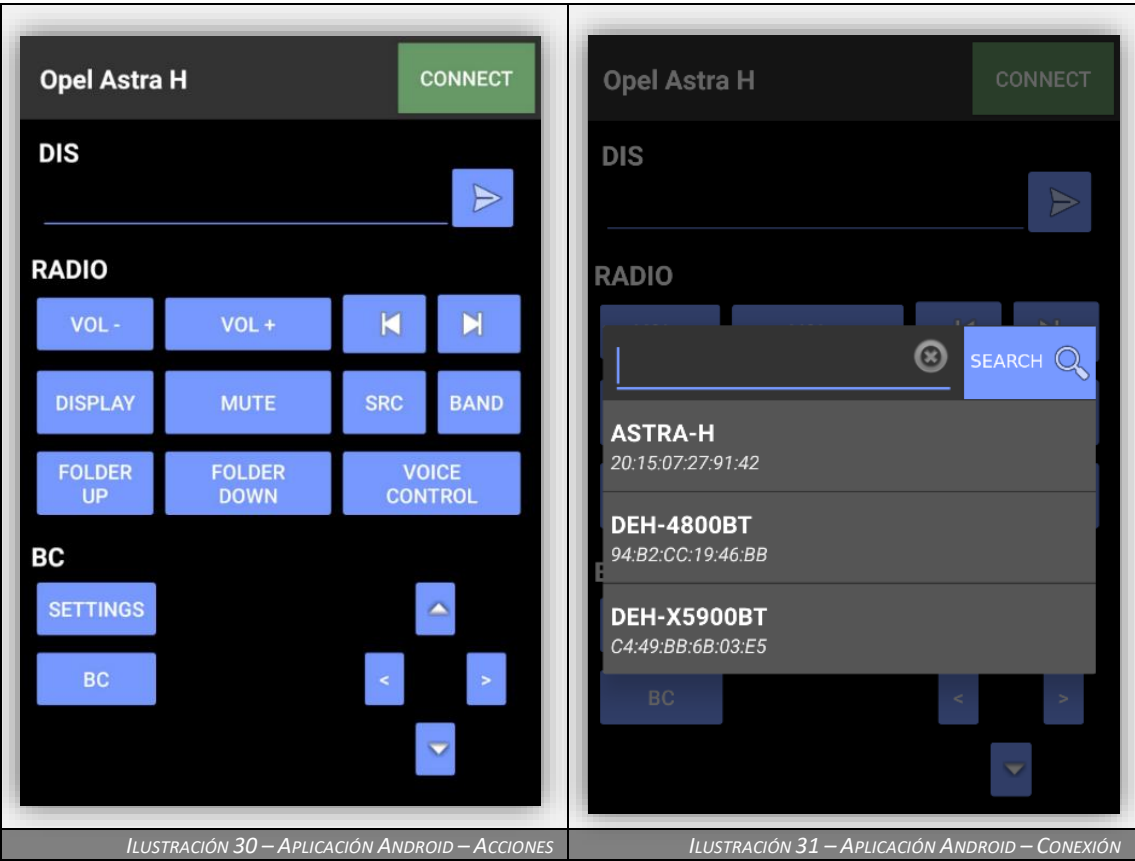




ILUSTRACIÓN 32 – APLICACIÓN ANDROID – CONFIGURACIÓN

ILUSTRACIÓN 33 – APLICACIÓN ANDROID – REGISTRO

4.3 – SOFTWARE DE SEGUIMIENTO – PLATAFORMA TRACCAR

Como se ha visto en puntos anteriores, la controladora se encarga de obtener la posición desde el módulo hardware GPS, transformarla al formato correcto y enviarla por internet hasta un servidor.

Este servidor ejecuta una plataforma de seguimiento llamada *Traccar* (ver ilustración 34), la cual gestiona todas las solicitudes y añade gran cantidad de funcionalidades como alarmas, histórico de posiciones, geocercas, etc...



ILUSTRACIÓN 34 –
ICONO OFICIAL DE TRACCAR

Traccar es un proyecto de software libre desarrollado y administrado por la compañía *Traccar Ltd* residente en San Petersburgo.

El modelo de negocio que sigue esta compañía se basa en ofrecer su plataforma como un servicio bajo un pago recurrente mensual. Pero la plataforma en sí es gratuita y está disponible para descargarla y ejecutarla en un servidor propio sin coste alguno.

Soporta prácticamente la totalidad de dispositivos GPS comerciales de rastreo para todo tipo de vehículos e incluso terminales móviles o mascotas.

La monitorización se realiza mediante una interfaz web (la cual se puede ver en la ilustración 35) que permite gestionar todos los dispositivos con pocas pulsaciones. Es posible configurar alertas de velocidad, de encendido

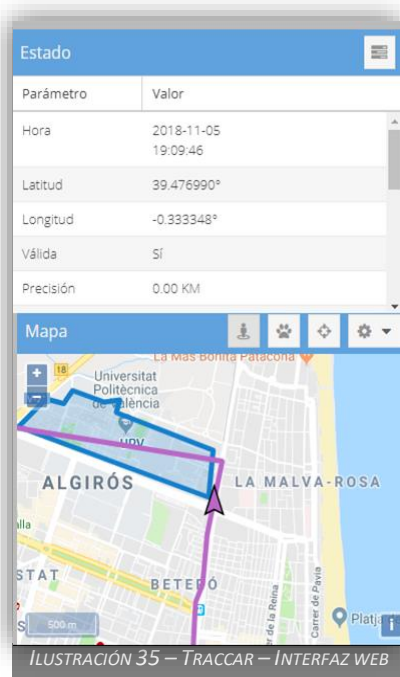


ILUSTRACIÓN 35 – TRACCAR – INTERFAZ WEB

y desconexión, delimitar zonas seguras, etc...

Además de la interfaz web, existe un aplicativo móvil (para *Android* e *iOS*) que permite acceder al panel de administración directamente sin necesidad de un navegador web.

En el caso de este proyecto se ha optado por la opción de instalar la plataforma de seguimiento bajo un servidor propio debido a la posibilidad de acceder directamente a las configuraciones y comprobar de primera mano los aspectos internos de la tecnología empleada.

En el caso de disponer de múltiples dispositivos y tener que realizar un seguimiento de todos, es preferible tomar la opción de pago ofrecida por la empresa para evitar tareas de mantenimiento y problemas derivados del servidor.

4.3.1 – CONFIGURACIÓN

Los parámetros por defecto que se especifican tras la instalación suelen ser suficientes para utilizar el servicio sin problemas. Aunque hay varios parámetros útiles que no son directamente configurables desde el panel de administración web, y deben ser escritos de forma manual en el fichero de configuración “*conf/traccar.xml*”. Algunos parámetros interesantes se pueden ver en la tabla 17.

Nombre	Tipo	Descripción
database.driver	Cadena de texto	Permite especificar el driver que se empleará para conectar a la base de datos.
database.url	Cadena de texto	La ruta a la base de datos.
database.user	Cadena de texto	Usuario de la base de datos.
database.password	Cadena de texto	Clave de acceso a la base de datos.
filter.enable	Booleano	Habilita el filtrado de posiciones.
filter.invalid	Booleano	Filtra las posiciones marcadas como inválidas.
filter.zero	Booleano	Filtra las posiciones en el punto (0,0).
filter.future	Número entero	Para una posición recibida, indica cuantos segundos puede estar en el futuro.
status.timeout	Número entero	Indica el número de segundos de inactividad antes de que el dispositivo se marque como desconectado.
processing.remoteAddress.enable	Booleano	Indica si se almacena la dirección IP con cada posición recibida.
mail.smtp.host	Cadena de texto	Indica el host del servidor de correo.
mail.smtp.port	Número entero	Indica el puerto del servidor de correo.
mail.smtp.starttls.enable	Booleano	Habilita el uso de la tecnología StartTLS.
mail.smtp.from	Cadena de texto	Indica la dirección de correo saliente.
mail.smtp.auth	Booleano	Indica si se utiliza autenticación en el servidor.
mail.smtp.username	Cadena de texto	Indica el nombre de usuario y la contraseña en el servidor de correo
mail.smtp.password	Cadena de texto	

TABLA 17 – TRACCAR – CONFIGURACIÓN

Existen más parámetros compatibles en la web oficial ⁹.

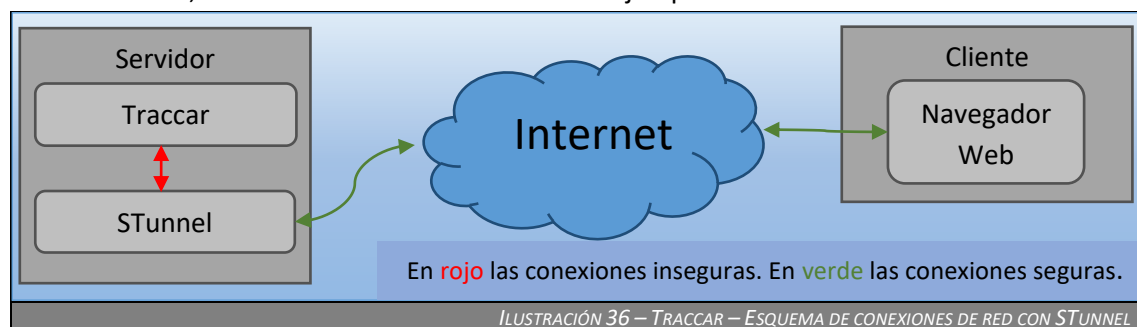
⁹ Fichero de configuración de Traccar - <https://www.traccar.org/configuration-file/>

4.3.2 – STUNNEL

Uno de los mayores inconvenientes de utilizar esta plataforma es que el servicio no incorpora el protocolo HTTPS (HTTP seguro) de forma nativa. Hoy en día es una característica imprescindible por motivos de seguridad. Más todavía si tenemos en cuenta que para acceder al panel de administración se envía un usuario y una contraseña sin cifrar por la red.

De forma opcional (aunque altamente recomendable) se puede emplear la utilidad *STunnel* para añadir la capa de seguridad de la que carece. Este programa encapsula todo el tráfico dirigido al servicio *Traccar* y lo envía a través de un canal seguro utilizando la tecnología *SSL*. De esta forma, el contenido inseguro nunca sale del servidor.

A continuación, en la ilustración 36 se muestra un ejemplo del funcionamiento de *STunnel*.



Para utilizar este programa necesitamos crear un certificado con su clave privada correspondiente. Para ello podemos utilizar el comando “*openssl*” descrito en la tabla 18.

Sin Passphrase : openssl req -new -x509 -nodes -keyout server.pem -out server.pem
Con Passphrase : openssl req -new -x509 -keyout user.pem -out user.pem

TABLA 18 – TRACCAR – CREACIÓN DE CERTIFICADOS MEDIANTE LA UTILIDAD OPENSSL

Para configurar *STunnel* es necesario escribir la configuración presentada en la tabla 19 dentro de un fichero de texto.

cert = server.pem pid = [TRACCAR] accept = 8083 connect = localhost:8082	Explicación simplificada: Utilizando el certificado “server.pem”, establecemos que STunnel se ponga a la escucha de forma segura en el puerto 8083 y redirija todo el tráfico al mismo servidor en el puerto 8082.
--------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Explicación simplificada:

TABLA 19 – TRACCAR – FICHERO DE CONFIGURACIÓN DE STUNNEL

Este fichero de configuración es el mínimo necesario, aunque la herramienta cuenta con un gran número de opciones y funciones. Toda la información está disponible en la página oficial¹⁰.

4.3.3 – PROTOCOLO OSMAND

Además de contar con compatibilidad para más de 170 protocolos comerciales y 1500 dispositivos de seguimiento GPS, *Traccar* incluye un protocolo propio llamado **OsmAnd**.

¹⁰ Página oficial de STunnel – https://www.stunnel.org/config_unix.html

Este protocolo es muy sencillo y se basa en el ampliamente conocido estándar HTTP, aunque de forma predeterminada se encuentra a la escucha en el puerto **TCP 5055**.

Simplemente hay que realizar una solicitud *GET* al puerto especificado con los parámetros descritos en la tabla 20 en la URL:

Parámetro	Tipo	Obligatorio	Descripción
id / deviceid	<i>Cadena de texto</i>	Sí	El identificador del dispositivo.
valid	<i>Booleano</i>	No	Si la posición se asume como válida o no. Por defecto se asume lo que se indique en el fichero de configuración.
timestamp	<i>Número entero</i>	No	La fecha y la hora correspondiente a la posición en formato <i>Unix Timestamp</i> . Si no se indica en la solicitud, se toma la hora del servidor en el momento de la recepción.
lat	<i>Número decimal</i>	Sí	La latitud y la longitud de la posición.
lon	<i>Número decimal</i>	Sí	
speed	<i>Número decimal</i>	No	La velocidad y altitud del dispositivo en el momento de tomar la posición. Por defecto se asume 0.
altitude	<i>Número decimal</i>	No	
batt	<i>Número decimal</i>	No	La carga de la batería del dispositivo. Al ser un número decimal se puede indicar un porcentaje o el voltaje de la misma.
cell	<i>Tupla de números enteros</i>	No	Una tupla de varios valores indicando los identificadores de la celda GSM a la que está conectado. Resulta útil para la geolocalización por antenas de telefonía.

TABLA 20 – TRACCAR – PARÁMETROS DEL PROTOCOLO OSMAND

Ejemplos:

`http://<hostTraccar>:5055/?id=ABC&lat=39.482832&lon=-0.347319&speed=31.2&batt=49.3`

`http://<hostTraccar>:5055/?id=DEF&lat=39.982182&lon=-0.748895×tamp=1550582433`

Se puede ver el listado completo de parámetros en el código fuente¹¹ de la clase *OsmAndProtocolDecoder* (archivo *traccar/src/org/Traccar/protocol/OsmAndProtocolDecoder.java*).

Los parámetros adicionales que no están establecidos en ese fichero se guardan como parámetros genéricos en una estructura de tipo “Clave-Valor”. Posteriormente pueden visualizarse o utilizarse como filtrado en el histórico de posiciones de la interfaz web.

¹¹ GitHub de Traccar – <https://github.com/traccar>

5 – ESPECIFICACIÓN Y ADAPTACIÓN DE VEHÍCULOS

Acorde al paradigma modular empleado, es necesario establecer una especificación para cada modelo de vehículo distinto. Esta especificación es, a grandes rasgos, la parte de software que se encuentra más próxima al vehículo y a sus protocolos internos.

Esta parte es especialmente compleja debido a que generalmente los protocolos internos de comunicaciones no están documentados para el público general. Esto desencadena en que prácticamente la única forma de obtener esta información es capturando los datos del vehículo en durante su funcionamiento y analizarlos aplicando técnicas de ingeniería inversa.

5.1 – ESQUEMA BÁSICO PARA DEFINIR UN VEHÍCULO

Cada modelo de automóvil tiene sus particularidades, por lo que es necesario realizar una especificación cambiando ciertos parámetros de configuración. En algunas ocasiones incluso puede ser necesario añadir código específico para facilitar la integración con el automóvil.

El proyecto cuenta con una estructura de carpetas donde se ubican los ficheros dedicados para cada modelo de automóvil. Estos son los pasos a seguir para crear una especificación nueva:

- Dentro de la carpeta “cars/” se crea una carpeta con un nombre significativo de la marca y modelo del vehículo. Dentro de esta carpeta se deben incluir todos los ficheros necesarios para la implementación del modelo de vehículo.
- Los únicos ficheros obligatorios son:
 - **car_configuration.h** – En este fichero van las configuraciones específicas del vehículo como por ejemplo los parámetros básicos del CAN BUS entre otros.
 - **main.c** – El código que recibirá los eventos del sistema. Estos eventos están descritos en la tabla 21.

Función / Evento	Descripción
<code>void carAfterMsgReceived(CAN_CONTROLLER *can, CAN_COMMAND *receivedMsg)</code>	Este evento se dispara con la recepción de un mensaje en la controladora CAN. Como parámetros recibe la controladora que recibió el mensaje además del mensaje.
<code>void carAfterLoop()</code>	Este evento se dispara tras cada iteración del programa principal.
<code>void carBeforeSetup()</code>	Este evento se dispara antes de comenzar la configuración del sistema.
<code>void carAfterSetup()</code>	Este evento se dispara tras configurar todo el sistema.

TABLA 21 – ESPECIFICACIÓN DE VEHÍCULOS – EVENTOS

- Para que el compilador acceda a la nueva especificación es necesario editar el fichero “configuration.h” y añadir al final del mismo la referencia a la configuración anterior. Se puede ver un ejemplo en la tabla 22.

```
#if(CAR_MODEL == OPEL_ASTRA_H_2004_CAR)
    #include "cars/OpelAstraH2004/car_configuration.h"
#elif(CAR_MODEL == FORD_FOCUS_2004_CAR)
    #include "cars/FordFocus2004/car_configuration.h"
#else
    #include "cars/DefaultCar/car_configuration.h"
#endif
```

TABLA 22 – ESPECIFICACIÓN DE VEHÍCULOS – FICHERO DE CONFIGURACIÓN

5.2 – CASO ESPECÍFICO – OPEL ASTRA H

5.2.1 – BREVE RESUMEN SOBRE EL VEHÍCULO



El Opel Astra es un automóvil diseñado por el fabricante alemán de automóviles Opel y vendido en numerosos países bajo las marcas Vauxhall, Chevrolet y todas del grupo industrial estadounidense General Motors. Se puede ver el aspecto del automóvil en la ilustración 37.

En las últimas tres décadas se han lanzado cinco generaciones de este modelo. Cada una se denomina por una letra en orden alfabético. Las letras empleadas son F, G, H, J y K.

Este documento recoge información sobre la tercera generación (H), lanzada en el año 2004 y que permaneció en el mercado hasta finales del año 2010.

Esta generación marcó un punto de inflexión a nivel electrónico, ya que fue la primera en incorporar una gran cantidad de microcontroladores repartidos por el vehículo, desplazando la mayoría de los controles mecánicos tradicionales. Esto le permitió añadir sistemas más robustos e inteligentes como el control de estabilidad (ESP) o el control de velocidad de crucero y así obtuvo la certificación de cinco estrellas Euro NCAP ¹² en la categoría de seguridad para ocupantes.

5.2.2 – CABLEADO

Aunque esta sección es específica del Opel Astra H, la información recogida a continuación es muy similar para todos los vehículos automóvil.

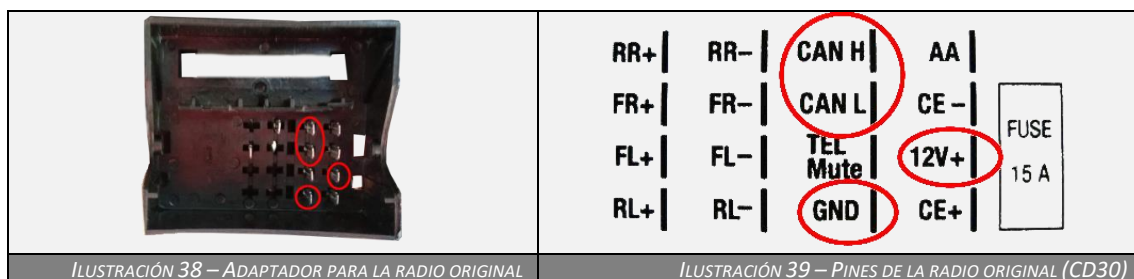
Para conectar la controladora genérica que se recoge en este documento es necesario conocer y modificar el cableado del vehículo [12] [17]. Es recomendable que dicha modificación la realice un profesional con los conocimientos necesarios para ello. Un conexionado incorrecto puede causar daños severos en los componentes del vehículo.

Los cables necesarios son:

- **+12V – Alimentación** – Se puede conectar directamente a la batería. También se puede emplear la alimentación del contacto (12V IGN) para que la controladora sólo funcione con el automóvil encendido. Ambos se pueden encontrar en la caja de fusibles o más fácilmente en el conector del mechero de la consola central [6].
- **GND – Masa** – Se encuentra en muchas partes del vehículo. Incluso la propia carrocería está conectada a masa.
- **CAN Low** – Bus de datos CAN – Cable Low.
- **CAN High** – Bus de datos Can – Cable High.

En el caso del Opel Astra H resulta sencillo obtener todas conexiones anteriores del conector de la radio. Es recomendable comprar algún adaptador o alargador y así realizar las conexiones sobre éste y no realizar ninguna modificación o corte al cableado original. Se puede ver un adaptador en la ilustración 38 y el esquema de conexiones en la ilustración 39.

¹² Euro NCAP – <https://www.euroncap.com/es>

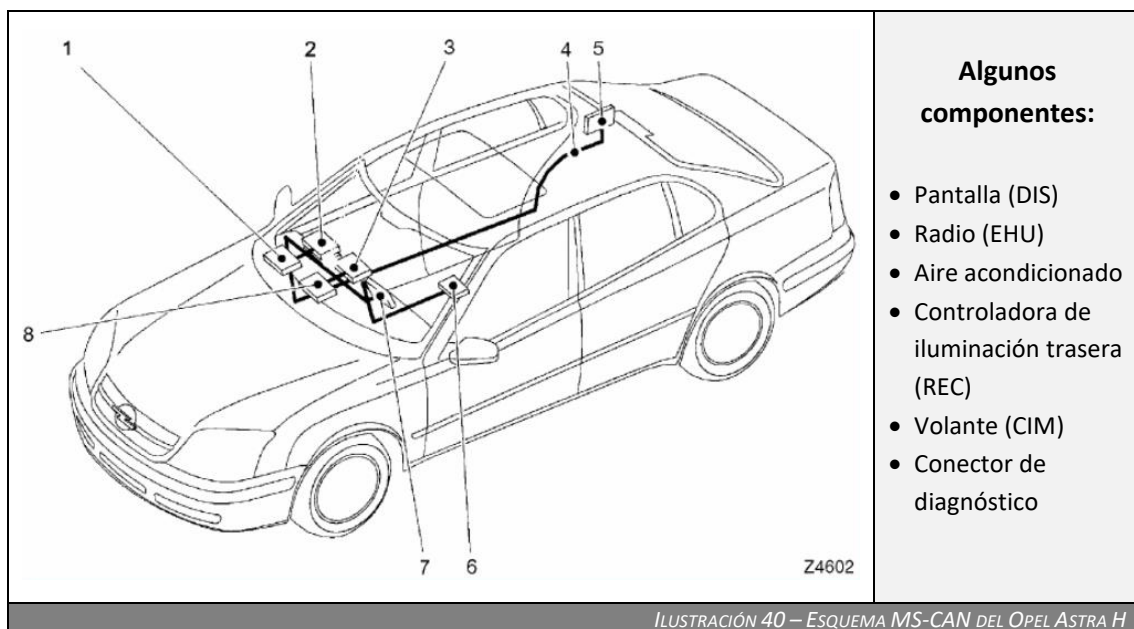


5.2.3 – BUS DE DATOS

Este automóvil cuenta con tres buses de datos distintos para gestionar todas las comunicaciones entre los dispositivos internos:

- Bus de alta velocidad **HS-CAN** – Este bus funciona a 500 Kbits/s y se emplea para las comunicaciones del bloque del motor. Por aquí viaja información como la velocidad del vehículo, temperatura del refrigerante, la marcha colocada o la activación del ABS.
- Bus de media velocidad **MS-CAN** – Este bus funciona a 95 Kbits/s. Este bus se emplea solamente para las comunicaciones entre los sistemas de entretenimiento y el aire acondicionado.
- Bus de baja velocidad **LW-CAN** – Este bus funciona a 33 KBits/s. Se utiliza para la comunicación del resto de sistemas auxiliares como puertas, luces o control de retrovisores.

El bus utilizado en esta controladora es el de media velocidad (MS-CAN). El principal motivo de esta elección es por razones de seguridad, ya que, en caso de fallo, los sistemas críticos no se ven afectados.



La ilustración 40 muestra algunos de los componentes que están conectados al MS-CAN y donde se encuentran ubicados. Es prácticamente la única información pública disponible acerca del bus de datos del Opel Astra H [9]. Se encuentra en un manual creado exclusivamente para los talleres autorizados de la marca.

La controladora puede conectarse en cualquier parte del bus de datos, aunque se ha decidido colocarla en la guantera o bajo la tapa del freno de mano por su facilidad de acceso y proximidad al conector de diagnóstico.

5.2.4 – COMPONENTES DEL VEHÍCULO

En la ilustración 34 se han citado distintos componentes del vehículo y su distribución por el interior del automóvil. En este punto vamos a ver en detalle qué funciones tienen los componentes principales [6].

5.2.4.1 – DIS –PANTALLA ORIGINAL

DIS (de *Display* o *Pantalla*) es el nombre que recibe este componente, el cual está formado por una unidad controladora y una pantalla para mostrar información. Dependiendo del año del vehículo y la gama seleccionada existen cuatro modelos distintos de la pantalla, los cuales están descritos en la tabla 23.

Nombre	Descripción
TID – Triple Info Display <i>(Ilustración 41)</i>	<ul style="list-style-type: none"> Este modelo no llegó a comercializarse en España y sólo se incluyó en los primeros modelos vendidos de la marca en el extranjero. También estaba presente en el modelo de vehículo anterior (Opel Astra G). Sólo dispone de dos líneas para mostrar información.
BID – Board Info Display <i>(Ilustración 42)</i>	<ul style="list-style-type: none"> Este fue el primer modelo en llegar a España. Se suministraba con los vehículos vendidos en los primeros años (alrededor de 2004) Incluye de tres líneas de información, lo que le permite mostrar información acerca del sistema de infoentretenimiento.
GID – Graphical Info Display <i>(Ilustración 43)</i>	<ul style="list-style-type: none"> Este modelo también se suministraba desde los primeros años en España, aunque se incluía en el modelo deportivo (GTC) y gamas medias / altas. Distribuye la información en pantalla de forma mucho más clara.
CID – Colour Info Display <i>(Ilustración 44)</i>	<ul style="list-style-type: none"> Soporte para mostrar contenido a color. Disponible a partir del año 2008. Al basarse en una pantalla LCD tiene mucho más potencial que los anteriores modelos. Aunque no de forma oficial, soporta entradas de vídeo analógicas de cámaras traseras y/o laterales.

TABLA 23 – OPEL ASTRA H – TIPOS DE PANTALLA



ILUSTRACIÓN 41 – MODELO DE PANTALLA TID



ILUSTRACIÓN 42 – MODELO DE PANTALLA BID



ILUSTRACIÓN 43 – MODELO DE PANTALLA GID



ILUSTRACIÓN 44 – MODELO DE PANTALLA CID

5.2.4.1.1 – ENVÍO DE TEXTO A LA PANTALLA

El método empleado para enviar textos a la pantalla es un poco complejo debido entre otras cosas, a la longitud variable de los textos y al tamaño máximo de mensaje CAN.

Por ello la pantalla emplea el estándar **ISO 15765-2**, el cual divide los datos en mensajes de menor tamaño con un identificador de secuencia para poder re-ensamblarlo en el otro extremo. A grandes rasgos realiza una función similar al protocolo TCP en la pila de protocolos TCP/IP.

Antes de comenzar a enviar texto es necesario realizar una solicitud a la pantalla para proceder con el envío. Esto se realiza mediante un mensaje al que se le denomina *DIS_TEXT_REQUEST*, el cual está descrito en la tabla 24.

El mensaje tiene los siguientes campos:

- Comando – Indica el comando para enviar un texto. Sólo se utiliza el 0xC000.
- Textos – En plural. Es posible con una única solicitud enviar varios textos a varias posiciones distintas.
- Tipo de texto – La categoría a la que corresponde el texto:
 - 0x01 – Sección de radio FM.
 - 0x03 – Sección de pantalla principal. (La única compatible con la pantalla BID).
 - 0x05 – Sección de origen CD.
 - 0x07 – Sección de teléfono móvil.
 - 0x09 – Sección de navegador GPS.
 - 0x0A – Sección del climatizador / aire acondicionado.
- Posición – A qué se corresponde el texto y donde debe mostrarse:
 - 0x01 – Nombre de la emisora de radio.
 - 0x10 – Nombre de la canción.
 - 0x24 – Temperatura del climatizador.

Mensaje <i>DIS_TEXT_REQUEST</i>								
ID	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]	data[7]
0x6C1	0x10 (Solicitud)	Longitud mensajes (bytes)	Comando (Siempre 0xC000)		Longitud Textos (bytes)	Tipo de texto	ID Posición	Longitud Posición

TABLA 24 – OPEL ASTRA H – FORMATO DEL MENSAJE *DIS_TEXT_REQUEST*

Tras el envío de este mensaje, instantáneamente recibiremos una confirmación para poder realizar el envío del texto. A esta confirmación se le denomina *DIS_REQUEST_ACK*, el cual está descrito en la tabla 25.

Mensaje <i>DIS_REQUEST_ACK</i>								
ID	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]	data[7]
0x2C1	0x30 ACK - OK	0x00	0x00	0x00	0x00	0x00	0x00	0x00

TABLA 25 – OPEL ASTRA H – FORMATO DEL MENSAJE DIS_REQUEST_ACK

Tras todas las investigaciones desconozco si hay alguna respuesta NACK, ya que nunca he recibido ningún valor distinto de 0x30 ni hay información pública al respecto. En cualquier caso, el desarrollo contempla esta posible situación y se reintenta el envío del texto pasados unos segundos en caso de recibir un valor distinto.

Por último, se procede al envío del texto en formato *Unicode-16*. Esto quiere decir que por cada carácter a enviar se van emplear 2 Bytes. Además, no hay que olvidar que al emplear el estándar ISO 15765-2 sólo tenemos 7 bytes de datos con cada envío, lo que quiere decir que realmente con cada mensaje se enviarán 3 caracteres y medio.

La tabla 26 muestra un ejemplo del envío del texto “1234567890” empleando los mensajes denominados *DIS_SEND_TEXT*.

Secuencia de mensajes <i>DIS_SEND_TEXT</i>								
ID	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]	data[7]
0x6C1	«Secuencia» 0x21	0x00	0x31	0x00	0x32	0x00	0x33	0x00
		'1'		'2'		'3'		½ '4'
0x6C1	«Secuencia» 0x22	0x34	0x00	0x35	0x00	0x36	0x00	0x37
		½ '4'	'5'		'6'		'7'	
0x6C1	«Secuencia» 0x23	0x00	0x38	0x00	0x39	0x00	0x30	-
		'8'		'9'		'0'		

TABLA 26 – OPEL ASTRA H – FORMATO DE LA SECUENCIA DE MENSAJES DIS_SEND_TEXT

Hay un inconveniente a la hora de enviar textos de mayor longitud que la admitida en línea principal de la pantalla. Al no caber entero, el texto simplemente se corta al ser recibido y se ignora la parte sobrante.

Esto está solventado en el software implementado en la controladora genérica que recoge este documento simulando el efecto de una marquesina en movimiento. De esta forma el texto va desplazándose poco a poco y se visualiza por completo.

5.2.4.1.2 – DIAGNÓSTICO DEL VEHÍCULO – “MODO TEST” O “MENÚ SECRETO”

El “Modo Test” es el nombre que recibe el menú oculto del Opel Astra H que permite visualizar información del vehículo en tiempo real.

Recibe este nombre ya que al activarse muestra el texto “Test Mode” por la pantalla, pero realmente lo único que hace es mostrar un menú oculto en las opciones de la pantalla. Está implementado en el hardware y en el software de la pantalla.

Toda la información que muestra está accesible por el bus de datos, por lo que realmente no se altera ningún parámetro interno del vehículo o se llega más allá que a la simple visualización de algunos valores internos.

Es posible habilitar este modo con la radio original manteniendo pulsado el botón «Settings» hasta que se escuche un pitido por los altavoces (alrededor de 8 o 10 segundos) y posteriormente pulsar el botón «BC». El problema reside en que si quitamos la radio original perdemos el acceso a estos botones y no es posible realizar la secuencia. La controladora genérica permite acceder de nuevo a este menú mediante la emulación del equipo de sonido original.

Este modo tiene una apariencia como la que podemos observar en las ilustraciones 45 y 46.

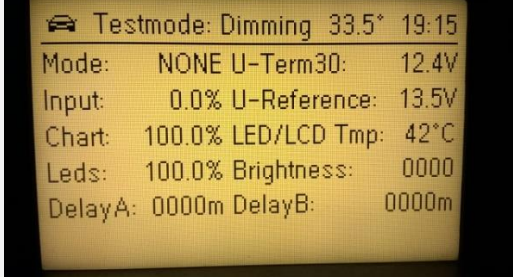
	<ul style="list-style-type: none"> • U-Term30: El voltaje de la batería (terminal 30) • LED/LCD Tmp: Temperatura de la pantalla. • Chart y LEDs: Brillo de la pantalla y de los instrumentos.
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ILUSTRACIÓN 45– “MODO TEST” O “MENÚ SECRETO” (1)

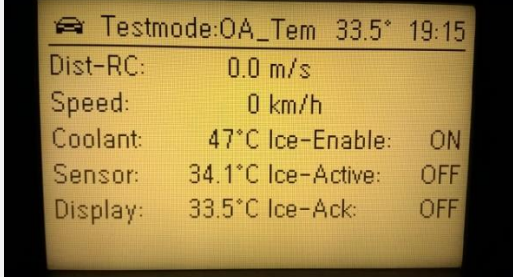
	<ul style="list-style-type: none"> • Speed: La velocidad actual del automóvil. • Coolant: La temperatura del líquido refrigerante. • Sensor: La temperatura del aire en la entrada al motor. • Ice-Active: Si se muestra o no el aviso por posibles placas de hielo en el asfalto.
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ILUSTRACIÓN 46 – “MODO TEST” O “MENÚ SECRETO” (2)

5.2.4.2 – EHU – ENTERTAINMENT UNIT

La “EHU” es el nombre interno que recibe el sistema de sonido original del vehículo.

En la ilustración 47 podemos ver una imagen del equipo de sonido CD30 MP3.

Existen múltiples sistemas de sonido originales al igual que las pantallas, pues varían dependiendo de la gama del vehículo y de la fecha de compra.

En este caso carece de sentido entrar en detalle en los distintos modelos, pues simplemente aumentan sus funcionalidades internas, pero no son pertinentes para el desarrollo de este proyecto.

Este componente está íntimamente ligado con la pantalla, pues incluye ciertos botones referentes a la misma. En la ilustración 47 podemos visualizar los botones «Settings» y «BC»



comentados anteriormente además de una cruceta de dirección que se utiliza para navegar y modificar algunos parámetros del vehículo como por ejemplo la fecha y la hora.

En circunstancias normales, si este componente es reemplazado por uno no oficial se vuelven inaccesibles todas las funcionalidades a las que se accede con esos botones. Pero gracias a este proyecto es posible simular los botones perdidos y no perder el acceso a unas configuraciones tan básicas como la fecha y la hora o el idioma del sistema.

Es importante tener en cuenta que, si reemplazamos este componente por otro no original, el nuevo no se conectará al bus de datos y el vehículo no será consciente de este cambio. Esto provoca el registro de un error interno debido a la falta del equipo, lo cual no resulta problemático, pero puede causar confusión en caso de averías.

Para subsanarlo disponemos de dos opciones:

- Desde la herramienta de configuración y diagnóstico oficial para talleres autorizados (OPCOM) es posible indicar que el equipo de infoentretenimiento está desconectado. Esto provoca que el “Modo Test” deje de estar disponible incluso aunque se envíe por el CAN BUS los mensajes para habilitarlo.
- La controladora genérica incluye un modo de simulación de EHU para que este error no se registre y se siga manteniendo el acceso a todas las funcionalidades.

5.2.4.3 – CIM – COLUMN INTEGRATED MODULE / STEERING COLUMN MODULE

Este componente recibe el nombre de “Column Integrated Module” en los vehículos fabricados hasta el año 2006, pero en adelante se denomina “Steering Column Module”. Se encuentra situado en el volante del automóvil.



También se encarga de tomar las pulsaciones de los botones situados en el volante (ver ilustración 48) y convertirlas en mensajes que se envían por el bus CAN.

Se podría decir que es el coordinador central del bus de datos de media velocidad, pues también está conectado al resto de buses de datos del automóvil y retransmite información importante entre los distintos buses de datos. Como por ejemplo el voltaje de la batería y la temperatura del líquido refrigerante.

Entre sus funciones podemos encontrar la comunicación del estado del contacto por el bus de datos. Es decir, es el encargado de notificar al resto de sistemas de que el vehículo tiene la llave correcta introducida y va a ser arrancado o incluso que ya está encendido.

Esos mensajes siguen el formato descrito a continuación en la tabla 27.

Mensaje de pulsación o liberación de botones en el volante			
ID	data[0]	data[1]	data[2]
0x206	<ul style="list-style-type: none"> • 0x01 – Pulsación • 0x00 – Liberación 	«Número de secuencia empezando en 0x01, sólo para pulsación»	<ul style="list-style-type: none"> • 0x84 – Botón rueda izquierda. • 0x91 – Botón superior derecho. • 0x92 – Botón inferior derecho. • 0x81 – Botón superior izquierdo. • 0x82 – Botón inferior izquierdo.

TABLA 27 – OPEL ASTRA H – FORMATO DEL MENSAJE DE LA PULSACIÓN O LIBERACIÓN DE LOS BOTONES DEL VOLANTE

Si la pulsación se mantiene, cada 100ms se enviará un nuevo mensaje incrementando el contador de secuencia. La tabla 28 muestra un ejemplo de pulsar y mantener el botón superior derecho durante 350ms.

ID	data[0]	data[1]	data[2]	Momento de envío
0x206	0x01 – Pulsación	0x01	0x91	0ms
0x206	0x01 – Pulsación	0x02	0x91	100ms
0x206	0x01 – Pulsación	0x03	0x91	200ms
0x206	0x01 – Pulsación	0x04	0x91	300ms
0x206	0x00 – Liberación	0x00	0x91	350ms

TABLA 28 – OPEL ASTRA H – EJEMPLO DE MENSAJES AL MANTENER UN BOTÓN DURANTE 350ms.

Además de botones, el volante cuenta con un par de ruedas a cada lado que también generan comandos en el bus de datos de forma muy similar a la ya vista. Se puede ver en detalle en la tabla 29.

Mensaje al mover las ruedas situadas en el volante			
ID	data[0]	data[1]	data[2]
0x206	0x08 Movimiento de rueda	<ul style="list-style-type: none"> 0x93 – Rueda derecha 0x83 – Rueda izquierda. 	<ul style="list-style-type: none"> 0x01 – Arriba 0xFF – Abajo

TABLA 29 – OPEL ASTRA H – FORMATO DEL MENSAJE AL MOVER LAS RUEDAS DEL VOLANTE

5.2.5 – ACCIONES IMPLEMENTADAS

La especificación de vehículos normalmente añade nuevas acciones. A continuación, en las tablas 30 a 33 se detallan los comandos del Opel Astra H.

Vehículo – Prefijo OPT		
Función	Parámetros	Descripción
get	1. Nombre del módulo	Obtiene o establece si se debe activar un módulo específico. <u>Módulos válidos:</u> <ul style="list-style-type: none"> SIM – Simulación de radio LED – Uso de LEDs adicionales. CAR – Monitor de estado del vehículo. TST – Modo “Test” DIS – Módulo de la pantalla.
set	1. Nombre del módulo 2. Valor	

TABLA 30 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – OPT

Vehículo – Prefijo LOG		
Función	Parámetros	Descripción
get	1. Nombre del módulo	Obtiene o establece si se debe mostrar un registro específico. <u>Registros válidos:</u> <ul style="list-style-type: none"> DIS – Pantalla principal
set	1. Nombre del módulo 2. Valor	

TABLA 31 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – LOG

Vehículo – Prefijo BC		
Función	Parámetros	Descripción
settings	-	Simula la pulsación del botón “Settings”.
bc	-	Simula la pulsación del botón “BC”.
up	-	Simula la pulsación de un botón de la cruceta de dirección.
down	-	
left	-	
right	-	
TABLA 32 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – BC		

Vehículo – Prefijo DIS		
Función	Parámetros	Descripción
set	1. Texto a mostrar	Establece el texto a mostrar por la pantalla principal.
TABLA 33 – ACCIONES – FUNCIONES ESPECÍFICAS DEL VEHÍCULO – DIS		

7 – PRESUPUESTO

En este punto se van a recoger tanto los elementos necesarios como recomendables para poder ensamblar el proyecto.

7.1 – HERRAMIENTAS

A continuación, en la tabla 34 se citan las herramientas empleadas para el montaje del proyecto.

Nombre	Precio medio (€)	Descripción
Polímetro digital	30,00 €	Un polímetro digital para realizar mediciones y comprobar la continuidad entre distintos puntos del circuito.
Osciloscopio USB	70,00 €	Necesario para analizar señales. Al ser USB es muy cómodo de llevar al coche junto al ordenador portátil.
Pelacables	15,00 €	Recomendable, especialmente para pelar muchos cables rápidamente y sin quitar ninguna vena del interior.
Soldador (punta 1mm)	45,00 €	Recomendable el JBC 14ST de 11W o equivalente.
Impresora 3D	250,00 €	Se puede encargar la impresión del mecanizado a un proveedor y ahorrarse la compra.
Pistola de silicona caliente	10,00 €	Se puede reemplazar por algún otro tipo de adhesivo.
Pistola de aire caliente	17,00 €	Se utiliza para contraer los tubos termo-retractiles. Es posible reemplazarlo por un mechero o el propio soldador, pero no es recomendable.
Limas metálicas de relojería	10,00 €	Recomendable para finalizar los acabados del mecanizado.

TABLA 34 – PRESUPUESTO – HERRAMIENTAS

7.2 – COMPONENTES

La tabla 35 contiene todos los componentes y piezas necesarias para ensamblar la controladora genérica al completo, con todos los módulos opcionales y su mecanizado.

Nombre	Precio unitario (€)	Cantidad	Subtotal
Placa base (Desde proveedor JLC PCB)	0,90 €	1	0,90 €
Arduino Nano v3	4,00 €	1	4,00 €
Módulo CAN MCP2515	4,30 €	1	4,30 €
Módulo GPS NEO-6MV2	15,00 €	1	15,00 €
Módulo GSM SIM800L	15,00 €	1	15,00 €
Módulo Bluetooth HC-05 / HC-06	8,00 €	1	8,00 €
Potenciómetro digital MCP4131	1,00 €	1	1,00 €
Fuente de alimentación conmutada 5A	2,80 €	2	5,40 €
Optoacoplador PC817	0,10 €	1	0,10 €
Mecanizado (Controladora + Fte. alimentación)	45,00 €	1	45,00 €

TABLA 35 – PRESUPUESTO – COMPONENTES

7.3 – CONSUMIBLES

La tabla 36 muestra los consumibles necesarios para ensamblar el circuito. Se asume como consumible los elementos de pequeño valor que simplemente se emplean para interconectar o ensamblar el resto de componentes.

Nombre	Precio unitario (€)	Cantidad	Subtotal
Resistencia (220Ω)	0,10 €	3	0,30 €
Resistencia (1000Ω)	0,10 €	1	0,10 €
Resistencia (2200Ω)	0,10 €	1	0,10 €
Diodo 1N914	0,10 €	1	0,10 €
Conmutador SPDT de 3 pines	0,05 €	2	0,10 €
Conector Jack 3.5mm hembra	0,65 €	1	0,65 €
Conectores de 3 pines macho y hembra (pack)	0,05 €	6	0,30 €
Conectores Dupont 2.54mm hembra (40 pines)	0,15 €	4	0,60 €
Tornillo M4	0,02 €	4	0,08 €
Espaciador / tuerca M4	0,02 €	4	0,08 €
Regleta de conexiones eléctricas AWG 22	1,15 €	1	1,15 €
Cable AWG 22 (preferible cobre) (metro)	2,00 €	5	10,00 €
Estaño 99% puro. 1mm de grosor (rollo)	12,00 €	1	12,00 €
Plástico para impresora 3D (1KG)	18,00 €	1	18,00 €
Tubos de adhesivo termofusible (silicona)	0,10 €	2	0,20 €
Bridas de sujeción	0,05 €	10	0,50 €
Tubo termo-retráctil	0,02 €	20	0,40 €

TABLA 36 – PRESUPUESTO – CONSUMIBLES

7.4 – DESARROLLO

En la tabla 37 se especifican la cantidad de horas empleadas para llevar a cabo el desarrollo completo y la documentación de este proyecto.

Nombre	Precio hora (€)	Cantidad	Subtotal
Hora de desarrollador	12,00 €	300	3.600,00 €
Hora de tutor	24,00 €	30	720,00 €

TABLA 37 – PRESUPUESTO – DESARROLLO

7.5 – COSTES PERIÓDICOS

En este punto se detallan los costes mensuales que se deben asumir para tener en funcionamiento la plataforma de seguimiento, los cuales están visibles en la tabla 38.

Nombre	Precio mes (€)
Plataforma Traccar (sin servidor dedicado) (1 cuenta)	10,00 €
Tarifa de datos GSM	6,00 €

TABLA 38 – PRESUPUESTO – COSTES PERIÓDICOS

7.6 – TOTAL

A continuación, en la tabla 39, se detalla el total del sumatorio de todas las partes del presupuesto acorde a si se adquiere una impresora 3D o no.

Apartado del presupuesto	Comprando Impresora 3D	Precio total del apartado (€)
Herramientas	Sí	447,00 €
	No	187,00 €
Componentes y piezas	Sí	51,10 €
	No	96,10 €
Consumibles	Sí	35,69 €
	No	17,69 €
Desarrollo	-	4320,00 €
Costes mensuales (1 año)	-	192,00 €
TOTAL	Sí	5.045,69 €
	No	4.812,69 €
PRECIO POR UNIDAD (Producción en masa)	-	40,00 €

TABLA 39 – PRESUPUESTO – TOTAL

8 – CONCLUSIÓN

Estoy muy satisfecho con el proyecto realizado, ya que es un ejemplo de los pasos necesarios para llevar a cabo el desarrollo de un producto comercial. Personalmente creo que ha tenido en mí un impacto mayor que el propio proyecto en sí, ya que me ha servido para no tenerle reparo a trabajar en entornos críticos o que puedan parecer complicados.

Inicialmente traté de dar forma a una vaga idea e investigué si era posible llevarla a cabo. Me encontré frente a una arquitectura completamente desconocida para mí como era la empleada en automoción. Es importante destacar que en estos casos no existe ningún tipo de información sobre la estructura interna del vehículo accesible al consumidor. Esto me forzó a analizar de primera mano el esquema del cableado, el diseño de los distintos componentes y las técnicas empleadas en su construcción entre otros.

Tras crear un pequeño prototipo funcional me centré en aprender gran cantidad de técnicas para mejorar la calidad y la eficiencia del software. Lo que me sirvió para seguir aumentando las funcionalidades implementadas sin cambiar el hardware base, ya que pude exprimir hasta el máximo sus capacidades tecnológicas. Por otro lado, también me sirvió para depurar al máximo el software y darle la robustez necesaria como para ser fiable en un entorno real.

Aprendí a diseñar placas de circuito impreso utilizando herramientas profesionales, lo cual creo que es un gran añadido a mi formación y puede serme de gran ayuda en el futuro con el desarrollo de prototipos y productos comerciales.

Finalmente, creo que el desarrollo de un componente de estas características resulta interesante y adecuado para aplicar los conocimientos obtenidos durante el estudio del grado directamente en infraestructuras desarrolladas por profesionales para el mundo real. Gracias a ellos pude darle forma y comprender el complejo sistema al que me estaba enfrentando.

8.1 – IMPACTO ESPERADO

Espero que este proyecto pueda ser útil para cualquier persona que esté considerando ampliar las características de su vehículo y necesite de una base fiable y robusta para llevarlo a cabo.

Pero aparte de la aplicación directa de la controladora en sí, también estaría gratamente satisfecho si este proyecto le sirviese a alguien como un ejemplo de las pautas a seguir y le ayudase a perder el miedo a la hora de enfrentarse a un entorno de estas características.

8.2 – POSIBLES MEJORAS

Al ser un proyecto genérico y modular, las posibilidades de ampliación son casi ilimitadas. Algunos ejemplos de temáticas susceptibles de mejora son:

- Soporte para nuevos modelos de vehículos.
- Mayor modularidad y parametrización en tiempo de ejecución.
- Nuevas funcionalidades.
- Integración completa con un software multimedia para reemplazar completamente al sistema de infoentretenimiento del vehículo por uno personalizado.

- El código binario para el Arduino Nano V3 ocupa alrededor del 95% de la memoria flash y el 80% de la memoria RAM. Sería recomendable cambiar a otro Arduino de mayor capacidad como el Mega 2560.

9 – BIBLIOGRAFÍA

- **[1] *An introduction to GSM***
Redl, Siegmund H. | Weber, Matthias K.; Oliphant, Malcolm W. | Boston ; London : Artech House, cop. 1995.
- **[2] *Android***
Burnette, Ed | Madrid : Anaya Multimedia cop. 2011.
- **[3] *Building apps for the universal Windows platform : explore Windows 10 Native, IoT, HoloLens, and Xamarin***
Chatterjee, Ayan, author. | Place of publication not identified : Apress, 2017.
- **[4] *C Programming for Embedded Systems***
Zurell, Kirk | Chapman and Hall/CRC | 2000
- **[5] *Electrónica básica***
Ballester Berman, Josep David | Martínez Marín, Tomás | San Vicente del Raspeig : Universidad de Alicante, D.L. 2006.
- **[6] *Estudio y manual de Taller del Opel Astra H Diésel 1.7 CDTI 100.***
Edición 2006-2007.
- **[7] *Exploring C for microcontrollers : a hands on approach***
Parab, Jivan S; Shelake, Vinod G.; Kamat, Rajanish K.; Naik, Gourish M. | Dordrecht; London : Springer, cop. 2007.
- **[8] *Guía práctica del GPS***
Correia, Paul | Barcelona : Marcombo, D.L. 2002.
- **[9] *Información oficial sobre el MS-CAN en el modelo Opel Astra H.***
<https://es.scribd.com/doc/211592390/Can-bus-Svel-02>
- **[10] *Learning Raspberry Pi***
Shah, Samarth | Olton Birmingham: Packt Publishing Ltd | 2015
- **[11] *Manual imprescindible de Arduino práctico***
Ribas Lequerica, Joan | Madrid : Anaya Multimedia, 2013.
- **[12] *Mecánica del automóvil actualizada***
Calvo Martín, Jesús | Miravete de Marco, Antonio; Universidad de Zaragoza. Centro Politécnico Superior | Zaragoza : Centro Politécnico Superior, Universidad de Zaragoza, 1997.
- **[13] *Microelectronics : from fundamentals to applied design***
Di Paolo Emilio, Maurizio, author. | Cham, Switzerland : Springer, 2016. | 1st ed. 2016.
- **[14] *New efficient communication services for ISO 11898 networks***
Cena, Gianluca ; Valenzano, Adriano
Computer Standards & Interfaces, 2000, Vol.22(1), pp.61-74
- **[15] *New version of NMEA 0183 standard released***
Ocean News & Technology, Sep 2012, Vol.18(8), pp.68-69
- **[16] *PetrolHeadGarage - Información general sobre mecánica***
<https://petrolheadgarage.com>

- **[17]** ClubAstraH.Net – Foro específico del vehículo Opel Astra H
<http://www.clubastrah.net>