



Universidad Popular Autónoma del Estado de Puebla

Centro Interdisciplinario de Posgrados
Investigación y Consultoría
Departamento de Ingeniería
Doctorado en Ingeniería Mecatrónica

“OBDII Plus - Desarrollo y estandarización de
protocolo de mensajes para periféricos del
automóvil en sistema OBDII / CAN-BUS”

Tesis que para obtener el grado de
Doctor en Ingeniería Mecatrónica

Presenta

Alejandro García Blanco

Asesor de tesis

Dr. Edgar Peralta Sánchez

Puebla, México

2018



UPAEP – Secretaría General

Dirección General de Apoyos Académicos

Dirección del Centro de Recursos para el Aprendizaje y la Investigación.

Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©

PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Página en blanco

I Carta de Autorización



Universidad Popular Autónoma del Estado de Puebla

Centro Interdisciplinario de Posgrados

Investigación y Consultoría

Departamento de Ingeniería

Doctorado en Ingeniería Mecatrónica

Se aprueba la tesis:

“OBDII Plus - Desarrollo y estandarización de
protocolo de mensajes para periféricos del
automóvil en sistema OBDII / CAN-BUS”

Alejandro García Blanco

Comité de Tesis Doctoral

Director de Tesis

Secretario

Vocal

Puebla, México

2018

Página en blanco

II Agradecimientos

Al Dr. Julio Cesar Ramírez, quien me dio el primer impulso para iniciar este sueño.
Sus primeros consejos aún perduran y cuentan

A la Dra. Rubí Salazar, quien me supo llevar de la mano por todo este camino
Hasta donde la salud le permitió

Al Dr. Edgar Peralta, quien supo tomar esta alma perdida
Y darle la última palmadita para poder cruzar la meta

A la Dra. Georgina Rosas, por saber ponerle la última cerecita a este proyecto

Y muy especialmente a mis dos pajaritas Alondra y Alotzin, y por sobre todo su mamá
mi novia de toda la vida
Gracias por su paciencia para seguir este sueño

Página en blanco

III Índice

I	Carta de Autorización.....	3
II	Agradecimientos	5
III	Índice	7
IV	Lista de Tablas.....	15
V	Lista de Figuras.....	17
VI	Abstract.....	21
VII	Resumen	23
1	Introducción	25
1.1	Establecimiento de la problemática.....	25
1.2	Justificación del Estudio	27
1.3	Objetivo Central de la Tesis	27
1.4	Alcance de la tesis	28
1.5	Preguntas de Investigación.....	30
1.6	Impacto Esperado	30
1.7	Organización del Estudio	30
2	Marco Teórico	33
2.1	Introducción.....	33
2.2	Historia del OBD	34
2.3	Arquitectura de un Automóvil	38
2.4	Modelo OSI del OBD II.....	46
2.5	El OBD En breve, incluyendo que son los Modos, los PID y los DTC.....	50
2.6	Los DTCs a fondo (SAE J2012 / ISO1031-6).....	54
2.7	Los PIDs a fondo (SAE J1979 / ISO15031-5)	57

2.8	Los Modos o Servicios de PID	61
2.9	El PID Modo 01- Datos Actuales.....	64
2.10	El PID Modo 02 Diagnostico	74
2.11	El PID Modo 03- DTC	75
2.12	El PID Modo 04- borrado	78
2.13	El PID Modo 05 - Sensores	79
2.14	El PID Modo 06- Pruebas Abordo.....	81
2.15	El PID Modo 07- DTC	81
2.16	El PID Modo 08- Prueba de Actuadores	81
2.17	El PID Modo 09- Información General del vehículo VIN	81
2.18	El PID Modo 10 (Modo 0A) – DTC códigos de falla permanentes	83
2.19	El PID Modo \$11 al \$15 (0x0A al 0x0F).....	83
2.20	Los PID Modos Extendidos.....	83
2.21	Los Header bytes.....	86
2.22	Protocolos por Fabricante.....	98
2.22.1	Protocolo SAE J1850 PWM – Capa Física y Datos en Ford / Mazda	98
2.22.2	Protocolo SAE J1850 VPW – Capa F. y D. en GM y camiones ligeros..	98
2.22.3	Protocolo ISO9141-2 – Capa Física y Datos en Chrysler	99
2.22.4	Protocolo ISO14230-1 a -4 – Capa Física y Datos KWP2000	99
2.22.5	Protocolo ISO 15765-1 a -4 – CAN-BUS.....	99
2.23	Protocolo SAE J1962 - Conector	100
2.24	El IC ELM327 – Enlace OBD a RS232	103
2.25	Herramientas de diagnóstico populares - Adaptadores	105
2.25.1	Interface Genérica ELM327 V1.5 OBDII	106
2.25.2	El BAFX Products Bluetooth OBDII Scan Tool.....	109

2.25.3	SparkFun OBD-II Diagnostics Kit	112
2.25.4	Freematics OBD-II Telematics DIY Kit	114
2.25.5	Innova 3100 Car Scanner	118
2.26	Tendencias a futuro en la tecnología de redes en vehículos	121
2.26.1	eCall El taller del futuro: comunicación directa con el auto	121
2.26.2	El Sistema AUTOSAR.....	123
3	Metodología de la Tesis	125
3.1	Etapas del proyecto	125
3.2	Detalle de la Metodología	125
4	Requerimientos de producto	129
4.1	Vehículos Más comunes en México.....	129
4.2	Módulos vehiculares comunes.....	135
4.3	Sistemas Seleccionados	135
4.4	Propósito del producto	137
4.5	Punto de vista del cliente	137
4.5.1	Perfil de usuarios.....	137
4.5.2	Expectativas de cliente	138
4.5.3	Objetivos.....	138
4.6	Principios Básicos del Producto	139
4.7	El PRD – Documento de Requerimientos de Producto	139
4.7.1	Propósito del Producto	139
4.7.1	Funciones y características	139
4.7.2	Especificaciones Técnicas.....	140
4.7.3	Especificaciones Técnicas – Software y protocolo	140
4.7.4	Especificaciones Técnicas – Hardware	143

5	Desarrollo de la contribución central de la tesis	151
5.1	Introducción.....	151
5.2	Análisis de la arquitectura de un automóvil	154
5.3	Módulos CAN-BUS, Alta y mediana velocidad.....	155
5.4	Los PIDs existentes no de control de emisiones.....	158
5.5	Tipos de mensajes y direccionamiento	159
5.6	El estándar SAE J2178 – Mensajes de comunicación	161
5.7	El estándar SAE J2178-1 – Direccionamiento Físico.....	162
5.8	El estándar SAE J2178-2 –Parámetros y Direccionamiento Funcional.....	165
5.9	El estándar SAE J2178-3 – Estructura para headers de un byte	169
5.10	El estándar SAE J2178-4 – Definición para headers de 3 bytes	169
5.11	Propuesta de PIDs no – estándar sobre CAN BUS – el OBDII <i>Plus</i>	172
5.11.1	Reglas para el OBDII Plus	173
5.11.2	Estructura del mensaje de OBDII Plus	178
5.11.3	Ejemplos para el OBDII Plus.....	179
5.12	Ventajas y desventajas del direccionamiento OBDII <i>plus</i>	183
6	Implementación de Hardware	185
6.1	Esquema General	185
6.2	Adaptador OBD.....	186
6.3	Emulador OBD	189
6.4	Central Electrónica del dominio de seguridad y confort CeBECU.....	190
6.5	Red Área Local CAN BUS	191
6.6	Subsistemas del Vehículo	193
6.7	Módulo de Luces.....	193
6.7.1	Circuito switch de luces	194

6.7.2	Circuito sensado de corriente	197
6.7.3	Circuito selector de switches	200
6.7.4	Modulo CAN BUS	202
6.7.5	Diseño del PCB	203
6.7.6	Implementación en PCB	204
6.8	Módulo HID	207
6.8.1	Circuito switch selectores	207
6.8.2	Circuito LEDS indicadores.....	208
6.8.3	Modulo CAN BUS	209
6.8.4	Modulo Central Virtual	210
6.9	Sistema Completo	211
7	Implementación de Software.....	213
7.1	Adaptador OBD -.....	214
7.1.1	Intérprete de PIDs	214
7.1.2	Decodificador del valor del PID	215
7.1.3	Leer PIDs.....	217
7.1.4	Display PIDs	218
7.2	Decoder del OBDII plus	220
7.3	Encoder del OBDII plus.....	222
7.4	Intérprete OBDII Plus	224
7.5	Mapa de comandos OBDII plus para luces.....	225
7.6	Interface(s) de usuario	227
7.7	Puente de comunicación UART / OBD/ CAN BUS/ GUI	227
7.8	Supervisor de sistema.....	229
7.9	Firmware específico para Hardware – Relay/Luces.....	230

7.9.1	Mapeo de Arduino Nano - Luces	230
7.9.2	Activación de relays.....	232
7.9.3	Sensor de luz en funcionamiento	236
7.10	Firmware específico para Hardware – HID	238
7.10.1	Mapeo de Arduino Nano - HID	238
7.10.2	Banderas de Estado.....	241
7.10.3	Main Loop	241
7.10.4	Lectura de botones.....	243
7.10.5	OBD Encode	245
7.10.6	OBD Decode	247
8	Validación	251
8.1	Simulación en ambiente virtual	251
8.1.1	Algoritmos de codificación y decodificación.....	251
8.1.2	10 ejemplos aleatorios.....	254
8.2	Aplicación en ambiente real	257
8.2.1	Prueba de luz de cuartitos (Park)	258
8.2.2	Prueba de luz de luces principales (headlamps)	260
8.2.3	Prueba de luz de luces altas (high beam).....	262
8.2.4	Prueba de luz de freno (Brake).....	264
8.2.5	Prueba de luz de Reversa	266
8.2.6	Prueba de luces intermitentes (hazard).....	268
8.2.7	Prueba de luces direccional izquierda (turn left)	270
8.2.8	Prueba de luces direccional derecha (turn right)	272
8.2.9	Prueba de monitoreo de estados.....	274
8.2.10	Prueba de puente de comunicación	275

9	Análisis e interpretación de resultados.....	277
10	Conclusiones.....	281
11	Contribuciones originales	285
12	Evidencia de la productividad académica resultado del proyecto de tesis	286
12.1	Coloquio UPAEP 2015.....	286
12.2	Coloquio UPAEP 2016.....	287
12.3	Coloquio IEEE/MAIS 2016	288
12.4	Wikilibros – El OBDII Completo	289
12.5	Artículo – Revista Ciencia y Desarrollo	290
12.6	Artículo – IEEE Capitulo Latinoamérica	291
13	Referencias	292
13.1	Estándares SAE sobre OBD-II.....	292
13.2	Estándares ISO.....	294
13.3	Referencias bibliográficas.....	297
14	Anexos	300
14.1	Tablas del SAE j2178-4	300

Página en blanco

IV Lista de Tablas

Tabla 2.4.1 - OBD especificaciones aplicables al modelo OSI (Fuente: Estándar ISO15031-6:2010).....	49
Tabla 2.4.2 Niveles de estándares en una aplicación (fuente: http://www.cnblogs.com/shangdawei/p/3570546.html)	50
Tabla 2.5.1 – Modos y DTCs.....	52
Tabla 2.6.1 – Especificaciones Generales de Código (fuente: SAE J2012).....	54
Tabla 2.7.1 – Formato de mensaje de requerimiento de datos (SAE, 2006).....	58
Tabla 2.7.2 – Formato de mensaje de respuesta de datos (SAE, 2006).....	59
Tabla 2.7.3 – Formato de mensaje de respuesta de datos negativa (SAE, 2006)	59
Tabla 2.7.4 – Servicios especificados (SAE, 2006).....	60
Tabla 2.8.1 – Mapa completo de modos o servicios (Fuente: SAE J2190:1993)	62
Tabla 2.9.1 - Modo 1 PIDs	73
Tabla 2.10.2 - Modo 1 PID 1C estándares soportados	74
Tabla 2.10.1 - Modo 2 PIDs	74
Tabla 2.11.1 - Modo 3 DTC.....	75
Tabla 2.11.2 – Primer carácter DTC.....	75
Tabla 2.11.3 – Segundo carácter DTC.....	75
Tabla 2.11.4 – Tercer carácter DTC.....	76
Tabla 2.12.1 - Modo 04 – DTC.....	78
Tabla 2.13.1 – Modo 05 Sensores	80
Tabla 2.17.1 – Modo 09 VIN	83
Tabla 2.20.1 Modos Extendidos (Fuente: SAE J12190, 1993).....	85
Tabla 2.21.1 byte del encabezado (fuente: SAE J1850)	89
Tabla 2.21.2 byte del encabezado (fuente: SAE J1850)	89
Tabla 2.21.4 Cuadro completo de datos (Fuente: SAE J2178)	91
Tabla 2.21.5 Encabezados reservados para diagnostico (fuente: SAE J1979).....	91
Tabla 2.21.6 Cuadro de datos CAN-BUS (fuente: ISO 15765-4)	92
Tabla 2.21.7 Requerimiento de dato en CAN-BUS	95
Tabla 2.21.8 Respuesta en CAN-BUS	96
Tabla 2.23.1 – Asignación de pines en el conector OBD (Fuente: SAE J1962).....	102

Tabla 4.1.1 – Ventas de autos en México (Fuente: AMDA, 2016)	131
Tabla 4.1.2 – Ventas de autos en México por segmento (Fuente: AMDA, 2016).....	133
Tabla 4.1.3 – Ventas de autos en México por modelo (Fuente: AMDA, 2016).....	133
Tabla 4.2.1 Sistemas presentes en los autos más vendidos (Fuente: Elaboración propia con información de cada fabricante)	134
Tabla 4.2.2 Otros Sistemas (Fuente: Elaboración propia con información de cada fabricante)	135
Tabla 4.3.1 – Listado condensado de sistemas (Fuente: Elaboración propia)	136
Tabla 4.7.4.1 – Direccionamiento físico seleccionado (Fuente: Elaboración propia) ..	144
Tabla 5.4.1 – Costo de membresía ETI (ETI, 2015).....	158
Tabla 5.4.2 – Costos de contrato con OEMs (ETI, 2016).....	158
Tabla 5.7.1 tipos de direcciones extendidas.	163
Tabla 5.7.2 - SAE J2178 – Direccionamiento Extendido Geográfico	163
Tabla 5.7.3 - SAE J2178 – Asignaciones validas	164
Tabla 5.7.3 - SAE J2178 – Direccionamiento Funcional, Ejemplo	165
Tabla 5.11.2.1 Estructura del mensaje en OBDII-plus (Fuente: Elaboración propia) ..	178
Tabla 7.5.1 Comandos OBDII plus para luces	226
Tabla 7.5.2 Respuestas de estatus OBDII plus para luces	226
Tabla 7.9.1.1 Comandos OBDII plus para luces	231
Tabla 8.1.1.1 Estructura del mensaje en OBDII-plus (Fuente: Elaboración propia)	252

V Lista de Figuras

Figura 2.3.1 – dominios de sistemas en el vehículo (Fuente: elaboración propia con imagen de publicidad Ford Fiesta © 2016).....	41
Figura 2.3.2 – Diagrama del Dominio del Tren Motriz (Fuente: elaboración propia con imagen de publicidad Ford Fiesta © 2016).....	42
Figura 2.3.3 – Diagrama del Dominio del chasis (Fuente: elaboración propia con imagen de publicidad Ford Fiesta © 2016).....	43
Figura 2.3.4 – Ejemplo de una puerta en el Diagrama del dominio de los sistemas auxiliares de seguridad y confort (Fuente: Navet & Simonot-Lion, 2009)	45
Figura 2.5.1 Estructura código PID OBD (González, 2008)	52
Figura 2.6.1 – Estructura de los DTC (Fuente: SAE J2012).....	55
Figura 2.6.2 – ejemplo DTC 2 bytes (Fuente: SAE J2012)	56
Figura 2.6.3 – ejemplo DTC 3 bytes (Fuente: SAE J2012)	56
Figura 2.21.1 Estructura del paquete de datos (Fuente: SAE J1850)	87
Figura 2.21.2 Estructura del cuadro de datos (Fuente: SAE J1979)	88
Figura 2.21.3 – Encabezado extendido según SAE J2178	89
Figura 2.23.1 – Conector OBD, medidas en mm (Fuente: SAE J1962)	101
Figura 2.23.2 – asignación de números de pines (Fuente: SAE J1962)	101
Figura 2.25.1.1 – Adaptador OBD Genérico	106
Figura 2.25.1.2 – Adaptador OBD Genérico - Circuito	108
Figura 2.25.4.1 – Tablero SparkFun OBDII.....	114
Figura 2.25.4.2 – Tablero SparkFun OBDII - Circuito.....	116
Figura 2.25.4.3 – Adaptación UART.....	117
Figura 2.25.5.1 – Innova 3100 Car Scanner	119
Figura 2.25.5.2 – INNOVA3100 - Circuito	120
Figura 2.26.1 - Botón eCall de BMW (Warnecke, 2016)	122
Figura 4.1.1 – Ranking de ventas de autos por marca (fuente: AMDA, 2016)	132
Figura 5.5.1 Estructura del paquete de datos (Fuente: SAE J1850)	160
Figura 5.7.1 formato de campo de datos funcional tipo 3.....	162
Figura 5.7.2 Direcccionamiento extendido	164
Figura 5.8.1 – Bytes PRN.....	166

Figura 5.8.2 – Grupos PRN y rangos de direcciones	167
Figura 5.10.1 – Estructura de mensaje OBD (Fuente: SAE J2178-1)	171
Figura 5.10.2 – Ejemplo de tabla de mensajes (Fuente: SAE J2178-4).....	172
Figura 5.11.1 – formato de header para CAN BUS (Fuente: ISO 15765-4)	173
Figura 5.11.1.1 – Tabla 11 SAE J2178-1 (Fuente: SAE J2178).....	174
Figura 5.11.1.2 – Bit Q (Fuente: SAE J2178-4).....	176
Figura 5.11.4 – Bit W (Fuente: SAE J2178-1)	176
Figura 5.11.1.3 – Bit C (Fuente: SAE J2178-4).....	176
Figura 5.11.14– Ejemplo de mensaje PID, PRN, SLOT (Fuente: SAE J2178-2, J2178-4)	177
Figura 5.11.3.1 – Ejemplo de mensaje (Fuente: SAE J2178-4)	179
Figura 5.11.3.2 – Ejemplo de mensaje (Fuente: SAE J2178-4)	180
Figura 5.11.3.3 – Ejemplo de mensaje (Fuente: SAE J2178-4)	181
Figura 5.11.3.4 – Ejemplo de mensaje (Fuente: SAE J2178-4)	182
Figura 5.11.3.5 – Ejemplo de mensaje (Fuente: SAE J2178-4)	182
Figura 5.11.3.6 – Ejemplo de mensaje (Fuente: SAE J2178-2)	183
Figura 6.1 – Esquema general de la estructura de Hardware	185
Figura 6.2.1 – Pines TX/RX UART del módulo OBD.....	187
Figura 6.2.2 – Circuito completo	187
Figura 6.2.3 – Display LCD	188
Figura 6.2.4 – Tablero implementado.....	188
Figura 6.3.1 – Emulador OBD sencillo	190
Figura 6.5.1 – Shield CAN-BUS linksprite	192
Figura 6.7.1.1 – Circuito implementado con relay	194
Figura 6.7.1.2 – Circuito implementado con MOSFET N	195
Figura 6.7.1.3 – Circuito implementado con MOSFET P.....	196
Figura 6.7.1.4 – Circuito con relay / mosfet.....	197
Figura 6.7.2.1 – Resistencias Shunt.....	198
Figura 6.7.2.2 – Circuito completo – Un Canal.....	199
Figura 6.7.2.3 – Relays y sus mosfets junto al procesador	199
Figura 6.7.3.1.- Switches pushbutton	201

Figura 6.7.3.2 – Switches selectores.....	201
Figura 6.7.4.1 – Modulo CAN-BUS conectado.....	202
Figura 6.7.5.1 – Circuito completo – Un Canal.....	203
Figura 6.7.5.2. – Circuito Impreso	204
Figura 6.7.6.1.- Primera implementación	204
Figura 6.7.6.2.- Primer Prototipo funcional.....	205
Figura 6.7.6.3.- Versión Final	206
Figura 6.7.6.4.- Versión Final - reverso	206
Figura 6.8.1.- Modulo HID	207
Figura 6.8.2. - Modulo HID	208
Figura 6.8.3.- Modulo HID - Bluetooth.....	209
Figura 6.9.1. - Sistema completo	211
Figura 7.1.4.1 – Implementación del código para lectura de PIDs	219
Figura 7.9.1.1 – Mapeo pines Arduino nano	230
Figura 7.10.1.1 – Mapeo pines Arduino nano	239
Figura 8.1.1.2 – Ejemplo de mensaje (Fuente: SAE J2178-4)	253
Figura 8.1.1.3 – Fragmento de la Tabla 1 de ID Primarios (Fuente: SAE J2178-4)	254
Figura 8.2.1.1 – Respuesta de los módulos ante el comando enviado	259
Figura 8.2.2.1 – Respuesta de los módulos ante el comando enviado	261
Figura 8.2.3.1 – Respuesta de los módulos ante el comando enviado	263
Figura 8.2.4.1 – Respuesta de los módulos ante el comando enviado	265
Figura 8.2.5.1 – Respuesta de los módulos ante el comando enviado	267
Figura 8.2.6.1 – Respuesta de los módulos ante el comando enviado	269
Figura 8.2.7.1 – Respuesta de los módulos ante el comando enviado	271
Figura 8.2.8.1 – Respuesta de los módulos ante el comando enviado	273

Página en blanco

VI Abstract

Vehicular emission control systems were addressed in the 1990s for the first time. There were basically 5 different systems until 2008 when the CAN BUS became the preferred standard. However, CAN BUS only standardized the physical communication between systems. With the growing inclusion of electronics subsystems none related to emission control, each vehicle maker developed its own set of protocols and intercommunication codes. This work proposes the basis of an Open hardware platform standard for non-emissions related systems based on the current OBDII structure from the standard SAE J2178 adapted over CAN BUS message frames.

Página en blanco

VII Resumen

Los sistemas de control de emisiones en los vehículos tuvieron su primera aparición en la década de 1990. Hubo básicamente 5 sistemas diferentes hasta el 2008 cuando el CAN BUS se volvió el estándar preferido. Sin embargo, CAN BUS sólo estandarizó la comunicación física entre sistemas. Con la creciente inclusión de sistemas electrónicos no relacionados al control de emisiones, cada fabricante de vehículos desarrollo su propio conjunto de protocolos y códigos de comunicación. Este trabajo propone las bases para un estándar en base a una plataforma de hardware abierto para sistemas vehiculares no relacionados al control de emisiones, basado en la estructura actual del OBDII del estándar SAE J2178 adaptado sobre la estructura de mensaje CAN BUS

Página en blanco

1 Introducción

La tecnología al ir avanzando ha permitido hacer más complejos y confiables los diversos dispositivos que se usan en la actualidad. En los automóviles, en la década de los '90 es cuando se señala como la era donde la electrónica llegó a estos. Iniciando con la incorporación cada vez más común de sensores eléctricos y en diversos lugares, hasta el desarrollo de una computadora que centralice toda esa información y coordine los actuadores correspondientes. Dicha computadora abrió la posibilidad de hacer más dinámicos e inteligentes los sistemas automotrices, permitiendo por ejemplo incrementar la eficiencia de combustible a la vez de reducir emisiones tóxicas.

En 1996, todos los autos vendidos a partir de ese año incorporaron el protocolo OBDII. Este estándar incorporó un set de mensajes fijos de estatus del tren motriz, el cual se podría decir universal dado que cualquier marca de auto maneja los mismos mensajes básicos. Esto permitió estandarizar las herramientas de diagnóstico, e incluir el famoso “check engine” en los carros, o luz indicador de problema en el motor. A principios de la década del 2000, la marca BOSCH lanzó al mercado la propuesta de un protocolo Ethernet, el CAN-BUS. Este protocolo contempla comunicación serial asíncrona entre dispositivos utilizando un solo canal a través de un único par trenzado que enlaza a todos los dispositivos. La ventaja es la reducción significativa de alambrado al dejar de usar un cable dedicado para cada función. La desventaja es la complejidad agregada y la inherente necesidad de módulos dedicados y especializados que decodifiquen dicha comunicación serial y la traduzcan nuevamente en cables dedicados a cada función. Los autos a partir del 2008 incorporan este protocolo (PR, 2015)

Con la entrada de Redes de datos a los vehículos a través del sistema CAN-BUS, esto permitió incorporar más módulos de control separados dentro del mismo vehículo, pero compartiendo información.

1.1 Establecimiento de la problemática

OBDII es la abreviatura de “On Board Diagnostics” versión II. Este sistema automotriz se desarrolló en la década de los 80's originalmente para monitorear los niveles de contaminantes emitidos por los automóviles. La primera versión se volvió mandatorio

para todos los automóviles vendidos en Estados Unidos a partir de 1991. La segunda versión marcó una mejora con respecto a su versión inicial. Incluyó estandarización del conector, señales, y una lista de parámetros estándar de señalización de estado del tren motriz. Dentro de estas características, la más reconocida es la aparición en los automóviles de la ahora común luz indicadora de problema, o “check engine” (Henderson, Haynes 2006)

La tecnología actual OBDII está definida por varios estándares SAE enfocados a los niveles de capa física (niveles y señales de voltaje) y capa de datos y red en cuanto a la estructura de los paquetes de datos y mensajes estándar (Carley, 2008).

Sin embargo, actualmente solo está definido para la computadora a bordo que maneja el motor y tren motriz, tal cual se definió en la década de los 90's. Actualmente los vehículos comunes de bajo y mediano nivel incorporan de manera estándar diversos elementos periféricos como son vidrios y seguros eléctricos, luces adicionales a los faros y cuartos, 3ra luz de freno y cajuela eléctrica, entre otros.

Las compañías automotrices grandes han empezado a implementar por medio del sistema CAN BUS (similar a Ethernet) protocolos propietarios buscando señalar estos periféricos. Sin embargo, no hay una normativa establecida que defina dichos mensajes y protocolos. Existen estándares como el SAE J2284-3 y el ISO 11898 los cuales definen la capa física y de enlace de datos del CAN. Algunos otros como el ISO 15765 CAN-BUS manejan comunicación entre dispositivos de diagnóstico y el automóvil, pero igualmente no entre periféricos del automóvil

La orientación de este trabajo de tesis es buscar normalizar los mensajes necesarios para todos los periféricos de un automóvil. Esto buscando expandir el protocolo ya conocido OBDII.

El impacto futuro buscado es, por medio de esta estandarización de protocolos, permitir el desarrollo de una base de hardware para periféricos en el automóvil, que a su vez permita aplicaciones de más alto nivel fácilmente logrables. Entre dichas aplicaciones puede ser enlace de la alarma o la posición de los cristales con el teléfono, cada luz independiente y con intensidad variable, altura de cristales y aire acondicionado dependiente de la velocidad del auto, sistemas de alarma inteligentes con aviso directo

al teléfono del usuario, envío de emails con estatus, entre otros (Actron, 2006), (Childers, 2013), (Moore, 2013), (Wojdyla 2013)

Este sistema podría también ser aplicable para vehículos anteriores a 2008 (implementación de CAN-BUS mandatorio) por medio de alambrar el mismo (Investor's business 2013)

1.2 Justificación del Estudio

De acuerdo a datos del Instituto de Estadística Geografía e Informática de México (INEGI) En el país hasta el 2008 (anterior a la implementación de CAN-BUS) había 19 millones de automóviles. Al 2014 se sumaron apenas 5 millones. Esto nos da un buen indicativo que, si bien en la Industria Automotriz hay tecnología de punta, la gran mayoría de los vehículos en circulación no la tienen. Esto representa un segmento de mercado potencial atractivo para crear tecnología fácilmente adaptable a estos vehículos ya en circulación.

Con respecto a los nuevos vehículos, prácticamente cada compañía automotriz ha desarrollado sus propios sistemas y protocolos siendo incompatibles entre sí. Uno de los objetivos principales del presente trabajo es el de buscar un sistema que sea lo más estándar posible para todo vehículo independiente de su marca y manufactura. De esta manera, al desarrollar módulos universales se puede potencializar su desarrollo y reducir costos por volumen

1.3 Objetivo Central de la Tesis

Como el título lo refleja, el objetivo central de la tesis es el desarrollo y estandarización de un protocolo de mensajes para periféricos del automóvil en sistema OBDII / CAN-BUS. Como objetivos más específicos se tienen los siguientes:

- Investigar el estado actual de los protocolos automotrices, para conocer la panorámica actual del estado de arte de los sistemas de señalización internos de los automóviles.
- investigar a fondo el protocolo OBDII, buscando agrupar los tipos de mensajes, establecer relaciones entre protocolos, conocer a detalle las señalizaciones eléctricas y

la estructura de los mensajes, entre otros. Lo anterior buscando dejar bases a opciones para expandir dicho protocolo.

- investigar todos los periféricos del auto candidato a incluir en el protocolo, por ejemplo los sistemas de luces y de las puertas.
- Mapear periféricos y organizarlos en grupos, con una orientación funcional hacia estructura de protocolos de señalización.
- Identificar maneras de expandir el actual OBDII, comparando opciones para seleccionar la más óptima al proyecto.
- Asignar claves a los periféricos. En esta etapa es ya definir propiamente la expansión del protocolo
- Desarrollo de una plataforma de hardware para pruebas. Dicha plataforma se pretende sea básica y mayormente orientada a auxiliar el desarrollo y puesta a punto de los protocolos. No obstante, se procurará sea funcional y accesible a ser duplicable fácilmente en base a tecnología comercial económica, como plataformas Arduino con conectividad CAN-BUS entre otros.
- Seleccionar una plataforma para dichas pruebas. Tentativamente seleccionar el sistema de luces por su simplicidad de hardware, pero que cubre diversos direccionamientos
- Prueba de subsecciones del protocolo en la base de hardware. Por cuestiones de tiempo se anticipa no poder probar todas las posibles combinaciones del protocolo, por lo que se buscara probar las más representativas y/o comunes y extrapolar resultados a la totalidad del protocolo.
- Pruebas finales y ajustes para definir completamente el proyecto.

1.4 Alcance de la tesis

El presente trabajo busca incorporar y estandarizar mensajes para la gran mayoría de los periféricos que puedan encontrarse dentro de un automóvil. El objetivo principal es el planteamiento del protocolo. Sin embargo, se busca además desarrollar las bases de hardware, sin limitarlo excesivamente, pero buscando algún punto de estandarización para producción en masa. Como ejemplo, desarrollar módulos universales para puertas que puedan servir para cualquiera de las 4 puertas y cualquier marca de auto.

Actualmente hay algunos módulos en el mercado, pero dedicados específicamente para cierta marca, modelo y lado de puerta.

Dada la esperada enorme cantidad de periféricos y combinaciones de estos, se espera poder abarcar solo una fracción de dichos mensajes en la aplicación de prueba de la tesis. Se buscará seleccionar los periféricos más comunes y representativos para su desarrollo, como son elevadores de cristales de puertas y sistemas de luces. De igual manera el potencial de hardware se espera sea la base para una gran cantidad de posibles aplicaciones de alto nivel.

El enfoque de la tesis es a bajos niveles, no a aplicaciones finales. Tomando como base el modelo OSI, la capa 1 física ya estaría definida por el protocolo CAN-BUS. La capa 2 de datos, 3 de red y 4 de transporte se tomaría como base la actual OBDII. Para la 5 de sesión y 6 de presentación a nivel bajo de códigos de protocolo es en donde este trabajo tendría su mayor enfoque al definir tanto la estructura y los paquetes de datos, así como los protocolos de comunicación tipo maestro/esclavo. No es el enfoque el desarrollar una aplicación de usuario final acorde con las capas 6 de presentación y 7 de aplicación. Sin embargo para fines de desarrollo se tendrá que desarrollar una aplicación de interface de usuario final que permita la evaluación del protocolo dentro de lo delineado en este trabajo.

Como investigación básica, se busca el expandir el protocolo actual hacia el área de periféricos no estandarizada actualmente. La investigación aplicada comprende el asentar el protocolo expandido en un nuevo protocolo funcional.

Una vez definido el protocolo, se busca su desarrollo tecnológico por medio de la plataforma de hardware, con una orientación de tecnología abierta al desarrollo compartido.

Para pruebas preliminares se planea utilizar algún simulador de protocolos como Gary (2009), Green (2015), y Gary (2015). Los resultados se buscará difundirlos por medio de foros de entusiastas automotrices y/o de la automatización, para ver de manera inicial el impacto que pudiese tener esta tecnología y su aceptación.

Así también, se buscará el acercamiento con empresas locales del ambiente automotriz, en busca de colaboración para el desarrollo del estado del arte actual

1.5 Preguntas de Investigación

Las siguientes preguntas se espera sean abordadas conforme se desarrolle esta tesis

- ¿Qué es el OBD y para qué sirve?
- ¿Qué hace y que no hace el OBD?
- Lo que no hace el OBD ¿Cómo se puede abordar?
- Los periféricos de un auto ¿es posible estandarizar sus comandos?
- ¿Es práctica y útil dicha estandarización?
- La base de hardware, ¿representa ventajas en costo?

1.6 Impacto Esperado

Se espera poder desarrollar una plataforma de protocolo de interconexión y control de periféricos comunes en los autos de hoy en día. Dicha plataforma se pretende ser una extensión de la muy popular OBDII en uso hoy en día, la cual actualmente se enfoca solamente al tren motriz y control de emisiones.

Esta plataforma se busca sea abierta, u open source, de manera que atraiga el interés de más desarrolladores a todos niveles que impulsen la plataforma.

Si bien no es el objetivo principal del presente trabajo el desarrollo de hardware asociado, también este se busca que sea tipo open source para hacerlo atractivo a su desarrollo popular

1.7 Organización del Estudio

El presente estudio se procura realizar en tres etapas de alrededor de seis meses cada una:

- La primera es realizar un estudio del estado del arte de las tecnologías actuales en periféricos automotrices. De ahí, realizar un estudio más a profundidad del protocolo actual OBDII y protocolos asociados resultantes del estudio del arte. En conjunto con este estudio, es determinar los subsistemas secundarios más populares de los automóviles, a llamar sistemas periféricos.

- La segunda etapa es, en base a lo aprendido en protocolos actuales, expandir los mismos a incluir los periféricos identificados. En esta etapa se desarrollaría el protocolo para incluir utilizando su misma estructura, o una mejorada dependiendo del estado del arte, instrucciones para control de los periféricos por este medio. Para realizar primeras pruebas se planea utilizar un simulador de OBD por medio de software, como los open source disponibles por parte de icculus.org o mp3car.com
- La tercera etapa es el desarrollo de una plataforma de hardware básica para pruebas, corrección de errores y puesta a punto del protocolo. Esta tercera etapa, su inicio podría irse empalmando con partes avanzadas de la segunda según se vea la necesidad.

Página en blanco

2 Marco Teórico

2.1 Introducción

La tecnología al ir avanzando ha permitido hacer más complejos y confiables los diversos dispositivos que se usan en la actualidad. En los automóviles, en la década de los '90 es cuando se señala como la era donde la electrónica llegó a estos. Iniciando con la incorporación cada vez más común de sensores eléctricos y en diversos lugares, hasta el desarrollo de una computadora que centralice toda esa información y coordine los actuadores correspondientes. Dicha computadora abrió la posibilidad de hacer más dinámicos e inteligentes los sistemas automotrices, permitiendo por ejemplo incrementar la eficiencia de combustible a la vez de reducir emisiones tóxicas.

A inicios de esta década de los 90's surgió el primer protocolo de comunicación, OBD (On Board Diagnostics) el cual estandarizó la manera de comunicarse en los autos a través de las diferentes compañías. Estos mensajes eran básicos y solamente de estados críticos del tren motriz (Roberts, 2009).

En 1996, todos los autos vendidos a partir de ese año incorporaron el protocolo OBDII bajo el estándar SAE J1962 y J1850 para capa física e ISO 15031 para el protocolo de comunicación. Este estándar incorporó un set de mensajes fijos de estatus del tren motriz, el cual se podría decir universal dado que cualquier marca de auto maneja los mismos mensajes básicos. Esto permitió estandarizar las herramientas de diagnóstico, e incluir el famoso "check engine" en los carros, o luz indicador de problema en el motor. Este estándar se encuentra en la gran mayoría de los carros circulando en la actualidad (Henderson, Haynes, 2006). A principios de la década del 2000, la marca BOSCH lanzó al mercado la propuesta de un protocolo Ethernet, el CAN-BUS. Este protocolo contempla comunicación serial asíncrona entre dispositivos utilizando un solo canal a través de un único par trenzado que enlaza a todos los dispositivos. La ventaja es la reducción significativa de alambrado al dejar de usar un cable dedicado para cada función. La desventaja es la complejidad agregada y la inherente necesidad de módulos dedicados y especializados que decodifiquen dicha comunicación serial y la traduzcan nuevamente en cables dedicados a cada función. Los autos a partir del 2008 incorporan este protocolo (PR, 2015)

Existe una propuesta de ley en el Estado de California, conocida popularmente como OBDIII. Actualmente no está completamente definido técnicamente aún y se mantiene solo como propuesta. El sistema conocido como OBDIII no es propiamente una siguiente revisión del sistema actual OBDII, sino una aplicación de la misma. El estado de California tiene como propuesta incluir en cada vehículo un dispositivo RFID que transmita a receptores de gobierno datos del vehículo como su condición de emisiones a la atmósfera, velocidad actual en zonas restringidas, fallas del vehículo entre otros incluyendo el VIN particular del vehículo. Esto con el fin de monitorear en tiempo real cada vehículo en el estado, y en caso de situación irregular detectada emitir la correspondiente multa automáticamente sin previo aviso al conductor (Wong, 2013). Esta propuesta ha desatado enorme polémica en el estado al estar vinculada a la discusión más general actual en los Estados Unidos de hasta donde el gobierno puede involucrarse en la privacidad de sus ciudadanos por diversas razones como la Seguridad Nacional

2.2 Historia del OBD

Para entender el nivel presente de los sistemas de diagnóstico en el vehículo, hay que remontarse algunas décadas atrás para poder apreciar cuanto ha avanzado la tecnología en los vehículos (Henderson, Haynes, 2006). Mientras que el diagnóstico en el vehículo, conocido como OBD no es un participante activo en las emisiones del vehículo, su desarrollo está relacionado estrechamente al sistema de control de emisiones de los autos y camiones

La década de los 80's, entre otros movimientos sociales interesantes se destacó por ser el inicio de un despertar de conciencia mundial en torno a los temas ecológicos y ambientales (Alberich, 1993). Es durante esta década que se iniciaron múltiples temas que son comunes hoy en día, como lo es la contaminación ambiental y el deterioro de la capa de ozono, así como el calentamiento global y su impacto en el clima.

Cuando los fabricantes de vehículos respondieron en los 90's al clamor público por un aire más limpio y pusieron más atención a las recomendaciones federales que evolucionaron más tarde en estándares y regulaciones de ley que había que cumplir, la ciencia del control de emisiones en motores de combustión estaba en sus primeros

inicios. Los inicios de la década de los 70's vieron algunos controles de emisiones incipientes, mas como parches agregados a los vehículos domésticos. Estos controles usualmente restaban poder al motor, malograban la economía de combustible y resultaban en dar al público una mala reputación al auto en los entonces llamados "controles de humo". Cuando el convertidor catalítico fue introducido a mediados de los 70's, las cosas mejoraron un poco dado que el convertidor limpiaba las emisiones del escape tan bien que los fabricantes de autos podían remover o modificar algunos de los anteriores controles ineficientes

El OBD (diagnóstico a bordo, del término On Board Diagnostics) no tuvo lugar hasta que los autos estuvieron equipados con controles computarizados. La compañía General Motors implemento una versión primeriza de OBD en algunos de sus vehículos de 1980. Conforme la inyección electrónica de combustible y otras funciones fueron siendo controladas por la computadora de abordó (El módulo de control de transmisión y motor, del termino Powertrain Module Control o PCM como es llamado hoy) la implementación del OBD se volvió práctica. El estado de California en Estados Unidos de América es quien ha liderado el camino en conseguir legislaciones cada vez más estrictas en el control de emisiones, tanto que para los fines de los 80's requirió que todos los vehículos vendidos en ese estado incorporaran un sistema OBD. Los fabricantes de autos y camiones ligeros tuvieron que desarrollar el software y hardware necesarios para poder proveer la capacidad de diagnóstico.

El objetivo del sistema OBD original era promover el aire limpio por medio de asegurar que los componentes del control de emisiones siguieran funcionando. Muchos estados de la unión americana empezaron a incluir una revisión de emisiones en su programa de revisión vehicular de seguridad anual o bienal. Pero, en la práctica mucho puede suceder al vehículo en un año entre revisiones, y las pruebas simplificadas de entonces solamente "olían" el escape mientras el vehículo estaba estacionado, no cuando estaba en movimiento. Además, estas pruebas eran de "pasa o falla". Entonces los automovilistas a los cuales fallaba su vehículo la prueba tenían que por su cuenta encontrar un taller que pudiese determinar que causo que las emisiones fueran más altas, para poder arreglarlo y presentarse a prueba nuevamente

La idea del OBD era tener un vehículo que realizara su propio monitoreo del control de emisiones, todo el tiempo, y más que eso asignar números de código que pudiesen identificar el problema, y finalmente guardar esos códigos de problema en la computadora. Una luz indicadora de problema en el tablero del vehículo podría indicar al conductor que hay un problema en el sistema de emisiones, y cuando el auto estuviese en el taller, el técnico podría extraer los códigos y saber que partes del sistema debería revisar y reparar.

Hubo algunas dificultades naturales con el sistema original OBD, hoy conocido como OBD-I asociadas a la curva de aprendizaje y desarrollo de toda nueva tecnología. Entre estas estaba la carencia de cooperación y estandarización entre fabricantes de vehículos. Cada vehículo tenía su propia leyenda para el aviso de falla, en algunos se leía “Service Engine Soon” (en inglés dar servicio al motor pronto), mientras que en otros decía “Check Engine” (en inglés revisar el motor). El conductor no siempre sabía que esta luz indicaba un problema solamente en el sistema de control de emisiones, y que debía ser reparado inmediatamente.

Cada fabricante tenía su propia versión de códigos, haciendo más difícil para los técnicos leer y diagnosticar. Si se le agrega que en ocasiones la luz se apagaba después de un tiempo, al conductor se le olvidaba pensando que tal vez el vehículo se había arreglado solo.

Cuando el Congreso de Estado Unidos paso la Iniciativa del aire limpio de 1990 (Clean Air Act Amendments, Viladrich, 2004) el siguiente nivel del OBD fue creado y mandatorio. Con esto, la mayoría de los problemas del anterior sistema fueron direccionados.

El ahora conocido como sistema OBDII requirió que todos los fabricantes utilizarían un sistema uniforme de letras y números para los códigos, compartieran la misma definición de un código, y tuvieran una luz estándar de aviso de problema. El conector en el vehículo donde la herramienta de diagnostica puede conectarse es ahora de un diseño uniforme entre fabricantes

Bajo el OBDII, no solo son monitoreados los controles de emisión del motor, también son monitoreados partes del sistema de combustible por escapes de vapor, y se incluyen sensores para darle seguimiento a la eficiencia del convertidor catalítico. Los

componentes del sistema de control de emisiones pueden ahora marcar un código incluso si no han fallado completamente, pero se encuentran al 50% de su efectividad. Una reparación temprana de estos componentes debe resultar en un aire más limpio para todos nosotros.

Las regulaciones del OBDII fueron mandatorias para todos los vehículos a partir de los modelos 1996, pero algunos modelo 1994 y la mayoría de los 1995 tenían ya el sistema operando en el mismo.

Regresando a febrero de 1986, la compañía alemana Robert Bosch GmbH introdujo el sistema de bus serial de Red de área con controladores (del Ingles CAN – Controlled Area Network) presentándolo en ese año en el Congreso de la Sociedad de Ingenieros Automotrices (del Ingles SAE – Society of Automotive Engineers) (CiA, 2016). Este fue el momento que marco el nacimiento de uno de los protocolos de red más exitosos de hoy en día. Este mismo es usado en la gran mayoría de los vehículos nuevos actuales así como en numerosas aplicaciones en la Industria.

A principios de los 80's los ingenieros en Bosch estaban evaluando sistemas seriales de la época con respecto a posibles aplicaciones en autos de pasajeros. Dado que ninguno de los protocolos disponibles en ese entonces cumplía con las expectativas automotrices, el grupo de Bosch desarrollo su propio protocolo. La idea principal detrás del desarrollo era la reducción drástica de cableado en el automóvil. De pasar a tener un cable dedicado para cada función, a tener un solo bus recorriendo el auto. Los ingenieros de Mercedes – Benz se sumaron al inicio con los primeros requerimientos del nuevo protocolo, así como Intel con miras a ser proveedor de los semiconductores. En febrero de 1986, el nuevo protocolo CAN nació de las manos del Dr, Uwe Kiencke durante el Congreso de la SAE en Detroit (Kiencke, Dais, Litschel, 1986). La anécdota cuenta (CiA, 2016) que después de la presentación del artículo hubo un silencio absoluto en la sala atestada de invitados, hasta que alguien gritó “ ! Lo hicieron!” (Del inglés “They’ve done it!”) Anticipándose al éxito que años después tendría este sistema. Después de esta presentación, un sinnúmero de artículos y presentaciones le siguieron. A mediados de 1987 Intel presento el primer chip CAN, el 82526. Poco después Phillips presento el 82C200. Estos fueron las primeras implementaciones físicas del nuevo protocolo con funciones básicas del mismo.

El estándar Bosch CAN en versión 2.0 (algunos detalles corregidos de la primera versión) fue presentada como propuesta de estándar internacional. En Noviembre de 1993 fue publicado como el estándar ISO11898. Otras revisiones y versiones le siguieron, como el ISO11898-1 para la capa de datos, ISO11898-2 para la capa física no tolerante a errores, el ISO11893-3 para capa física tolerante a errores, ISO11992 para camiones y el ISO11783 para maquinaria agrícola.

En Europa el OBD tomo la forma de EOBD y se hizo mandataria para el 2001. En vehículos de carga pesados en Europa para el 2005 y en Estados Unidos en 2008. Brasil en 2012 e India en 2013. (Posada, Bandivadekar, 2015). Para mediados de la década del 2000, en los automóviles particulares había básicamente 4 diferentes estándares OBD: SAE J1850 (FORD – PWM & GM - VPW), ISO9141-2 (Chrysler, Europa y Asia), ISO14230 (KWP2000) e ISO 15765 (CAN BUS). Para el 2008 y delante se hizo mandatorio que el único protocolo a utilizar sea el ISO15765 – 4.3, la cual es una variante del estándar CAN a 500 kbps (Cassias, Kun2007). Este es el estándar en uso hoy en día.

A inicios del 2011, Bosch y General Motors iniciaron el desarrollo de un CAN de velocidad variable, pero que además permita mayor velocidad. Esto para poder soportar la creciente demanda interna de compartición de datos entre equipos (CiA, 2016). A la fecha aún no se logra una estandarización adecuada con los demás protocolos ISO, pero se espera este sea el próximo paso

2.3 Arquitectura de un Automóvil

El automóvil es una de las invenciones de uso masivo que ha acompañado a la humanidad a través de los últimos poco más de 100 años. Hoy en día se ha convertido en parte indispensable de la sociedad civilizada junto a otras invenciones como el teléfono y la televisión. Es difícil imaginar actualmente una sociedad urbana sin el mismo. Sin embargo, a pesar de su gran impacto y de haber estado presente por más de un siglo, no es sino hasta las últimas dos décadas que ha tenido un gran avance tecnológico en los modelos comercialmente disponibles.

Por décadas, los sistemas de control interno de los automóviles fueron mayormente mecánicos. Se puede hacer memoria y recordar las viejas camionetas del abuelo, con

nada más que el motor adentro. Cuando ocupaba reparación, el abuelo se sentaba en el guardafangos con los pies adentro y sobraba espacio. En esos entonces, alrededor de los 70's los controles avanzados eran por medio de mangueras de vacío y válvulas. En especial los motores Chrysler se volvieron complejos, para su época tecnología avanzada pero delicada en cuanto a si se salía una manguera estaba en problemas siendo una pesadilla el poder encontrar la causa. Hoy en día apenas cabe una mano dentro del motor con todos los sistemas que trae.

Los primeros sistemas automatizados de control aparecieron en los 80's junto con las iniciativas globales Ecológicas. Los primeros sistemas electrónicos se dieron buscando controlar las emisiones contaminantes de los vehículos. Dichos primeros sistemas se colocaron en el motor y sistemas de emisiones. Esta entonces nueva plataforma de sistema en el motor dio paso a la siguiente gran iniciativa, la eficiencia y por ende la reducción de consumo de combustible. Para esto, se incorporaron más sistemas ya no solo en la emisión del combustible utilizado sino también en la manera que entra y se utiliza. De este punto en adelante fueron incorporándose sistemas cada vez más diversos al vehículo. Dichos sistemas no podrían ser posibles sin la automatización digital iniciada por estas iniciativas.

Entre estos nuevos sistemas está el control de contaminación ambiental por medio de algoritmos complejos, nuevos conceptos de motor como los híbridos o de reducción de potencia dinámico, y sistemas de tracción y estabilidad avanzados que involucran motor, transmisión, frenos, dirección y otros en coordinación. Se estima que gracias a los sistemas digitales, el valor agregado en un automóvil esta alrededor del 80% (Leen & Heffernan, 2002). Así también, la incursión de los sistemas digitales en los vehículos hizo permisible otros sistemas no relacionados con el tren motriz como lo son los sistemas de entretenimiento digital, guías de navegación, controles eléctricos de asientos y ventanas, entre otros.

Todos estos sistemas diferentes se pueden agrupar en dominios dentro del automóvil. De acuerdo a la definición de la Real Academia Española, un dominio es un ámbito de una actividad, o dentro de la informática su significado es la denominación que identifica a un sitio en la red y que indica su pertenencia a una categoría determinada.

Así pues, se puede utilizar el término de dominio para diferenciar grupos de sistemas mecánicos o eléctricos dentro del automóvil.

Históricamente se han identificado 5 dominios (Navet & Simonot-Lion, 2009): Tren motriz, chasis, sistemas auxiliares de seguridad y confort (en inglés “body”), interfaces de usuario, y telemática. El tren motriz está relacionado con los sistemas que participan en la propulsión longitudinal del vehículo, incluyendo el motor, la transmisión y sus componentes subsidiarios. El chasis se refiere a las cuatro ruedas y su relativa posición y movimiento. En este dominio, los sistemas son principalmente la dirección y los frenos. Los sistemas auxiliares de seguridad y confort agrupan los sistemas que no están relacionados a la dinámica de movimiento del vehículo, sino al soporte al conductor como lo son la bolsa de aire, los limpiadores de parabrisas, luces, ventanas eléctricas, aire acondicionado entre otros. La interface de usuario incluye el equipo que permite el intercambio de información entre los sistemas y el conductor como lo es interruptores y paneles de información. Finalmente, el dominio de la telemática está relacionado con componentes que permiten el intercambio de información del vehículo con el mundo exterior, como lo son el radio, sistemas de navegación, internet, pago remoto de casetas, entre otros.

De un dominio a otro, los sistemas electrónicos difieren en sus características. Por ejemplo, el tren motriz y el chasis tienen requerimientos muy estrictos de datos en tiempo real y elevados niveles de poder de computación. Sin embargo, la arquitectura del hardware del dominio del chasis está completamente distribuida a través de todo el vehículo, cubriendo los sistemas dispersos a través de todo el vehículo. El dominio de la telemática, en contraste presenta requerimientos de alto manejo de datos contenidos usualmente en un solo equipo físico. Para estos dos sistemas, las soluciones tecnológicas difieren tanto en los sistemas embebidos requeridos como sus necesidades de interconexión y manejo de datos.

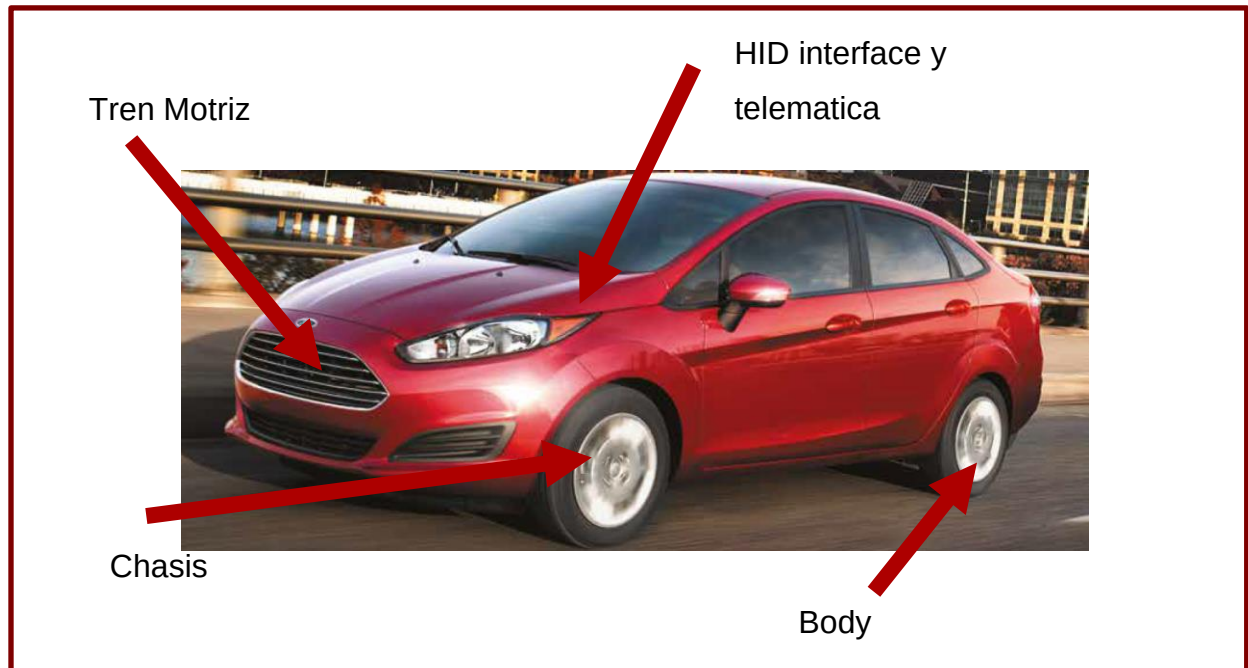


Figura 2.3.1 – dominios de sistemas en el vehículo (Fuente: elaboración propia con imagen de publicidad Ford Fiesta © 2016)

De acuerdo a Navet & Simonot-Lion, (2009) el dominio del tren motriz, como se mencionaba anteriormente, representa los sistemas que controlan el motor de acuerdo a los requerimientos del conductor, por decir acelerar o reducir velocidad de acuerdo a la posición del pedal del acelerador, o conforme a otros requerimientos de otros sistemas como lo son el aire acondicionado. El controlador también reacciona a otros factores naturales como son temperatura del aire y nivel de oxígeno, así como otros factores ambientales como son contaminación y ruido. Por otro lado, también está diseñado para optimizar otros parámetros como son facilidad de manejo y confort para el conductor, así como economía de combustible. Esto último puede ser controlado por medio de cuanto combustible es inyectado en cada cilindro en cada ocasión dependiendo de las revoluciones del motor (RPM) y la posición del acelerador. Por otro lado esto lo balancea con otros factores como posición de las válvulas admisión escape, potencia requerida, emisiones, y la lista continua. Parte de esta información es enviada a otros dominios, como lo son las revoluciones al domino de interfaces para desplegar en el tacómetro del tablero.

Desde el punto de vista funcional, el dominio del tren motriz toma en cuenta los diferentes modos de funcionamiento del motor como son baja velocidad, poca o mucha carga. Esto corresponde a velocidades de muestreo de la información en el orden de los milisegundos. Desde el punto de vista del hardware esta velocidad de muestreo requiere sensores y microcontroladores que sean capaces de satisfacer esta demanda de resolución y velocidad, a la vez de poder balancearse con el costo.

En este dominio se encuentran sistemas continuos, muestreados y discretos todos al mismo tiempo. Las reglas que los controlan tienen hasta 2000 parámetros de calibración. Esto conlleva a grandes complejidades en el control, como lo es el manejo de punto flotante con suficiente resolución para captar los cambios demandados. Además está la necesidad de manejar simultáneamente y en tiempo real muchos de estos sistemas

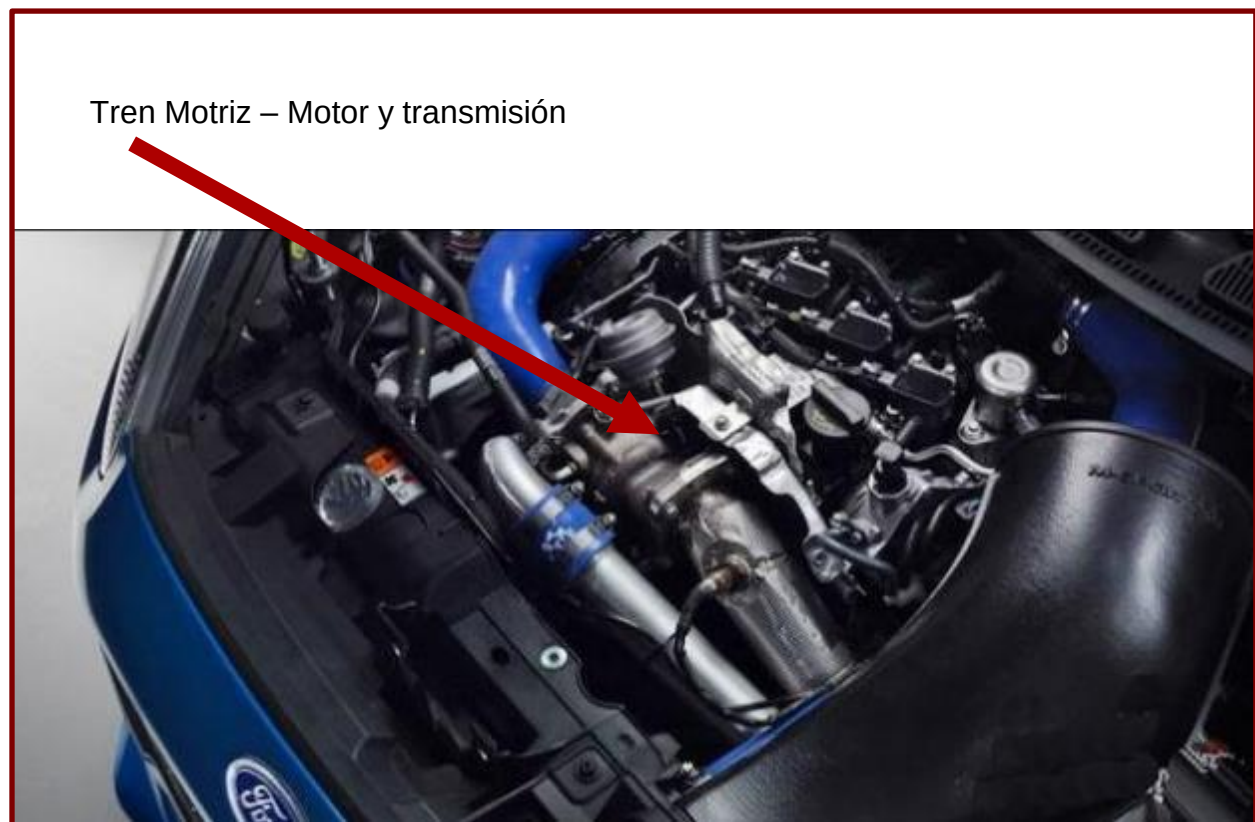


Figura 2.3.2 – Diagrama del Dominio del Tren Motriz (Fuente: elaboración propia con imagen de publicidad Ford Fiesta © 2016)

De acuerdo a Navet & Simonot-Lion, (2009) el dominio de chasis está compuesto de sistemas cuyo objetivo es el control de la interacción del vehículo con el camino, como lo son las ruedas y la suspensión. Los controladores toman en cuenta los requerimientos emitidos por el conductor, como lo es la posición del volante, frenado y aceleración, las condiciones del camino y ambientales.

Estos deben asegurar la seguridad y confort de los viajeros. En el dominio del chasis recaen en primera instancia condiciones de seguridad para los pasajeros y el vehículo en sí. Los requerimientos de estos sistemas son similares a los del tren motriz en cuanto a la velocidad a la que hay que procesar los datos en el orden de los milisegundos, así como manejar diferentes sistemas en tiempo real al mismo tiempo. La diferencia con el dominio del tren motriz está en que los sistemas del chasis están distribuidos a todo lo largo del automóvil, en vez de estar concentrados en el motor y transmisión. Así por ejemplo, se encuentran sistemas de frenado en cada rueda, los cuales junto al sistema de dirección son capaces de frenar independientemente cada una para evitar descontrolar el vehículo ante una situación inesperada por parte del conductor.



Figura 2.3.3 – Diagrama del Dominio del chasis (Fuente: elaboración propia con imagen de publicidad Ford Fiesta © 2016)

De acuerdo a Navet & Simonot-Lion, (2009) el dominio de los sistemas auxiliares de seguridad y confort controlan funciones embebidas en un vehículo que no están relacionados al control de su dinámica. Hoy en día, los limpiadores, luces, puertas, ventanas, asientos y espejos están controlados cada vez más por sistemas automatizados. En general no están sujetos a situaciones tan rigurosas como en el dominio del tren motriz y el chasis, por lo que no representan sistemas críticos. Sin embargo, hay sistemas como el de alarma de seguridad para el acceso al interior que tiene que respetar reglas de tiempo real.

Estos sistemas por lo general por estar distribuidos en todo el vehículo involucran una red de comunicación entre todos ellos haciendo más compleja su estructura. Es en este dominio donde emerge la noción de subsistemas basados en redes locales de sensores y actuadores de bajo costo, por ejemplo en red LIN (del acrónimo en inglés Local Interconnect Network o red local interconectada), la cual conecta los módulos contruidos como sistemas mecatrónicas integrados. Por ejemplo, diversas funciones pueden estar presentes en una puerta: poner o quitar seguros de acuerdo a una señal transmitida por una red inalámbrica, control de ventanas de acuerdo a señales del conductor o pasajero, así como la posición de los espejos.

Una posible aplicación de estas funciones puede ser un solo control central ECU (del inglés Electronic Control Unit, unidad de control electrónico) el cual centraliza la recepción de los sistemas que requieran la acción, mientras que los controles para los motores que realizan las acciones en el dispositivo físicamente como el espejo o la ventana son soportados por otros tres ECUs. Estos cuatro ECUs están conectados en una LIN. Como algunos requerimientos involucran varias puertas como lo es el quitar los seguros, el ECU principal de cada puerta está conectado a una red CAN de baja velocidad. Finalmente para presentar el estatus de las puertas al conductor, la información es transmitida por el ECU principal al ECU del tablero por medio de CAN-BUS.

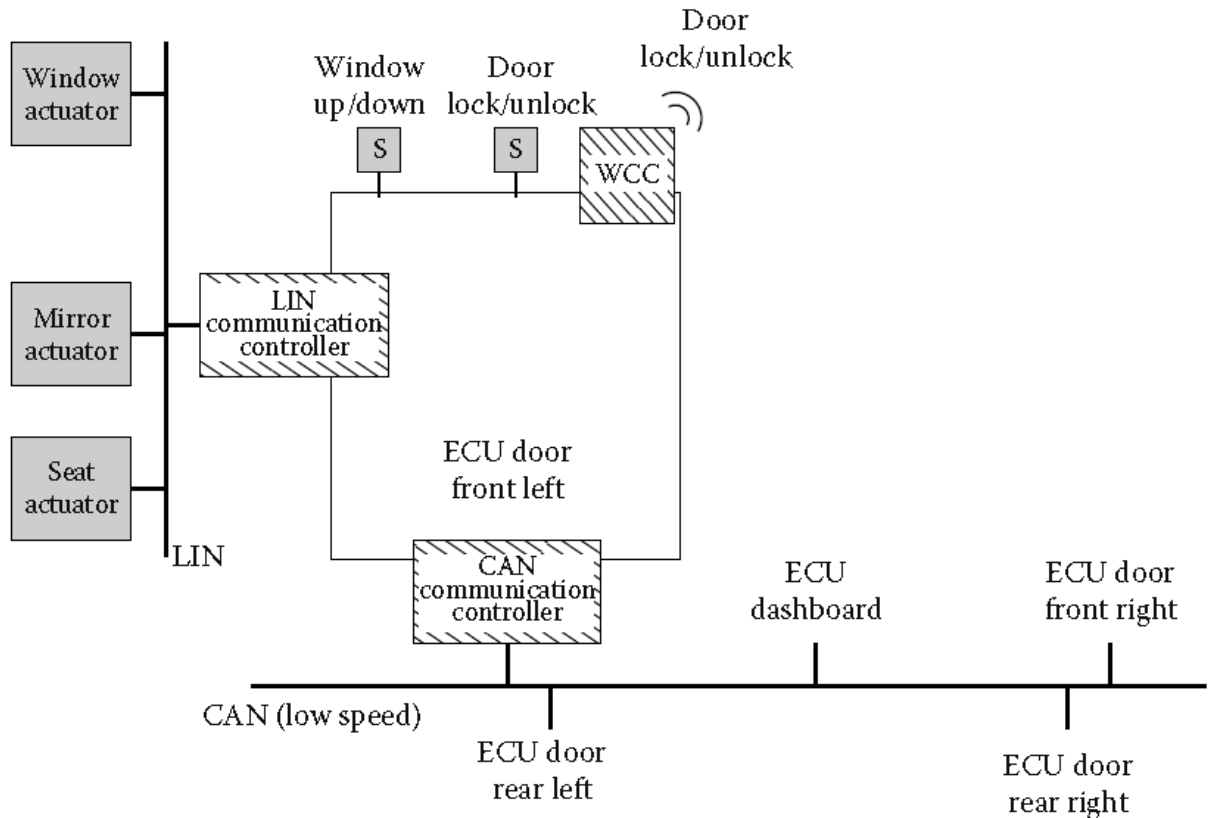


Figura 2.3.4 – Ejemplo de una puerta en el Diagrama del dominio de los sistemas auxiliares de seguridad y confort (Fuente: Navet & Simonot-Lion, 2009)

Por otro lado, este dominio también contiene un subsistema central, cuya funcionalidad principal es asegurar la transmisión de mensajes entre diferentes sistemas o dominios. Este sistema es reconocido como una entidad central crítica. Las funciones de este dominio están relacionadas mayormente a aplicaciones con eventos discretos por lo que su diseño y validación dependen en máquinas de estado y modelos síncronos. La implementación como se mencionaba implica una arquitectura distribuida. Para este sistema central se requiere alto poder de cómputo, así como tolerancia a fallas y alta confiabilidad.

Aunque no se requieren niveles de tiempo real tan estrictos como en el dominio del tren motriz, si se deben de tener en cuenta tiempos de respuesta y desempeño de diversos componentes de hardware, tiempos de acceso y llamada de cada uno y el protocolo de red entre ellos.

Todos estos se deben de balancear de manera adecuada para poder tener la flexibilidad de poder acomodar múltiples plataformas y sistemas, sin perder de vista el desempeño y confiabilidad requeridos.

2.4 Modelo OSI del OBD II

El sistema OBD en completo es en síntesis un sistema de comunicación entre módulos del vehículo y a su vez estos con el exterior a través de otros equipos o interfaces de usuario. Como todo sistema de comunicación, este puede ser modelado utilizando los niveles OSI (de las siglas en inglés Open Systems Interconnect) definido por el estándar ISO/IEC 7498-1 y el ISO/IEC 15031, el cual es universalmente conocido y empleado para estructurar sistemas de comunicación como este. La ventaja de definir los niveles en estos sistemas estriba en facilitar el delimitar los diferentes módulos y subsistemas, ayudando así a definir interfaces estándar entre estos. Así, el desarrollo de dichos módulos y subsistemas por separado se facilita y abre las posibilidades de enfocarse, por decir en una interface de usuario robusta y de gran utilidad en el mercado, sin necesidad de preocuparse por definir la comunicación entre tarjetas electrónicas.

El estándar ISO15031 consiste en una serie de secciones, las cuales juntas proveen una serie de especificaciones las cuales son orientadas al diagnóstico de emisiones en los vehículos. La parte 1 es una introducción a los estándares internacionales. Las partes 2 a 7 están basadas en prácticas recomendadas por la organización SAE.

La sección 6 está basada en el estándar SAEJ2012. Esta sección en especial contempla la comunicación entre equipos externos de prueba y las unidades a bordo del vehículo para diagnóstico (Sistemas OBD) contemplados en las diversas legislaciones.

Para dar una estructura coherente, esta sección del estándar como se mencionaba se apoya en la estructura OSI de 7 niveles. Cuando se mapean de esta manera, los servicios especificados por este estándar ISO 15031 para definir la interacción del OBD con otros sistemas, visto desde la aplicación del usuario final hacia dentro del sistema queda como sigue:

- Capa 7 – Servicios de diagnóstico especificados en:
 - ISO 15031-5 comunicación entre vehículos y equipo externo para diagnóstico de emisiones – Parte 5: Servicios de diagnóstico relacionados a emisiones;
 - ISO 27145-3 Implementación del diagnóstico armonizado mundial (del inglés World Wide Harmonized On board Diagnostics WWH-OBD) Parte 3- Diccionario de mensajes comunes;

- Capa 6 – Servicios de presentación especificados en:
 - ISO 15031-2, SAE J1930-DA Comunicación entre el vehículo y equipo externo para diagnóstico de emisiones Parte 2 – Guía de términos, definiciones, abreviaciones y acrónimos;
 - ISO 15031-5, SAE J1979-DA Modos de Prueba (del inglés Parameter Identification o PIDs);
 - ISO 15031-6, SAE J2012-DA (OBD) Comunicación entre el vehículo y equipo externo para diagnóstico de emisiones Parte 6 – Definiciones de códigos de falla (del inglés DTC Diagnostic Trouble Codes);
 - ISO 27145-2, SAE J2012-DA Implementación del diagnóstico armonizado mundial (del inglés World Wide Harmonized On board Diagnostics WWH-OBD) Parte 2-Diccionario de Datos (códigos) Común;
 - ISO 14230-3 sistemas de diagnóstico para intercambio de información por KWP2000 sección 3 – Capa de aplicación;

- Capa 5 – Servicios de sesión, especificados en:
 - ISO 14229-2 soportando los protocolos ISO 15765-4 DoCAN e ISO 14230-4 DoK-Line ;
 - ISO 14229-2 el cual no aplica en los protocolos SAE J1850 e ISO 9141-2
 - Servicios de diagnóstico unificado (del inglés UDS unified diagnostic services) en capa de servicios de sesión (del inglés Session Layer Services);

- Capa 4 – Servicios de transporte, especificados en:
 - Los siguiente estándares prácticamente definen todo de la capa 4 a la 1
 - ISO 15765-2 Comunicación de diagnóstico en redes de controladores parte 2 – servicios de transporte y red (del inglés Diagnostic communication over Controller Area Network DoCAN -- Part 2: Transport protocol and network layer services);
 - SAE J1850 definido en ISO 15031-5 Interface de comunicación de datos clase B para vehículos terrestres, con dos versiones PWM y VPW;
 - ISO 9141-2 definido en ISO 15031-5 sistemas de diagnóstico para intercambio de información serial asíncrono, similar al RS232 ;
 - ISO 14230-4 definido en ISO 15031-5 sistemas de diagnóstico para intercambio de información por KWP2000, similar al ISO9141-2;
 - SAE J1939 comunicación en vehículos de carga pesada

- Capa 3 – Servicios de Red (del inglés network layer services), especificados en:
 - ISO 15765-2;
 - SAE J1850 definido en ISO 15031-5;
 - ISO 9141-2 definido en ISO 15031-5;
 - ISO 14230-4 definido en ISO 15031-5;

- Capa 2 – Servicios de enlace de datos (del inglés data link layer), especificado en:
 - ISO 15765-4, ISO 11898-1 e ISO 11898-2; Requerimientos Generales para sistemas de control de emisiones
 - SAE J1850;
 - ISO 9141-2;
 - ISO 14230-2 comunicación de diagnóstico utilizando línea K (DoK-Line) Parte 2 – Servicios de enlace de datos;

- Capa 1 – Capa física, especificado en:
 - ISO 15765-4, ISO 11898-1 e ISO 11898-2;
 - SAE J1850;
 - ISO 9141-2;
 - ISO 14230-1 comunicación de diagnóstico utilizando línea K (DoK-Line)
- Parte 1 – Capa física;
- SAE J1962 Conector OBD

La siguiente tabla realiza un resumen de lo anterior expuesto, así como ilustrando el comparativo con la iniciativa de unificación de códigos global (WWH-OBD)

Aplicación	Capa OSI	Requerimientos comunicación OBD de emisiones				Requerimientos de comunicación WWH-OBD de emisiones		
7 capas según ISO/IEC 7498-1 & ISO/IEC10731	capa7 - aplicación	ISO 15031-5				ISO 27145-3		
	capa 6- presentación	ISO 15031-2, ISO 15031-5, ISO 15031-6				ISO 27145-2		
		SAE J1930-DA/SAEJ1979-DA/ SAE J2012-DA(OBD)				SAE J2012-DA (WWH-OBD)		
	capa 5 - sesión	ISO 14229-2		NO APLICA		ISO 14229-2		
	capa 4 - transporte	ISO 15765-2	ISO 15765-4	ISO 15031-5			ISO 14230-4	
	capa 3- red							
	capa 2 - datos	ISO 11898-1		SAE J1850	ISO 9141-2	ISO14230-2		
	capa 1 - física	ISO 11898-2				ISO14230-1		

Tabla 2.4.1 - OBD especificaciones aplicables al modelo OSI (Fuente: Estándar ISO15031-6:2010)

De una manera más simplificada, los estándares se pueden representar de la siguiente manera para ilustrar los niveles dentro de una aplicación de diagnóstico

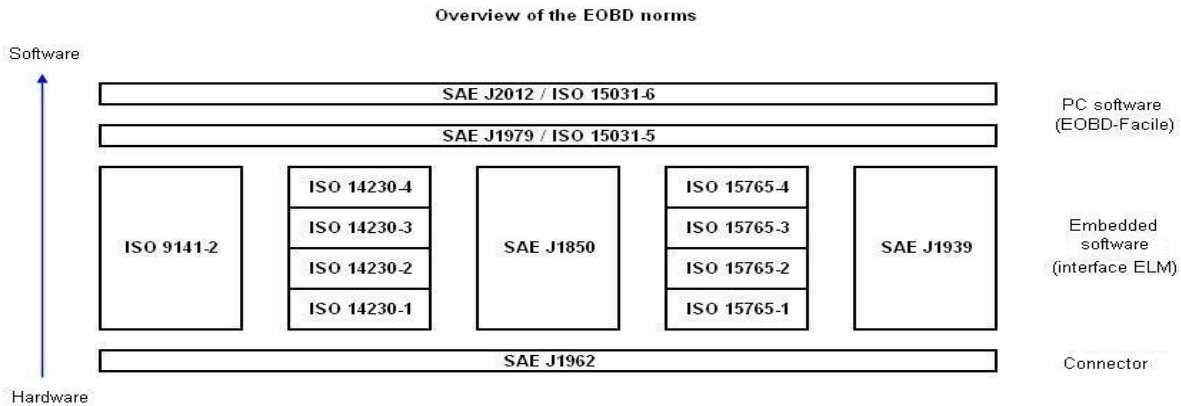


Tabla 2.4.2 Niveles de estándares en una aplicación (fuente: <http://www.cnblogs.com/shangdawei/p/3570546.html>)

2.5 El OBD En breve, incluyendo que son los Modos, los PID y los DTC

Tomado de DEC (2000), López (2008) y González (2008)

El estado de California en los EEUU fue uno de los primeros en esforzarse en limitar por ley las emisiones nocivas del automóvil. La razón fue que en las ciudades grandes como Los Ángeles, por su situación geográfica, los gases de escape no eran alejados por el viento. De modo que quedaban suspendidos sobre la ciudad como una gran cúpula de humo, siendo nocivo para la salud e impedía seriamente la visibilidad. La normativa CARB (California Air Resources Board) sobre gases de escape fue el inicio para que los países más industrializados tomaran medidas parecidas para corregir las emisiones. Para cumplir con estas normativas se obliga a los fabricantes de automóviles que incorporen en la UCE unas funciones de diagnóstico que reconozcan los fallos de sistemas relevantes a los gases de escape de forma adicional a los códigos de avería del propio fabricante. En cada país se fijan valores límite específico de tal forma que si se sobrepasan éstos, se indica el defecto al conductor mediante un testigo de averías (MIL, del término en inglés Malfunction Indicator Lamp o check engine)

Los parámetros principales que dictan como debe estar funcionando el motor, y verifican si todo funciona correctamente son:

- Velocidad
- Carga
- Temperatura del motor
- Consumo de combustible
- Temperatura ambiente
- Caudal de aire
- Emisiones

Para conocerlos, los automóviles actuales, incorporan una gran cantidad de sensores, que permiten conocer a las ECU, cuales son las condiciones externas, y decidir cómo actuar sobre el motor. En caso de que alguno de los parámetros se salga de los rangos marcados, el sistema OBD II, es el encargado de almacenar esta información, y avisar al conductor de que algo sufre un mal funcionamiento, señalizando con un indicador luminoso que es recomendable ir al taller a revisar que error se ha producido.

Cuando se detecta un fallo el testigo de avería MIL se enciende a más tardar en el tercer ciclo de marcha. Si el fallo desaparece (fallo intermitente), éste se conserva en la memoria de averías durante 40 ciclos de marcha. El MIL se apaga de nuevo al tercer ciclo de marcha sin averías. Si el fallo puede ocasionar daños en el catalizador el MIL parpadea.

La lectura de los códigos de avería se realiza a través de un interfaz que se comunica con las unidades de control del vehículo, además de la gestión del motor, (por ejemplo para el ABS, ESR, ESP, airbag, etc.). Estos puertos en serie funcionan con una velocidad de entre 5 baudios y 10 Kbaudios. Con estos interfaces se puede leer-borrar códigos de avería, leer valor de sensores y activar actuadores. Los protocolos utilizados a nivel mundial son:

- ISO 9141-2 (turismos europeos)
- SAE J1850 (turismos americanos)
- ISO 14230-4(KWP2000)(turismos e industriales europeos)
- SAE J1708 (industriales EEUU)
- A partir de 2008 bus CAN (ISO 15765-4).

Los códigos de avería OBD conocidos como DTCs (del inglés Diagnostic Trouble Code) se dividen en cuatro clases:

Modo	Clase
P - Motor.	0 - SAE
C - Chasis	1 - Fabricante del vehículo
B - Carrocería	
U - Comunicaciones	

Tabla 2.5.1 – Modos y DTCs

Así por ejemplo, el código de avería DTC P0003, es un código de avería del motor
La estructura de un código DTC esta estandarizada y responde al esquema siguiente

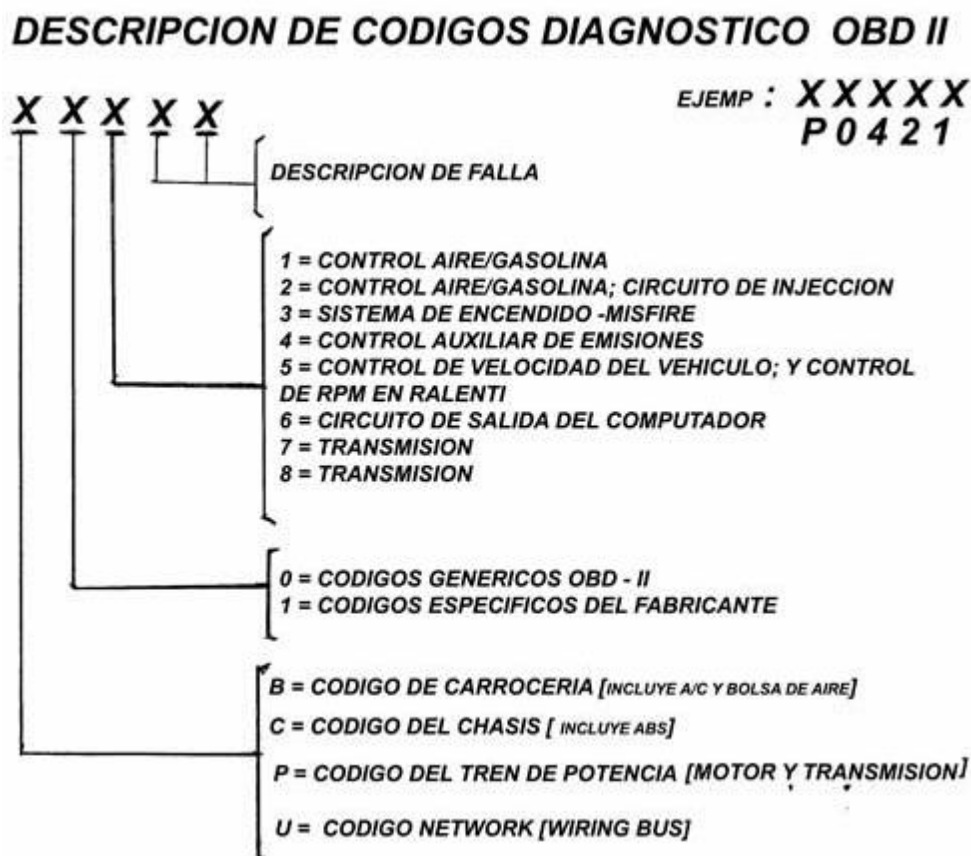


Figura 2.5.1 Estructura código PID OBD (González, 2008)

Los códigos DTC son los más populares códigos de información que el vehículo entrega. El segundo grupo de información que el vehículo es capaz de proveer son los

códigos de información de parámetros (del inglés Parameter IDs o PIDs). Los PIDs entregan información de las condiciones u operación del vehículo, muchos de los cuales son en tiempo real. Por decir, dan la información de la velocidad actual, las revoluciones del motor, temperatura de los diversos sensores, entre otros.

Estos se agrupan en conjuntos por funciones, conocidos como modos OBD. Los modos estándar principales en que se presenta la información son los siguientes:

- **Modo 1.** Identificación de Parámetro (PID, del inglés Process Identification), es el acceso a datos en vivo de valores analógicos o digitales de salidas y entradas a la ECU.
- **Modo 2.** Acceso a Cuadro de Datos Congelados (del inglés Freeze Frame Data). Ésta es una función muy útil del OBD-II porque la ECU toma una muestra de todos los valores relacionados con las emisiones, en el momento exacto de ocurrir un fallo...
- **Modo 3.** Este modo permite extraer de la memoria de la ECU todos los códigos de fallo (del inglés DTC - Data Trouble Code) almacenados.
- **Modo 4.** Con este modo se pueden borrar todos los códigos almacenados en la PCM, incluyendo los DTCs y el cuadro de datos grabados.

La comunicación de comandos OBD se realiza enviando bytes en hexadecimal. El primer byte corresponde al modo OBD en el que se quiere trabajar, el segundo byte corresponde a un parámetro de identificación del modo previsto (PID).

Por ejemplo, Si se quiere preguntar cuál es la temperatura del motor, el primer byte será el modo "01" que muestra flujo de datos y el segundo byte es el PID "05". La comunicación a nivel de bytes se realiza de la siguiente manera:

>01 05 Pregunta

>41 05 7B Respuesta

En este ejemplo, el valor regresado es 0x7B en hexadecimal. En decimal es un valor 123. De acuerdo al SAE J1979, la temperatura en centígrados es el valor menos 40, por lo que la temperatura es 83 grados centígrados.

2.6 Los DTCs a fondo (SAE J2012 / ISO1031-6)

El estándar SAE J2012 (SAE, 2007) se enfoca a estandarizar los Códigos de Diagnostico de Falla (del inglés DTC Diagnostic Trouble Code) que los sistemas OBD de los vehículos son requeridos de reportar cuando hay una falla en el funcionamiento. Este estándar incluye el formato del DTC así como un listado de códigos comunes.

El estándar SAE J2012 fue originalmente desarrollado para cumplir con los requerimientos de OBD de la legislación de los E.U.A. para los vehículos de 1996 en adelante. El estándar ISO 15031-6 se basó en este mismo para los vehículos Europeos a partir del 2000, por lo que ambos estándares son equivalentes.

La siguiente tabla especifica los sistemas, categorías de códigos, valores en hexadecimal y secciones en particular de los sistemas de diagnóstico:

Sistema	Categoría del Código	Valor Hex	Apéndice
Body	B0xxx – B3xxx	8xxx – Bxxx	B0
Chassis	C0xxx – C3xxx	4xxx – 7xxx	C0
Powertrain	P0xxx – P3xxx	0xxx – 3xxx	P0
Network	U0xxx – U3xxx	Cxxx - Fxxx	U0

Tabla 2.6.1 – Especificaciones Generales de Código (fuente: SAE J2012)

Los códigos DTC recomendados consisten de un código hexadecimal de tres dígitos precedidos de un designador alfanumérico. Los designadores alfanuméricos son B0-B3, C0-C3, P0-P3 y U0-U3. Estos corresponden a cuatro sets de códigos de diagnóstico cada uno para los cuatro dominios del automóvil Body, Chassis, tren motriz y dispositivos en red. La estructura de código en si es parcialmente abierta. Una porción de estos códigos está abierta para actualizaciones futuras. Entre estos están porciones de B0, C0, P0, P2, P3, U0 y U3. La mayoría de las fallas en sensores y sistemas están relacionadas en cuatro categorías:

- Circuito abierto o falla general
- Falla de rango o desempeño
- Circuito bajo
- Circuito alto

La gran mayoría de los códigos están ubicados orientados a estas fallas.

Respecto a la estructura del código, el DTC consiste de un designador alfanumérico ej. B0 seguido de un número hexadecimal. La designación del designador alfa debe estar determinada por el área más apropiada para esa función. En la mayoría de los casos el designador alfanumérico está implícito dependiendo del controlador. Los códigos y estructuras están como en la siguiente figura:

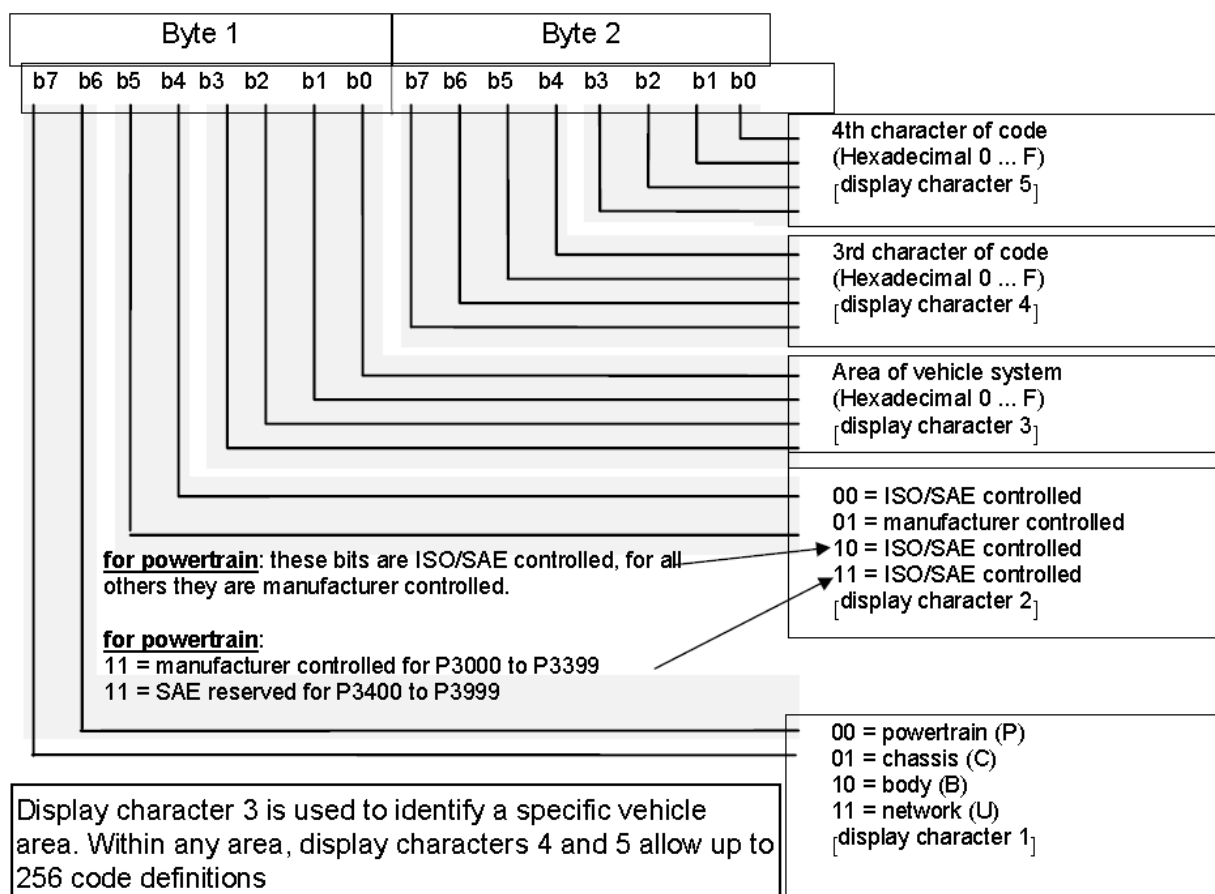


Figura 2.6.1 – Estructura de los DTC (Fuente: SAE J2012)

Ejemplo: el DTC de 2 bytes con valor en el bus de datos \$9234 se despliega a los técnicos como el código DTC B1234 como se muestra en la siguiente figura

DTC HIGH BYTE								DTC LOW BYTE							
\$9				\$2				\$3				\$4			
1	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
B	1			2				3				4			

Figura 2.6.2 – ejemplo DTC 2 bytes (Fuente: SAE J2012)

Ejemplo: el DTC de 3 bytes con valor de bus de datos \$92 34 00 se mostraría a los técnicos como el DTC con código B1234-00, como se muestra en la siguiente figura. El byte menor (LOW Byte) se debe desplegar en hexadecimal, ej. \$1A se debe desplegar como 1^a

DTC HIGH BYTE								DTC MIDDLE BYTE								DTC LOW BYTE							
\$9				\$2				\$3				\$4				\$0				\$0			
1	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	
B		1		2				3		4				0		0							

Figura 2.6.3 – ejemplo DTC 3 bytes (Fuente: SAE J2012)

Este mismo estándar SAE J2012 contempla cuales grupos de DTCs controlar y cuales dejar abiertos a los fabricantes para definirlos de acuerdo a sus necesidades particulares. Sin embargo, establece lineamientos a respetar como lo es mantener consistencia para un mismo fabricante a través de todos sus vehículos, y respetar grupos actuales de datos por decir los códigos 100s y 200s para medición de aire y gasolina y 300s para problemas de ignición.

Para el sistema de Body:

- B0xxx controlados por ISO/SAE
- B1xxx Controlados por cada fabricante
- B2xxx Controlados por cada fabricante
- B3xxx Reservados por el mismo estándar

Para el sistema de Chasis:

- C0xxx controlados por ISO/SAE
- C1xxx Controlados por cada fabricante
- C2xxx Controlados por cada fabricante
- C3xxx Reservados por el mismo estándar

Para el sistema de Tren motriz:

- P0xxx controlados por ISO/SAE
- P1xxx Controlados por cada fabricante
- P2xxx Controlados por ISO/SAE
- P3xxx Controlados por cada fabricante y Reservados por el mismo estándar

Para sistemas en Red:

- U0xxx controlados por ISO/SAE
- U1xxx Controlados por cada fabricante
- U2xxx Controlados por ISO/SAE
- U3xxx Controlados por cada fabricante y Reservados por el mismo estándar

Este estándar en su apéndice A0 establece nomenclaturas guía para los sistemas y nombres comunes. Por ejemplo, para la temperatura del motor la establece por medio del término en inglés Engine Coolant Temperature con siglas ECT

En el apéndice B0 establece códigos DTC estándar para los diferentes dominios. El estándar contempla alrededor de 100 páginas de códigos. Algunos selectos se incluyen como referencia en el anexo 1 de este trabajo

2.7 Los PIDs a fondo (SAE J1979 / ISO15031-5)

El estándar SAE J1979 y su equivalente técnico europeo el ISO15031-5 están enfocados a los requerimientos de reporte de datos de OBD. El J1939 guarda similitudes, pero este último está enfocado a los vehículos de carga pesada. Este estándar está enfocado a ser independiente de región, modelo del vehículo, tipo de motor y de vehículo.

El SAE J1979 especifica:

- Formatos de los mensajes para pedir y recibir mensajes
- Requerimientos de tiempos y secuencia entre requerimientos de mensajes desde equipos externos de prueba y mensajes de respuesta de los vehículos, así como entre estos mensajes y los subsecuentes requerimientos
- Comportamiento entre el vehículo y el equipo externo de prueba si no hay datos disponibles.
- Un conjunto de servicios de diagnóstico, con los correspondientes contenido de mensajes de requerimiento y respuesta, para satisfacer las regulaciones OBD en los centros de verificación.

El sistema está especificado de manera que los mensajes de requerimiento son mensajes funcionales, es decir que el equipo de prueba externo va a solicitar el dato sin saber que ECUs le van a contestar. En algunos vehículos, múltiples ECUs pueden contestar con la información requerida. Cualquier equipo externo debe ser capaz de procesar múltiples respuestas. Todos los sistemas deben ser capaces de procesar y responder al servicio \$01 y PID \$00. Este servicio \$01 y PID \$00 son definidos como el mensaje universal de “ping” el cual en los sistemas que contemplan time out se utiliza para mantener viva la conexión.

Este protocolo maneja una extensa guía de los requerimientos del orden de los mensajes requerida y respuesta, con sus tiempos limites dependiendo del estándar utilizado en capas inferiores así como la condición en la que se encuentra, por decir si no hay respuesta como proceder. La longitud de mensaje máximo es de 7 bytes. Dependiendo del estándar en capas inferiores, esta longitud puede ser fija o variable.

El formato del mensaje para requerir información es como en la tabla siguiente

Data Byte	Parameter Name	Cvt	Hex Value	Mnemonic
#1	Request Service Identifier	M	xx	SIDRQ
#2	service-specific data byte#1	U	xx	—
#3	service-specific data byte#2	U	xx	—
#4	service-specific data byte#3	U	xx	—
#5	service-specific data byte#4	U	xx	—
#6	service-specific data byte#5	U	xx	—
#7	service-specific data byte#6	U	xx	—

Tabla 2.71 – Formato de mensaje de requerimiento de datos (SAE, 2006)

El estándar ISO15765-4 permite algunos bits opcionales extra en el mensaje de requerimiento. Para un mensaje de respuesta positiva, el formato es como en la tabla siguiente

Data Byte	Parameter Name	Cvt	Hex Value	Mnemonic
#1	Positive Response Service Identifier	M	xx	SIDPR
#2	service-specific data byte#1	U	xx	—
#3	service-specific data byte#2	U	xx	—
#4	service-specific data byte#3	U	xx	—
#5	service-specific data byte#4	U	xx	—
#6	service-specific data byte#5	U	xx	—
#7	service-specific data byte#6	U	xx	—

Tabla 2.7.2 – Formato de mensaje de respuesta de datos (SAE, 2006)

Si la respuesta retornada es negativa, el formato es como sigue

Data Byte	Parameter Name	Cvt	Hex Value	Mnemonic
#1	Negative Response Service Identifier	M	7F	SIDNR
#2	Request Service Identifier	M	xx	SIDRQ
#3	ResponseCode	M	xx	RC_

Tabla 2.7.3 – Formato de mensaje de respuesta de datos negativa (SAE, 2006)

Algunos otros mensajes mnemotécnicos como bus ocupado o servicio no soportado son también definidos en este estándar.

Los servicios están definidos del \$01 al \$09, aunque el mismo estándar se reserva del \$01 al \$0F.

La comunicación de comandos OBD se realiza enviando bytes en hexadecimal. El primer byte corresponde al modo OBD en el que se quiere trabajar, el segundo byte corresponde a un parámetro de identificación del modo previsto (PID).

Los modos se son los mencionados anteriormente, por ejemplo 1º Byte 0x01 Muestra flujo de datos en “tiempo real”. El segundo byte, tercer byte, etc., especifican la información que se requiere. Por ejemplo:

Si se quiere preguntar cuál es la temperatura del motor, el primer byte será el modo “01” que muestra flujo de datos y el segundo byte es el PID “05”.

Data	Services	Display Format
Device ID – source address of response	all	ISO 9141-2: Hexadecimal (00 to FF) ISO 14230-4: Hexadecimal (00 to FF) SAE J1850: Hexadecimal (00 to FF) ISO 15765-4: Hexadecimal (11 bit or 29 bit CAN Identifier)
Parameter ID (PID)	\$01 & \$02	Hexadecimal (00 to FF) description (see Annex B)
Frame number	\$02	Decimal (0 to 255)
Data values	\$01 & \$02	See Annex B
Diagnostic trouble codes	\$03 & \$07	“P”, “B”, “C” or “U”, plus 4 hexadecimal characters and/or DTC definition - see ISO 15031-6
Test ID	\$05, \$06 & \$08	Hexadecimal (00 to FF)
Test value and test limits	\$05	Engineering units for Test IDs less than \$80 (see Annex C) - Decimal (0 to 255) for test IDs greater than \$80
Test value and test limits	\$06	Decimal (0 to 65535)
Component ID	\$06	Hexadecimal (00 to 7F)
Optional data bytes	\$08	4 bytes, each decimal (0 to 255) (see Annex F)
Vehicle information type	\$09	Hexadecimal (00 to 7F) (see Annex G)
Vehicle information data	\$09	ASCII for information types \$02 and \$04; Hexadecimal for information type \$06; Decimal for information type \$08 (see Annex G)

Tabla 2.7.4 – Servicios especificados (SAE, 2006)

En el documento SAE J1979 se establecen todas las regulaciones sobre OBD. Así siguiendo el ejemplo, para conocer la temperatura la comunicación a nivel de bytes se realiza de la siguiente manera:

>01 05 Pregunta

>41 05 7B Respuesta

En la respuesta recibida, se suma en hexadecimal 40+01=41, a continuación el PID =05 y después la respuesta recibida será 7B. Este número en decimal es el 123, el cual corresponde en la fórmula de obtención del dato al byte A. Por lo que:

Temperatura motor=A-40=123-40=83°C

No todos los PIDs son soportados por las ECUs por lo que en el modo 01 con PID 00 se informa de qué PIDs son soportados por el vehículo. De esta forma, accediendo a cada PID mediante un puerto COM, se puede acceder a todos los datos soportados por el vehículo.

2.8 Los Modos o Servicios de PID

La información viva de operación que el vehículo es capaz de proveer son los códigos de información de parámetros (del inglés Parameter IDs o PIDs). Los PIDs entregan información de las condiciones u operación del vehículo, muchos de los cuales son en tiempo real. Por decir, dan la información de la velocidad actual, las revoluciones del motor, temperatura de los diversos sensores, entre otros.

Estos se agrupan en conjuntos por funciones, conocidos como servicios o modos OBD. El estándar SAE J1979 y su equivalente técnico europeo el ISO15031-5 definen los primeros 10 modos o servicios de los PIDs, del \$01 al \$0A o 10. Posteriormente surgió el estándar SAE J2190, el cual describe la implementación de modos de prueba extendidos con la intención de suplementar al SAE J1979. En este estándar, se definen modos relacionados con el sistema de emisiones (cubiertos por el J1979) y los extendidos los cuales cubren modos no necesariamente relacionados con control de emisiones. La estructura es similar a la establecida para PIDS y modos por el J1979, solamente extendiéndose al rango completo de direccionamiento de bits.

El J2190 fue cancelado por la SAE en 2008 debido a poco uso. Partes del mismo han sido cubiertos nuevamente por algunos estándares ISO relacionados a OBD.

Este estándar J2190 establecía el siguiente cuadro de mapeo de modos o servicios

Test Mode Value	Message type (bit 6)	Where defined
00 - 0F	Request (bit 6 = 0)	SAE J1979
10 - 1F		SAE J2190
20 - 2F		
30 - 3F		
40 - 4F	Response to Modes \$00-\$3F (bit 6 = 1)	SAE J1979
50 - 5F		SAE J2190
60 - 6F		
70 - 7F		
80 - 8F	Request (bit 6 = 0)	Reserved for future expansion as needed
90 - 9F		
A0 - AF		Defined by vehicle manufacturer
B0 - BF		
C0 - CF	Response to Modes \$80-\$BF (bit 6 = 1)	Reserved for future expansion as needed
D0 - DF		
E0 - EF		Defined by vehicle manufacturer
F0 - FF		

Tabla 2.8.1 – Mapa completo de modos o servicios (Fuente: SAE J2190:1993)

La tabla 2.8.1 indica la asignación de modos de prueba, e indica el tipo de mensaje y donde está definido, ya sea en J1979, este estándar J2190 o algún otro. Los modos \$00-\$0F y \$40-\$4F están reservados por el SAE J1979, el cual solo incluye mensajes de funcionamiento mayormente relacionados a sistemas de control de emisiones. Hay una correspondencia uno a uno entre los mensajes que requieren información y la respuesta a dichos mensajes, donde el bit 6 del modo indica el tipo de mensaje. Todos los mensajes de requerimiento de dato tienen por estándar que tener una respuesta, con excepción del mensaje “Equipo de prueba presente” modo \$3F en el cual la respuesta es opcional. El mismo estándar SAE J1979 reserva los modos \$21 a \$3F y sus respuestas \$61 a \$7F para ser definidos por el SAE J2190 como modos extendidos.

De acuerdo a SAE J1979, los modos o servicios estándar en que se presenta la información son los siguientes:

- **Modo 1.** Identificación de Parámetro (PID, del inglés Process Identification), es el acceso a datos en vivo de valores analógicos o digitales de salidas y entradas a la ECU. Este modo es también llamado flujo de datos. Aquí es posible ver, por ejemplo, la temperatura de motor o el voltaje generado por una sonda lambda.
- **Modo 2.** Acceso a Cuadro de Datos Congelados (del inglés Freeze Frame Data). Ésta es una función muy útil del OBD-II porque la ECU toma una muestra de todos los valores relacionados con las emisiones, en el momento exacto de ocurrir un fallo. De esta manera, al recuperar estos datos, se pueden conocer las condiciones exactas en las que ocurrió dicho fallo. Sólo existe un cuadro de datos que corresponde al primer fallo detectado.
- **Modo 3.** Este modo permite extraer de la memoria de la ECU todos los códigos de fallo (del inglés DTC - Data Trouble Code) almacenados.
- **Modo 4.** Con este modo se pueden borrar todos los códigos almacenados en la PCM, incluyendo los DTCs y el cuadro de datos grabados.
- **Modo 5.** Este modo devuelve los resultados de las pruebas realizadas a los sensores de oxígeno para determinar el funcionamiento de los mismos y la eficiencia del convertidor catalítico
- **Modo 6.** Este modo permite obtener los resultados de todas las pruebas de abordó.
- **Modo 7.** Este modo permite leer de la memoria de la ECU todos los DTCs pendientes.
- **Modo 8.** Este modo permite realizar la prueba de actuadores. Con esta función, el mecánico puede activar y desactivar actuadores como bombas de combustible, válvula de ralentí, etc.
- **Modo 9.** Información del número de serie del vehículo o VIN (del inglés Vehicle Identification Number)

Los modos descritos en el estándar J1979 del \$01 al \$09 son por mucho los más comúnmente utilizados por las herramientas de diagnóstico comerciales, siendo los modos extendidos desconocidos o solamente utilizados por herramientas avanzadas o concesionarias. Existen otros modos del 10 hacia arriba definidos por SAE J2190, los cuales son funciones avanzadas como códigos de falla permanentes, acceso a

funciones internas de la computadora de abordo ECU, entre otros. Estos modos usualmente no están presentes o accesibles desde las herramientas de escáner comerciales, dado que permiten acciones avanzadas como lo es la modificación de parámetros de la computadora para aumentar el rendimiento del motor a costa de reducción de control de emisiones, lo cual en algunas partes es ilegal

2.9 El PID Modo 01- Datos Actuales

El modo 1 contempla acceso a datos actuales del motor (Fuente: Wikipedia, con traducción obtenida de Copro, 2016. Traducción auditada contra los estándares SAE correspondientes)

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
00	4	PID apoyado [01-20]				Bit codificado [A7...D0] == [PID \$01.. PID \$20] Vea a continuación
01	4	Estado del monitor desde DTCs borra. (Incluye estado (MIL) la lámpara indicadora de malfuncionamiento y número de DTC).				Bit codificado. Vea a continuación
02	2	Congelación DTC				
03	2	Estado del sistema de combustible				Bit codificado. Vea a continuación
04	1	Calcula el valor de la carga del motor	0	100	%	$A * 100/255$
05	1	Temperatura del refrigerante del motor	-40	215	° C	$A-40$
06	1	A corto plazo combustible % ajuste — Banco 1	-100 restando combustible (condición Rich)	99.22 agregando el combustible (condición sin grasa)	%	$(A-128) * 100/128$
07	1	Largo plazo combustible % ajuste — Banco 1	-100 restando combustible (condición Rich)	99.22 agregando el combustible (condición sin grasa)	%	$(A-128) * 100/128$

08	1	A corto plazo combustible % ajuste — banco 2	-100 restando combustible (condición Rich)	99.22 agregando el combustible (condición sin grasa)	%	$(A-128) * 100/128$
09	1	Largo plazo combustible % ajuste — banco 2	-100 restando combustible (condición Rich)	99.22 agregando el combustible (condición sin grasa)	%	$(A-128) * 100/128$
0A	1	Presión de combustible	0	765	kPa (calibre)	$A * 3$
0B	1	Presión absoluta del múltiple de admisión	0	255	kPa (absoluta)	A
0C	2	RPM del motor	0	16,383.75	rpm	$((A*256) + B) / 4$
0D	1	Velocidad del vehículo	0	255	km/h	A
0E	1	Avance de sincronización	-64	63.5	° en relación con cilindro #1	$(A-128) / 2$
0F	1	Temperatura del aire de admisión	-40	215	° C	A-40
10	2	Flujo de aire MAF	0	655.35	gramos/seg.	$((A*256) + B) / 100$
11	1	Posición del acelerador	0	100	%	$A * 100/255$
12	1	Estado de aire secundario ordenada				Bit codificado. Vea a continuación
13	1	Sensores de oxígeno presentes				[A0...A3] == Banco 1, sensores 1-4. [A4...A7] == Banco 2...
14	2	Bancada 1, Sensor 1: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)
15	2	Bancada 1, Sensor 2: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)
16	2	Bancada 1, Sensor 3: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	$A/200$ $(B-128) * 100/128$ (si B == \$FF, sensor no se utiliza en calc trim)

17	2	Bancada 1, Sensor 4: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	A/200 (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
18	2	Bancada 2, Sensor 1: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	A/200 (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
19	2	Bancada 2, Sensor 2: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	A/200 (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
1A	2	Bancada 2, Sensor 3: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	A/200 (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
1B	2	Bancada 2, Sensor de 4: Voltaje del sensor de oxígeno, Reajuste de combustible de corto plazo	0 -100(lean)	1,275 99.2(Rich)	Voltios %	A/200 (B-128) * 100/128 (si B == \$FF, sensor no se utiliza en calc trim)
1C	1	Este vehículo se ajusta a las normas OBD				Bit codificado. Vea a continuación
1D	1	Sensores de oxígeno presentes				Similar a PID 13, pero [A0...A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
1E	1	Estado de la entrada auxiliar				A0 == estado poder tomar Off (PTO) (1 == activo) [A1...A7] no se utiliza
1F	2	Tiempo de ejecución desde el motor	0	65.535	segundos	(A * 256) + B
20	4	PID apoyado [21-40]				Bit codificado [A7...D0] == [PID \$21.. PID \$40] Vea a continuación
21	2	Distancia recorrida con lámpara indicadora de malfuncionamiento	0	65.535	km	(A * 256) + B

		(MIL) en				
22	2	Presión de carril de combustible (en relación al vacío del múltiple)	0	5177.265	kPa	$((A*256) + B) * 0.079$
23	2	Combustible presión del carril (diésel o gasolina directo inyectar)	0	655.350	kPa (calibre)	$((A*256) + B) * 10$
24	4	O2S1_WR_lambda(1): Relación de equivalencia Voltaje	0 0	1.999 7.999	N / A V	$((A*256) + B) * 2/65535$ o $((A*256) + B) / 32768$ $((C*256) + D) * 8/65535$ o $((C*256) + D) / 8192$
25	4	O2S2_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
26	4	O2S3_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
27	4	O2S4_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
28	4	O2S5_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
29	4	O2S6_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
2A	4	O2S7_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
2B	4	O2S8_WR_lambda(1): Relación de equivalencia Voltaje	0 0	2 8	N / A V	$((A*256) + B) * 2/65535$ $((C*256) + D) * 8/65535$
2C	1	Ordenado EGR	0	100	%	$A * 100/255$
2D	1	Error EGR	-100	99.22	%	$(A-128) * 100/128$
2E	1	Purga ordenada por evaporación	0	100	%	$A * 100/255$
2F	1	Entrada de nivel de combustible	0	100	%	$A * 100/255$
30	1	# de calentamiento ya que borra códigos	0	255	N / A	A

31	2	Distancia recorrida desde que borra códigos	0	65.535	km	$(A * 256) + B$
32	2	Presión de Vapor de gas sistema	-8,192	8.192	Pa	$((A*256) + B) / 4$ (A y B son complemento a dos firmado)
33	1	Presión barométrica	0	255	kPa (absoluta)	A
34	4	O2S1_WR_lambda(1): Relación de equivalencia Corriente	0 -128	1.999 127.99	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
35	4	O2S2_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
36	4	O2S3_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32768$ $((C*256) + D) / 256-128$
37	4	O2S4_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
38	4	O2S5_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
39	4	O2S6_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
3A	4	O2S7_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
3B	4	O2S8_WR_lambda(1): Relación de equivalencia Corriente	0 -128	2 128	N / A mA	$((A*256) + B) / 32.768$ $((C*256) + D) / 256-128$
3C	2	Temperatura del catalizador Bancada 1, Sensor 1	-40	6,513.5	° C	$((A*256) + B) / 10-40$
3D	2	Temperatura del catalizador Bancada 2, Sensor 1	-40	6,513.5	° C	$((A*256) + B) / 10-40$
3E	2	Temperatura del catalizador Bancada 1, Sensor 2	-40	6,513.5	° C	$((A*256) + B) / 10-40$
3F	2	Temperatura del catalizador Bancada 2, Sensor 2	-40	6,513.5	° C	$((A*256) + B) / 10-40$

40	4	PID apoyado [41-60]				Bit codificado [A7...D0] == [PID \$41... PID \$60] Vea a continuación
41	4	Supervisar el estado de este ciclo de coche				Bit codificado. Vea a continuación
42	2	Voltaje del módulo de control	0	65.535	V	$((A*256) + B) / 1000$
43	2	Valor absoluto de la carga	0	25.700	%	$((A*256) + B) * 100/255$
44	2	Relación de equivalencia de comandos	0	2	N / A	$((A*256) + B) / 32768$
45	1	Posición relativa del acelerador	0	100	%	$A * 100/255$
46	1	Temperatura del aire ambiente	-40	215	° C	A-40
47	1	Posición del acelerador absoluta B	0	100	%	$A * 100/255$
48	1	Posición absoluta del acelerador C	0	100	%	$A * 100/255$
49	1	Posición del pedal acelerador D	0	100	%	$A * 100/255$
4A	1	Posición del pedal del acelerador E	0	100	%	$A * 100/255$
4B	1	Posición del pedal acelerador F	0	100	%	$A * 100/255$
4C	1	Actuador de válvula reguladora ordenada	0	100	%	$A * 100/255$
4D	2	Ejecutan con MIL	0	65.535	minutos	$(A * 256) + B$
4E	2	Tiempo que borra los códigos de problemas	0	65.535	minutos	$(A * 256) + B$
4F	4	Valor máximo para la relación de equivalencia, voltaje del sensor de oxígeno, actual del sensor del oxígeno y la presión absoluta del múltiple de admisión	0, 0, 0, 0	255, 255, 255, 2550	V, mA, kPa	A, B, C, D * 10
50	4	Valor máximo de caudal de masa de aire sensor de flujo de aire	0	2550	g/s	A * 10, B, C y D están reservados para uso futuro
51	1	Tipo de combustible				De tabla de tipo de combustible vea a continuación
52	1	% Etanol combustible	0	100	%	$A * 100/255$

53	2	Sistema Evap absoluto presión de Vapor	0	327.675	kPa	$((A*256) + B) / 200$
54	2	Presión de vapor del sistema EVAP	-32,767	32.768	Pa	$((A*256) + B) - 32767$
55	2	Sensor de oxígeno secundario recortar Banco 1 y 3 a corto plazo	-100	99.22	%	$(A-128) * 100/128$ $(B-128) * 100/128$
56	2	Largo plazo sensor de oxígeno secundario ajuste Banco 1 y 3 del Banco	-100	99.22	%	$(A-128) * 100/128$ $(B-128) * 100/128$
57	2	Sensor de oxígeno secundario recortar banco 2 y 4 a corto plazo	-100	99.22	%	$(A-128) * 100/128$ $(B-128) * 100/128$
58	2	Largo plazo sensor de oxígeno secundario ajuste banco 2 y 4 del Banco	-100	99.22	%	$(A-128) * 100/128$ $(B-128) * 100/128$
59	2	Presión del carril del combustible (absoluta)	0	655.350	kPa	$((A*256) + B) * 10$
5A	1	Posición del pedal acelerador relativa	0	100	%	$A * 100/255$
5B	1	Batería híbrida vida restante	0	100	%	$A * 100/255$
5C	1	Temperatura de aceite del motor	-40	210	° C	$A - 40$
5D	2	Tiempo de inyección de combustible	-210.00	301.992	°	$((A*256) + B) - 26,880 / 128$
5E	2	Tasa de combustible del motor	0	3212.75	L. / h	$((A*256) + B) * 0.05$
5F	1	Requisitos de emisiones que el vehículo está diseñado				Bit codificado
60	4	PID apoyado [61-80]				Bit codificado [A7...D0] == [PID \$61.. PID \$80] Vea a continuación
61	1	Motor de la demanda del conductor - por ciento del esfuerzo de torsión	-125	125	%	A-125
62	1	Motor real - por ciento del esfuerzo de torsión	-125	125	%	A-125
63	2	Par motor referencia	0	65.535	Nm	$A * 256 + B$
64	5	Datos de porcentaje de esfuerzo de torsión del motor	-125	125	%	A-125 inactivo Punto B-125 motor 1 Punto C-125 motor 2

						Punto D-125 motor 3 Punto E-125 motor 4
65	2	Auxiliar de entrada / salida apoyada				Bit codificado
66	5	Sensor de flujo de masa de aire				
67	3	Temperatura del refrigerante del motor				
68	7	Sensor de temperatura de aire de admisión				
69	7	EGR ordenada y Error de EGR				
6A	5	Control de flujo de aire de admisión Diésel ordenada y admisión relativa posición de flujo de aire				
6B	5	Temperatura de recirculación de gases de escape				
6C	5	Control del actuador del acelerador ordenada y la posición relativa del acelerador				
6D	6	Sistema de control de presión de combustible				
6E	5	Sistema de control de presión de inyección				
6F	3	Presión de entrada del turbocompresor				
70	9	Control de refuerzo de presión				
71	5	Control de turbo de geometría variable (VGT)				
72	5	Control de la válvula de derivación				
73	5	Presión de escape				
74	5	Turbocompresor RPM				
75	7	Temperatura del turbocompresor				
76	7	Temperatura del turbocompresor				
77	5	Carga la temperatura del aire del refrigerador (CACT)				
78	9	(EGT) temperatura del Gas de escape Banco 1				Especial PID. Vea a continuación

79	9	(EGT) temperatura del Gas de escape banco 2				Especial PID. Vea a continuación
7A	7	Filtro de partículas diésel (DPF)				
7B	7	Filtro de partículas diésel (DPF)				
7C	9	Diésel Particulate temperatura filtro (DPF)				
7D	1	Estado de zona de control de NOx NTE				
7E	1	Estado de zona de control PM NTE				
7F	13	Motor de tiempo de ejecución				
80	4	PID apoyado [81 - A0]				Bit codificado [A7...D0] == [PID \$81.. PID \$A0] Vea a continuación
81	21	Motor de tiempo de ejecución para auxiliar Device(AECD) de Control de emisiones				
82	21	Motor de tiempo de ejecución para auxiliar Device(AECD) de Control de emisiones				
83	5	Sensor de NOx				
84		Temperatura de superficie múltiple				
85		Sistema reactivo de NOx				
86		Sensor de partículas (PM)				
87		Presión absoluta del múltiple de admisión				
A0	4	PID apoyado [A1 - C0]				Bit codificado [A7...D0] == [PID \$A1..PID \$C0] Vea a continuación
C0	4	PID apoyado [C1 - E0]				Bit codificado [A7...D0] == [PID \$C1..PID \$E0] Vea a continuación
C3	?	?	?	?	?	Devuelve datos numerosos, incluyendo conducir

						condición ID y motor de velocidad *
C4	?	?	?	?	?	B5 es ocioso solicitud de motor B6 es petición parada motor *
PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula^[a]

Tabla 2.9.1 - Modo 1 PIDs

Modo 1 PID 1C - Una solicitud de este PID devuelve un único byte de datos que describe que esta ECU fue diseñada para cumplir con los estándares OBD. A continuación, se muestran los distintos valores que del octeto de datos puede contener junto a lo que significan:

Valor	Descripción
1	OBD-II según lo definido por la CARB
2	OBD según lo definido por la EPA
3	OBD y OBD-II
4	OBD-I
5	No compatibles con OBD
6	EOBD (Europa)
7	EOBD y OBD-II
8	EOBD y OBD
9	EOBD, OBD y OBD II
10	JOBD (Japón)
11	JOBD y OBD II
12	JOBD y EOBD
13	OBD II, EOBD y JOBD
14	Reservados
15	Reservados
16	Reservados
17	Diagnóstico de fabricante del motor (EMD)
18	Diagnóstico de fabricante del motor mejorado (EMD +)
19	Heavy Duty diagnóstico a bordo (niño/parcial) (HD OBD-C)
20	Heavy Duty diagnóstico a bordo (OBD HD)
21	Todo el mundo Armonizado OBD (WWH OBD)
22	Reservados

23	Pesado deber Euro OBD etapa sin NOx control (EOBD HD-yo)
24	Pesado deber Euro OBD etapa con NOx control (HD EOBD-I N)
25	Pesado deber Euro OBD II etapa sin control de NOx (HD EOBD-II)
26	Pesado deber Euro OBD II etapa con control de NOx (HD N EOBD-II)
27	Reservados
28	Brasil OBD fase 1 (OBDBr-1)
29	Brasil OBD fase 2 (OBDBr-2)
30	OBD Coreano (KOBD)
31	India OBD I (IOBD I)
32	India OBD II (IOBD II)
33	Heavy Duty Euro OBD etapa VI (HD EOBD-IV)
34-250	Reservados
251-255	No está disponible para su asignación (SAE J1939 significado especial)

Tabla 2.10.2 - Modo 1 PID 1C estándares soportados

2.10 El PID Modo 02 Diagnostico

Modo 02 acepta los mismos PID como modo de 01, con el mismo significado, pero la información dada es desde cuando se creó el Acceso a Cuadro de Datos Congelados (del inglés Freeze Frame Data). Ésta es una función muy útil del OBD-II porque la ECU toma una muestra de todos los valores relacionados con las emisiones, en el momento exacto de ocurrir un fallo. De esta manera, al recuperar estos datos, se pueden conocer las condiciones exactas en las que ocurrió dicho fallo. Sólo existe un cuadro de datos que corresponde al primer fallo detectado. En este modo se tiene que enviar el número del cuadro de datos congelados en la sección de datos del mensaje motor (Fuente: Wikipedia, con traducción obtenida de Copro, 2016. Traducción auditada contra los estándares SAE correspondientes)

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
02	2	DTC que causó Congelar fotograma que se almacenará.				BCD codificado. Decodificada como en el modo 3

Tabla 2.10.1 - Modo 2 PIDs

2.11 El PID Modo 03- DTC

Este modo permite extraer de la memoria de la ECU todos los códigos de fallo (del inglés DTC - Data Trouble Code) almacenados. (Fuente: Wikipedia, con traducción obtenida de Copro, 2016. Traducción auditada contra los estándares SAE correspondientes)

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
N / A	n * 6	Solicitud de códigos de avería				3 códigos por cuadro de mensaje. Vea a continuación

Tabla 2.11.1 - Modo 3 DTC

Una solicitud de este modo devuelve una lista de los DTC que se han definido. La lista está encapsulada utilizando el ISO 15765-2 Protocolo.

Si hay dos o menos DTC (4 bytes) se volvieron en un marco único de ISO-TP (SF). DTC tres o más en la lista se divulgan en varios marcos, con la cuenta exacta de frames dependen del tipo de comunicación y abordar los detalles.

Cada código de problemas requiere de 2 bytes para describir. La descripción de texto de un código de apuro puede ser decodificada como sigue. El primer carácter del código de problemas está determinado por los dos primeros bits del primer byte:

A7-A6	Primer carácter DTC
00	P -Powertrain
01	C -Chasis
10	B -Cuerpo
11	U -Red

Tabla 2.11.2 – Primer carácter DTC

Los cuatro dígitos siguientes están codificados como 2 bits. El segundo personaje en el DTC es un número definido de la siguiente tabla:

A5-A4	Segundo personaje DTC
00	0
01	1
10	2
11	3

Tabla 2.11.3 – Segundo carácter DTC

El tercer personaje en el DTC es un número definido por:

A3-A0	Tercer personaje DTC
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Tabla 2.11.4 – Tercer carácter DTC

Los cuarto y quinto caracteres se definen de la misma manera como el tercero, pero usando pedacitos B7-B4 y B3-B0. El código resultante de cinco caracteres debería verse como "U0158" y pueden consultarse en una tabla de OBD-II DTC. Hexadecimales (0-9, A-f), mientras que relativamente raro, se permiten caracteres en las últimas 3 posiciones del Código mismo

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
0100		OBD Monitor IDs apoyado (\$01 – \$20)				
0101		O2 Sensor Monitor Banco 1 Sensor 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0102		O2 Sensor Monitor Banco 1 Sensor 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0103		O2 Sensor Monitor Banco 1 Sensor 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0104		Sensor de O2 Sensor Monitor Banco 1 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0105		O2 Sensor Monitor banco 2 Sensor 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0106		O2 Sensor Monitor banco 2 Sensor 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0107		O2 Sensor Monitor banco 2 Sensor 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0108		Sensor de O2 Sensor Monitor banco 2 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0109		Sensor de O2 Sensor Monitor Banco 3 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010A		Sensor de O2 Sensor Monitor Banco 3 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010B		Sensor de O2 Sensor Monitor Banco 3 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010C		Sensor de O2 Sensor Monitor Banco 3 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010D		Sensor de O2 Sensor Monitor Banco 4 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010E		Sensor de O2 Sensor Monitor Banco 4 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010F		Sensor de O2 Sensor Monitor Banco 4 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0110		Sensor de O2 Sensor Monitor Banco 4 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0201		O2 Sensor Monitor Banco 1 Sensor 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0202		O2 Sensor Monitor Banco 1 Sensor 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje

0203		O2 Sensor Monitor Banco 1 Sensor 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0204		Sensor de O2 Sensor Monitor Banco 1 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0205		O2 Sensor Monitor banco 2 Sensor 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0206		O2 Sensor Monitor banco 2 Sensor 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0207		O2 Sensor Monitor banco 2 Sensor 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0208		Sensor de O2 Sensor Monitor banco 2 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0209		Sensor de O2 Sensor Monitor Banco 3 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020A		Sensor de O2 Sensor Monitor Banco 3 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020B		Sensor de O2 Sensor Monitor Banco 3 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020C		Sensor de O2 Sensor Monitor Banco 3 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020D		Sensor de O2 Sensor Monitor Banco 4 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020E		Sensor de O2 Sensor Monitor Banco 4 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020F		Sensor de O2 Sensor Monitor Banco 4 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0210		Sensor de O2 Sensor Monitor Banco 4 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]

2.12 El PID Modo 04- borrado

Con este modo se pueden borrar todos los códigos almacenados en la PCM, incluyendo los DTCs y el cuadro de datos grabados. (Fuente: Wikipedia, con traducción obtenida de Copro, 2016. Traducción auditada contra los estándares SAE correspondientes)

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
N / A	0	Borrar códigos de avería / mal funcionamiento (MIL) Indicador luz del motor del cheque				Borra todos los códigos almacenados y apaga el millón.

Tabla 2.12.1 - Modo 04 – DTC

2.13 El PID Modo 05 - Sensores

Este modo devuelve los resultados de las pruebas realizadas a los sensores de oxígeno para determinar el funcionamiento de los mismos y la eficiencia del convertidor catalítico (Fuente: Wikipedia, con traducción obtenida de Copro, 2016. Traducción auditada contra los estándares SAE correspondientes)

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
0100		OBD Monitor IDs apoyado (\$01 – \$20)				
0101		O2 Sensor Monitor Banco 1 Sensor 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0102		O2 Sensor Monitor Banco 1 Sensor 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0103		O2 Sensor Monitor Banco 1 Sensor 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0104		Sensor de O2 Sensor Monitor Banco 1 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0105		O2 Sensor Monitor banco 2 Sensor 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0106		O2 Sensor Monitor banco 2 Sensor 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0107		O2 Sensor Monitor banco 2 Sensor 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0108		Sensor de O2 Sensor Monitor banco 2 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0109		Sensor de O2 Sensor Monitor Banco 3 1	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010A		Sensor de O2 Sensor Monitor Banco 3 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010B		Sensor de O2 Sensor Monitor Banco 3 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010C		Sensor de O2 Sensor Monitor Banco 3 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010D		Sensor de O2 Sensor	0.00	1,275	Voltios	Rich 0,005 a voltaje del

		Monitor Banco 4 1				umbral del sensor de inclinación
010E		Sensor de O2 Sensor Monitor Banco 4 2	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
010F		Sensor de O2 Sensor Monitor Banco 4 3	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0110		Sensor de O2 Sensor Monitor Banco 4 4	0.00	1,275	Voltios	Rich 0,005 a voltaje del umbral del sensor de inclinación
0201		O2 Sensor Monitor Banco 1 Sensor 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0202		O2 Sensor Monitor Banco 1 Sensor 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0203		O2 Sensor Monitor Banco 1 Sensor 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0204		Sensor de O2 Sensor Monitor Banco 1 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0205		O2 Sensor Monitor banco 2 Sensor 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0206		O2 Sensor Monitor banco 2 Sensor 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0207		O2 Sensor Monitor banco 2 Sensor 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0208		Sensor de O2 Sensor Monitor banco 2 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0209		Sensor de O2 Sensor Monitor Banco 3 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020A		Sensor de O2 Sensor Monitor Banco 3 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020B		Sensor de O2 Sensor Monitor Banco 3 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020C		Sensor de O2 Sensor Monitor Banco 3 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020D		Sensor de O2 Sensor Monitor Banco 4 1	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020E		Sensor de O2 Sensor Monitor Banco 4 2	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
020F		Sensor de O2 Sensor Monitor Banco 4 3	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
0210		Sensor de O2 Sensor Monitor Banco 4 4	0.00	1,275	Voltios	Lean 0.005 a Rich sensor umbral de voltaje
PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula^[a]

Tabla 2.13.1 – Modo 05 Sensores

2.14 El PID Modo 06- Pruebas Abordo

El propósito de este servicio es de permitir acceso a los resultados de pruebas de monitoreo de auto diagnóstico a bordo para sistemas o componentes específicos. Ejemplos de estos son monitoreo del convertidor catalítico y el sistema evaporativo.

El fabricante del vehículo es el responsable de asignar IDs de prueba e IDs a los componentes para las respectivas pruebas. Los últimos resultados validos son retenidos hasta ser reemplazados por otros válidos. Una función opcional es la de indicar los modos disponibles para pruebas.

2.15 El PID Modo 07- DTC

Este modo permite leer de la memoria de la ECU todos los DTCs pendientes, detectados durante el actual o el último ciclo de manejo para sistemas de emisiones. Este servicio \$07 es requerido para todos los DTCs y es independiente del servicio \$03. La intención de este servicio es proveer información al técnico después de una reparación, reportando códigos de falla después de un solo ciclo de manejo. Dado que son códigos que pueden aparecer en un solo ciclo, el indicador MIL puede no encenderse. Este modo tiene la misma estructura y protocolo que el modo \$03

2.16 El PID Modo 08- Prueba de Actuadores

Este modo permite realizar la prueba de actuadores. Con esta función, el mecánico puede activar y desactivar actuadores como bombas de combustible, válvula de ralentí, etc. Como en otros modos, con el PID \$00 el sistema regresa los PIDs soportados

2.17 El PID Modo 09- Información General del vehículo VIN

Información del número de serie del vehículo o VIN (del inglés Vehicle Identification Number) entre otras informaciones (Fuente: Wikipedia, con traducción obtenida de Copro, 2016. Traducción auditada contra los estándares SAE correspondientes)

PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]
00	4	9 el modo compatible PIDs (01 a 20)				Bit codificado. [A7...D0] = [PID \$01.. PID \$20] Vea a continuación
01	1	VIN mensaje Conde en el PID 02. Sólo para ISO 9141-2, ISO 14230-4 y SAE J1850.				Generalmente el valor será 5.
02	17-20	Número de identificación del vehículo (VIN)				17-char VIN, codificación ASCII e izquierda-rellenado con caracteres nulos (0 x 00) si necesario.
03	1	Conde de mensaje de ID de calibración para el PID 04. Sólo para ISO 9141-2, ISO 14230-4 y SAE J1850.				Será un múltiplo de 4 (4 mensajes son necesarios para cada ID).
04	16	ID de calibración				Hasta 16 caracteres ASCII. Bytes de datos no utilizados serán reportados como bytes nulos (0 x 00).
05	1	Número de verificación de calibración (CVN) mensaje cuenta por PID 06. Sólo para ISO 9141-2, ISO 14230-4 y SAE J1850.				
06	4	Número de verificación de calibración (CVN)				Izquierda-rellenado con caracteres nulos (datos en bruto 0 x 00). Generalmente aparece como cadena hexadecimal.
07	1	Conde de mensaje por PID de seguimiento del rendimiento en uso 08 y 0B. Sólo para ISO 9141-2, ISO 14230-4 y SAE J1850.	8	10		8 si dieciséis 16 valores están obligados a informar, 9 18 18 valores están obligados a informar, y 10 si veinte 20 valores deben ser registrados (un mensaje informes dos valores, cada uno que consiste en dos bytes).
08	4	Seguimiento para los vehículos de ignición de chispa del rendimiento en uso				4 o 5 mensajes, cada uno conteniendo 4 bytes (dos valores). Vea a continuación
09	1	ECU nombre mensaje no cuentan para PID 0A				
0A	20	Nombre ECU				Código ASCII. Derecho- rellenado con caracteres nulos (0 x 00).
0B	4	Seguimiento para los vehículos de ignición de compresión del				5 mensajes, cada uno conteniendo 4 bytes (dos valores). Vea a continuación

		rendimiento en uso				
PID (hexadecimal)	Bytes de datos devuelto	Descripción	Valor mín.	Valor máx.	Unidades	Fórmula ^[a]

Tabla 2.17.1 – Modo 09 VIN

2.18 El PID Modo 10 (Modo 0A) – DTC códigos de falla permanentes

El propósito de este servicio es habilitar al equipo de prueba externo obtener todos los DTCs que tengan estatus de permanente. Estos DTCs fueron previamente borrados por una ejecución satisfactoria del modo \$04 de borrado de códigos, pero permanecen en la memoria no volátil del controlador. No pueden borrarse manualmente. Estos son borrados por el controlador hasta que un número apropiado de ciclos de manejo han pasado sin marcar nuevos códigos. Tiene la misma estructura que el modo \$03

2.19 El PID Modo \$11 al \$15 (0x0A al 0x0F)

Estos modos están reservados por el SAE J1979. No se encontró mayor información al respecto de estos modos

2.20 Los PID Modos Extendidos

El estándar SAE J2190 especifica los demás modos de la tabla de posibles servicios, desde el punto de vista de los bits empleados. Estos siguen la misma estructura en cuanto a que el bit 6 establece si es un requerimiento de dato o una respuesta de datos. Por ejemplo, el requerimiento \$01 00 tiene respuesta \$04 00. La tabla de PIDs completos la especifica el estándar:

Mode	Message Name	Mode	Message Name
00		40	
01	Request Current Powertrain Diagnostic Data	41	Report Current Powertrain Diagnostic Data
02	Request Powertrain Freeze Frame Data	42	Report Powertrain Freeze Frame Data
03	Request Powertrain Diagnostic Trouble Codes	43	Report Powertrain Diagnostic Trouble Codes
04	Request to Clear/Reset Diagnostic Trouble Codes	44	Report Diagnostic Trouble Codes Cleared
05	Request Oxygen Sensor Monitoring Test Results	45	Report Oxygen Sensor Test Results
06	Request On-Board Monitoring Test Results	46	Report On-Board Monitoring Test Results
07	Request Pending Powertrain Trouble Codes	47	Report Pending Powertrain Trouble Codes
08		48	
09		49	
0A		4A	
0A		4A	
0B		4B	
0C		4C	
0D		4D	
0E		4E	
0F		4F	
10	Initiate Diagnostic Operation	50	Confirm Diagnostic Operation
11	Request Module Reset	51	Confirm Module Reset
12	Request Diagnostic Freeze Frame Data	52	Report Diagnostic Freeze Frame Data
13	Request Diagnostic Trouble Code Information	53	Report Diagnostic Trouble Code Information
14	Clear Diagnostic Information	54	Confirm Diagnostic Information Cleared
15		55	
16		56	
17	Request Status of Diagnostic Trouble Codes	57	Report Status of Diagnostic Trouble Codes
18	Request Diagnostic Trouble Codes by Status	58	Report Diagnostic Trouble Codes by Status
19		59	
1A		5A	
1B		5B	
1C		5C	
1D		5D	
1E		5E	
1F		5F	
20	Return to Normal Operation	60	Confirm Normal Operation
21	Request Diagnostic Data by Offset	61	Report Diagnostic Data by Offset
22	Request Diagnostic Data by PID	62	Report Diagnostic Data by PID
23	Request Diagnostic Data by Memory Address	63	Report Diagnostic Data by Memory Address
24	Request Scaling and Offset/PID	64	Report Parameter Offset and Scaling/Mask
25	Request to Stop Transmitting Data	65	Confirm Data Stopped
26	Specify Setting of Data Rates	66	Confirm Data Rates Set
27	Data Link Security Access	67	Confirm Security Access

Mode	Message Name	Mode	Message Name
28	Disable Normal Message Transmission	68	Confirm Normal Message Transmission Disabled
29	Enable Normal Message Transmission	69	Confirm Normal Message Transmission Enabled
2A	Request Diagnostic Data Packet(s)	6A	Report Diagnostic Data Packet(s)
2B	Define Diagnostic Data Packet by Offset	6B	Confirm Data Packet Defined by Offset
2C	Define Diagnostic Data Packet	6C	Confirm Data Packet Defined
2D		6D	
2E		6E	
2F	Input/Output Control by PID	6F	Confirm Input/Output Control by PID
30	Input/Output Control by Data Value ID	70	Confirm Input/Output Control by Data Value ID
31	Request Start Diagnostic Routine by Test No.	71	Confirm Diagnostic Routine Started by Test No.
32	Request Stop Diagnostic Routine by Test No.	72	Confirm Diagnostic Routine Stopped by Test No.
33	Request Diagnostic Routine Results by Test No.	73	Report Diagnostic Routine Results by Test No.
34	Request Download	74	Report Download Status
35	Request Upload	75	Report Upload Status
36	Transfer Data	76	Data Transfer
37	Request Stop Data Transfer	77	Confirm Data Transfer Stopped
38	Request Start Diagnostic Routine by Address	78	Confirm Diagnostic Routine Started by Address
39	Request Stop Diagnostic Routine by Address	79	Confirm Diagnostic Routine Stopped by Address
3A	Request Diagnostic Routine Results by Address	7A	Report Diagnostic Routine Results by Address
3B	Write Data Block	7B	Confirm Data Block Written
3C	Read Data Block	7C	Report Contents of Data Block
3D		7D	
3E		7E	
3F	Test Device Present - No Operation Performed	7F	General Response Message (accept/reject)

Tabla 2.20.1 Modos Extendidos (Fuente: SAE J12190, 1993)

Algunos de estos modos se especifican de la siguiente manera:

0x10 – Iniciar operaciones de diagnostico

0x11 – Requerir reset del módulo ECU

0x12 – Requerir datos congelados (del inglés Diagnostic Freeze Frame Data)

0x13 – Requerir información de códigos de diagnóstico DTC

0x14 – Borrar información de diagnostico

0x17 – Requerir el estado de los DTC

0x18- Requerir los códigos DTC por estado

0x20 – Regresar a operaciones normales

0x21 al 0x23 Requerir datos de diagnostico

0x21 – Offset de un byte

0x22 – Leer dato PID, 2 bytes

0x23 – Leer dirección de memoria, 3 bytes

0x24 – Requerir escalamiento y offset de los PIDs en memoria

0x25 – Detener la transmisión de datos
0x26 – Especificar los rates de transmisión
0x27 – Modo de seguridad de acceso
0x28 – Deshabilitar transmisión normal de mensajes
0x29 – Habilitar transmisión normal de mensajes
0x2A – Requerir paquetes de datos de diagnostico
0x2B – Definir dinámicamente los paquetes de datos por offsets de un solo byte
0x2C – Definir dinámicamente el paquete de datos de diagnostico
0x2D – Leer datos por ID
0x2E – Grabar datos por ID
0x2F – Control de Entrada / Salida por PID
0x30 – Control de Entrada / Salida por valor de dato de PID
0x31 al 0x33 – Realizar rutina de diagnóstico por número de prueba
0x34 al 0x37 – Transferencia de datos
0x34 - Request Download
0x35 - Request Upload
0x36 - Transfer Data
0x37 - Request Transfer Exit
0x38 – al 0x3A – Realizar rutina de diagnóstico en la dirección especificada
0x3B – Grabar un bloque de datos
0x3C – Leer un bloque de datos
0x3d - Write Memory By Address
0x3E y 0x3F – Equipo de prueba presente (no realiza operación)

2.21 Los Header bytes

Todo paquete de datos debe contener un inicio del mismo a la hora de su transmisión. Este es conocido como el encabezado, del término en inglés header. Dependiendo del modo de transmisión o protocolo empleado, este puede contener un patrón de bytes fijo que permita reconocer el inicio de una transmisión, un patrón de bytes en transmisiones de un hilo que permita recuperar la señal de reloj y sincronizarse, o en

sistemas de direccionamiento a múltiples componentes el direccionamiento hacia quien se dirige la comunicación. Este último es el caso del sistema OBD.

Hasta este punto, se ha estado manejando solamente los modos, PIDs y sus respuestas. Los códigos de diagnóstico DTC son un caso incluido en el modo 6. Estos casos prácticamente se manejan con un solo formato de header establecido, el cual depende del protocolo físico utilizado. Por decir, para J1979 de 10.4Kbps se utiliza un header 68 6A F1 y para el mismo en 41.6Kbps se utiliza 61 6A F1. Naturalmente, el manejar distintos headers abre aún más las posibilidades de comunicación.

El estándar SAE J1850 tiene dos tipos de paquete de datos, uno si solo se requiere enviar un mensaje y otro si se requiere respuesta a dicho mensaje. Las estructuras de los mismo es como en la siguiente figura

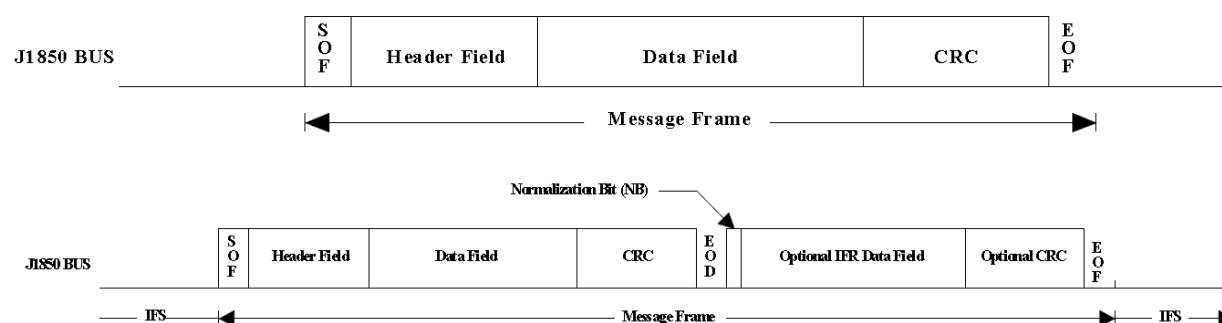


Figura 2.21.1 Estructura del paquete de datos (Fuente: SAE J1850)

La estructura completa de un paquete de datos (del inglés message frame) queda compuesto por diferentes elementos, cada uno con su propia estructura interna y sus reglas. Entre estos elementos se tienen:

- SOF – Inicio del paquete, del inglés Start Of Frame. Dependiendo del protocolo este puede ser un simple bit que jale físicamente la línea a nivel alto, o alguna cadena de bits fácilmente reconocibles
- Header – El primer byte propiamente del paquete de datos es el encabezado o header. Sin embargo este puede ser de uno o tres bytes. La información contenida en este es crucial para dar a entender al sistema que es lo que se espera a continuación. Por decir, cuantos bytes se espera contenga el paquete completo.

- Data Field – Son propiamente los bytes de la información que se pretende transmitir. El SAE J1979 define la estructura de dicha estructura de 7 bytes para los estándares ISO 9141-2, ISO 14230-4, SAE J1850, ISO 15765-4

Data Byte	Parameter Name	Cvt	Hex Value	Mnemonic
#1	Request Service Identifier	M	xx	SIDRQ
#2	service specific data byte#1	U	xx	---
#3	service specific data byte#2	U	xx	---
#4	service specific data byte#3	U	xx	---
#5	service specific data byte#4	U	xx	---
#6	service specific data byte#5	U	xx	---
#7	service specific data byte#6	U	xx	---

Figura 2.21.2 Estructura del cuadro de datos (Fuente: SAE J1979)

- CRC – El código redundante de verificación (del inglés Cyclical Redundancy Check) es quizás una de las facultades más poderosas de los estándares OBD de capas bajas. El mensaje completo se trata como una sola palabra o sucesión de unos y ceros, el cual se divide entre un polinomio especial y se envían los coeficientes en el byte CRC. Al recibir el paquete, se hace la misma operación y se compara con el CRC. Si hay algún error este se detecta inmediatamente.
- EOD – Después del CRC, se envía un fin de datos (del inglés End Of Data). Esta es la liberación oficial del canal de comunicación por lo que la respuesta se puede dar inmediatamente después de este.
- IFR – Si se requirió una respuesta, este es el momento en el cual se emite la respuesta, después del EOD. De esta manera se hace más eficiente la comunicación al compartir el inicio de los datos entre emisor y receptor.

El estándar SAE J2178 en su sección primera expande la manera en que se especifica los encabezados en J1850 y describe dos formas para los mensajes, de un byte y de formato consolidado. El formato consolidado tiene a su vez dos formas, de un byte y de tres bytes. A su vez, también define la estructura general de los mensajes.

La estructura presentada en la figura 2.21.3 muestra la estructura planteada a partir de J1850. Muestra las dos variantes de un byte y de consolidado, el cual en un bit tiene las dos opciones de un byte y de tres.

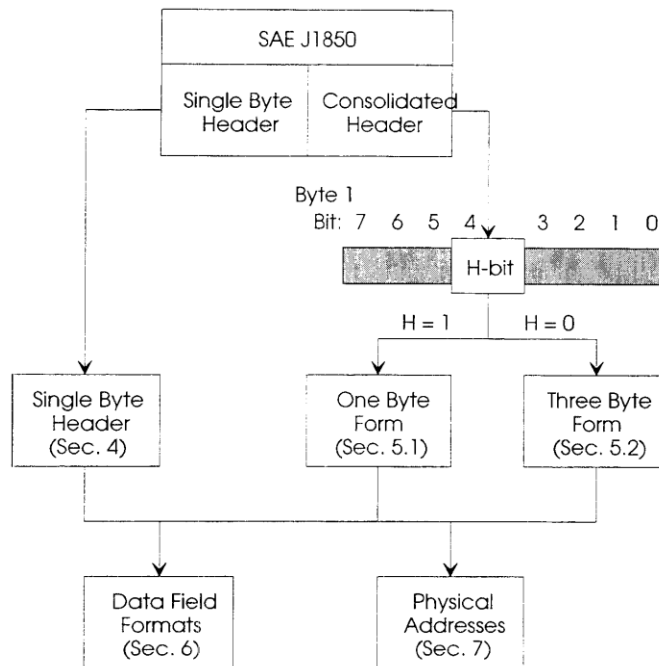


Figura 2.21.3 – Encabezado extendido según SAE J2178

El SAE J1850 define el encabezado de la siguiente manera

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P₂	P₁	P₀	H-bit	K-bit	Y-bit	Z₁	Z₀

Tabla 2.21.1 byte del encabezado (fuente: SAE J1850)

Estos se explican con la siguiente tabla

Bit	Significado	Valor	Significado
PPP	Prioridad del mensaje	000	Alta prioridad
		111	Baja Prioridad
H	Tipo de encabezado	0	De tres bytes
		1	De un byte
K	Respuesta incluida en El paquete (IFR)	0	IFR requerido
		1	IFR no permitido
Y	Modo de direccionamiento	0	Direccionamiento Funcional
		1	Direccionamiento físico
ZZ	Tipo de mensaje específico	00	Estos dependen de los anteriores
		01	K & Y. Se especifican en
		10	SAE J2178-1
		11	

Tabla 2.21.2 byte del encabezado (fuente: SAE J1850)

Los tres primeros bits designan ocho niveles de prioridad y sirven para ayudar a todo el sistema a establecer prioridades en los mensajes cuando los canales están saturados.

El bit H indica si el header va a ser de un byte o de tres. El bit K indica si el mensaje enviado debe contener información de regreso. El bit Y indica el tipo de direccionamiento. Si el bit se hace “1” es direccionamiento físico o dirigido a un destino con una localización física determinada, como lo es dirigirse a un módulo en específico. Si se hace “0” se utiliza direccionamiento funcional, significando que los mensajes están identificados dependiendo del contenido de los mismos o su función como el caso de los PIDs vistos con información de estado de sensores y sistemas del auto. Dado que un equipo externo no sabe físicamente que equipo tiene la información que requiere, por eso se maneja como direccionamiento funcional. Los últimos dos bits ZZ determinan la forma que va a tener el resto del mensaje. Z1 determina si va a haber direccionamiento extendido o con un quinto byte. El Z0 indica si hay datos contenidos en el mensaje. Estos cuatro bytes KYZZ determinan 16 posibles tipos de mensajes que se pueden transmitir con el header de un byte

Msg	Response	Addressing	IFR	
Type	KYZZ	(K bit)	(Y bit)	Type
0	0000	Required	Functional	2
1	0001	Required	Functional	1
2	0010	Required	Functional	2
3	0011	Required	Functional	3
4	0100	Required	Physical	1
5	0101	Required	Physical	—
6	0110	Required	Physical	—
7	0111	Required	Physical	—
8	1000	Not Allowed	Functional	0
9	1001	Not Allowed	Functional	0
10	1010	Not Allowed	Functional	0
11	1011	Not Allowed	Functional	0
12	1100	Not Allowed	Physical	0
13	1101	Not Allowed	Physical	0
14	1110	Not Allowed	Physical	0
15	1111	Not Allowed	Physical	0

NOTE 1—Functional addresses for the three byte form of the consolidated header are defined in SAE J2178-4.

Physical address ranges are defined in Section 7 of this part.

NOTE 2—Message types 8 and 9 use functional data format #2 only; message types 10 and 11 use functional data format #3 only.

NOTE 3—Reserved - SAE indicates reserved for future SAE Committee action. Reserved - MFG indicates that manufacturers are allowed to allocate these definitions without requiring any commonality between motor vehicle manufacturers.

Tabla 2.21.3 Bits KYZZ (Fuente: SAE J2178)

El header de 3 bytes es muy parecido. El bit H se hace “0” para indicar que es un header de 3 bytes. Para los mensajes de diagnóstico de los estándares ISO 9141-2, ISO14230-4 y SAE J1850 se utilizan headers de 3 bytes.

Byte 1	Byte 2	Byte 3
PPPHKYZZ	ID destino	ID fuente

Tabla 2.21.4 Cuadro completo de datos (Fuente: SAE J2178)

El primer byte para ISO 9141-2 y SAE J1850 determina el bit rate y el tipo de mensaje, mientras que para ISO 9141-2 indica el tipo de direccionamiento y la longitud de bytes de datos.

El segundo byte contiene una dirección destino, ya sea funcional o física. El tercer byte contiene la dirección física de quien originó el mensaje. Para equipos de prueba externos esta se asigna como \$F1. Algunos otros equipos pueden utilizar direcciones en el rango \$F0 a FD. La respuesta a todos los mensajes es independiente de las direcciones del equipo de prueba externo que realiza la petición.

El estándar SAE J1979 especifica como reservados los siguientes headers para propósitos solamente de mensajes de diagnostico

Header bytes (Hex)			Data bytes								
Priority/Type	Target address (hex)	Source address (hex)	#1	#2	#3	#4	#5	#6	#7	ERR	RESP
Diagnostic request at 10.4 kbit/s: SAE J1850 and ISO 9141-2											
68	6A	F1	Maximum 7 data bytes							Yes	No
Diagnostic response at 10.4 kbit/s: SAE J1850 and ISO 9141-2											
48	6B	ECU addr	Maximum 7 data bytes							Yes	No
Diagnostic request at 10.4 kbit/s (ISO 14230-4)											
11LL LLLLb	33	F1	Maximum 7 data bytes							Yes	No
Diagnostic response at 10.4 kbit/s (ISO 14230-4)											
10LL LLLLb	F1	addr	Maximum 7 data bytes							Yes	No
Diagnostic request at 41.6 kbit/s (SAE J1850)											
61	6A	F1	Maximum 7 data bytes							Yes	Yes
Diagnostic response at 41.6 kbit/s (SAE J1850)											
41	6B	addr	Maximum 7 data bytes							Yes	Yes

Tabla 2.21.5 Encabezados reservados para diagnostico (fuente: SAE J1979)

CAN-BUS utiliza un formato algo diferente. El estándar ISO 15765-4 identifica cada cuadro de datos por un identificador CAN conocido como CAN ID, que viene siendo el equivalente al header mencionado anteriormente. El tamaño del identificador es de 11 o 29 bits. El identificador debe siempre ir seguido de un cuadro de datos de 8 bytes. Dependiendo del tipo de mensaje, hasta tres bytes son utilizados para control de información de protocolo (del inglés PCI Protocol Control Information), antes del identificador de servicio y los bytes de datos. El total de PCI y datos debe ser siempre de 8 bytes

Header bytes	CAN frame data field							
CAN Identifier (11 or 29 bit)	#1	#2	#3	#4	#5	#6	#7	#8

Tabla 2.21.6 Cuadro de datos CAN-BUS (fuente: ISO 15765-4)

En los bytes de datos, para los estándares no-CAN, el primer byte de datos después del header es el identificador de servicio o modo, y el resto de bytes son dependientes de dicho modo. En CAN-BUS, el primer byte después del identificador es el PCI de diferentes longitudes de bytes, dependiendo del tipo de cuadro. Después sigue el identificador de servicio o modo, siendo el resto de bytes dependiendo del modo requerido.

Como ejemplo, utilizando algún adaptador OBD basado en el IC ELM327 se pueden definir los headers que sean necesarios. Siguiendo J2178 se encuentra que para un protocolo J1850 VPW recomienda una prioridad y tipo de byte \$A8. El mismo estándar indica que el refrigerante del motor es la dirección funcional (no física) \$48. Finalmente como dirección de quien envía se selecciona \$F1, la cual es la indicada para un equipo de prueba. Por lo tanto, se le indica al ELM327 que configure su header utilizando el comando AT SH

➤ AT SH A8 48 F1
OK

Estos bytes permanecen así hasta que se resetee el IC o se seleccione otro header. Por default tiene apagados el eco de los headers, por lo que una conversación para saber la temperatura del refrigerante usando PID 10 sería

```
➤ 10
    10 2E
```

El primer byte hace eco del requerimiento, mientras el segundo es la respuesta requerida. Para hacer direccionamiento físico hacia un ECU determinado, se requiere saber la dirección de dicho ECU. Esto se puede lograr empíricamente capturando paquetes del CAN-BUS del vehículo y observando las direcciones del segundo y tercer byte, mientras se opera repetidamente alguna función de dicho ECU, por decir el radio u otro control. Por decir, si se determina que se quiere accesar al ECU con dirección \$10 se utiliza este mismo valor en el segundo byte del header. Asumiendo por ejemplo un protocolo J1850 PWM, el SAE J2178 recomienda una prioridad de \$E4. Como es recomendable, se asume el rol de equipo de prueba por lo que la dirección de quien envía se establece como \$F1. Por lo tanto, se le indica al ELM327 que configure su header utilizando el comando AT SH

```
➤ AT SH E4 10 F1
    OK
```

A partir de este momento, el ELM327 utiliza este header para todas sus comunicaciones. Usualmente los fabricantes utilizan el modo 22 para sus PIDs propietarios, por lo que siguiendo el ejemplo se intenta el siguiente mensaje en el mismo formato que los PIDs

```
➤ 22 11 6B
    62 11 6B 00 00
```

La respuesta a este comando sigue la estructura del OBD en cuanto a que la respuesta es un eco de la pregunta, más un offset de \$40 en el modo, y los datos de respuesta en este caso 00 00. Siendo estos propietarios de cada marca de vehículo, usualmente el conocer que significan estos supone una gran cantidad de prueba y error con el vehículo.

CAN-BUS, como se mencionaba anteriormente utiliza IDs o headers de 11 o 29 bits. Para configurar por ejemplo un header \$7DF se utiliza la misma instrucción

```
➤ AT SH 7DF
OK
```

El estándar ISO 15765-4 para 11 bits establece para direccionamiento funcional un ID/header de \$7DF, mientras que para direccionamiento físico requiere un ID/Header de \$7E8 para solicitar algún dato a un ECU en específico. El \$7DF se utiliza como default para comunicación, siendo como una llamada general en la cual cualquier ECU puede contestar, y de hecho pueden recibirse múltiples respuestas. Para una llamada con un CAN ID \$7DF se espera recibir respuestas con ID \$7E8 a \$7EF. Los ECUs que pueden responder a llamados de OBD están en escucha de un ID \$7DF y de su propio ID asignado en el rango \$7E0 a \$7E7. Su respuesta tiene un ID que es el suyo ID propio asignado más 8, por lo que la respuesta está en el rango \$7E8 a \$7EF. De esta manera quedan especificados 8 posibles ECUs. Sin embargo, CAN-BUS en general permite cualquier IDs fuera de los especificados. Así por ejemplo Ford utiliza el ID \$720 para su panel de instrumentos, con sus datos propietarios.

Por ejemplo, con el despliegue de todos los datos visibles activado en el ELM327, para un requerimiento general de 01 00, se podría ver

```
➤ 01 00
7E8 06 41 00 BE 3F B8 13 00
```

El 7E8 significa que el ECU #1 contesto la llamada. Por lo tanto para hablarle a este ECU se configura el header al valor 7E0 como establece ISO 15765-4. La manera de comunicarse con este ECU queda

```
➤ AT SH 7E0
    OK
➤ 01 00
7E8 06 41 00 BE 3F B8 13 00
```

De esta manera se establece comunicación directa con una ECU en particular.

Por lo tanto, para establecer un requerimiento a un sistema utilizando el ID \$7DF, el formato a utilizar es el siguiente

Byte								
Tipo PID	0	1	2	3	4	5	6	7
SAE Standard	Numero de bytes adicionales:2	Mode 01 = datos actuales; 02 = freeze frame; etc.	PID code (e.g.: 05 = Engine coolant temperature)	No usado, puede ser 55h				
Propietario por marca	Numero de bytes adicionales:3	Modo propietario Ej 22	PID code (e.g.: 4980h)	No usado, puede ser 00 o 55h				

Tabla 2.21.7 Requerimiento de dato en CAN-BUS

El vehículo responde al requerimiento de PID en el bus CAN con un mensaje de ID que depende de cual módulo respondió. El ECU del motor típicamente responde al ID \$7E8. Otros módulos, como el controlador híbrido o el de la batería en un Toyota Prius responden al \$7E9h, \$7EAh, y \$7EB. Estos son 8h arriba de la dirección física en la cual el módulo responde. Aunque la respuesta puede variar en el largo de sus bytes, el cuadro de datos CAN es siempre de 8 bytes. Estos son

Byte								
PID Type	0	1	2	3	4	5	6	7
SAE Standard 7E8h, 7E9h, 7EAh, etc.	Numero de adicionales bytes: 3 to 6	Modo propietario, igual que la pregunta, solo que se agrega 40h al modo. Ej 41h = datos actuales; 42h = freeze frame; etc.	PID code (e.g.: 05 = Engine coolant temperature)	Valor del parámetro requerido, byte 0	Valor, byte 1 (opcional)	Valor, byte 2 (opcional)	Valor, byte 3 (opcional)	No usado, puede ser 00h o 55h)
Especifico del vehículo 7E8h, o 8h + ID físico del modulo	Numero de adicionales bytes: 4 to 7	Modo propietario, igual que la pregunta, solo que se agrega 40h al modo. Ej 62h = respuesta al modo 22h	PID code (e.g.: 4980h)		Valor del parámetro requerido, byte 0	Valor, byte 1 (opcional)	Valor, byte 2 (opcional)	Valor, byte 3 (opcional)
Especifico del vehículo 7E8h, o 8h + ID físico del módulo.	Numero de adicionales bytes: 3	7Fh Esta es una respuesta general indicando que no se entendió el requerimiento	Custom mode: (e.g.: 22h = enhanced diagnostic data by PID, 21h = enhanced data by offset)	31h	No usado, puede ser 00h			

Tabla 2.21.8 Respuesta en CAN-BUS

El sistema OBD comprende básicamente 5 estándares. Los protocolos utilizados a nivel mundial son:

- ISO 9141-2 (turismos europeos)
- SAE J1850 (turismos americanos)
- ISO 14230-4(KWP2000)(turismos e industriales europeos)
- SAE J1708 (industriales EEUU)
- A partir de 2008 bus CAN (ISO 15765-4).

Los 4 primeros conservan mucha similitud a nivel del header. CAN BUS marca la diferencia extendiendo su capacidad de headers. El estándar SAE J2178 vino a englobar todos estos anteriores y estandarizarlos en un solo documento. Comprende 4 partes, de las cuales la primera sección marca los lineamientos básicos comprendidos en los 5 anteriores estándares. Este estándar SAE J2178-1 (primera sección) se convirtió en la referencia de-facto para especificar las reglas a seguir a la hora de especificar un header. La gran mayoría del uso comercial del OBD está ligado a su origen, el control de emisiones. Por lo tanto, los headers para comunicación con el tren motriz son la configuración por default de toda comunicación OBD con el vehículo.

Así como CAN BUS vino a llevar los 4 anteriores OBD a un siguiente nivel común en el direccionamiento, el J2178 lleva la definición de headers y datos a un siguiente nivel funcional.

La parte 1 además de englobar los 5 estándares y sus headers para sistemas de control de emisiones (motor, transmisión, etc) define lineamientos para especificar headers en sistemas no de control de emisiones. Divide el vehículo en áreas funcionales y físicas, como por decir lado izquierdo de la cajuela o control de luces.

La parte 2 engloba el SAE J1850 en la parte de la estructura del intercambio de datos. Los primeros 256 PIDs definidos en SAE J1979 (un byte) los engloba aquí y los extiende un byte más, renombrándolos como numero de referencia paramétrico (Del inglés PRN, Parameter Reference Number, contrastado con el inglés Parameter ID identification)

La parte 3 y 4 definen la asignación de mensajes para headers de un byte y de tres bytes respectivamente.

Curiosamente este estándar nació en 1992, antes que muchas otras iniciativas como CAN BUS en el 2002 reafirmado como estándar obligatorio en 2008. Su última actualización fue en 1999. Se revisó sin cambios en 2004 y se declaró estable en 2011, significando que no se le ve uso en la industria ni futuros cambios. Permanece activo como referencia a la especificación de headers, más las demás partes especificadas como lo son mensajes no-de emisiones no se les encontraron referencia alguna en otros sitios.

2.22 Protocolos por Fabricante

Hay esencialmente 5 protocolos por fabricante, y normalmente cada fabricante utiliza solo un protocolo. En algunos casos después de la implementación del CAN-BUS en 2008, algunos continúan soportando los protocolos anteriores. Los protocolos principales son:

2.22.1 Protocolo SAE J1850 PWM – Capa Física y Datos en Ford / Mazda

- J1850 PWM (del inglés pulse width modulation)
- Utilizado por Ford Motor Company y Mazda
- Conector: pin 2 bus (-); pin 10 Bus (+)
- El voltaje alto es +5V
- El rate de datos es de 41.6 KBaudios
- Longitud de mensaje restringida a 12 bytes incluyendo CRC
- Utiliza una arbitración multi-maestro CSMA/NDA (del inglés Carrier Sense Multiple Access with Non-Destructive Arbitration)

2.22.2 Protocolo SAE J1850 VPW – Capa F. y D. en GM y camiones ligeros

- J1850 VPW (del inglés variable width modulation)
- Usado por General Motors y en algunos camiones o camionetas de carga ligera
- Conector: pin 2 bus (+); El bus en reposo está en LOW
- El voltaje ALTO es +7V, el punto de decisión es +3.5V
- La longitud del mensaje está restringida a 12 bytes, incluido CRC

- Utiliza una arbitración multi-maestro CSMA/NDA (del inglés Carrier Sense Multiple Access with Non-Destructive Arbitration)
- El rate de datos es de 10.4K o 41.6 KBaudios

2.22.3 Protocolo ISO9141-2 – Capa Física y Datos en Chrysler

- ISO9141-2; similar al RS-232.
- Protocolo anterior de Chrysler, así como de vehículos Europeos y asiáticos entre 2000-2004
- Conector: pin 7 - Línea K; pin 15 – Línea L. Se conoce como “K dual”.
- La línea K en reposo está en ALTO. Voltaje alto es VBateria
- Señalización tipo UART
- Utiliza un rate de datos de 10.4KBauds
- Longitud del mensaje es de 12 bytes máximo, incluyendo CRC

2.22.4 Protocolo ISO14230-1 a -4 – Capa Física y Datos KWP2000

- ISO14230-4 KWP2000 (del inglés keyword protocol 2000)
- Común entre diversos fabricantes Europeos y Asiáticos después del 2003. Alfa Romeo, Audi, BMW, Citroen, Fiat, Honda, Hyundai, Jaguar (X300, XK), Jeep desde 2004, Kia, Land Rover, Mazda, Mercedes, Mitsubishi, Nissan, Peugeot, Renault, Saab, Skoda, Subaru, Toyota, Vauxhall, Volkswagen (VW) desde 2001, Volvo hasta 2004
- Conector: pin 7 - Línea K; pin 15 – Línea L. Se conoce como “K dual”.
- La capa física es idéntica al ISO 9141-2
- Rate de datos entre 1.2 y 10.4 KBauds
- El mensaje puede tener hasta 255 bytes de datos

2.22.5 Protocolo ISO 15765-1 a -4 – CAN-BUS

- ISO 15765-4 CAN-BUS = Introducido en el 2003, siendo mandatorio para todos los vehículos después del 2008. Hay 4 variantes del CAN-BUS con diferentes longitudes de palabra y velocidad:

- ISO 15765-4 CAN (11 bit ID,500 Kbaud)
 - ISO 15765-4 CAN (29 bit ID,500 Kbaud)
 - ISO 15765-4 CAN (11 bit ID,250 Kbaud)
 - ISO 15765-4 CAN (29 bit ID,250 Kbaud)
- Conector: pin 6 CAN High; pin 14 CAN Low

2.23 Protocolo SAE J1962 - Conector

El punto de acceso al vehículo es por medio del conector OBD. Este conector está regulado por medio del estándar SAE J1962. Este estándar es equivalente a su contraparte europea IS/DIS 15031-3. Entre otros puntos, dicho estándar especifica la posición preferente, tamaño y forma del conector, tamaño y fuerza requeridos para cada pin, y asignación de funciones para cada pin.

Respecto a la ubicación, el estándar recomienda que este ubicado del lado del conductor, fije al panel de instrumentos, entre la columna de dirección y el centro del vehículo, y ser de fácil acceso. Usualmente los fabricantes lo ponen sobre la línea de la columna de dirección, en la parte de abajo del tablero.

La forma del conector es del tipo “D”, similar a los utilizados en RS-232 serial y el VGA. Esto facilita la conexión con la polaridad adecuada. Además, el tipo “D” está pensado para poder conectarse con una mano sin necesidad de ver los conectores. El estándar especifica un área libre alrededor del mismo (Access área) para facilitar la conexión de equipos externos. El mismo estándar también especifica la contraparte, la cual es el conector macho del lado del equipo de prueba.

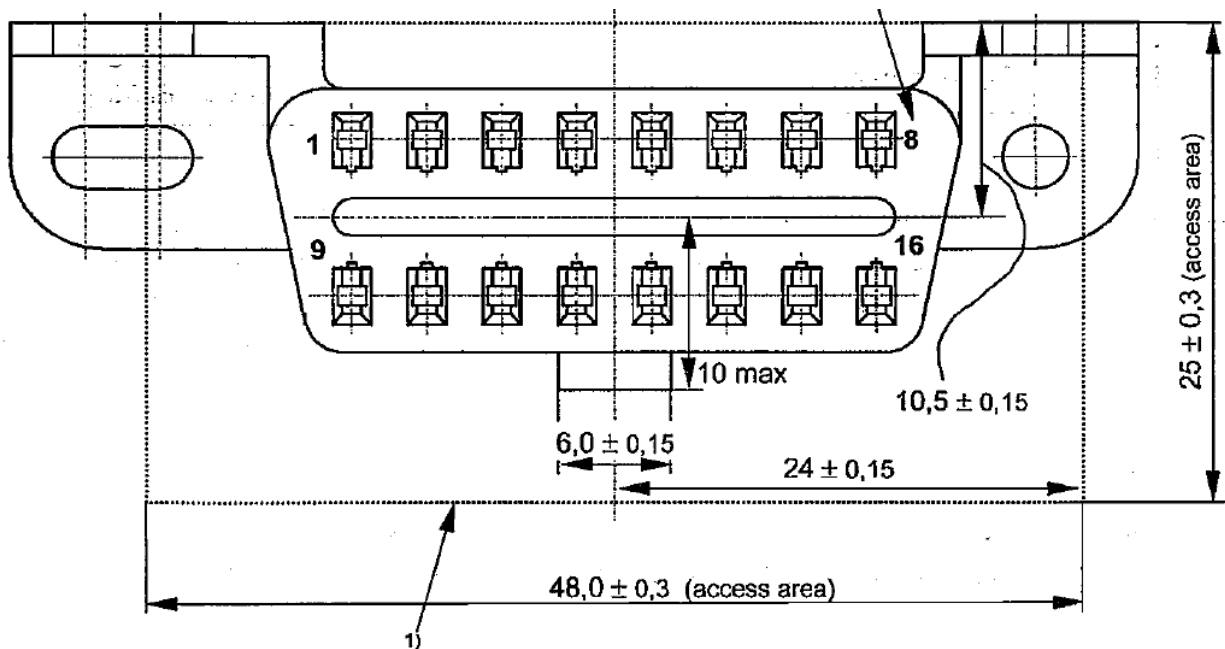


Figura 2.23.1 – Conector OBD, medidas en mm (Fuente: SAE J1962)

La numeración de los pines está dado por la figura. Viendo el conector del vehículo de frente, la numeración va de izquierda a derecha. La fila de arriba va del 1 al 8 y la inferior del 9 al 16

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

Figura 2.23.2 – asignación de números de pines (Fuente: SAE J1962)

Cada línea tiene una asignación descrita por el estándar. Los pines 1, 3, 8, 9, 11, 12 y 13 se dejan a discreción de cada fabricante. Por ejemplo, los pines 3 y 11 son utilizados por Ford y Chrysler para diagnósticos propietarios avanzados. No es requerido que todos los pines estén instalados. Solamente los utilizados pueden instalarse. Ejemplo, si el vehículo es asiático solamente requiere los correspondientes al ISO9141-2. El conector solamente tendría instalados los pines 7 y 15, además de 4,5, y 16 de tierra y voltaje. De hecho, esta es una manera usual de identificar el

estándar soportado por el vehículo, por medio de ver cuales pines tiene instalados. Las asignaciones se sumarizan en la siguiente tabla.

Contacto	Asignación General
1	Asignado por el Fabricate
2	SAE J1850 – Bus positivo
3	Asignado por el Fabricate / Ford DCL – Chrysler CCD
4	Tierra chasis
5	Tierra señal
6	ISO 15765-4 – Línea CAN_H
7	ISO9141-2 / ISO14230-4 – Línea K
8	Asignado por el Fabricate
9	Asignado por el Fabricate
10	SAE J1850 – Bus Negativo
11	Asignado por el Fabricate / Ford DCL – Chrysler CCD
12	Asignado por el Fabricate
13	Asignado por el Fabricate
14	ISO 15765-4 – Línea CAN_L
15	ISO9141-2 / ISO14230-4 – Línea L
16	Voltaje batería positivo permanente

Tabla 2.23.1 – Asignación de pines en el conector OBD (Fuente: SAE J1962)

2.24 El IC ELM327 – Enlace OBD a RS232

Prácticamente todos los automóviles en producción hoy en día tienen por ley una interface para conexión de equipo de diagnóstico. La transferencia de datos de estas interfaces obedece a diversos estándares, pero ninguno de ellos es directamente utilizable por una PC o dispositivo inteligente. El circuito integrado ELM 327 (ELM Electronics, 2014) basado en un PIC de Microchip está diseñado para actuar como un puente entre estos puertos OBD y una interface estándar serial. Además de poder detectar automáticamente e interpretar nueve protocolos OBD, el ELM327 también provee soporte para comunicaciones de alta velocidad, un modo de bajo consumo de energía y el estándar J1939 para vehículos de carga pesada.

La relevancia de este IC para cualquier proyecto que involucre OBD es, como la compañía lo publicita permite delegar al IC toda la comunicación a bajo nivel para el estándar que sea, para poder enfocarse en alto nivel al desarrollo del proyecto.

Si bien está pensado para comunicación RS232, la salida corresponde a UART a la cual se le puede conectar interfaces RS232, USB, Bluetooth o similares. El default es 38400bps, 8 bits, no parity, 1 stop. Si todo es correcto, en reset el IC prende sus 4 leds en secuencia (lamp test) y envía el mensaje

ELM 321 v2.1

> .

El IC responde con su nombre de IC y la versión de Software. El símbolo > es el prompt del mismo. El comando más sencillo para verificar comunicación es enviar un reset por medio del comando AT Z terminado con un retorno de carro 0x0D. El IC contesta con la secuencia descrita anteriormente

Al 2016 la versión más reciente publicada es la 2.1, siendo las anteriores 1.0 a 1.4. En el mercado hay algunos dispositivos que se publican como versión 1.5. Desafortunadamente estas son versiones clonadas con algunos errores. La primera versión 1.0 salió al mercado desprotegida, por lo que en China se extrajo el código de esta primera versión y se comercializa por su cuenta como 1.5.

El IC tiene varios comandos de configuración, accesibles por el prefijo AT. Los más usuales son:

- AT <CR> - Repite el ultimo comando
- AT E0, AT E1 – Eco encendido o apagado
- AT Z – Reset
- AT @1 – Regresa la descripción del componente
- AT @2 – Regresa la identificación del componente
- AT RV – Lee el voltaje del vehículo
- AT DP – Describe el protocolo actual del vehículo
- AT DPH – Describe cual protocolo se está usando de acuerdo a la tabla
- AT SP h – Utilizar el protocolo h de la tabla y grabarlo
- AT SP Ah – Búsqueda automática de protocolo a partir del numero h
- AT TP Ah – intenta comunicarte por protocolo numero h y sigue buscando
- AT D – Regresa el IC a sus valores por default

La lista de números de protocolos soportados (comando h) es la siguiente:

- 0 – automático
- 1 – SAE J1850 PWM (41.6 Kbaud)
- 2 - SAE J1850 VPW (10.4 Kbaud)
- 3 – ISO9141-2 (5 bauds inicio, 10.4 Kbauds)
- 4 - ISO14230-4 KWP (5 bauds inicio, 10.4 Kbauds)
- 5 - ISO14230-4 KWP (inicio rapido, 10.4 Kbauds)
- 6 – ISO15765 -4 CAN (11 bits ID, 500 Kbauds)
- 7 – ISO15765 -4 CAN (29 bits ID, 500 Kbauds)
- 8 – ISO15765 -4 CAN (11 bits ID, 250 Kbauds)
- 9 – ISO15765 -4 CAN (29 bits ID, 250 Kbauds)
- A – SAE J1939 CAN (29 bits ID, 250 Kbauds)
- B – USER1 CAN (11 bits ID, 125 Kbauds)
- C – USER2 CAN (11 bits ID, 50 Kbauds)

El primer protocolo listado (0) es una manera conveniente de indicarle al ELM327 que no se conoce el protocolo del vehículo, por lo tanto debe iniciar una búsqueda.

Para comunicarse por el OBD a través del ELM27 simplemente se le envía en caracteres ASCII por el serial el par hexadecimal correspondiente. Como ejemplo, para conocer las revoluciones del motor, este es el PID 0C del modo 1 por lo que en un hyperterminal conectado se teclearía

>01 0C

Si el motor esta encendido, la respuesta seria similar a

41 0C 1A F8

El valor retornado (1A F8) es en realidad un valor entero hexadecimal de dos bytes. Convertido a decimal es 6904. De acuerdo al estándar, el valor retornado está en cuartos de vuelta, por lo que al dividir $6904/4$ las rpm del motor son 1726. Es de notar que esta comunicación que se acaba de realizar fue completamente independiente del tipo de vehículo y protocolo utilizado.

2.25 Herramientas de diagnóstico populares - Adaptadores

Los adaptadores OBDII comercialmente disponibles son esencialmente puentes de comunicación entre el ECU del vehículo y un dispositivo externo. Para comunicarse con el vehículo utiliza alguno de los 5 protocolos OBDII conectándose al mismo a través del conector OBDII. La comunicación hacia el dispositivo externo la realiza utilizando algún protocolo serial como WiFi, Bluetooth, USB, RS232 o UART.

La gran mayoría de los adaptadores o prácticamente todos utilizan alguna versión del chip ELM 327 (ELM Electronics, 2014) o en menor cantidad el STN1110 de OBD Solutions el cual es equivalente al ELM327. De acuerdo a información de ELM Electronics al 2016 la versión más reciente publicada es la 2.1, siendo las anteriores 1.0 a 1.4. En el mercado hay algunos dispositivos que se publican como versión 1.5. Desafortunadamente estas son versiones clonadas con algunos errores. La primera versión 1.0 salió al mercado desprotegida, por lo que en China se extrajo el código de esta primera versión y se comercializa por su cuenta como 1.5. Para cuestiones

prácticas, la diferencia entre versiones es la cantidad de comandos soportados. La primera versión V1.0 soporta 52 comandos “AT” o comandos especiales como son de configuración. Mientras tanto la versión más reciente V2.1 soporta 126 comandos AT. Todos los adaptadores utilizan la misma estructura de comandos AT V1.0 desarrollada por ELM, por lo que para las aplicaciones es prácticamente transparente cual chip se utilice. Los comandos a los cuales no va a responder son los presentes en versiones superiores a la declarada por el chip. La excepción son los chips declarados como V1.5, los cuales en realidad son V1.0. Estos no responden a comandos como el AT @2 para ver el identificador, el cual es un comando disponible solamente a partir de la versión V1.3.

2.25.1 Interface Genérica ELM327 V1.5 OBDII

Estos adaptadores son los más populares y económicos en venta por internet, estando disponibles en numerosos sitios. Su precio al 2016 anda entre \$2.00 y \$6.00 USD. Algunos sitios su precio se eleva hasta los \$20.00 USD. Los gráficos de la etiqueta tienen ciertas variaciones pero todos conservan la misma similitud y colores azul y amarillo parecidos. El diseño de la caja es prácticamente el mismo, en forma de caja rectangular alargada con los 5 Leds de estatus en hilera por un costado.



Figura 2.25.1.1 – Adaptador OBD Genérico

Características de Fabricante:

Los diversos sitios muestran para el mismo adaptador diversas características. Entre las características en común se muestran las siguientes:

- Disponible mayormente con interface Bluetooth, algunas versiones con USB o WiFi
- Lee códigos de diagnóstico DTC
- Borra códigos o la luz de Check Engine llamada también MIL
- Muestra datos de sensores y parámetros del vehículo como RPM y velocidad. Algunos sitios hacen un listado de parámetros soportados.
- Soporta TODOS los protocolos OBDII. Algunos buscando entre especificaciones en letra chica mencionan en realidad cuales son los que soportan.
- Incluye un CD con programas OBDII para computadora.

Evaluación y Comentarios

Estos adaptadores genéricos son apenas buenos dispositivos de entrada para iniciarse a explorar el mundo del OBDII. Representan una baja inversión inicial, la cual se puede considerar de riesgo dado que por ser de baja calidad hay una probabilidad mayor de falla, con respecto a otros adaptadores de gama más alta.

La primera consideración, siendo tal vez la más relevante es tener cuidado en la leyenda “Soporta TODOS los protocolos OBDII” lo cual es falso. En ocasiones buscando entre especificaciones vienen los nombres de los protocolos que realmente soporta. El adaptador adquirido después de pruebas solo soporta SAEJ1850VPW (GM-Chrysler) e ISO15765 CAN (modelos recientes). La especificación de internet declara soportar también ISO14230-4 (KWP2000) e ISO9141-2 utilizados por Nissan (KWP después del 2002) pero resultó ser falso, no logró conectarse en dos Sentras de diferentes modelos. En el análisis del circuito se observó que carece del hardware necesario para soportar dichos protocolos. Para vehículos a partir del 2008 se utiliza el CAN-Bus, siendo este protocolo el dominante a partir de ese modelo. Por los comentarios de numerosos y diversos foros de entusiastas del OBDII casero, estos adaptadores genéricos son funcionales en modelo 2008 y posteriores, considerando que tengan CAN-BUS.

Respecto a los comandos soportados, solamente responde a los comandos AT declarados por ELM Electronics pertenecientes a la revisión V1.0 a pesar de declarar

tener versión V1.5. No responde a comandos de versiones más recientes. Esto refuerza el comentario de ELM Electronics de los chips clonados.

Estos adaptadores, por su bajo costo son utilizados por entusiastas de proyectos caseros. Hay diversos ejemplos en internet de adaptaciones realizadas a partir de los mismos. Entre las más comunes adaptaciones está el utilizar la salidas UART del IC para conectarse directo con un Arduino o similar. En algunos adaptadores, utilizan un PCB aparte para soldar encima un módulo comercial Bluetooth como el HC-05, o módulos USB o WiFi dependiendo del adaptador final. Conociendo los módulos soldados, es fácil ubicar el puerto UART para comunicarse con el Arduino.

El CD acompañante contiene programas para Windows de OBD. Usualmente son versiones anteriores de programas de acceso libre en internet. Sin embargo, para el principiante la mayor ventaja es el incluir una selección recomendada de entre la gran cantidad de opciones de programas disponibles

La conexión del adaptador con algún dispositivo como un teléfono o computadora es a través de protocolos serial conocidos como bluetooth o USB. El adaptador aparece en las búsquedas como “ELM327 V1.5 device”. Para dicha conexión los pasos son los mismos que en cualquier otro dispositivo sin mayores problemas.

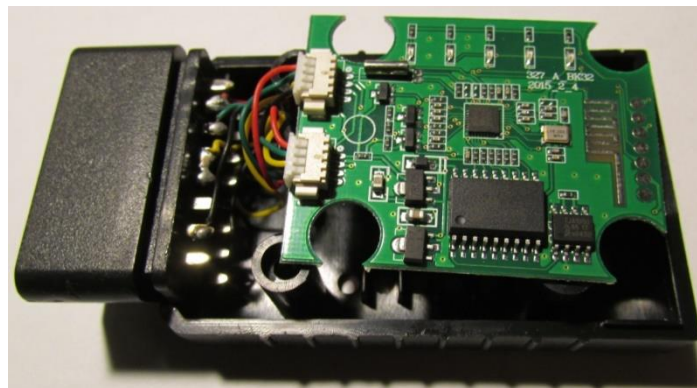


Figura 2.25.1.2 – Adaptador OBD Genérico - Circuito

El circuito interno está basado en un PIC Microchip MCP2515. El Bluetooth está en el tablero con el IC BK3231. El CAN-BUS lo maneja el IC TJA1050. No se le ven componentes en el PCB faltantes o no soldados, por lo que se intuye el soporte para los otros estándares nunca fue contemplado.

Conclusiones

Estos adaptadores genéricos son dispositivos apenas buenos para iniciarse en comunicaciones OBDII (del inglés Entry Level Device). Son de nivel básico, propensos a fallas, limitados en funcionamiento y capacidad, aunque económicos. Un aspecto muy importante a considerar es el protocolo OBDII soportado. En modelos anteriores al 2008 no es garantía de funcionamiento. Para modelos CAN-BUS posteriores al 2008 hay mayor oportunidad de conexión. Con respecto a funcionalidad, las limitaciones de comandos lo restringen a funciones básicas. Sin embargo, para el usuario principiante estas son suficientes para iniciarse en su funcionamiento.

2.25.2 El BAFX Products Bluetooth OBDII Scan Tool

El adaptador BAFX 3415 es perteneciente a la gama media con precios entre \$20 y \$60 USD. Este adaptador se vende en promedio en \$22 en diversos sitios. Su mayor publicidad es la de “realmente” soportar todos los protocolos OBDI. Si bien conserva el diseño de gabinete de los adaptadores genéricos al ser rectangular con los 5 leds de estado a un costado, incluye completo el hardware recomendado por el chip. Esto le permite soportar los diversos protocolos OBD.



Figura 2.25.2.1 – Adaptador OBDII BAFX

Características de Fabricante:

Las características más relevantes publicitadas por el fabricante son:

- Disponible con interface Bluetooth solamente. Recomienda solamente Android o Windows, no compatible con iOS “por el momento”

- Lee códigos de diagnóstico DTC
- Borra códigos o la luz de Check Engine llamada también MIL
- Muestra datos de sensores y parámetros del vehículo como RPM y velocidad en tiempo real. Soporta aplicaciones que grafican los datos. Enlista algunos sensores de ejemplo
- Prepara el vehículo para la prueba/verificación oficial de emisiones. Esto es, soporta aplicaciones que realizan una pre-verificación
- Soporta todos los protocolos OBDII disponibles en USA. Hace publicidad de incluir el hardware que lo soporta a diferencia de otros que no lo incluyen. En el caso de lugares fuera de USA recomienda contactarlos para asegurar la compatibilidad.
- Otorga 60 días para devolución “sin preguntas”, dos años de garantía contra defectos. Pasados los dos años promociona como quiera contactarlos.
- Incluye un CD con programas OBDII para computadora.

Evaluación y Comentarios

Este adaptador de gama media se instaló en diversos vehículos, con lo que se verifico que efectivamente soporta diversos protocolos no soportados por los de gama baja. Se puede considerar como plug and play, dado que solamente se requiere conectar al vehículo y emparejar con un dispositivo bluetooth para que funcione. El dispositivo se encarga de manera automática de seleccionar el protocolo adecuado. Responde a comandos de versiones ELM327 posteriores a la V1.0 aunque al preguntarle por el ID responde como “ELM327 V1.5” de manera similar a los ICs clonados. En funcionalidad, se vuelve prácticamente transparente para la aplicación OBD, haciendo la función aparente como si la aplicación estuviese conectada directa al vehículo.

El consumo de corriente es bajo, relativo a la capacidad de batería de un vehículo. Este puede quedarse conectado en el vehículo permanentemente sin problemas.

Entre las desventajas, está el costo moderado y el tamaño físico del dispositivo. Por reglamentación ISO y SAE, el conector OBDII del vehículo debe estar ubicado en la parte media del área del conductor, por debajo de la columna de la dirección. En algunos vehículos esto hace que el adaptador quede enfrente del pedal de freno.



Figura 2.25.2.2 – Adaptador OBDII BAFX - circuito

El circuito interno está basado en in PIC Microchip PIC18F2480. El Bluetooth esta manejado a través de un plug-in soldado estilo HC-05. Tiene el mismo gabinete de plástico que un adaptador genérico. Sin embargo el PCB está notoriamente más populado que su contraparte genérica, respaldando la afirmación que soporta todos los protocolos OBDII

Conclusiones

Este adaptador es una opción recomendable para usuarios que deseen hacer uso continuo de su sistema OBD. La inversión modesta que representa queda a bien justificada de esta manera. Es un dispositivo sencillo de uso, estilo plug and play. Al soportar todos los estándares el usuario no tiene que preocuparse por compatibilidad de marcas. El dispositivo al conectarse escanea de manera automática los protocolos OBDII y selecciona el utilizado por el vehículo conectado de manera transparente para el usuario. La conexión serial bluetooth utilizada es prácticamente universal por lo que tampoco representa problemas de emparejamiento. Aparece solamente como "OBDII" para emparejamiento. Tiene un mayor soporte de comandos por lo que su funcionalidad permisible es mayor.

2.25.3 SparkFun OBD-II Diagnostics Kit

El tablero Sparkfun OBDII-UART es un PCB orientado específico al mercado de los desarrolladores de sistemas embebidos como el Arduino o Raspberry. Maneja los 5 protocolos OBDII y cuenta con una salida UART. Tiene puertos y conectores específicos para facilitar su conexión con un arduino. Se alimenta directo del voltaje del vehículo a través del conector OBD. Se vende en diversos sitios a un precio de \$50USD. El mismo se vende también en kit junto con un cable OBDII-DB9 a la medida del tablero por un precio de \$60USD.

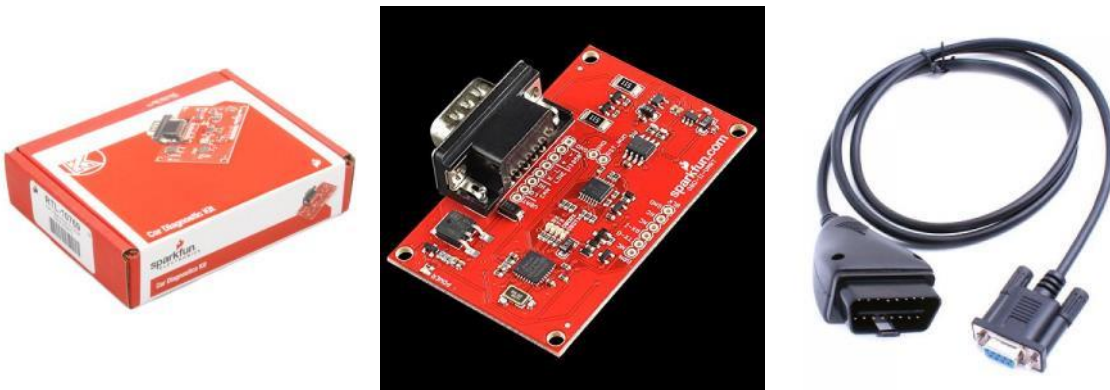


Figura 2.25.3.1 – Tablero SparkFun OBDII

Características de Fabricante:

Las características más relevantes publicitadas por el fabricante son:

- Disponible con interface UART solamente para conexión con sistemas embebidos
- Compatible con los comandos “AT” del ELM327 y los comandos extendidos ST
- Soporta todos los protocolos OBDII
- No incluye CD. Sin embargo en su página oficial incluye enlaces a esquemáticos, hojas de datos y código ejemplo para Arduino. <https://www.sparkfun.com/products/9555>
- Se encuentra también disponible en kit junto con un cable dedicado OBDII – DB9 para conexión directa

Evaluación y Comentarios

Este tablero, junto al Freemantle mencionado en el siguiente punto son los únicos encontrados disponibles para desarrolladores de sistemas OBDII. La orientación del mismo es precisamente para el mercado de personas que desee desarrollar un sistema por su cuenta. Cuenta con interface UART para conectarse directamente con sistemas embebidos como el Arduino. Como otros tableros de desarrollo o shields, es un tablero abierto sin gabinete. Esto lo hace propenso a accidentes si no se maneja con cuidado. Cuenta con un conector DB9 con pinout propietario para conectarse con el OBDII. Aparte de este, no cuenta con otros conectores por lo que requiere soldarle ya sea cables, pines o algún conector. El PCB cuenta con pin holes dedicados para su interface. Se alimenta con el voltaje de la batería del vehículo, cuenta con un regulador 5V onboard. Sin embargo, no provee alimentación o salida alguna para el sistema embebido a conectar por lo que se le debe proporcionar el suyo propio. El tablero está basado en un STN110, el equivalente al ELM327. Es un tablero abierto sin gabinete.

Es un tablero relativamente sencillo de funciones. Viene por default configurado para buscar automáticamente el estándar OBD adecuado. Durante la evaluación, el problema encontrado fue en la interface UART. De fábrica está configurado en 9600bps, una velocidad no estándar para interfaces USB o bluetooth por lo que hubo que hacer adecuaciones y reconfiguraciones. Una vez lograda la comunicación, no se comportó muy estable. Constantemente marcaba caracteres inválidos, aunque se le puede dar beneficio de la duda de haber sido un agente externo no identificado durante la evaluación.

Una vez lograda la comunicación a través de un arduino, el tablero permite la comunicación directa con el ECU del vehículo de manera transparente de manera similar a los otros adaptadores OBDII comerciales. No provee mayor funcionalidad, la misma queda en manos del desarrollador de la aplicación.

Conclusiones

Este tablero es la opción común encontrada entre entusiastas desarrolladores de sistemas OBD. Provee una conexión entre cualquiera de los 5 sistemas OBDII y una

interface UART. La interface UART permite la conexión directa con sistemas embebidos como lo es un Arduino o Raspberry.

Entre las desventajas está el ser un tablero abierto, propenso a accidentes. Cuenta con un conector OBDII a través de un DB9 con pinout propietario haciendo “recomendable” el comprar el cable dedicado. No cuenta con otros conectores por lo que requiere soldarle interfaces al gusto del cliente. No proporciona mayores funcionalidad que ser una interface OBDII-UART, por lo que pare el precio ofertado \$50USD comparado con otros adaptadores es un precio elevado. En diversos foros de entusiastas hay opciones desarrolladas con otros adaptadores comerciales más económicos, los cuales en algunos incluso se conectan internamente directo al UART del adaptador.

2.25.4 Freematics OBD-II Telematics DIY Kit

Este kit desarrollado por Freematics de Australia es un paquete completo de solución de desarrollo para un sistema de monitoreo de OBDII. Cuenta con Arduino y pantalla así como un adaptador OBDII y sensor GPS. Viene pre-programado por lo que puede instalarse directo saliendo de la caja. En pantalla muestra los parámetros más relevantes a partir de los sensores. El precio dependiendo de las opciones y el sitio de venta es entre \$80USD a \$185USD

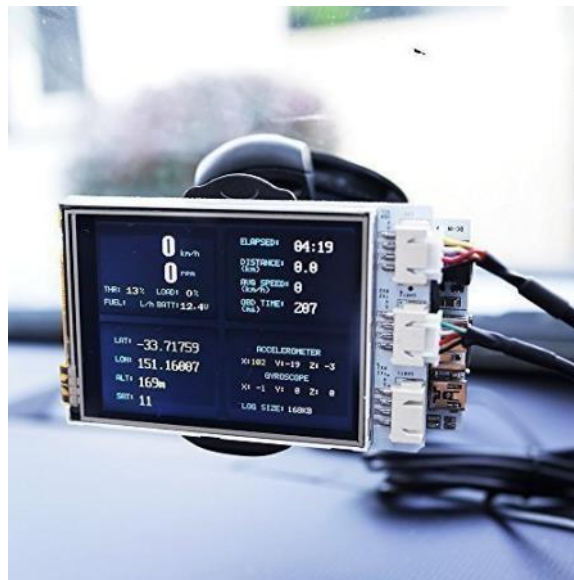


Figura 2.25.4.1 – Tablero SparkFun OBDII

Características de Fabricante:

Las características más relevantes publicitadas por el fabricante son:

- Incluye:
 - un Arduino MEGA 2560,
 - pantalla LCD 3.2" touchscreen con microSD, Bluetooth, puertos I/O, conectores, montable directo en el arduino MEGA
 - Adaptador OBD-II UART. Disponible versión I2C con sensores MEMS de 6-axis.
 - Sensor GPS M8030 GLONASS
 - Soporte estilo teléfono para todo el sistema
- Incluye extensas librerías de código abierto de desarrollo disponibles en GitHub, actualizadas periódicamente.
- El kit basado en Arduino es ideal para construir un sistema de telemetría del vehículo hecho a medida con registro de datos de OBD y GPS. Con display de datos en vivo y conectividad Bluetooth. Todo es alimentado por el mismo puerto OBDII. Las librerías arduino constantemente actualizadas lo hacen un kit perfecto para hacer un sistema por cuenta de uno.
- El adaptador es puente OBDII-UART con librerías open source disponibles. La versión I2C incluye sensores de 6 o 9 ejes MEMS y un voltímetro para leer el voltaje del vehículo. Provee voltaje al sistema embebido (Arduino). Soporta solo CAN-BUS y KWP2000. Compatible con comandos ELM327 AT.
- Cuenta con un shield microSD en el LCD para hacer registro de datos paramétricos y de posición. Estos pueden cargarse en Google Maps y mostrar el recorrido realizado.

Evaluación y Comentarios

Este sistema es para desarrolladores medio a avanzados por la relativa complejidad del hardware y lo completo del código y el sistema en sí. Sin embargo, el kit tal cual saliendo de la caja se puede conectar y funciona. Viene pre-programado con el código Megalogger disponible en GitHub. Este despliega algunos parámetros como lo es velocidad, revoluciones, carga del motor, posición GPS, tiempo y distancia recorrida. El

soporte incluido es del tamaño justo del Arduino Mega por lo que lo vuelve en un sistema listo para instalación. Solo se arma el arduino con la pantalla, se conecta el GPS y el OBD-UART, se coloca el tablero en el soporte y está listo para funcionar.

Las librerías disponibles son bastante extensas, con diversos comentarios. Se actualizan periódicamente. Por lo extenso de las mismas, requiere paciencia y conocimientos de programación para poder darles seguimiento y adaptarlos a las propias necesidades. Una vez comprendido lo básico, es relativamente sencillo ir haciendo modificaciones al programa.

Entre las desventajas está el adaptador incluido de solo soportar CAN-BUS y KWP2000 (modelos asiáticos recientes). Sin embargo, se puede conectar otra interface al mismo puerto UART. Otra desventaja está en el alto costo, sin embargo considerando todo el hardware que se incluye y las librerías ya desarrolladas el precio queda razonable.

El tablero UART-OBDII está basado en un uP STM32F. La conexión CAN-BUS está a través de un IC MCP2551. El modelo UART adquirido tiene provisión para conexión I2C y módulos de señores axiales como el MCP9050

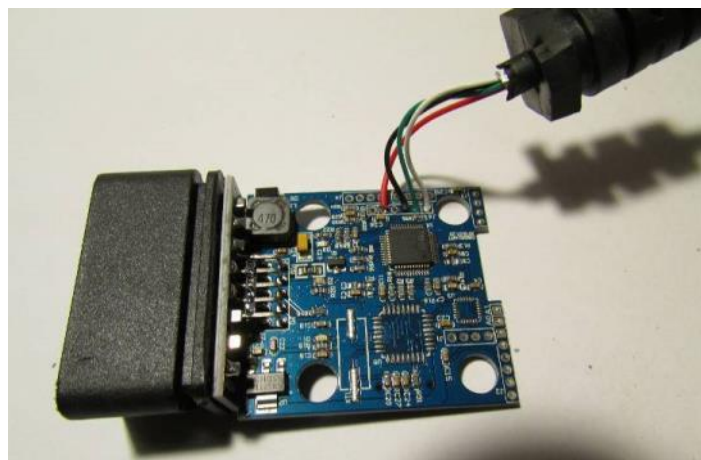


Figura 2.25.4.2 – Tablero SparkFun OBDII - Circuito

Para poder utilizarlo con otro protocolo que no son los soportados por el adaptador, se hizo la modificación de un adaptador genérico, tal como se mencionó anteriormente. Se retiró el modulo Bluetooth. De este módulo, se conectaron cables en el Tx/Rx y estos se utilizaron como puertos UART.

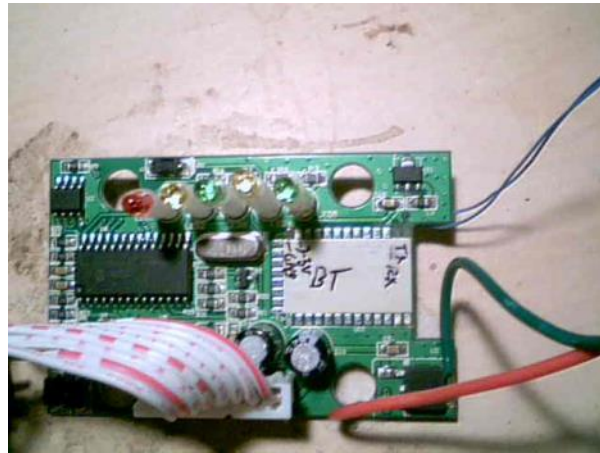


Figura 2.25.4.3 – Adaptación UART

Respecto al código, el software Megalogger está basado en una librería OBD para todas las aplicaciones Freemantle. El adaptador recibe todos los comandos (hasta 8) simultáneamente. Después, con el comando “AT” regresa todos los valores. Esto no es soportado por un adaptador ELM327 normal

Para modificar el código, en la librería OBD.cpp se fuerza a leer uno por uno realizando la siguiente modificación:

```
byte COBD::readPID(const byte pid[], byte count, int result[])
{
    // send a multiple query command
    char buffer[128];
    char *p = buffer;
    byte results = 0;
    /*    Codigo original, comentado para removerlo
    for (byte n = 0; n < count; n++) {
        p += sprintf(p, "%02X%02X\r", dataMode, pid[n]);
    }

    write(buffer);
    // receive and parse the response
    for (byte n = 0; n < count; n++) {
        byte curpid = pid[n];
        if (getResult(curpid, result[n]))
            results++;
    }
    */
}
```

```

// codigo agregado para forzar con el FOR a leer uno por uno
DEBUG.println("READPID \n\r");
for (byte n = 0; n < count; n++) {
    *p = sprintf(p, "%03X%02X\r", dataMode, pid[n]);
    DEBUG.println(buffer);
    write(buffer);
    byte curpid = pid[n];
    if (getResult(curpid, result[n]))
        DEBUG.println(result[n]);
    results++;
}
return results;
}

```

Conclusiones

Este kit de desarrollo es sin duda una muy buena alternativa para desarrolladores de sistemas OBDII que deseen hacer sistemas más avanzados. Lo más valioso además del hardware incluido son las librerías. Las mismas son bastante extensas y con diversos comentarios incluidos lo que las hacen entendibles y relativamente fáciles de modificar al gusto y necesidades del desarrollador. Una desventaja está en el alto costo, sin embargo considerando todo el hardware que se incluye y las librerías ya desarrolladas el precio queda razonable. Otra desventaja relevante a considerar son los protocolos soportados por el adaptador incluido. Solo soporta CAN-BUS (posteriores a 2008) y KWP2000 (asiáticos posteriores a 2002).

2.25.5 Innova 3100 Car Scanner

De acuerdo a su fabricante, la computadora de taller INNOVA 3100 ha sido la herramienta de diagnóstico más vendida, con un millón de unidades vendidas al 2011 desde su introducción en 2001. Sigue en el mercado en venta en diversos lugares con precio en el rango \$100 a \$140USD. Es una herramienta amigable para rápidamente diagnosticar problemas en los vehículos.

Los reparadores profesionales podrán reconocer sus muchas ventajas, mientras que los usuarios domésticos lo encontrarán fácil de usar.



Figura 2.25.5.1 – Innova 3100 Car Scanner

Características de Fabricante:

Las características más relevantes publicitadas por el fabricante son:

- Funciona con cualquier vehículo, camión ligero, SUV o híbrido fabricado posterior a 1996 (OBDII)
- Pantalla patentada que muestra 20 funciones de información simultaneas, incluyendo códigos de falla DTC
- Menú seleccionable inglés, francés, español
- Leds rojo, amarillo y verde indican si el vehículo está listo para prueba de emisiones.
- Datos de cuadro fijo indican datos del motor al momento de la última falla.
- Por medio de conexión USB a computadora, se conecta a internet y despliega en computadora soluciones comunes al problema registrado.

Evaluación y Comentarios

La mayor ventaja de este sistema es ser stand-alone o de ser una unidad sola contenida. No requiere otro tipo de conexiones ni interfaces como lo es un teléfono externo. Está contenido en una carcasa robusta hecho para manejo rudo dentro de un taller mecánico. Como el fabricante lo indica, está diseñado pensando en el taller de gama media que desea un diagnostico por computadora rápido de la falla del “check engine” entre otros datos. Además está pensado para el consumidor casero que desea una herramienta fácil de diagnóstico para su vehículo. Solamente se conecta con el motor encendido y automáticamente gestiona comunicaciones y descarga de datos. En un momento despliega en pantalla el estatus de fallas (si hay) y condiciones de los sensores más comunes. La navegación por las teclas es intuitiva. Puede borrar códigos, con lo que se apaga el “check engine”. Es una herramienta útil para quien desea enfocarse al diagnóstico solamente sin involucrar circuitos, aplicaciones externas o conexiones bluetooth

La desventaja, además del costo es el ser una herramienta solo para diagnostico por medio de los DTC. No tiene datos en tiempo real ni maneja parámetros. No tiene conexión de los mismos a una computadora por lo que no hay manera de hacer interface directo con el ECU. Hay otras soluciones que permiten la conexión con el ECU o datos en tiempo real por medio de los cuales se puede ver por decir en grafica tiempo real el desbalance entre cilindros. Dichas soluciones van en costos por arriba de los \$250USD. El circuito es completamente propietario. El procesador principal es un Atmel AT91SAM7X256

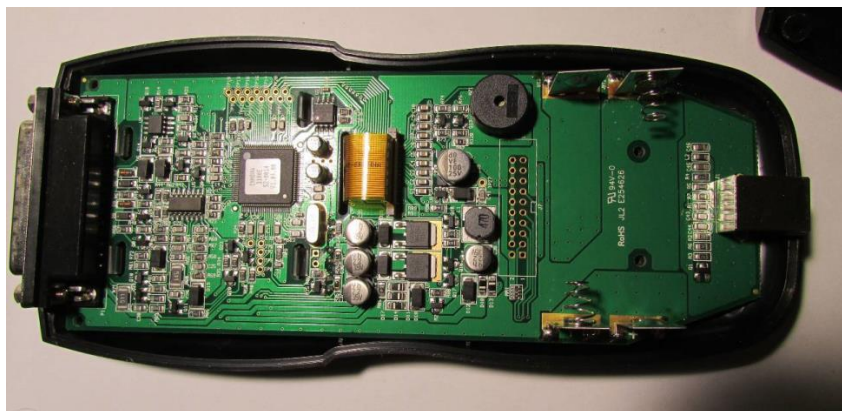


Figura 2.25.5.2 – INNOVA3100 - Circuito

Conclusiones

La mayor ventaja de este sistema es ser stand-alone o de ser una unidad sola contenida. No requiere otro tipo de conexiones ni interfaces como lo es un teléfono externo. Su desventaja precisamente, además del costo es el ser una herramienta solo para diagnóstico por medio de los DTC. No tiene datos en tiempo real ni maneja parámetros. No tiene conexión de los mismos a una computadora por lo que no hay manera de hacer interface directo con el ECU. Sin embargo, sigue siendo una herramienta muy sencilla, robusta y recomendable para el taller de gama media o usuario casero que desee hacer diagnósticos por computadora de códigos de falla sin tener que lidiar con circuitos o aplicaciones en el teléfono.

2.26 Tendencias a futuro en la tecnología de redes en vehículos

2.26.1 eCall El taller del futuro: comunicación directa con el auto

Tomado de Warnecke, A. (2016). Dpa News

“Hola, taller, soy yo, el auto”. La frase, que parece salida de una película de ciencia ficción de los años 70, es casi una realidad. La comunicación entre el auto y el concesionario es cada vez mayor. Los datos definen cada vez más el día a día en los talleres, donde los códigos de error reemplazan al medidor de aceite.

Las tarjetas SIM pueden transmitir distinto tipo de informaciones, desde datos técnicos hasta la cantidad de pasajeros de un auto, que en un accidente puede ser registrada a través de los sensores de los airbags.

De todas formas, la transmisión automática de datos sólo se produce con la función de llamada de emergencia, la eCall. “Para los demás usos el dueño del coche debe autorizar primero que los datos de su auto puedan enviarse o recibirse a través de la tarjeta SIM”, explica Gollwitzer. Si el conductor está de acuerdo, puede generarse un intercambio activo entre el auto por un lado y la automotriz y el taller por el otro. BMW, por ejemplo, envía actualizaciones para su sistema de navegación a través de las direcciones SIM. Por el otro lado, los talleres pueden recibir códigos de error de los autos. “Si se priorizan determinados códigos de error, el vehículo los comunica

automáticamente a la empresa de servicios, que puede contactar a los clientes para fijar una cita”, relata Gollwitzer. “El auto prácticamente habla directo con el experto”.



Figura 2.26.1 - Botón eCall de BMW (Warnecke, 2016)

A partir de esto, cuando llega el día de la cita el taller ya cuenta con los repuestos que se puedan necesitar. “Todo el funcionamiento del taller puede planificarse mejor y el cliente recibe una estimación de costos precisa cuando arregla la cita por teléfono”, dice Joseph Schlobmacher, de Audi. “El trabajo en sí en el vehículo no cambia demasiado, mientras que las herramientas y los aparatos para reparaciones se siguen desarrollando”, apunta Ulrich Köster, de la Asociación Alemana de Talleres y Reparaciones (ZDK). Los expertos podrán realizar más trabajos de mantenimiento a distancia en los talleres. “El conductor tendrá la posibilidad de recibir un consejo con fundamento técnico sobre un error del vehículo sin tener que ir corriendo al taller”, señala Johannes Boos, del Automóvil Club Alemán (ADAC).

El proveedor de autopartes Continental también parte de esta base. “En el futuro se tratará más de un mantenimiento preventivo y de reconocer un error tempranamente a partir de los datos, para que se pueda evitar el verdadero problema”, considera Patricia Stich, de Continental. Stich resalta que lo decisivo será la calidad de estos datos, que en cada caso deben ofrecer información básica como la velocidad o las revoluciones.

De todas formas, aún no existe una plataforma telemática unificada, así que los dueños de los coches deben recurrir en la mayoría de los casos a la red de talleres de sus marcas. La idea es que haya una plataforma estandarizada. “Si una interfaz de telemática es controlada por un único representante de las partes, como el fabricante de automóviles, no hay libertad de elección para el consumidor”, advierte Boos. “El

cliente debe poder elegir a qué proveedor quiere ofrecerle acceso a su vehículo”, considera Köster.

Los vehículos de gama media y alta son los que cuentan actualmente con una tarjeta SIM incorporada. Ya se pueden hacer varias cosas, como enviar los datos de localización a través de una aplicación que cuenta con un botón para llamar al taller, hasta contar con un hotspot WLAN.

2.26.2 El Sistema AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) AUTOSAR(2016), al español "Arquitectura de Sistemas Abierta para el sector del Automóvil" es una arquitectura de software abierta y estandarizada desarrollada conjuntamente por fabricantes de automóviles, proveedores y desarrolladores de herramientas de software. El objetivo de esta colaboración es crear y establecer estándares abiertos para arquitecturas de componentes electrónicos en el sector automotriz, que provea una infraestructura básica a partir de módulos, interfaces al usuario, y control para los diferentes dominios. Esto incluye la estandarización de funciones de sistema básicas, la portabilidad a diferentes variantes de vehículos y plataformas, la portabilidad a lo largo de la red, la integración de múltiples proveedores y el mantenimiento y nuevas versiones del software a lo largo del ciclo de vida del vehículo.

Metas de AUTOSAR:

- Funda los cimientos para sistemas electrónicos innovadores que mejoran las prestaciones, seguridad y el impacto medioambiental;
- Es un fuerte vínculo global que crea un estándar común: "Cooperar en estándares, competir en su implementación";
- Es una tecnología clave para tener bajo control la creciente complejidad de componentes eléctricos / electrónicos. El objetivo es estar preparados para futuras tecnologías y optimizar los costes sin un deterioro del nivel de calidad;
- Hace posible el intercambio y mejora del software y del hardware a lo largo del ciclo de vida del vehículo.

AUTOSAR se basa en el concepto de arquitectura por niveles. La solución total del software cuenta con 3 niveles conocidos como Software de aplicación, Runtime Environment y Software básico. Todas las funcionalidades se encuentran en el software de aplicación, Runtime environment actúa como un bus virtual de funciones para la centralita electrónica y el software básico se usa para el procesamiento de los mensajes.

Los miembros centrales incluyen a PSA Peugeot Citroën, Toyota, Volkswagen, BMW, Daimler AG, Ford, General Motors, y proveedores de la automoción como Bosch y Continental. En septiembre de 2005 se contaban ya 35 miembros corporativos.

Los objetivos divulgados son la modularidad, escalabilidad, portabilidad y reusabilidad de funciones para proveer una plataforma estandarizada en el sector de la automoción. Esto permitirá una amplia configuración de los sistemas y una optimización del tiempo de ejecución para centralitas electrónicas de uso en los automóviles. Muchos de las interfaces de los módulos cercanos al hardware (principalmente el sistema operativo y el nivel de comunicación) se basan en OSEK.

3 Metodología de la Tesis

3.1 Etapas del proyecto

El presente estudio se procura realizar en tres etapas generales:

- La primera es realizar un estudio de arte de las tecnologías actuales en periféricos automotrices. De ahí, realizar un estudio más a profundidad del protocolo actual OBDII y protocolos asociados resultantes del estudio del arte. En conjunto con este estudio, es determinar los subsistemas secundarios más populares de los automóviles, a llamar sistemas periféricos. Esta etapa se cubre por medio del marco teórico.
- La segunda etapa es, en base a lo aprendido en protocolos actuales, expandir los mismos a incluir los periféricos identificados. En esta etapa se desarrollaría el protocolo para incluir utilizando su misma estructura, o una mejorada dependiendo del estado del arte, instrucciones para control de los periféricos por este medio. Para realizar primeras pruebas se planea utilizar un simulador de OBD por medio de software. La etapa de desarrollo perteneciente a este trabajo es la encargada de cubrir esta parte
- La tercera etapa es el desarrollo de una plataforma de hardware básica para pruebas, corrección de errores y puesta a punto del protocolo. Esta tercera etapa, su inicio podría irse empalmando con partes avanzadas de la segunda según se vea la necesidad. Es entonces que la etapa de implementación y validación del presente trabajo es la idónea para cubrir esta parte.

3.2 Detalle de la Metodología

En pasos más específicos, la primera etapa se cubriría por los siguientes puntos:

- Análisis de la arquitectura de un automóvil
- Definición de los 5 dominios funcionales: Tren motriz, chasis, sistemas auxiliares de seguridad y confort (en inglés “body”), interfaces de usuario, y telemática.
- Con respecto al tren motriz y el chasis, realizar una investigación de los subsistemas, modos y PIDs que ya existen, en base a los estándares J2012 y J1979

- De los sistemas de seguridad, confort e interfaces de usuario hacer un listado de los subsistemas más comunes presentes en vehículos comerciales
- Investigar de dichos subsistemas los listados existentes de PIDs.
- El enfoque principal es en listar los modos. Respecto a los PIDs se buscaría listar los más relevantes para dar contexto al desarrollo.
- Los sistemas del dominio de la telemática como lo son Radio, GPS, centro multimedia y comunicación al exterior no entran dentro del alcance presente trabajo, el cual es enfocarse a sistemas presentes en vehículos comerciales de gama baja. Dependiendo del avance se podrá cubrir parte de estos.
- El trabajo se enfoca a los códigos de comunicación entre sistemas, por lo que capas bajas como la física y capas altas como aplicaciones finales de interface no son la orientación principal. Sin embargo, se tendrán que cubrir las bases para poder dar contexto al trabajo principal

Para la Segunda etapa de desarrollo, una vez listados los modos PIDs existentes se podrá ubicar lagunas y oportunidades en los mismos, como son códigos disponibles y subsistemas no cubiertos.

Detectados estos se procede a crear los mismos utilizando la estructura OBD existente y buscando la simplicidad del mismo que facilite la implementación final.

Es en la tercera etapa, la implementación donde se prueba el sistema y se pone a punto. Esta comprende dos áreas: Hardware y Software

- El hardware comprende dos vertientes: El subsistema de comunicación con el tren motriz y el subsistema de comunicación con periféricos de los otros dominios. Estos convergen en una unidad central del dominio de los sistemas auxiliares de seguridad y confort (en inglés “body”)
- Para la comunicación con el tren motriz, se pretende usar un conector adaptador de OBD (del inglés OBD scanner). Los datos del tren motriz se pueden leer directamente de un vehículo de prueba. Entre estos datos esta por decir la velocidad actual del mismo.

- Para pruebas de escritorio, se contempla la posibilidad de crear un emulador de códigos de OBD, el cual se conectaría el adaptador al mismo.
- Del adaptador saldría una comunicación UART al microcontrolador principal
- La parte central de este sistema estaría conformada por la Unidad controladora Central Electrónica del dominio de seguridad y confort (CeBECU, por las siglas en inglés Central Body Electronic Control Unit). En primera intención se planea utilizar una plataforma mBED por su bajo costo y gran velocidad de procesamiento
- Adicional a esta comunicación con el tren motriz, el CeBECU es la unidad central para la red local de comunicación de subsistemas a enlazar.
- Esta red tiene que implementarse en CAN-BUS para compatibilidad con sistemas modernos
- Los subsistemas contemplados de inicio para incluir en los protocolos son al menos los siguientes:
 - o Las 4 puertas, los cuales cubren las ventanas y los seguros.
 - o El sistema de luces de la parte frontal
 - o Sistema de luces de la parte trasera
 - o Limpiaparabrisas
 - o El HID para conexión con el switches de las luces y de los limpiaparabrisas
 - o Un sistema de display de información, posible de contenido multimedia
 - o Para los sistemas anteriores se pretende en base Arduino, por económico y dado que no requieren mayor capacidad de procesamiento y velocidad. Con excepción del sistema de display de información el cual se pretende sea en base a Raspberry por sus capacidades multimedia
 - o Por la capacidad limitada de desarrollo, Se contempla excluir de la implementación física, mas no del protocolo, a los siguientes subsistemas: espejos, asientos, aire acondicionado y calefacción, sistemas interiores, radio, quemacocos, sensores de reversa, entre otros menores presentes en gamas de vehículos más exclusivos

- Por parte de la implementación de Software, se tendrían que cubrir al menos los siguientes módulos
 - Intérprete de PIDs
 - Puente de comunicación UART / OBD/ CAN BUS
 - Emisor múltiple (del inglés Broadcaster) de PIDs selectos, como la velocidad y temperatura ambiente
 - Interface(s) de usuario
 - Supervisor de aspectos de seguridad (ej alarma, seguros, etc)
 - Supervisor de sistema (ej vehículo encendido, en movimiento, etc)
 - Funciones de aplicación de usuario avanzadas
 - Despliegue de códigos de falla DTC (similar a la función en inglés “Check Engine”)
 - Puente de comunicación entre subsistemas
 - Supervisor de estado de subsistemas

Para la implementación física, se buscaría hacer el desarrollo mínimo suficiente de hardware y software que permita establecer con un buen grado de confianza la correcta función de la propuesta de protocolo

Una vez implementado, se procede a la validación. Esta se planea hacer en dos ambientes: simulación y aplicación real con ambiente reducido. Debido a la gran diversidad de sistemas y la dificultad de probar físicamente todos así como su combinación, es por esto que se opta por estas dos situaciones

La primera es una validación de escritorio, es decir presentar el problema y la solución (universo de sistemas posibles y su codificación) en papel uno frente a otro y analizar punto a punto dicha solución

El siguiente paso es una validación en un ambiente de simulación de software o hardware emulador de los sistemas más comunes. Enseguida se hace una aplicación en un vehículo real, limitado a las capacidades de desarrollo disponibles

En cada etapa se analiza y se presentan resultados. Si hay oportunidades de mejora se atienden antes de pasar a la siguiente etapa. Por último, se presenta la conclusión y resultados finales

4 Requerimientos de producto

El presente trabajo de tesis está enfocado en el desarrollo de un protocolo de comunicación y control de periféricos en el automóvil que sea estandarizada. El trabajo contempla también el desarrollo de una plataforma de hardware con el doble propósito de ayudar al desarrollo del protocolo y de ser en sí también una propuesta de hardware estándar.

En cuestión de hardware, es en este punto donde mayor diversidad se encuentra en el mercado. Para una misma funcionalidad hay innumerables combinaciones de marcas y modelos de vehículos, dentro de las cuales también hay diversas soluciones electromecánicas para una misma función. Es por esta diversidad que se busca hacer una definición más concreta de la plataforma de tesis por medio de un procedimiento estándar de la industria conocido como el desarrollo de producto.

En todo desarrollo de un producto, el elemento clave alrededor del cual se mueve todo el proyecto es el Documento de Requerimientos de Producto (del inglés PRD – Product Requirements Document). El PRD describe el producto que se va a construir. Enfoca todos los esfuerzos del equipo y la compañía que lo va a comercializar así como sus áreas de soporte. Es en definitiva la parte más importante y valiosa en el desarrollo y comercialización de un producto (Cagan, 2005).

El propósito del PRD o Especificación del Producto es de articular claramente y sin ambigüedades el propósito, funciones y comportamiento del producto. Para el desarrollo del producto descrito es clave el utilizar este documento para construir y probar el producto final, por lo que requiere que este suficientemente completo para proveerles la información necesaria para su desarrollo.

4.1 Vehículos Más comunes en México

El primer paso a la hora de desarrollar un producto para un mercado seleccionado, es revisar las condiciones de dicho mercado. El enfoque de este trabajo puede aplicarse a todo automóvil de casi cualquier marca, dado que casi todos funcionan bajo los mismos principios básicos e incorporan las mismas funcionalidades básicas.

Sin embargo, un enfoque global haría demasiado extenso el trabajo de análisis del mercado, por lo que se elige solamente el mercado mexicano como base.

En México, la clasificación de los autos está definida por la Asociación Mexicana de la Industria Automotriz (AMIA), la cual se presenta a continuación:

- AUTOS SUBCOMPACTOS: son unidades con una distancia entre ejes hasta 2,475 mm; con un motor de 4 cilindros de hasta 1800 cm³ de desplazamiento y potencia hasta 110 HP.
- AUTOS COMPACTOS: son unidades con una distancia entre ejes de 2,476 hasta 2,700 mm; con motores de 4 o 6 cilindros de 2500 hasta cm³ de desplazamiento y potencia de 110 hasta 135 HP.
- De manera informal, los subcompactos y compactos se les clasifica también como cupé (dos puertas), Sedan (4 puertas con cajuela) y hatchback o lift back (2 o 4 puertas con puerta/cristal trasero abatible)
- AUTOS DE LUJO: son unidades con una distancia entre ejes de 2,601 hasta 3,000 mm; con motores de 4, 6, 8 o 12 cilindros de 3200 a 6000 cm³ de desplazamiento y potencia de 136 hasta 394 HP.
- AUTOS DEPORTIVOS: son unidades con una distancia entre ejes de 2,601 hasta 3,000 mm; con motores de 4, 6 u 8 cilindros de 2000 a 4600 cm³ de desplazamiento y potencia de 160 a 295 HP, estas unidades también poseen dispositivos especiales en su motor como: turbo cargadores o un mayor número de válvulas en cada cilindro.
- Camiones ligeros (CL1): Camiones ligeros (grupo uno) cuyo peso bruto vehicular es de hasta 2,722 Kg. y con peso de prueba (PP) de hasta 1,701 kg.
- Camiones ligeros (CL2): Camiones ligeros (grupo dos) cuyo peso bruto vehicular es de hasta 2,722 kg. y con peso de prueba (PP) mayor de 1,701 y hasta 2,608 kg.
- Camiones ligeros (CL3): Camiones ligeros (grupo tres) cuyo peso bruto vehicular es mayor de 2,722 y hasta 3,856 kg y con peso de prueba (PP1) de hasta 2,608 kg.

- Camiones ligeros (CL4): Camiones ligeros (grupo 4) cuyo peso bruto vehicular es mayor de 2,722 y hasta 3,856 kg y con peso de prueba (PP1) mayor de 2,608 y hasta 3,856 kg.
- Vehículo de uso múltiple o utilitario (VU): Vehículo automotor diseñado para el transporte de personas y/o productos, con o sin chasis o con equipo especial para operar ocasionalmente fuera del camino. En esta categoría entran los vehículos todoterreno y familiares

De acuerdo al Reporte de Mercado Interno Automotor de Abril de 2016, por parte de la Asociación Mexicana de Distribuidores de Automóviles (AMDA), las ventas totales de vehículos ligeros en el país de Enero a Mayo de 2016 estuvo compuesto de la siguiente manera

Segmento	2016	%
Subcompactos	206,344	35.1%
Compactos	142,633	24.3%
De-Lujo	29,232	5.0%
Deportivos	4,985	0.8%
Usos-Múltiples	122,531	20.9%
Camiones-ligeros	80,908	13.8%
Camiones-pesados	687	0.1%
Total	587,320	100.0%

Tabla 4.1.1 – Ventas de autos en México (Fuente: AMDA, 2016)

De la anterior tabla se desprende que el 80.3% de los autos vendidos en México pertenecen a las categorías de Subcompactos, Compactos y de usos Múltiples (SUV). Del mismo reporte se desprende el segmento de mercado por marca:

Ranking de marcas enero-mayo de 2016

Participación de las marcas con mayor volumen

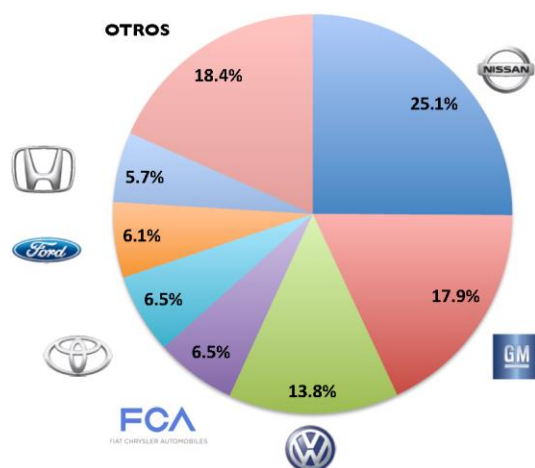


Figura 4.1.1 – Ranking de ventas de autos por marca (fuente: AMDA, 2016)

De acuerdo a la misma AMDA, En Subcompactos, compactos y de usos múltiples los vehículos más vendidos de Enero a Mayo 2016 por segmento son:

Subcompactos			
Modelo	Marca	Unidades	%
Vento	VW	24,231	12%
Aveo	GM	23,490	11%
Tsuru	Nissan	20,647	10%
March	Nissan	19,782	10%
Spark	GM	18,890	9%
Resto segmento		99,304	48%
Total		206,344	100%

Compactos			
Modelo	Marca	Unidades	%
Versa	Nissan	32,409	23%
Nuevo Jetta	VW	21,785	15%
Sentra 2.0	Nissan	15,206	11%
Tiida Sedán	Nissan	7,964	6%
Mazda 3 Sedan	Mazda	6,852	5%
Resto segmento		58,417	41%
Total		142,633	100%

Usos Múltiples			
Modelo	Marca	Unidades	%
Xtrail	Nissan	11,366	9%
Trax	GM	9,584	8%
CR-V-	Honda	8,112	7%
HR-V	Honda	7,045	6%
Sportage	KIA	6,954	6%
Resto segmento		79,470	65%
Total		122,531	100%

Tabla 4.1.2 – Ventas de autos en México por segmento (Fuente: AMDA, 2016)

Por lo tanto, la lista de vehículos más vendidos en México, conjuntando la información anterior con la lista oficial de la AMDA es como sigue:

Posición	Modelo	Marca	Segmento	Unidades	%mercado
1	VERSA	Nissan	Compactos	32,409	5.50%
2	VENTO	VW	Subcompactos	24,231	4.10%
3	AVEO	GM	Compactos	23,490	4.00%
4	NUEVO_JETTA	VW	Subcompactos	21,785	3.70%
5	TSURU	Nissan	Subcompactos	20,647	3.50%
6	MARCH	Nissan	Subcompactos	19,782	3.40%
7	SPARK	GM	Subcompactos	18,890	3.20%
8	SONIC	GM	Compactos	15,303	2.60%
9	SENTRA2.0	Nissan	Subcompactos	15,206	2.60%
10	CHASISLARGO	Nissan	Camiones Ligeros	12,824	2.20%
11	Xtrail	Nissan	Usos Múltiples	11,366	1.93%
12	Trax	GM	Usos Múltiples	9,584	1.63%
13	CR-V-	Honda	Usos Múltiples	8,112	1.38%
14	Tiida Sedán	Nissan	Compactos	7,964	1.36%
15	HR-V	Honda	Usos Múltiples	7,045	1.20%
16	Sportage	KIA	Usos Múltiples	6,954	1.18%
17	Mazda 3 Sedan	Mazda	Compactos	6,852	1.17%

Tabla 4.1.3 – Ventas de autos en México por modelo (Fuente: AMDA, 2016)

	version	L7Z	L7Z Automático	L7Z Automático	L7Z	Epic CVT	EXL Navi 4WD	ACTIVE CARGO TM ABS '17	GSIEQP. T/M MY'17	SEDAN ADVANCE TA 1.8 '17	EXCLUSIVENAVI A/T A/C 1.6L '16	EXCLUSIVCUT MY'17	EXCLUSIV 3 ROW '17	Highline	Sportline 2016
Marca	Chevrolet	Chevrolet	Chevrolet	Chevrolet	Honda	Honda	Nissan	Nissan	Nissan	Nissan	Nissan	Nissan	Nissan	Volkswagen	Volkswagen
Modelo	Spark	Aveo	Sonic	Trax	HR-V	CR-V	March	Tsuru	TIIDA	Versa	sentra	X-TRAIL	vento	Jetta	
Precio	\$ 198,100	\$ 230,800	\$ 253,200	\$ 362,200	\$ 329,900	\$ 439,900	\$ 154,700	\$ 159,300	\$ 214,700	\$ 242,000	\$ 326,500	\$ 460,200	\$ 219,200	\$ 303,900	
Aire acondicionado	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Alarma a control remoto	X		X		X		X	X	X		X	X			
Alarma acústica y óptica									X			X			X
Asiento con ajuste de altura	X		X	X	X	X	X	X		X	X	X			X
Aviso de olvido de llave, luces encendidas y nivel de combustible, cinturón seguridad	X					X	X		X						X
Botón de encendido y apagado de motor					X	X	X			X		X			X
Cámara de reversa					X	X	X			X	X	X			
Computadora de viaje	X		X	X	X	X	X			X	X	X	X	X	X
Control de velocidad crucero				X	X	X				X		X	X	X	X
Controles de audio, y velocidad crucero al volante	X	X	X	X	X	X	X			X	X	X	X		X
Cristales eléctricos	X	X	X	X	X		X			X	X	X	X	X	X
Desempañador trasero eléctrico con temporizador	X	X			X	X	X	X	X	X	X	X	X	X	X
Espejos exteriores abatibles con calefacción	X			X		X					X	X			
Espejos exteriores con ajuste eléctrico	X	X	X	X	X	X	X			X	X	X	X	X	X
Faros de halógeno	X			X	X	X	X		X	X	X				X
Faros de niebla	X	X	X	X	X	X	X			X	X	X			X
Función de verano (cristales por control remoto)														X	X
Indicador de temperatura exterior						X									X
Inmovilizador remoto	X	X	X				X		X	X	X	X			X
Liberador de cajuela										X	X	X			X
Liberador de toma de combustible	X					X	X		X	X					X
Limpiador en cristal posterior	X			X	X		X					X			
Limpiaparabrisas intermitente/con velocidad variable	X	X		X	X		X		X	X	X	X	X	X	
Llave inteligente con función de apertura/cierre de puertas/cajuela y alarma de pánico		X	X		X	X	X			X	X	X	X	X	X
Luces de lectura delanteras														X	
Luces direccionales en espejos	X									X	X				X
Luz en cajuela / área de carga					X	X	X			X	X	X			X
Seguros eléctricos	X	X		X	X		X			X	X	X	X	X	X
Sistema de audio	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Sistema de manos libres (Bluetooth®)	X		X	X	X	X	X			X	X	X			
Sistema de Navegación				X		X	X			X	X	X			
Sistema de pantalla táctil	X		X	X		X	X			X	X	X			X
Tacómetro				X			X		X	X					
Techo Corredizo			X	X	X	X					X	X			X
Tercera luz de freno			X				X				X	X			X
Tomacorriente de 12V					X	X	X	X	X	X	X		X	X	X

Tabla 4.2.1 Sistemas presentes en los autos más vendidos (Fuente: Elaboración propia con información de cada fabricante)

4.2 Módulos vehiculares comunes

Como se ha mencionado anteriormente, el objetivo del presente trabajo es el desarrollo de un protocolo estándar para los sistemas periféricos presentes en un vehículo. Por lo tanto, se requiere definir dicha lista de sistemas periféricos.

De la anterior lista de vehículos más vendidos, para cada modelo se selecciona la versión más completa. De cada uno de estos se consiguió su hoja de especificación, o el listado de sistemas y características con el que cuenta. Estos se pusieron juntos en la anterior tabla comparativa 4.2.1.

La tabla 4.2.2. Contiene algunos otros sistemas de los anteriores vehículos, los cuales son interesantes de mencionar para su posible aplicación

Alerta de Punto Ciego (BSW)/Alerta de Tráfico Cruzado (RCTA)	Indicador de cambio de velocidades para transmisión manual, reflejado en el tablero de instrumentos
Apertura eléctrica de cajuela “No Touch” con sensor - -	Luces de marcha diurna (DRL)
Around View Monitor (AVM) - -	Regulador de intensidad de luz en tablero, Faros de halógeno con función de encendido automático
Botón de encendido y apagado de motor (ligado a llave inteligente)	Limpiaparabrisas delantero intermitente y con velocidad variable sensible a la velocidad
Centro de información con indicadores de odómetro, tacómetro, brújula, consumo de combustible y velocímetro digital	Cristales eléctricos con apertura y cierre de un solo toque y sistema antiopresión (conductor)
Cierre automático de seguros sensible a la velocidad con sensor de impacto (con cierre centralizado)	Compensación de Volumen sensible a la Velocidad (SVC)
Radio Estéreo AM/FM con buscador e identificador de señal, control automático de sintonía y reloj digital y entradas auxiliares para dispositivos MP3, USB, iPod®, audio y video streaming, proyección de fotos, pantalla táctil de 7” con opción de hacer llamadas telefónicas vía Bluetooth® y radio vía internet Interface para Siri Eyes Free de iOS y para navegación vía BringGo app. Apple CarPlay® y Android Auto® y reconocimiento de voz	

Tabla 4.2.2 Otros Sistemas (Fuente: Elaboración propia con información de cada fabricante)

4.3 Sistemas Seleccionados

En base a los anteriores resultados, el enfoque del presente trabajo se puede limitar y orientar hacia los vehículos Subcompactos, Compactos y SUV, de las marcas Nissan, General Motors y Volkswagen. Al analizar solamente los anteriores, se tendrá una cobertura de alrededor del 80% de los vehículos en México. El listado condensado de estos sistemas se enlista en la tabla 4.3.1

Aire acondicionado	Liberador de toma de combustible
Alarma acústica y óptica a control remoto con inmovilizador	Limpiador en cristal posterior intermitente/con velocidad variable
Asiento con ajuste de altura	Limpiaparabrisas intermitente/con velocidad variable
Aviso de olvido de llave, luces encendidas y nivel de combustible, cinturón seguridad	Llave inteligente con función de apertura/cierre de puertas/cajuela y alarma de pánico
Botón de encendido y apagado de motor	Luces exteriores
Computadora de viaje (rendimiento, distancia)	Luz en cajuela / área de carga
Control de velocidad crucero	Seguros eléctricos
Controles de audio, aire y velocidad crucero al volante	Sistema de audio (funciones básicas)
Cristales eléctricos	Espejos exteriores con ajuste eléctrico, calefacción y luces direccionales
Desempañador trasero eléctrico con temporizador	Sistema de telemetría (Tacómetro, velocidad, etc.)
Sistema de pantalla táctil	Techo Corredizo
Faros de niebla	Tercera luz de freno
Liberador de cajuela	Tomacorriente de 12V

Tabla 4.3.1 – Listado condensado de sistemas (Fuente: Elaboración propia)

La anterior tabla es válida para los vehículos más vendidos en México. Con el fin de hacer un comparativo a nivel internacional, se comparó con vehículos similares de otras marcas con presencia mundial como Acura, Audi, BMW, Dodge, Hyundai, Jeep, KIA, Mazda, Scion y Toyota. Estos mismos sistemas se observaron presentes y comunes en los mismos, por lo que dicha tabla tiene relevancia no solo local en México.

Entre otros sistemas de interés de mencionar encontrados en estas marcas están sensores perimetrales de tráfico (aviso de proximidad y/o colisión cambio de carril, tráfico cercano, asistencia de estacionamiento, tráfico en punto ciego, etc), indicadores de presión de neumáticos y espejos abatibles eléctricamente.

4.4 Propósito del producto

Esta es la primera sección del Documento de requerimiento del Producto o PRD (Cagan, 2005). La intención de este es buscar en un solo párrafo condensar la idea del producto final. Para este trabajo se define como:

“Sistema automotriz de plataforma abierta para manejo de sub-sistemas no críticos pertenecientes al dominio de seguridad y confort (body), con un protocolo y plataforma de hardware estandarizados y funcionales para la mayoría de vehículos independientemente de la marca, de fácil implementación en vehículos nuevos o ya existentes, con enfoque al bajo costo por medio de alto volumen derivado de la estandarización y plataforma abierta. “

4.5 Punto de vista del cliente

En esta sección se requiere especificar el Perfil de usuarios, expectativas de cliente y objetivos. Para el presente trabajo se definen los siguientes:

4.5.1 Perfil de usuarios

En base a los anteriores resultados, el perfil del usuario se puede limitar y orientar hacia los actuales propietarios de vehículos Subcompactos, Compactos y SUV, de las marcas Nissan, General Motors y Volkswagen. Al analizar solamente los anteriores, se tendrá una cobertura de alrededor del 80% de los vehículos en México.

Este trabajo inherentemente crea dos grupos de usuarios: Desarrolladores y consumidores

- Los desarrolladores son los usuarios que tomen los conceptos y principios delineados por el protocolo planteado y elaboren productos finales que puedan instalarse en vehículos nuevos o existentes. Dado que uno de los principios es de ser código abierto, el nivel de programación y desarrollo requerido para los desarrolladores no puede ser elevado. Se busca abarcar al desarrollador casero o de empresa chica-mediana los cuales no tienen acceso a elevados recursos.
- Los consumidores son los usuarios que toman el producto desarrollado y lo instalan en su vehículo. Al igual que los desarrolladores de código abierto, se busca abarcar

el mercado del consumidor casero o de empresa chica-mediana. Por lo tanto, el nivel de habilidades y herramientas para su instalación no debe ser complicado ni especializado.

4.5.2 Expectativas de cliente

El proyecto se presentó a 12 personas de profesión ingeniería para obtener retroalimentación. Al presentarles el concepto, tanto en el rol de desarrolladores como de consumidores coincidieron en el perfil buscado. Por lo tanto, las expectativas de cliente quedaron definidas como sigue:

- Facilidad de desarrollo
- Facilidad de instalación
- No requerir herramientas especializadas para desarrollo ni instalación
- Poder adaptarse fácilmente a cualquier vehículo
- Bajo costo
- Cuestiones avanzadas de desarrollo como los detalles del protocolo que sean lo más transparentes posibles.
- Proveer un valor agregado a las soluciones actuales. Que el sistema tenga alguna funcionalidad o ventaja adicional sobre los sistemas actuales.

4.5.3 Objetivos

Por lo tanto, desde el punto de vista del consumidor se tienen los siguientes objetivos, los cuales se requiere que el producto los cumpla para satisfacción del consumidor

- Facilidad de desarrollo e instalación. No requerir herramientas especializadas
- Poder adaptarse fácilmente a cualquier vehículo
- Bajo costo
- Proveer funcionalidades adicionales a las soluciones actuales.

4.6 Principios Básicos del Producto

Los objetivos, funciones, o características básicas y elementales que deben regir todos los aspectos del producto final son:

- Seguro, tanto de uso como de proveer seguridad ante agentes externos
- Mantenerlo explícitamente sencillo ante el consumidor. Buscar dejar la complejidad inherente en capas internas del mismo
- Bajo costo tanto de desarrollo como del producto final
- Proveer funcionalidades adicionales a las soluciones actuales.
- Plataforma de desarrollo abierta

4.7 El PRD – Documento de Requerimientos de Producto

4.7.1 Propósito del Producto

Repitiendo del punto 4.4. El propósito del producto es:

“Sistema automotriz de plataforma abierta para manejo de sub-sistemas no críticos pertenecientes al dominio de seguridad y confort (body), con un protocolo y plataforma de hardware estandarizados y funcionales para la mayoría de vehículos independientemente de la marca, de fácil implementación en vehículos nuevos o ya existentes, con enfoque al bajo costo por medio de alto volumen derivado de la estandarización y plataforma abierta. “

4.7.1 Funciones y características

A partir de los puntos anteriores, se desprenden las siguientes funciones y características específicas, desde el punto de vista del cliente. Dicho punto de vista toma el producto desde una perspectiva total de producto final sin tomar en cuenta detalles de sus subsistemas. Estas se enlistan para mayor claridad y facilidad de seguimiento. Similar a los objetivos a cumplir en un trabajo académico

- Capaz de comunicar, controlar y operar los sistemas del dominio automotriz de seguridad y confort (conocido en inglés como Body)

- Capaz de Interconectar el dominio del body con funciones de los otros dominios como lo es el tren motriz. Esto a través de uno o varios módulo bridge por dominio, el cual puede ser virtual embebido en algún ECU
- Ser plataforma abierta, con código abierto a desarrollo particular
- Seguro, tanto de uso como de proveer seguridad ante agentes externos
- Capaz de integrar y proveer funcionalidades adicionales a las soluciones actuales.
- Capaz de instalarse de forma sencilla y económica en vehículos ya existentes en el mercado

4.7.2 Especificaciones Técnicas

A partir de las funciones y características como producto final mencionadas en el punto anterior, se trabaja ahora en definir los detalles internos del producto. Se establecen entonces requerimientos y especificaciones técnicas detalladas tanto para su comportamiento como producto final, como para sus subsistemas y requerimientos internos. Es aquí donde se hacen notar cumplimientos y requerimientos técnicos como lo son el cumplir con determinados estándares industriales o internos. Así también, cada requerimiento es delineado y expandido al detalle de una manera técnica que proporcione los delineamientos específicos necesarios para facilitar el diseño técnico y fabricación del mismo sin ambigüedades que causen contratiempos durante su desarrollo y puesta en mercado. El sistema por lo tanto debe poder cumplir con los siguientes requerimientos y excepciones notadas:

4.7.3 Especificaciones Técnicas – Software y protocolo

En Alto nivel, el sistema en general debe ser capaz de comunicar, controlar y operar los sistemas del dominio automotriz de seguridad y confort (conocido en inglés como Body).

- El protocolo de comunicación y mensajes se debe realizar adaptando la estructura de headers establecida en los estándares SAE J1850 y SAE J2178 para headers de 3 bytes consolidado. El direccionamiento de acuerdo a los mismos estándares

se realiza funcional y físico según se requiera de acuerdo a esta misma especificación.

- Los bytes de datos acompañantes de los mensajes se deben estructurar de acuerdo a lo establecido en los estándares SAE J1850 y SAE J2178 para headers de 3 bytes consolidado.
- La comunicación física se debe realizar utilizando protocolo CAN-Bus por ISO 15765-4 con formato de header de 29 bits

La manera de implementar el protocolo utilizando J2178-4 sobre CAN BUS utilizando el header de 29 bits es de la siguiente manera:

- Los primeros bits se utilizan en la misma manera que el OBDII para direccionamiento funcional. Estos por lo tanto quedan fijos en 0x18DB
- Para la dirección destino usar de las tablas J2178-4 el ID primario de comando.
- La dirección de la fuente es la dirección física asignada al módulo ECU que emite el mensaje. EL único reservado es 0xF1 para equipo externo. Para otros módulos se recomienda utilizar la llamada general a todos los nodos 0xFE
- Cuando se conteste un mensaje, la dirección destino es el ID primario de estatus, el cual está desplazado un bit al ID primario de comando.
- El primer byte de datos está compuesto de los bits Q y C, y la dirección secundaria ID de 6 bits.
- El bit Q está definido en las mismas tablas primarias de ID del J2178-4 y es dependiente de la función utilizada.
- El bit C queda fijo a un valor 0. Cuando el bit C es combinado con el bit W del header este representa las acciones de cargar un valor (reemplazar) o leer un estado, el cual es conveniente para esta aplicación. Recordando el bit W es el LSB y causa el efecto del desplazamiento en uno entre ID comando e ID estatus. Otras combinaciones de W con C son valores para incrementar / decrementar registros, los cuales complican esta propuesta por lo que no se les ve necesarios.
- En caso de que sea requerido direccionamiento extendido o geográfico, este es enviado como el segundo byte de datos. Estos son definidos en el J2178-4 y son referenciados en cada tabla de ID primario.

- El siguiente byte de datos después del direccionamiento (segundo o tercer byte según si incluye direccionamiento extendido) es el primer byte de datos paramétricos. Por ejemplo, los RPMs son retornados en este byte.
- La decodificación de los bytes de datos enviados a parámetros reales, o función de transferencia es realizada utilizando los PRN o Número de Parámetro de Referencia. Este es un ID de referencia para aquellas funciones que requieren otro byte como cuando se regresa la temperatura.
- No todas las funciones requieren regresar un dato. Por ejemplo el ID del Odómetro al requerírsele puede retornar el kilometraje. O simplemente requerir resetear el mismo a cero para lo cual no es necesario retornar valor alguno. Para eventos como el anterior de set/reset o similares comandos solamente se necesita el bit Q y no es necesario un byte de dato adicional.
- El número de PRN y las funciones de transferencia están definidos en J2178-2. Los primeros 256 códigos (\$00 a \$FF) son los mismos de los ya conocidos SAE J1979 PIDs. Por ejemplo, la temperatura de admisión es referenciada usando el ID primario \$0A, ID secundario \$20, y PRN \$00 0F. El PID J1979 en modo 1 es \$01 0F
- Para codificar una respuesta, o leer la misma, el J2178-2 especifica para cada PRN de qué manera se realiza la codificación. Siguiendo el mismo ejemplo para la temperatura del aire de admisión, el PRN \$00 0F hace referencia a un SLOT UNM-08-102. La tabla correspondiente a este SLOT UNM-08-102 indica que es un número sin signo (del inglés Unsigned Number) de 8 bits, donde el renglón 102 indica que al valor del byte (N) se le deben restar - 40 para tener el valor ingenieril (E) correspondiente a la temperatura en grados centígrados.

El sistema debe ser capaz de Interconectar el dominio del body con funciones de los otros dominios como lo es el tren motriz. Esto a través de uno o varios módulos bridge por dominio, el cual puede ser virtual embebidos en algún ECU.

- Para interconectarse con el dominio del tren motriz debe ser capaz de comunicarse por medio de algún estándar actual OBD con el vehículo, utilizando los protocolos adecuados

- Debido a la gran diversidad de vehículos existentes en el mercado destino, se requiere que el módulo de lectura de códigos OBD sea capaz de soportar los 5 diferentes estándares.
- La comunicación del módulo OBD con el ECU puede ser a conveniencia dependiendo del módulo OBD comercial adquirido o propietario desarrollado. El sistema comercial debe ser capaz de soportar al menos uno de los siguientes: UART, Bluetooth, USB, o CAN-BUS. A nivel de software, la selección de cada uno debe ser lo más fácil e intuitiva posible. Se recomienda el uso de programación orientada a objetos con polimorfismo para facilitar dicha selección. Los comandos deben ser basados en el set de instrucciones del IC ELM 327
- El bridge puede ser un módulo ECU independiente o ser virtual embebido en algún ECU, a conveniencia de la plataforma. Se recomienda por costo un módulo OBD comercial que se comuniquen con el CeBECU por alguno de los métodos indicados.

4.7.4 Especificaciones Técnicas – Hardware

El hardware de los dominios de seguridad y confort del automóvil involucra diversos sistemas a lo largo de todo el vehículo, tal como se vio en el análisis de los sistemas más comunes en el punto 5.2. Si se buscara direccionar cada elemento por separado, por decir cada luz el sistema se vuelve impráctico. Por lo tanto, se propone el delinear módulos que agrupen diversos componentes selectos basados en su localización geográfica.

Para la selección de estos, dado que se busca proponer módulos estandarizados se utiliza direccionamiento físico (basado en localidad geográfica, no en funcionamiento). El estándar SAE J2178 propone una matriz de 7x7 áreas como mapeo físico del vehículo. Esta misma de manera reducida es la que se toma para este trabajo, quedando de la siguiente manera

Frontal izquierdo	Central Frontal	Frontal Derecho
Conductor delantero	Central Cabina	Pasajero delantero
Conductor Trasero		Pasajero Trasero
Trasero izquierdo	Central Cajuela	Trasero derecho

Tabla 4.7.4.1 – Direccionamiento físico seleccionado (Fuente: Elaboración propia)

Estos 11 módulos más el Modulo central (12 en total) pueden cubrir 19 de los 26 sistemas comunes descritos en la tabla 4.3.1, por lo que se consideran una selección bastante representativa para su implementación. Otros sistemas como el aire acondicionado tienen funciones específicas y particulares los cuales por si mismos pueden considerarse un módulo dedicado.

Los módulos seleccionados son unidades centrales actuadoras, en cuanto a que reciben señales externas de sensores o actuadores HID, o de comandos por CAN-BUS, para posteriormente activar algún dispositivo. Para esto, se definen en alto nivel para cada módulo los siguientes requerimientos:

- Entradas
CAN BUS, Interruptores y Sensores
- Funciones
Interpretación de instrucciones y generación de respuesta, de acuerdo al protocolo establecido
- Salidas
Controles de relay seco y/o estado sólido de potencia, preferente este ultimo

Si bien se está recomendado dividir el vehículo en 11 zonas físicas con módulos independientes, se deja opcional el integrar dos o más zonas en un solo modulo físico. Por ejemplo integrar los frontales en un módulo, o los centrales de cabina y cajuela. La condición es que internamente permanezcan como módulos virtuales separados. Esto es, el direccionamiento y respuesta debe realizarse tal y como si fueran módulos físicos independientes.

Los requerimientos para cada módulo se especifican a continuación:

El modulo Central CeBECU (del Inglés Central Body Electronic Control Unit) tiene como funciones principales el servir de enlace entre los diferentes sistemas del auto, así como fungir de repositorio de los estados de los demás subsistemas. Este módulo no tiene una ubicación geográfica definida, sin embargo se sugiere cerca del tablero para facilidad de interconexión con los subsistemas que comprende. Por la definición del proyecto está dividido en los siguientes subsistemas:

- Módulo OBD interface de comunicación con el ECU del vehículo
- Computadora Central del dominio o CeBECU (del inglés Central Body Electronic Control Unit). Esta sección es el repositorio de estado y comandos de los demás subsistemas presentes. Funge también como repositorio de algunos algoritmos que cubren todo el sistema, como lo es la lógica del sistema de alarmas o de compensación de luces.
- Interfaz de comunicación CAN BUS hacia los subsistemas
- Interfaz HID para interacción con el usuario. Se sugiere pantalla LCD independiente de 3" con botonera de navegación o touch. Son admisibles otros esquemas como comunicación a través de bluetooth con una aplicación propietaria residente en un sistema Android onboard.
- Interfaz con los dispositivos HID presentes en el vehículo como interruptores, indicadores y sistema de alarma. La modalidad de este módulo es de funcionar como puente entre dichos sistemas. En modo transparente recibe un estímulo por decir de un switch de luz y envía el correspondiente comando de la luz al módulo indicado. Este módulo puede ser el hardware implementado independientemente pero tener la funcionalidad en modo esclavo sujeto a comandos del CeBECU para lo cual requiere un puerto de comunicación dedicado
- La plataforma se sugiere Android Mega o superior por la facilidad de programación, bajo requerimiento de tiempo real y la capacidad de implementar varios puertos de comunicación.

De la lista de sistemas seleccionados en el capítulo anterior, se observa que todos los vehículos tienen al menos módulos de luces y puertas. Por lo tanto, se delinearán los requerimientos de funcionalidad para dichos módulos.

Los módulos frontal/trasero - izquierdo/derecho tienen como función principal las luces correspondientes. Los módulos de las luces deben contar con los siguientes elementos

- Puerto de comunicación CAN-Bus capaz de soportar los anteriores
- Alimentación de 12V nominal protegida con fusible on board de acceso externo, intercambiable de acuerdo a la carga conectada
- Dip Switch de 2 posiciones para indicar su instalación geográfica (ej. Chofer-delantero)
- Conectores adecuados para uso en interiores automotrices, por mejor practica
- Protección ambiental para uso en interior automotriz, por mejor practica
- Al menos 2 puertos de salidas independientes de al menos 25W c/u para luces de posición o direccionales. Esto son para el lateral y Frontal/trasero
- Al menos 3 puertos de salida independientes de al menos 100W c/u para luces faro. En el frente estos son Luz baja, luz alta y luz faro de niebla
- Los anteriores deben contar con algún tipo de sensado de corriente el cual pueda ser capaz de detectar una luz no operacional (foco fundido)
- Opcional al menos un puerto de salida independiente de al menos 25W genérica para expansión.
- Opcional al menos una entrada digital open collector genéricas para expansión.

Los módulos Conductor/Pasajero – Frontal/trasero tienen como función principal dar soporte a las funciones de las puertas. Los módulos de las puertas deben contar con los siguientes elementos

- Puerto de comunicación CAN-Bus capaz de soportar los anteriores
- Alimentación de 12V nominal protegida con fusible on board de acceso externo, intercambiable de acuerdo a la carga conectada
- Conectores adecuados para uso en interiores automotrices, por mejor practica
- Protección ambiental para uso en interior automotriz, por mejor practica

- Dip Switch de 2 posiciones para indicar su instalación geográfica (ej. Chofer-delantero)
- Se requieren dos versiones: Básica y Extendida. Para la Básica se requiere lo siguiente:
 - Un controlador tipo H para motor DC capaz de manejar 10Amps con sensor de corriente. Velocidad variable opcional.
 - Un controlador para el seguro eléctrico. Este es similar al puente H, solamente que debe manejar impulsos de corriente de alrededor de 5 amps.
 - Una entrada digital para detectar puerta cerrada
 - Al menos dos entradas digitales para poner o quitar los seguros eléctricos
 - Al menos dos entradas digitales para actuación de la ventana subir-bajar. Opcional si es el modulo trasero controlar las ventanas de ambos lados. Si es uno delantero controlar todas las ventanas. Este puede ser un tablero matricial codificado para minimizar pines del uP
- Para la Extendida se requiere lo siguiente:
 - Un controlador tipo H para motor DC capaz de manejar 10Amps con sensor de corriente. Velocidad variable opcional.
 - Un controlador para el seguro eléctrico. Este es similar al puente H, solamente que debe manejar impulsos de corriente de alrededor de 5 amps.
 - Una salida digital 12V para desempañante del espejo, capaz de manejar al menos 1 amp
 - Una salida digital 12V para luz direccional, capaz de manejar al menos 25 watts
 - Ocho entradas digitales para actuación de la ventana subir-bajar para cada ventana del vehículo (4x). Este puede ser un tablero matricial codificado para minimizar pines del uP
 - Dos entradas digitales para poner o quitar los seguros eléctricos. Este puede ser parte del tablero matricial
 - Dos entradas digitales para poner o quitar bloqueo de ventanas. Este puede ser parte del tablero matricial

- Una entrada digital para detectar puerta cerrada
- Opcional Al menos dos entradas digitales genéricas para expansión
- Opcional Dos salidas digital 12V genéricas para expansión, capaz de manejar al menos 25 watts
- Opcional dos salidas digitales 12V para X, Y del espejo eléctrico, capaz de manejar un circuito driver. Los mecanismos de espejos dependen en gran medida del fabricante por lo que es difícil estandarizarlos.

El hardware de los módulos centrales puede ser idéntico a los módulos de las luces. Solamente requiere cambiar su direccionamiento ya sea por firmware o por medio de un pin adicional hardwired.

Estos módulos se sugiere deben soportar al menos los siguientes sistemas:

Central Frontal:

- Salida de activación de claxon
- Salida de activación de limpiador de parabrisas
- Driver para motor de limpiador de parabrisas de velocidades
- Entrada digital de detección de cofre abierto
- Luces adicionales opcionales
- Opcional entradas y/o salidas adicionales

Central Cabina:

- Salida de activación de luces interiores, 2 o 3 zonas
- Salida para activación de desempañado de cristal trasero
- Driver para el techo corredizo (quemacocos)
- Activación o desactivación de puertos de carga/tomacorriente de 12V

Central Cajuela:

- Salida de activación de luces interiores
- Liberador de cajuela
- Sensado de cajuela cerrada

- Driver para motor de limpiador de parabrisas trasero de velocidades
- Tercera luz de freno
- Liberador/ sensor de toma de combustible
- Luz de la placa
- Luces adicionales opcionales
- Sensores de reversa
- Opcional entradas y/o salidas adicionales

Página en blanco

5 Desarrollo de la contribución central de la tesis

5.1 Introducción

La tecnología al ir avanzando ha permitido hacer más complejos y confiables los diversos dispositivos que se usan en la actualidad. En los automóviles, en la década de los '90 es cuando se señala como la era donde la electrónica llegó a estos. Iniciando con la incorporación cada vez más común de sensores eléctricos y en diversos lugares, hasta el desarrollo de una computadora que centralice toda esa información y coordine los actuadores correspondientes. Dicha computadora abrió la posibilidad de hacer más dinámicos e inteligentes los sistemas automotrices, permitiendo por ejemplo incrementar la eficiencia de combustible a la vez de reducir emisiones tóxicas.

A inicios de esta década de los 90's surgió el primer protocolo de comunicación, OBD (On Board Diagnostics) el cual estandarizó la manera de comunicarse en los autos a través de las diferentes compañías. Estos mensajes eran básicos y solamente de estados críticos del tren motriz (Roberts, 2009).

En 1996 las diversas iniciativas de los fabricantes de autos se agruparon en los estándares SAE J1962 y J1850 para capa física e ISO 15031 para el protocolo de comunicación. Todos los autos vendidos a partir de este año incorporaron estos estándares, en lo que en adelante se conoció genéricamente como OBDII, en referencia al original OBD en segunda versión.

Este nuevo grupo de estándares, denominado simplemente OBDII incorporó un set de mensajes fijos de estatus del tren motriz, el cual se podría decir universal dado que cualquier marca de auto maneja los mismos mensajes básicos. Esto permitió estandarizar las herramientas de diagnóstico, e incluir el famoso "check engine" en los carros, o luz indicador de problema en el motor. Este estándar se encuentra en la gran mayoría de los carros circulando en la actualidad (Henderson, Haynes, 2006). A principios de la década del 2000, la marca BOSCH lanzó al mercado la propuesta de un protocolo Ethernet, el CAN-BUS. Este protocolo contempla comunicación serial asíncrona entre dispositivos utilizando un solo canal a través de un único par trenzado que enlaza a todos los dispositivos. La ventaja es la reducción significativa de alambrado al dejar de usar un cable dedicado para cada función. La desventaja es la

complejidad agregada y la inherente necesidad de módulos dedicados y especializados que decodifiquen dicha comunicación serial y la traduzcan nuevamente en cables dedicados a cada función. Los autos a partir del 2008 incorporan este protocolo (PR, 2015)

Para entender el nivel presente de los sistemas de diagnóstico en el vehículo, hay que remontarse algunas décadas atrás para poder apreciar cuanto ha avanzado la tecnología en los vehículos (Henderson, Haynes, 2006). Mientras que el diagnóstico en el vehículo, conocido como OBD no es un participante activo en las emisiones del vehículo, su desarrollo está relacionado estrechamente al sistema de control de emisiones de los autos y camiones. Cuando los fabricantes de vehículos respondieron en los 90's al clamor público por un aire más limpio y pusieron más atención a las recomendaciones federales que evolucionaron más tarde en estándares y regulaciones de ley que había que cumplir, la ciencia del control de emisiones en motores de combustión estaba en sus primeros inicios.

El OBD (diagnóstico a bordo, del término On board Diagnostics) no tuvo lugar hasta que los autos estuvieron equipados con controles computarizados. La compañía General Motors implemento una versión primeriza de OBD en algunos de sus vehículos de 1980. Conforme la inyección electrónica de combustible y otras funciones fueron siendo controladas por la computadora de abordo (El módulo de control de transmisión y motor, del termino Powertrain Module Control o PCM como es llamado hoy) la implementación del OBD se volvió práctica. El estado de California en Estados Unidos de América es quien ha liderado el camino en conseguir legislaciones cada vez más estrictas en el control de emisiones, tanto que para los fines de los 80's requirió que todos los vehículos vendidos en ese estado incorporaran un sistema OBD. Los fabricantes de autos y camiones ligeros tuvieron que desarrollar el software y hardware necesarios para poder proveer la capacidad de diagnóstico.

El objetivo del sistema OBD original era promover el aire limpio por medio de asegurar que los componentes del control de emisiones siguieran funcionando. Muchos estados de la unión americana empezaron a incluir una revisión de emisiones en su programa de revisión vehicular de seguridad anual o bienal. Pero, en la práctica mucho puede suceder al vehículo en un año entre revisiones, y las pruebas simplificadas de entonces

solamente “olían” el escape mientras el vehículo estaba estacionado, no cuando estaba en movimiento. Además, estas pruebas eran de “pasa o falla”. Entonces los automovilistas a los cuales fallaba su vehículo la prueba tenían que por su cuenta encontrar un taller que pudiese determinar que causó que las emisiones fueran más altas, para poder arreglarlo y presentarse a prueba nuevamente.

La idea del OBD era tener un vehículo que realizara su propio monitoreo del control de emisiones, todo el tiempo, y más que eso asignar números de código que pudiesen identificar el problema, y finalmente guardar esos códigos de problema en la computadora. Una luz indicadora de problema en el tablero del vehículo podría indicar al conductor que hay un problema en el sistema de emisiones, y cuando el auto estuviese en el taller, el técnico podría extraer los códigos y saber que partes del sistema debería revisar y reparar.

A principios de los 80's los ingenieros en Bosch estaban evaluando sistemas seriales de la época con respecto a posibles aplicaciones en autos de pasajeros. Dado que ninguno de los protocolos disponibles en ese entonces cumplía con las expectativas automotrices, el grupo de Bosch desarrolló su propio protocolo. La idea principal detrás del desarrollo era la reducción drástica de cableado en el automóvil. De pasar a tener un cable dedicado para cada función, a tener un solo bus recorriendo el auto. Los ingenieros de Mercedes – Benz se sumaron al inicio con los primeros requerimientos del nuevo protocolo, así como Intel con miras a ser proveedor de los semiconductores. En febrero de 1986, el nuevo protocolo CAN nació.

El estándar Bosch CAN en versión 2.0 (algunos detalles corregidos de la primera versión) fue presentada como propuesta de estándar internacional. En Noviembre de 1993 fue publicado como el estándar ISO11898. Otras revisiones y versiones le siguieron, como el ISO11898-1 para la capa de datos, ISO11898-2 para la capa física no tolerante a errores, el ISO11893-3 para capa física tolerante a errores, ISO11992 para camiones y el ISO11783 para maquinaria agrícola.

En Europa el OBD tomó la forma de EOBD y se hizo mandataria para el 2001. En vehículos de carga pesados en Europa para el 2005 y en Estados Unidos en 2008. Brasil en 2012 e India en 2013. (Posada, Bandivadekar, 2015). Para mediados de la década del 2000, en los automóviles particulares había básicamente 4 diferentes

estándares OBD: SAE J1850 (FORD – PWM & GM - VPW), ISO9141-2 (Chrysler, Europa y Asia), ISO14230 (KWP2000) e ISO 15765 (CAN BUS). Para el 2008 y delante se hizo mandatorio que el único protocolo a utilizar sea el ISO15765 – 4.3, la cual es una variante del estándar CAN a 500 kbps (Cassias, Kun2007). Este es el estándar en uso hoy en día.

5.2 Análisis de la arquitectura de un automóvil

Históricamente se han identificado 5 dominios (Navet & Simonot-Lion, 2009): Tren motriz, chasis, sistemas auxiliares de seguridad y confort (en inglés “body”), interfaces de usuario, y telemática. El tren motriz está relacionado con los sistemas que participan en la propulsión longitudinal del vehículo, incluyendo el motor, la transmisión y sus componentes subsidiarios. El chasis se refiere a las cuatro ruedas y su relativa posición y movimiento. En este dominio, los sistemas son principalmente la dirección y los frenos. Los sistemas auxiliares de seguridad y confort agrupan los sistemas que no están relacionados a la dinámica de movimiento del vehículo, sino al soporte al conductor como lo son la bolsa de aire, los limpiadores de parabrisas, luces, ventanas eléctricas, aire acondicionado entre otros. La interface de usuario incluye el equipo que permite el intercambio de información entre los sistemas y el conductor como lo es interruptores y paneles de información. Finalmente, el dominio de la telemática está relacionado con componentes que permiten el intercambio de información del vehículo con el mundo exterior, como lo son el radio, sistemas de navegación, internet, pago remoto de casetas, entre otros.

De un dominio a otro, los sistemas electrónicos difieren en sus características. Por ejemplo, el tren motriz y el chasis tienen requerimientos muy estrictos de datos en tiempo real y elevados niveles de poder de computación. Sin embargo, la arquitectura del hardware del dominio del chasis está completamente distribuida a través de todo el vehículo, cubriendo los sistemas dispersos a través de todo el vehículo. El dominio de la telemática, en contraste presenta requerimientos de alto manejo de datos contenidos usualmente en un solo equipo físico. Para estos dos sistemas, las soluciones tecnológicas difieren tanto en los sistemas embebidos requeridos como sus necesidades de interconexión y manejo de datos.

De acuerdo a Navet & Simonot-Lion, (2009) el dominio de los sistemas auxiliares de seguridad y confort controlan funciones embebidas en un vehículo que no están relacionados al control de su dinámica. Hoy en día, los limpiadores, luces, puertas, ventanas, asientos y espejos están controlados cada vez más por sistemas automatizados. En general no están sujetos a situaciones tan rigurosas como en el dominio del tren motriz y el chasis, por lo que no representan sistemas críticos. Sin embargo, hay sistemas como el de alarma de seguridad para el acceso al interior que tiene que respetar reglas de tiempo real.

Estos sistemas por lo general por estar distribuidos en todo el vehículo involucran una red de comunicación entre todos ellos haciendo más compleja su estructura. Es en este dominio donde emerge la noción de subsistemas basados en redes locales de sensores y actuadores de bajo costo, por ejemplo en red LIN (del acrónimo en inglés Local Interconnect Network o red local interconectada), la cual conecta los módulos contruidos como sistemas mecatrónicas integrados. Por ejemplo, diversas funciones pueden estar presentes en una puerta: poner o quitar seguros de acuerdo a una señal transmitida por una red inalámbrica, control de ventanas de acuerdo a señales del conductor o pasajero, así como la posición de los espejos.

Una posible aplicación de estas funciones puede ser un solo control central ECU (del inglés Electronic Control Unit, unidad de control electrónico) el cual centraliza la recepción de los sistemas que requieran la acción, mientras que los controles para los motores que realizan las acciones en el dispositivo físicamente como el espejo o la ventana son soportados por otros tres ECUs. Estos cuatro ECUs están conectados en una LIN. Como algunos requerimientos involucran varias puertas como lo es el quitar los seguros, el ECU principal de cada puerta está conectado a una red CAN de baja velocidad. Finalmente para presentar el estatus de las puertas al conductor, la información es transmitida por el ECU principal al ECU del tablero por medio de CAN-BUS.

5.3 Módulos CAN-BUS, Alta y mediana velocidad

De acuerdo a Breeves (2013), en los autos Ford y similares a partir del 2004 es cuando se empezó a utilizar el CAN BUS. Anterior a estos en Ford se utilizaba SAE J1850PWM. Anterior a 2013, hay básicamente dos CAN buses separados, llamados de

alta velocidad y mediana velocidad (del inglés HS-CAN high speed CAN, y MS-CAN Medium Speed CAN). El HS-CAN se utiliza para datos con prioridad, como los parámetros del módulo del motor PCM y comunicación crítica entre módulos de seguridad como los frenos ABS y el módulo RCM de las bolsas de aire. El otro bus MS-CAN se utiliza para otras cosas en el interior del auto como lo es el radio, el aire acondicionado y los controles de las luces

En los autos Ford a partir del 2013, prácticamente cada función y control pasa por CAN-BUS. Estos tienen cuatro buses: tres de alta velocidad y uno de mediana velocidad, debido a la gran cantidad de datos que circulan. Algunos elementos conectados al bus pueden ser el switch de las luces, los de las puertas, ajustes de los espejos y prácticamente cada botón que este a la vista.

Dado que la información que circula en los buses es propietaria de cada fabricante, Breeves (2013) realizó un escaneo de un Ford Taurus 2013 para enlistar los módulos existentes. Los dos buses HS-CAN y MS-CAN se conectan juntos en el cluster del panel de instrumentos (del inglés IPC – Instrument Panel Cluster). Es en este módulo donde se comparten información entre los diferentes buses, como lo es por ejemplo la velocidad del vehículo.

Los módulos que encontró en el bus HS-CAN son:

- PCM – Módulo del tren motriz (del inglés Powertrain Control Module)
- ABS - Sistema antibloqueo de frenos (del inglés Anti-lock Brake System, incluye AdvanceTrac and Traction Control)
- RCM – Módulo de control de restricciones (del inglés Restraint Control Module, las bolsas de aire y los cinturones de seguridad)
- AWD – Tracción en todas las ruedas (del inglés All Wheel Drive module)
- OCSM – Módulo de clasificación de los ocupantes (del inglés Occupant Classification System Module, o sensores de los asientos de pasajeros)
- PAM – Módulo de auxilio para estacionamiento (del inglés Parking Aid Module)
- IPC – Tablero de instrumentos (del inglés Instrument Panel Cluster)
- PSCM – Módulo de control de la dirección hidráulica (del inglés Power Steering Control Module, Hydraulic PS only)

- SECM – Modulo de control de la dirección eléctrica (del inglés Steering Effort Control Module, Electric PS only)
- CCM - Modulo de control de cruceo (del inglés Cruise Control Module)
- APIM – Modulo de interface de protocolos (del inglés Accessory Protocol Interface Module AKA SYNC)

A su vez, los módulos presentes en el MS-CAN son:

- SJB/BCM Modulo de control del Body (del inglés Smart Junction Box/Body Control Module)
- HVAC – Aire Acondicionado (del inglés Heating, Ventilation & Air Conditioning module)
- ACM – Modulo de controles de audio (del inglés Audio Control Module)
- DSP – Modulo de audio digital (del inglés Audio Digital Signal Processing Module)
- DSM – Modulo del asiento del conductor (del inglés Driver Seat Module)
- DDM – Modulo de la puerta del conductor (del inglés Driver Door Module)
- RFA - Control Remoto (del inglés Remote Function Actuator Module)
- DCSM – Modulo de calefacción del asiento (del inglés Dual Climate Controlled Seat Module)
- SDARS – Radio digital satelital (del inglés Satellite Digital Audio Radio Service)
- FCIM – Modulo de controles frontales (del inglés Front Controls Interface Module)
- FDIM – Modulo de interface indicadora frontal (del inglés Front Display Interface Module)
- ILCM – Modulo de luces interiores (del inglés Interior Lighting Control Module)
- HCM-2 – Modulo de luces faros frontales (del inglés High Beam Control Module)
- SOD-R/L – Modulo de detección de obstáculos a los lados, uno en cada lado (del inglés Side Obstacle Detection Control Module AKA BLIS)
- IPC-MS – Panel de instrumentos (del inglés Instrument Panel Cluster)
- GPSM – Modulo de posicionamiento global (del inglés Global Positioning System Module)
- APIM – Modulo de interface de protocolos (del inglés Accessory Protocol Interface Module AKA SYNC)

5.4 Los PIDs existentes no de control de emisiones

La gran mayoría de los PIDs del OBDII en uso son no-regulados por algún estándar de oficina reguladora como SAE o ISO. Para la mayoría de los vehículos modernos, hay muchas más funciones soportadas en la interface OBDII que las cubiertas por PIDs estándares, y hay un relativo poco parecido entre fabricantes respecto a estos PIDs no-estándar (The Wikimedia Foundation, 2016)

Hay muy poca información y muy limitada en el dominio público para PIDs no-estándar. La fuente primaria de información de PIDs no-estándar entre los diferentes fabricantes es mantenida y administrada por el Equipment and Tool Institute de Estados Unidos, el cual solo da acceso a dicha información a los miembros del mismo instituto. El costo de una membresía para acceder a dichos códigos varia basado en el tamaño de la compañía definido por sus ventas anuales de equipos en Norteamérica (ETI, 2015):

Ventas anuales en Norte América	Costo Anual
hasta \$10,000,000	\$5,000
\$10,000,000 - \$50,000,000	\$7,500
Más de \$50,000,000	\$10,000

Tabla 5.4.1 – Costo de membresía ETI (ETI, 2015)

Sin embargo, aún con esta membresía, el Instituto no proporciona toda la documentación para PIDs no-estándar. Algunos fabricantes OEM rechazan utilizar el instituto ETI como una fuente única para información de herramientas de diagnóstico. Dichos fabricantes prefieren hacer negocios con cada compañía por separado. Estas compañías también requieren un contrato muy exclusivo con ellos. Algunos de estos costos que cargan dichas compañías por el contrato son (ETI, 2016):

GM	\$50,000
Honda	\$5,000
Suzuki	\$1,000
BMW	\$25,500 plus \$2,000 por actualización anual.

Tabla 5.4.2 – Costos de contrato con OEMs (ETI, 2016)

5.5 Tipos de mensajes y direccionamiento

Los códigos de avería OBD conocidos como DTCs (del inglés Diagnostic Trouble Code) son por mucho los mensajes más conocidos y populares entre los sistemas OBD. Estos se dividen en cuatro clases: P – Motor, C – Chasis, B – Carrocería y U – Comunicaciones. Así por ejemplo, el código de avería DTC P0003, es un código de avería del motor

El segundo grupo de información que el vehículo es capaz de proveer son los códigos de información de parámetros (del inglés Parameter IDs o PIDs). Los PIDs entregan información de las condiciones u operación del vehículo, muchos de los cuales son en tiempo real. Por decir, dan la información de la velocidad actual, las revoluciones del motor, temperatura de los diversos sensores, entre otros.

Estos se agrupan en conjuntos por funciones, conocidos como modos OBD. Los modos estándar principales en que se presenta la información son:

- **Modo 1** - Identificación de Parámetro (PID, del inglés Process Identification)
- **Modo 2.** Acceso a Cuadro de Datos Congelados (del inglés Freeze Frame Data)
- **Modo 3.** Este modo permite extraer de la memoria de la ECU todos los códigos de fallo (del inglés DTC - Data Trouble Code) almacenados.
- **Modo 4.** Con este modo se pueden borrar todos los códigos almacenados en la PCM, incluyendo los DTCs y el cuadro de datos grabados.

La comunicación de comandos OBD se realiza enviando bytes en hexadecimal. El primer byte corresponde al modo OBD en el que se quiere trabajar, el segundo byte corresponde a un parámetro de identificación del modo previsto (PID).

Por ejemplo, Si se quiere preguntar cuál es la temperatura del motor, el primer byte será el modo "01" que muestra flujo de datos y el segundo byte es el PID "05". La comunicación a nivel de bytes se realiza de la siguiente manera:

>01 05 Pregunta

>41 05 7B Respuesta

Por último, el mensaje completo OBD incluye un header el cual indica el tipo de mensaje. Los códigos DTC y los PIDs comunes utilizan un header fijo dependiente del estándar de comunicación utilizado. Por decir, para J1979 de 10.4Kbps se utiliza un header 68 6A F1 y para el mismo en 41.6Kbps se utiliza 61 6A F1.

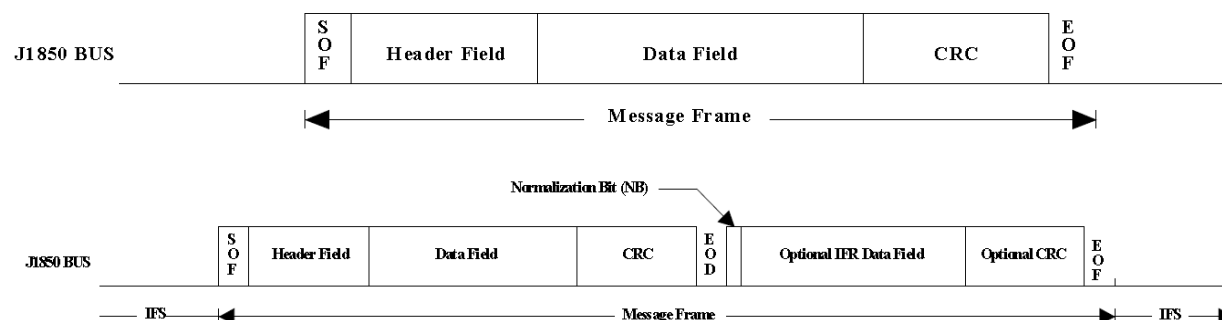


Figura 5.5.1 Estructura del paquete de datos (Fuente: SAE J1850)

Los sistemas OBD nacieron a partir de las iniciativas de control de emisiones. Estos fueron los primeros encargados de monitorear el estado del tren motriz a través de diversos sensores. A partir de esta información, los sistemas embebidos realizan los controles y ajustes necesarios para regular dichas emisiones. Estos primeros sistemas fueron evolucionando incorporando cada vez más sensores y más actuadores, ejerciendo controles más avanzados como algoritmos para balancear mayor potencia con mejor rendimiento de combustible. Una vez iniciado la incorporación de sistemas electrónicos en el automóvil, otros sistemas se fueron incorporando como lo es el tablero de instrumentos digital y módulos conectados en red.

Los mensajes digitales que circulan dentro del automóvil se pueden entonces distinguir en dos grandes grupos: los relacionados al control de emisiones (del inglés emission control messages) y los que no están relacionados (del inglés non-standard messages). Todos los mensajes tienen un origen y un destinatario. Hay dos maneras en que los destinatarios pueden ser localizados en un sistema OBD: Por medio de la ubicación física del módulo destino, o por la función requerida por el mensaje mismo independiente de cual modulo es el que debe de contestar.

El estándar SAE J2178 hace una distinción entre las dos maneras de dirigir los mensajes, o de realizar el direccionamiento de la comunicación:

Tipos de Direccionamiento (SAE J2178):

- Direccionamiento físico o dirigido a un destino con una localización física determinada, como lo es dirigirse a un módulo en específico.
- Direccionamiento funcional, significando que los mensajes están identificados dependiendo del contenido de los mismos o su función como el caso de los PIDs vistos con información de estado de sensores y sistemas del auto.

5.6 El estándar SAE J2178 – Mensajes de comunicación

El estándar SAE J2178 define la información contenida en el encabezado y los campos de datos de mensajes que no son de diagnóstico para comunicación serial basada en redes SAE J1850. Este documento describe los campos de los encabezados, datos, tamaño de los campos, escalamiento, representaciones y posición de los datos.

El SAE J2178 se divide en 4 partes:

La sección primera describe las dos formas de encabezados de los mensajes, de un byte y de formato consolidado. El formato consolidado tiene a su vez dos formas, de un byte y de tres bytes. Esta sección también cubre la manera de identificar el contenido de los mensajes

La sección segunda define los datos paramétricos estándar que pueden transmitirse en estas redes. La intención de esta sección es que al menos una de estas definiciones de parámetros debe coincidir con los especificados por SAE J1979

La sección tercera asigna los mensajes para el header de un byte y el consolidado de un byte

La sección cuarta asigna los mensajes para el header consolidado de tres bytes

En la literatura relacionada a los sistemas automotrices, como en Breeves (2013), López (2008), Posada, F & Bandivadekar (2015) y otros el estándar SAE J2178 es la referencia usual para definir el encabezado del mensaje OBD. Sin embargo, este estándar incluye varias más definiciones como las especificaciones para parámetro.

Todas estas demás funciones que incluye el estándar, no se encontró alguna literatura o sistema que haga referencia al mismo.

La organización SAE (del inglés Society of Automotive Engineers) realizó la última actualización a este estándar en 1999, lo revisó en 2004 sin modificaciones, y lo declaró estable en 2012 significando que no considera ser un estándar el cual tenga uso significativo en la industria.

El estándar, como otros estándares SAE es definida como una “práctica recomendada”, es decir es un documento el cual busca estandarizar diversos sistemas a una sola práctica. Sin embargo, no es obligatorio por ley como otros, es solo recomendado su seguimiento. Los diversos fabricantes han desarrollado sus propios sistemas de protocolos de mensajes propietarios no compatibles entre sí, y en ocasiones tampoco entre modelos de la misma marca.

Es bajo lo anterior en lo que se basa esta propuesta de protocolo, buscando sea estándar y en modalidad de sistema abierto.

5.7 El estándar SAE J2178-1 – Direccionamiento Físico

La sección 1 cubre dos áreas: la definición del header, direccionamiento extendido físico (geográfico) y asignaciones de direcciones físicas. El header quedó explicado en la sección 2.21 de este trabajo. Comprende dos formas: la de un solo byte y la de byte consolidado el cual se divide en dos, de un byte y de tres bytes.

Recordando, el header de 3 bytes para direccionamiento funcional incluye una dirección ID primaria destino en el segundo byte. El tercer byte es la dirección origen. El primer byte de datos es la dirección ID secundaria. Los siguientes bytes son datos, excepto el formato 3. En el formato tipo 3 para direccionamiento funcional, el segundo byte es el direccionamiento extendido. El resto de bytes son de datos.

..	Secondary Address	Extended Address	Data 1	...	Data n	...
----	-------------------	------------------	--------	-----	--------	-----

Figura 5.7.1 formato de campo de datos funcional tipo 3

El direccionamiento extendido define una localización geográfica en el vehículo. Los primeros dos bits (MSB) del modo extendido RR son usados para permitir cuatro diferentes tipos de direccionamiento extendido. Para RR -00 es el caso de direccionamiento geográfico.

RR	Type
00	Geographical Addressing
01	Reserved - SAE
10	Reserved - MFG
11	Reserved - MFG

Tabla 5.7.1 tipos de direcciones extendidas.

Left Side Front Surface	Driver Side Front Surface	Left Center Front Surface	Center Point Front Surface	Right Center Front Surface	Passenger Side Front Surface	Right Side Front Surface
Left Side Under Hood	Driver Side Under Hood	Left Center Under Hood	Center Point Under Hood	Right Center Under Hood	Passenger Side Under Hood	Right Side Under Hood
Left Side Aft Bulkhead	Driver Side Aft Bulkhead	Left Center Aft Bulkhead	Center Point Aft Bulkhead	Right Center Aft Bulkhead	Passenger Side Aft Bulkhead	Right Side Aft Bulkhead
Left Side Aft "A" Pillar	Driver Side Aft "A" Pillar	Left Center Aft "A" Pillar	Center Point Aft "A" Pillar	Right Center Aft "A" Pillar	Passenger Side Aft "A" Pillar	Right Side Aft "A" Pillar
Left Side Mid Vehicle	Driver Side Mid Vehicle	Left Center Mid Vehicle	Center Point Mid Vehicle	Right Center Mid Vehicle	Passenger Side Mid Vehicle	Right Side Mid Vehicle
Left Side Trunk	Driver Side Trunk	Left Center Trunk	Center Point Trunk	Right Center Trunk	Passenger Side Trunk	Right Side Trunk
Left Side Rear Surface	Driver Side Rear Surface	Left Center Rear Surface	Center Point Rear Surface	Right Center Rear Surface	Passenger Side Rear Surface	Right Side Rear Surface

Tabla 5.7.2 - SAE J2178 – Direccionamiento Extendido Geográfico

El formato de direcciones extendidas para direccionamiento geográfico se codifica de la siguiente manera

Bit 7	6	5	4	3	2	1	0
RR = 00		XXX = Row #			YYY = Col #		

Figura 5.7.2 Direccionamiento extendido

Los códigos XXX = 000 (para renglones) y YYY = 000 (para columnas) indican que Todos los renglones y/o Todas las columnas son seleccionados. En otras palabras, para dirigirse todos los ítems de un renglón en específico, independiente de que columna, se utiliza XXX = 000. Para todos los ítems en la misma columna, independiente de cual renglón, utilizar YYY = 000. Utilizando todas las combinaciones de XXX y YYY hay 64 zonas geográficas en total que pueden ser direccionadas. La tabla 4.7.3 muestra estas posibles combinaciones.

La tabla 4.7.2 tomada del mismo SAE J2178 especifica las 49 áreas físicas en las que divide el vehículo formando una matriz de 7x7 secciones. Representa de manera general una vista superior del vehículo y soporta las referencias generales izquierda derecha (del inglés left/right) y conductor/pasajero (del inglés driver/passenger).

XXX Range	YYY Range	Description	# areas
000	000	ALL Rows and Columns	1
000	001 - 111	ALL Rows x 7 Columns	7
001 - 111	000	ALL Columns x 7 Rows	7
001 - 111	001 - 111	49 Distinct Areas	49

Tabla 5.7.3 - SAE J2178 – Asignaciones validas

Por otro lado, en la tabla 5.7.3 se ve un ejemplo de direccionamiento de los mensajes dependiendo de la función o contenido requerido por el mismo mensaje independiente de cual modulo es el encargado de realizar dicha acción.

La asignación de direcciones físicas para los nodos en una red está especificada en esta misma tabla 5.7.3. Es importante recalcar que cada asignación de dirección de nodo en cualquier red debe ser única. En otras palabras, dos nodos y por lo tanto dos ECUs no pueden tener la misma asignación de dirección física.

Node Category	Address (hex)	# of Nodes
Powertrain Controllers:	00 - 1F	32
Integration/Manufacturer Expansion	00 - 0F	16
Engine Controllers	10 - 17	8
Transmission Controllers	18 - 1F	8
Chassis Controllers:	20 - 3F	32
Integration/Manufacturer Expansion	20 - 27	8
Brake Controllers	28 - 2F	8
Steering Controllers	30 - 37	8
Suspension Controllers	38 - 3F	8
Body Controllers:	40 - C7	136
Integration/Manufacturer Expansion	40 - 57	24
Restraints	58 - 5F	8
Driver Information/Displays	60 - 6F	16
Lighting	70 - 7F	16
Entertainment/Audio	80 - 8F	16
Personal Communications	90 - 97	8
Climate Control (HVAC)	98 - 9F	8
Convenience (Doors, Seats, Windows, etc.)	A0 - BF	32
Security	C0 - C7	8

Tabla 5.7.3 - SAE J2178 – Direccionamiento Funcional, Ejemplo

5.8 El estándar SAE J2178-2 –Parámetros y Direccionamiento Funcional

La parte 2 de este estándar define los datos paramétricos estándar que pueden intercambiarse en estas redes OBDII. De estos parámetros, se definen el escalamiento, rangos y funciones de transferencia. Los mensajes que se refieren a estas definiciones de parámetros deben apegarse a estas definiciones. La intención de este estándar J2178-2 es que los primeros parámetros de la lista corresponden con los PIDs definidos en J1979.

Los parámetros usados para describir variables de datos son la parte importante de esta sección. Para lograr un entendimiento de mensajes entre nodos, los parámetros de datos de los bytes enviados y recibidos son especificados. Al usar definiciones

comunes en la red para mensajes de diagnóstico y no diagnóstico, los modules que forman la red pueden mantener una misma descripción de cada parámetro.

La definición de cada parámetro consiste de dos partes, el PRN y el SLOT. El PRN o Número de Parámetro de Referencia (del inglés Parameter Reference Data) es un número usado para identificar un parámetro específico por nombre, unidad de medida, y su asociado SLOT. El SLOT (del inglés Scaling, Limits, Offset, Trasfer function) define las características matemáticas de los parámetros en términos de su representación (Binario, Numérico sin signo, ASCII, BCD, etc). Su escalamiento (1 bit = nn, 1 byte = mmm), sus límites, su offset, y su función de transferencia.

Los PRNs se utilizan dentro de este estándar para facilitar la documentación. Por si solo el número no tiene mayor significado, salvo los primeros 256 de ser equivalentes a los PIDs del J1979. Todos los PRNs son de dos bytes de longitud, siendo el primero para identificar el grupo o clasificación. El segundo byte es un consecutivo sin ningún orden en particular. Los primeros son idénticos al J1979, por lo que el primer byte para este caso es 0x00. Del primer byte, los tres primeros bits (MSB) determinan el dominio al cual se hace referencia (chasis, body, etc). Los siguientes dos son el grupo de sistemas del dominio seleccionado. Los últimos tres (LSB) son 000 para SAE, y otras combinaciones son propietarias para cada fabricante. Los dos bytes del PRN quedan como en la figura 5.8.1

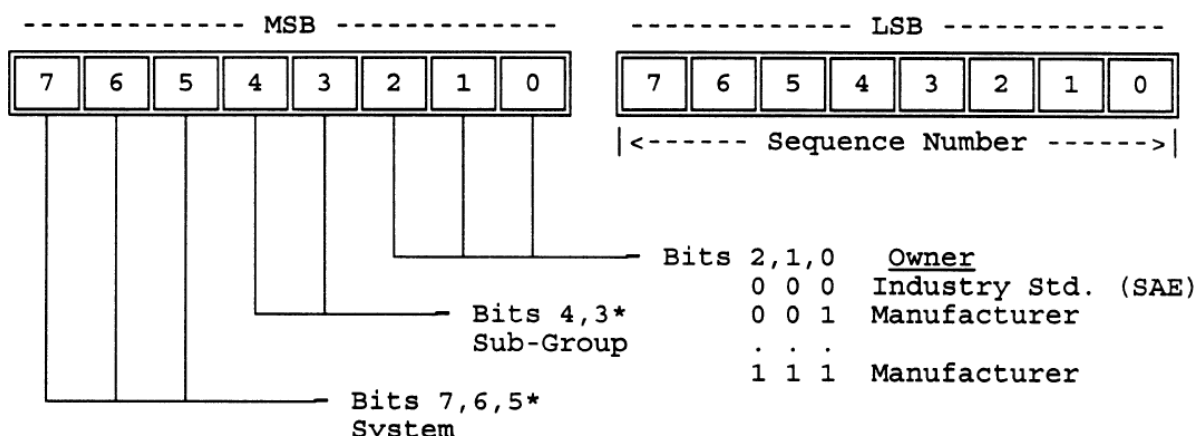


Figura 5.8.1 – Bytes PRN

7 6 5	System	4 3	Subgroup	Address (Hex)
0 0 0	Powertrain	0 0	SAE J1979 Compatible	0000 - 00FF
		0 0	Reserved	0100 - 07FF
		0 1	Reserved	0800 - 0FFF
		1 0	Engine	1000 - 17FF
		1 1	Transmission	1800 - 1FFF
0 0 1	Chassis	0 0	Reserved	2000 - 27FF
		0 1	Brakes/Tires/Wheels	2800 - 2FFF
		1 0	Steering	3000 - 37FF
		1 1	Suspension	3800 - 3FFF
0 1 0	Body 1	0 0	Reserved	4000 - 47FF
		0 1	Reserved	4800 - 4FFF
		1 0	Reserved	5000 - 57FF
		1 1	Restraints	5800 - 5FFF
0 1 1	Body 2	0 0	Driver Info.	6000 - 67FF
		0 1	Reserved	6800 - 6FFF
		1 0	Lighting	7000 - 77FF
		1 1	Reserved	7800 - 7FFF
1 0 0	Body 3	0 0	Audio	8000 - 87FF
		0 1	Reserved	8800 - 8FFF
		1 0	Pers. Comm.	9000 - 97FF
		1 1	HVAC	9800 - 9FFF
1 0 1	Body 4	0 0	Convenience	A000 - A7FF
		0 1	Reserved	A800 - AFFF
		1 0	Reserved	B000 - B7FF
		1 1	Reserved	B800 - BFFF
1 1 0	Other 1	0 0	Security	C000 - C7FF
		0 1	EV-ETS	C800 - CFFF
		1 0	Reserved	D000 - D7FF
		1 1	Reserved	D800 - DFFF
1 1 1	Other 2	0 0	Config. Codes	E000 - E7FF
		0 1	Reserved	E800 - EFFF
		1 0	Tester/Diag.	F000 - F7FF
		1 1	Miscellaneous	F800 - FFFF

Figura 5.8.2 – Grupos PRN y rangos de direcciones

Las definiciones de los SLOT o Escala, Límites, Offset y función transferencia fueron agrupadas para evitar duplicidad entre los PRNs. Estos fueron definidos ampliamente en el J2178-2. La función de transferencia define la relación entre unidades de computadora (N) en decimal, y unidades de ingeniería (E) de los datos.

Los SLOTS fueron agrupados en categorías, identificados por 3 letras

- PKT – Paquete de datos múltiples paramétricos. Envía varios parámetros en paquete, por decir el VIN del vehículo o fecha/hora.
- BMP – Mapeo de bits sin máscara. Estos se utilizan para codificar datos que típicamente contienen varios datos binarios como son bits o banderas. Pueden utilizarse para salidas discretas como las diversas luces donde cada una es un bit. Se limita a 4 bytes sin mascara.
- BMM – Mapeo de bits con mascara. Similar al BMP, pero incluye hasta 2 de los 4 bytes con una máscara de bits para indicar cuales son válidos.
- UNM – Numérico sin signo. Estos son usados para codificar datos que son típicamente asociados con información como temperatura, velocidad y porcentaje. Estos pueden ser de 8, 16, 24,...,56 bits (1 a 7 bytes) y pueden o no tener offset. Cada SLOT de UNM contiene también una definición de resolución por bit, mínimo y máximo valor, y función de transferencia para relacionar las unidades de computadora (N - valor en bytes) con el valor de ingeniería (E – valor del parámetro físico real que representa)
- SNM – Numérico con signo en complemento de dos. Estos son representados en complemento de dos. Si el MSB bit es uno, entonces el número es negativo y el valor absoluto se encuentra haciendo el complemento a dos es decir invirtiendo cada bit y después sumando un uno binario (1).
- SED – Estado Codificado. Los SLOT SED son usados para datos que pueden obtener uno de diferentes estados como lo son los días de la semana o el estado de los limpiaparabrisas. Cada definición de SLOT contiene un campo para describir cada estado posible. Puede tener entre 1 a 8 bits con sus combinaciones.
- ASC – Codificado ASCII. Se envían hasta 7 caracteres en codificación ASCII estándar formando un string
- BCD – Codificado binario decimal. Utiliza codificación BCD de manera similar al ASCII. Va desde un dígito con 4 bits, hasta 14 BCD dígitos con 56 bits.

- SFP – Punto flotante con signo. Se utiliza cuando es necesario manejar datos en punto flotante aritmético con un carácter por delante con el signo. Este sigue el IEEE std 754-1985 con 32 bits o 4 bytes.
- Múltiples mensajes – Codifica información extensa en varios mensajes

Una lista abreviada de PRNs y SLOTS del J2178 se incluye en la sección de apéndices.

5.9 El estándar SAE J2178-3 – Estructura para headers de un byte

Esta sección define los mensajes del header de un byte y del consolidado, la versión de un byte también. Esta sección codifica en un solo byte tipo de header/mensaje, direcciones de fuente y destino. Para los datos maneja el PRN y respuesta en el mensaje IFR. Tiene tablas definidas para las combinaciones

La ventaja es su reducida longitud en bits. Para sistemas con bajo baud rate esto significa una mayor velocidad en los mensajes. Sin embargo, esta misma reducida cantidad de bits lo hace muy limitado en la capacidad de diversidad de mensajes

El presente trabajo está orientado a una aplicación en CAN BUS con un relativo alto baud rate, por lo que esta modalidad de un byte representa más desventajas que ventajas y se opta por un header consolidado de 3 bytes como en la sección J2178-4

5.10 El estándar SAE J2178-4 – Definición para headers de 3 bytes

La sección 4 del SAE J2178 define la asignación de mensajes para la forma de tres bytes del header en su forma consolidada. Contiene 51 tablas de mensajes y su estructura de cómo comunicarse con diversos sistemas como las ruedas, el a/c, los asientos, etc

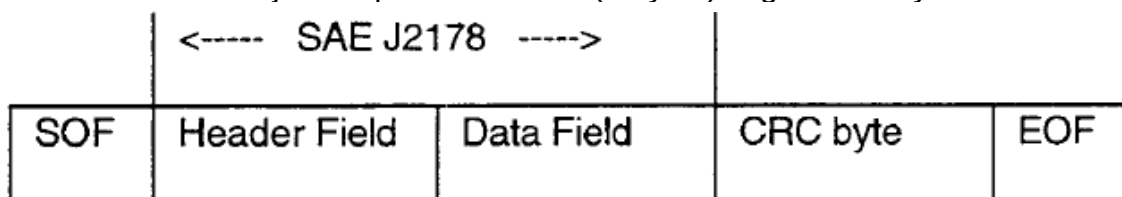
En breve, el primero de los tres bytes define la prioridad del mensaje, tipo de header (uno o tres bytes), respuesta en mensaje requerida IFR, direccionamiento funcional o físico, y tipo de mensaje (broadcast, comando, estatus, entre otros). El segundo byte contiene la dirección de destino. Esta puede ser funcional o dirigida a cualesquier modulo que maneje la función requerida tal como el estado de un sensor determinado,

o física siendo dirigido a un módulo en específico. En el segundo byte, el mensaje requerido (comando) y la respuesta están desplazados por un bit consecutivo. Por ejemplo, en J1850 la dirección en el segundo byte para requerir un mensaje de diagnóstico es 6A, mientras que la respuesta se utiliza 6B como destino. El tercer byte es el remitente o la dirección de quien envía el mensaje. Para este byte se utiliza siempre direccionamiento físico. Para equipo de diagnóstico externo esta es siempre F1. Cuando un ECU responde a un equipo externo, poner un F1 en el segundo byte como destino, y coloca su propia dirección física asignada en el tercer byte como remitente.

El estándar SAE J2178, como el mencionado J2190 fue creado para atender funcionalidades extendidas en los sistemas OBD de los 90's. Sin embargo, este estándar estuvo también adelantado a su tiempo al direccionar sistemas que no existían en la tecnología comercial de su tiempo, y fue removido más tarde. Parte de la sección 1 y de la 2 a la 4 especificaron un direccionamiento para mensajes no de emisiones usando una estructura nacida y optimizada para mensajes de diagnóstico de emisiones. Este estándar no fue cancelado dado que integró los headers especificados en los diferentes estándares en un solo esquema en su sección primera, convirtiéndose en la referencia de facto para la definición del header en OBD.

El estándar SAE J2178 contempla diversas estructuras de mensajes y modos de direccionamientos, de los cuales el contemplado en esta sección 4 para headers de 3 bytes es el más completo. La sección cuarta establece 45 tablas (de la 6 a la 51 en el estándar) con mensajes funcionales para todos los dominios del vehículo. Los mensajes están organizados en grupos definidos con una dirección ID primaria. Este ID primario o PID tiene dos modos: ID de comando e ID de estado dependiendo si fue un requerimiento (comando) o una respuesta a un comando (estado). Los mismos están desplazados por un bit en el LSB (bit W). En cada grupo de PIDs, hay una dirección secundaria ID de 6 bits para una función particular en dicho grupo. Por ejemplo, para un tipo particular de sensores referenciado por su ID primario, el ID secundario puede indicar el número de sensor por decir el 1 o el 2 o algún otro.

Mensaje Completo – Header (3 bytes) seguido de Bytes de datos



Header - Forma consolidada del header de tres bytes

Byte 1	Byte 2	Byte 3
See Figure 8	Target Address	Source Address

Byte 1 del header consolidado de tres bytes

Bit 7	6	5	4	3	2	1	0
P	P	P	0	K	Y	Z	Z
Priority (0 to 7)			H=0	Message Type (see below)			
Bit	Definition			Value	Meaning		
K	In-Frame Response (IFR)			0	IFR Required		
				1	IFR Not Allowed		
Y	Addressing Mode			0	Functional Addressing		
				1	Physical Addressing		
ZZ	Specific Message Type			0 0	See Table 6		
				0 1			
				1 0			
				1 1			

Byte 2 del header - Dirección funcional destino

Bit 7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	W
Primary ID (128 pairs)							

Bytes de datos

...	Secondary Address	Data 1	...	Data n	...
-----	-------------------	--------	-----	--------	-----

Primer byte de datos – dirección ID secundaria

Bit 7	6	5	4	3	2	1	0
Q	C	Secondary ID (6 bits)					

Figura 5.10.1 – Estructura de mensaje OBD (Fuente: SAE J2178-1)

Los otros dos bits que acompañan en el primer byte de datos a la dirección secundaria ID de 6 bits son el bit Q (MSB – bit 7) y el bit C (bit 6). El bit Q codifica una acción de set/reset, abrir/cerrar o lo que corresponda a la función direccionada por el grupo de dirección primaria ID. El bit C junto con el último bit del header (bit W) define si es un mensaje con un comando o es usado para regresar un estado. Estas combinaciones

están definidas en el J2178 sección primera. En algunos casos el segundo byte de datos se utiliza como un tercer byte de dirección, el cual es la dirección geográfica dentro del vehículo. Por ejemplo, el comando para prender una luz debe incluir cual luz es la que se está referenciando (delantera, derecha, adentro, o similar)

Engine Air Intake—(See Table 7.)

Primary ID: Engine Air Intake
 Command ID: \$0A
 Status ID: \$0B

TABLE 7—ENGINE AIR INTAKE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Boost On/Off	01	L / R	On	Off	--	--
Boost Value	02	R	--	x	--	1021
Mass Air Flow Rate	08	R	--	x	--	0010
Manifold Absolute Pressure	11	R	--	x	--	000B
Intake Air Temperature	20	R	--	x	--	000F

Figura 5.10.2 – Ejemplo de tabla de mensajes (Fuente: SAE J2178-4)

5.11 Propuesta de PIDs no – estándar sobre CAN BUS – el OBDII Plus

La tecnología actual OBDII está definida por diferentes estándares SAE e ISO desde capas físicas hasta capas de red con respecto a niveles de voltaje, manera de codificar bits y bytes y estructura de los mensajes. Sin embargo, los mismos están enfocados casi completamente a sistemas de emisiones basados en los dominios de tracción y chasis, tal como se inició en los 90's. Los vehículos comunes modernos de bajo y mediano rango incorporan ahora como equipo estándar de fábrica diversos sistemas periféricos como ventanas eléctricas y alarma. Los fabricantes de vehículos a partir del 2008 implementaron a través de CAN BUS protocolos propietarios para manejar la señalización entre estos nuevos sistemas. Los formatos de los mensajes y las tablas en si fueron desarrollados por cada fabricante como propietarias. Las organizaciones SAE e ISO solo definen a través de J2284-3 e ISO11898 los requerimientos de la capa física para CAN de manera genérica. Otros estándares como ISO 15765 maneja la comunicación entre vehículo y equipo de prueba y diagnóstico, pero no entre sistemas dentro del vehículo.

La propuesta OBDII plus es respecto a estandarizar en un set de mensajes comunes los mensajes relacionados a sistemas no de emisiones, expandiendo el actual sistema OBDII. El impacto general es desarrollar una plataforma de hardware estándar común para sistemas del dominio del Body que permitan una implementación sencilla de aplicaciones de más alto nivel.

En CAN BUS, el header para OBDII según ISO 15765-4 tiene dos formatos: 11 bits y 29 bits. Estos son parecidos a las formas de uno y tres bytes del J2178. Para la forma de 29 bits, este contempla una primera sección la cual codifica el formato del mensaje, además de un par de direcciones de destino y fuente. En suma, para comunicarse con un equipo de prueba usando direccionamiento funcional el header queda 18 DB 33 F1. Si se utiliza direccionamiento físico, un bit cambia así como las direcciones de destino y fuente. Para un mensaje de requerimiento o comando el mensaje es 18 DA nn F1 y su respuesta o estatus es 18 DA F1 nn, donde nn es la dirección del ECU con el cual se está comunicando.

CAN Id bit position	28 24	23 16	15 8	7 0
Functional CAN Id	18 hex	DB hex	TA	SA
Physical CAN Id	18 hex	DA hex	TA	SA
NOTE The CAN identifier values given in this table use the default value for the priority information in accordance with ISO 15765-2.				

CAN identifier	Target Address (TA)	Source Address (SA)	TA type (TAtype)	Message type (Mtype)
Functional request	Legislated OBD system = 33 hex	External test equipment = F1 hex	functional	diagnostics
Physical response	External test equipment = F1 hex	Legislated-OBD ECU = xx hex	physical	diagnostics
Physical request	Legislated OBD ECU = xx hex	External test equipment = F1 hex	physical	diagnostics
xx hex ECU physical diagnostic address				
NOTE For detailed descriptions of parameters TA, SA, TAtype and Mtype, see ISO 15765-2.				

Figura 5.11.1 – formato de header para CAN BUS (Fuente: ISO 15765-4)

5.11.1 Reglas para el OBDII Plus

La manera de implementar la propuesta de OBDII plus utilizando J2178-4 sobre CAN BUS utilizando el header de 29 bits es de la siguiente manera:

- Los primeros bits se utilizan en la misma manera que el OBDII para direccionamiento funcional. Estos por lo tanto quedan fijos en 18 DB

- Para la dirección destino usar en el SAE J2178-4 de las tablas 6 a la 51 el ID primario de comando
- La dirección de la fuente es la dirección física asignada al módulo ECU que emite el mensaje. EL único reservado es F1 para equipo externo. Para otros módulos se puede seguir la recomendación establecida en el estándar SAE J2178-1 en la tabla 11

TABLE 11—NODE PHYSICAL ADDRESS ASSIGNMENTS

Node Category	Address (hex)	# of Nodes
Powertrain Controllers:	00 - 1F	32
Integration/Manufacturer Expansion	00 - 0F	16
Engine Controllers	10 - 17	8
Transmission Controllers	18 - 1F	8
Chassis Controllers:	20 - 3F	32
Integration/Manufacturer Expansion	20 - 27	8
Brake Controllers	28 - 2F	8
Steering Controllers	30 - 37	8
Suspension Controllers	38 - 3F	8
Body Controllers:	40 - C7	136
Integration/Manufacturer Expansion	40 - 57	24
Restraints	58 - 5F	8
Driver Information/Displays	60 - 6F	16
Lighting	70 - 7F	16
Entertainment/Audio	80 - 8F	16
Personal Communications	90 - 97	8
Climate Control (HVAC)	98 - 9F	8
Convenience (Doors, Seats, Windows, etc.)	A0 - BF	32
Security	C0 - C7	8
Electric Vehicle Energy Transfer System (EV-ETS)	C8 - CB	4
Utility Connection Services (FN#1)	C8	1
AC to AC Conversion (FN#2)	C9	1
AC to DC Conversion (FN#3)	CA	1
Energy Storage Management (FN#4)	CB	1
Future Expansion:	CC - CF	4
Manufacturer Specific:	D0 - EF	32
Off-Board Testers/Diagnostic Tools:	F0 - FD	14
All Nodes:	FE	1
Null Node:	FF	1

Figura 5.11.1.1 – Tabla 11 SAE J2178-1 (Fuente: SAE J2178)

- Para el caso del presente trabajo, se pretende utilizar direccionamiento funcional en los sistemas del body. Por lo tanto siguiendo la misma tabla se recomienda utilizar la llamada general a todos los nodos 0xFE. Cada nodo dependiendo del mensaje (la función requerida en el mismo) decide si lo atiende o lo ignora.
- El direccionamiento Funcional planteado contempla el utilizar un byte adicional como dirección extendida, para los casos donde se requiera saber físicamente cual función debe responder. Ejemplo, para reportar un foco fundido, la tabla 9.8 del J2178-4 establece dicha dirección extendida para un foco en determinada localización física.
- Solamente en los casos donde haga valor agregado un direccionamiento físico se puede seguir lineamiento descrito abajo, utilizando dirección de las manecillas de un reloj:
 - Delantero Derecha (Conductor) 0xn1,
 - Delantero Izquierda (Pasajero) 0xn2
 - Trasero Derecha 0xn3
 - Trasero Izquierda 0xn4
 - Central o alguna posición no especifica geográfica 0xn0
- Un ejemplo del anterior es al reportar el estado de un nodo de luces delanteras del lado del pasajero. Por la tabla 11 anterior, el rango de direcciones de nodos para las luces es 0x70 – 0x7F. El nodo para este ejemplo sería entonces 0x72. El ID primario para estado de red es 0xFE para comando y 0xFF para estado. Por lo tanto, para reportar el estado de red (0xFF) dicho nodo (0x72) que está OK (C=1, 0x8) respecto a Nodo vivo (0x4) el string quedaría 18DB FF 72 84
- Cuando se conteste un mensaje, la dirección destino es el ID primario de estatus, el cual está desplazado un bit al ID primario de comando.
- El primer byte de datos está compuesto de los bits Q y C, y la dirección secundaria ID de 6 bits.
- El bit Q está definido en las mismas tablas primarias de ID del J2178-4 y es dependiente de la función utilizada. En general dicho bit es “1” Para verdadero, habilitado, etc y “0” para falso, deshabilitado, etc.

TABLE 5—Q BIT DEFINITIONS

Q bit = "1"	Q bit = "0"	Meaning
--	x	Q bit not used, Defaults to logic "0"
x	–	Q bit not used, Defaults to logic "1"
Set	–	Q bit must be logic "1" and means Set
E	D	Enable(d) / Disable(d)
Inc	Dec	Increment / Decrement
O	C	Open(ed) / Close(d)
L	U	Lock(ed) / Unlock(ed)
On	Off	On / Off
R	–R	Reset / Not Reset
Y	N	Yes / No

Figura 5.11.1.2 – Bit Q (Fuente: SAE J2178-4)

- El bit C queda fijo a un valor 0. Cuando el bit C es combinado con el bit W del header este representa las acciones de cargar un valor (reemplazar) o leer un estado, el cual es conveniente para esta aplicación. Recordando el bit W es el LSB y causa el efecto del desplazamiento en uno entre ID comando e ID estatus. Otras combinaciones de W con C son valores para incrementar / decrementar registros, los cuales complican esta propuesta por lo que no se les ve necesarios.

TABLE 7—W BIT DEFINITION

W Bit	Functional Target Address Type
0	Command
1	Status

Figura 5.11.4 – Bit W (Fuente: SAE J2178-1)

TABLE 4—MESSAGE OPERATIONS

Operation	Msg Op	Primary ID	C-bit (bit 6)
Report Data	R	Status	0
Load Data	L	Command	0
Modify State	M	Command	1

Figura 5.11.1.3 – Bit C (Fuente: SAE J2178-4)

- En caso de que sea requerido direccionamiento extendido o geográfico, este es enviado como el segundo byte de datos. Estos son definidos en el J2178-4 y son referenciados en cada tabla de ID primario.

- El siguiente byte de datos después del direccionamiento (segundo o tercer byte según si incluye direccionamiento extendido) es el primer byte de datos paramétricos. Por ejemplo, los RPMs son retornados en este byte.
- La decodificación de los bytes de datos enviados a parámetros reales, o función de transferencia es realizada utilizando los PRN o Número de Parámetro de Referencia. Este es un ID de referencia para aquellas funciones que requieren otro byte como cuando se regresa la temperatura.
- No todas las funciones requieren regresar un dato. Por ejemplo el ID del Odómetro al requerírselo puede retornar el kilometraje. O simplemente requerir resetear el mismo a cero para lo cual no es necesario retornar valor alguno. Para eventos como el anterior de set/reset o similares comandos solamente se necesita el bit Q y no es necesario un byte de dato adicional.

Engine Air Intake—(See Table 7.)

Primary ID: Engine Air Intake
 Command ID: \$0A
 Status ID: \$0B

TABLE 7—ENGINE AIR INTAKE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Boost On/Off	01	L / R	On	Off	--	--
Boost Value	02	R	--	x	--	1021
Mass Air Flow Rate	08	R	--	x	--	0010
Manifold Absolute Pressure	11	R	--	x	--	000B
Intake Air Temperature	20	R	--	x	--	000F

**TABLE 4—SAE J1979 COMPATIBLE PRN ASSIGNMENTS
(PRNS 0000 - 00FF)**

PRN	Parameter Name	Resolution (1 Bit =)	Units of Measure	SLOT # (F-N-#)
0000	PIDs Supported (01h - 20h)	—	Bit Mapped	BMP-32-1
000F	Intake Air Temperature	1	Degrees Centigrade	UNM-08-102

TABLE 24—UNM-08, 8 Bit SLOTS

SLOT #	Scaling / Resolution (1 Bit =)	Minimum Limit	Maximum Limit	Invalid Range	Transfer Function N =	Transfer Function E =	Comment
1	1/8000	0	0.031875	—	E * 8000	N / 8000	=0.000125
102	1	-40	+215	—	E + 40	N - 40	1 w/ Offset

Figura 5.11.14– Ejemplo de mensaje PID, PRN, SLOT (Fuente: SAE J2178-2, J2178-4)

- El número de PRN y las funciones de transferencia están definidos en J2178-2. Los primeros 256 códigos (\$00 a \$FF) son los mismos de los ya conocidos SAE J1979 PIDs. Por ejemplo, la temperatura de admisión es referenciada usando el ID primario \$0A, ID secundario \$20, y PRN \$00 0F. El PID J1979 en modo 1 es \$01 0F
- Para codificar una respuesta, o leer la misma, el J2178-2 especifica para cada PRN de qué manera se realiza la codificación. Siguiendo el mismo ejemplo para la temperatura del aire de admisión, el PRN \$00 0F hace referencia a un SLOT UNM-08-102. La tabla correspondiente a este SLOT UNM-08-102 indica que es un número sin signo (del inglés Unsigned Number) de 8 bits, donde el renglón 102 indica que al valor del byte (N) se le deben restar - 40 para tener el valor ingenieril (E) correspondiente a la temperatura en grados centígrados.

5.11.2 Estructura del mensaje de OBDII Plus

A manera de sumario de la propuesta, se presenta la siguiente tabla con la estructura del mensaje del OBDII-Plus

HEADER – 29 bits			3 Bytes de datos (1 datos + 2 si se requieren)				
Header	Target Address	Source Address	Bit 7 “Q”	Bit 6 “C”	Bit 5-0	Dirección extendida	Byte de datos
0x18DB	1 Byte 0xNN	0xFE	1/0	0	0x0N	1 Byte 0xNN	1 Byte 0xNN
Header - Fijo	ID Primario por J2178-4	Dirección física – 0xFE llamada a todos los nodos	1- Activo 0-Inactivo	Fijo en 0	ID secundario Por J2178-4	Direccionamiento extendido (Default 0x00 todos)	Dato paramétrico (Si se requiere)

Tabla 5.11.2.1 Estructura del mensaje en OBDII-plus (Fuente: Elaboración propia)

5.11.3 Ejemplos para el OBDII Plus

El presente trabajo está orientado a los sistemas automotrices del body. Por lo tanto, se incluyen algunos ejemplos relacionados a dichos sistemas.

El primer ejemplo mostrado es la posición de la llave de encendido. La tabla 28 SAE J2178-4 lo indica solo para reportar estado (R). El ID primario es 0x86 para comando y 0x87 para estado. Por lo tanto, para preguntar el estado (0x86) dirigiéndose a ningún nodo en particular (0xFE) si está en un estado activo (C=1, 0x8) de la llave (0x05) el string quedaría 18DB 86 FE 85

La respuesta o estado actual (0x87) dirigiéndose a ningún nodo en particular (0xFE) suponiendo un estado activo (C=1, 0x8) de la llave (0x05) el string quedaría 18DB 87 FE 85

8.23 Ignition Switch/Starter—(See Table 28.)

Primary ID: Ignition Switch / Starter
Command ID: \$86
Status ID: \$87

TABLE 28—IGNITION SWITCH/STARTER MESSAGES

Secondary ID Name			Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Ignition Switch Position			04	R	--	x	--	1047
Key-In-Ignition			05	R	Y	N	--	--
102	1	-40	+215	--	E + 40	N - 40	1 w/ Offset	

Figura 5.11.3.1 – Ejemplo de mensaje (Fuente: SAE J2178-4)

Ejemplo 2. Para apagar la luz MIL (Check engine) se utiliza la tabla 29 SAE J2178-4, la cual mapea las luces. Este ID permite cargar valores (L-Load) o leer (R-Read). El ID primario es 0x88 para comando y 0x89 para estado. Por lo tanto, para enviar el comando (0x88) dirigiéndose a ningún nodo en particular (0xFE) de pasar a estado inactivo (C=0, 0x0) el check engine (0x03) el string quedaría 18DB 88 FE 03

8.24 Tell Tales—(See Table 29.)

Primary ID: Tell Tales
 Command ID: \$88
 Status ID: \$89

TABLE 29—TELL TALES MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Seatbelt	01	L/R	On	Off	--	--
Service Engine Soon	02	L/R	On	Off	--	--
Check Engine (MIL)	03	L/R	On	Off	--	--
High Beam Indicator	04	L/R	On	Off	--	--
Left Turn Indicator	05	L/R	On	Off	--	--
Right Turn Indicator	06	L/R	On	Off	--	--
Airbag	07	L/R	On	Off	--	--
Anti-Lock Brake System Failed	08	L/R	On	Off	--	--
Traction Control System Failed	09	L/R	On	Off	--	--
Security	0A	L/R	On	Off	--	--
Low Fuel	0B	L/R	On	Off	--	--
Low Coolant	0C	L/R	On	Off	--	--
Low Oil	0D	L/R	On	Off	--	--
Low Voltage	0E	L/R	On	Off	--	--
Upshift	0F	L/R	On	Off	--	--
Low Washer Fluid	10	L/R	On	Off	--	--
Traction Control Active	11	L/R	On	Off	--	--
Alternator Failure	12	L/R	On	Off	--	--
Low Brake Fluid	13	L/R	On	Off	--	--
Overdrive	14	L/R	On	Off	--	--
Traction Control Disabled	15	L/R	On	Off	--	--
Convertible Latch Warning	21	L/R	On	Off	--	--
Super Lock System Warning	22	L/R	On	Off	--	--
Catalyst Overtemperature	23	L/R	On	Off	--	--
Vehicle Speed Control	24	L/R	On	Off	--	--

Figura 5.11.3.2 – Ejemplo de mensaje (Fuente: SAE J2178-4)

Ejemplo 3. En la mayoría de los casos, los sistemas del Body son tan diversos y numerosos que requieren un direccionamiento extendido adicional. Este se implementa por medio de agregar un byte al string. Para el caso de prender las luces delanteras del coche (Headlamps) se utiliza la tabla 41 SAE J2178-4. El ID primario es 0xDA para comando y 0xDB para estado. Esta tabla además llama a la 9.8 (tabla 59) la cual especifica físicamente cual luz deseamos. Por lo tanto, para enviar el comando (0xDA) dirigiéndose a ningún nodo en particular (0xFE) de pasar a estado activo (C=1, 0x8) el headlamps (0x01) agregando dirigirse a todas las lámparas (0x00) el string quedaría 18DB DA FE 81 00. Igualmente pudiéramos habernos dirigido solamente a las lámparas de enfrente utilizando 18DB DA FE 81 08. Dado que no hay (normalmente) luces de este tipo en la parte de atrás se tendría el mismo efecto.

8.36 Exterior Lamps—(See Table 41.)

Primary ID: Exterior Lamps
 Command ID: \$DA
 Status ID: \$DB

TABLE 41—EXTERIOR LAMPS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Headlamp On/Off	01	L/R	On	Off	8.8	--
Tail Lamp On/Off	02	L/R	On	Off	8.8	--
Brake Lamp On/Off	03	L/R	On	Off	8.8	--
Park Lamp On/Off	04	L/R	On	Off	8.8	--
Turn Lamp On/Off	05	L/R	On	Off	8.8	--
High Beam Lamp On/Off	06	L/R	On	Off	8.8	--
Hazard Lamp On/Off	07	L/R	On	Off	8.8	--
Reverse Lamp On/Off	08	L/R	On	Off	8.8	--
Fog Lamp On/Off	09	L/R	On	Off	8.8	--
Daytime Running Lamp On/Off	0A	L/R	On	Off	8.8	--
Spot Lamp On/Off	0B	L/R	On	Off	8.8	--
Cargo Lamp On/Off	0C	L/R	On	Off	8.8	--
Cornering Lamp On/Off	0D	L/R	On	Off	8.8	--
Driving Lamp On/Off	0E	L/R	On	Off	8.8	--
Coach Lamp On/Off	0F	L/R	On	Off	8.8	--
Autolamp Delay	10	L/R	E	D	8.8	A014
Flash-to-Pass	11	L/R	E	D	8.8	--
Remote Headlamp On/Off w/Transmitter ID	12	L/R	On	Off	8.8	C001
Remote Headlamp On/Off	13	L/R	On	Off	8.8	--
Headlamp Sw. Active	21	R	Y	N	8.8	--
Right Turn Sw. Active	22	R	Y	N	8.8	--
Park Lamp Sw. Active	24	R	Y	N	8.8	--
Left Turn Sw. Active	25	R	Y	N	8.8	--
High Beam Sw. Active	26	R	Y	N	8.8	--
Hazard Sw. Active	27	R	Y	N	8.8	--
Fog Lamp Sw. Active	28	R	Y	N	8.8	--
Driving Lamp Sw. Active	29	R	Y	N	8.8	--

9.8 External Lamps—The extended address assignments for external lamps are in Table 59.

TABLE 59—EXTENDED ADDRESS ASSIGNMENTS FOR EXTERNAL LAMPS

Address	Description
00	ALL
01	Left Side (Turn Signal)
07	Right Side (Turn Signal)
08	All Front
09	Left Front
0F	Right Front
38	All Rear
39	Left Rear
3C	CHMSL
3F	Right Rear

Figura 5.11.3.3 – Ejemplo de mensaje (Fuente: SAE J2178-4)

Ejemplo 4. Para los sistemas del motor y transmisión (powertrain) es común reportar un byte adicional con el valor del parámetro en cuestión. En los sistemas del body son contados los ID que manejan un valor numérico que no sea binario lógico 1/0, activo/desactivo, etc. Con el fin de ilustrar un ejemplo se muestra la manera de reportar la posición de la ventana. De la tabla 36 SAE J2178-4, el ID para comando es 0xCA y para estado es 0xCB. El ID secundario para la posición es 0x1A. La dirección extendida para identificar cual ventana está dado por la sección 9.7 tabla 58 SAE J2178-4. Para

interpretar el byte adicional con el dato de la posición de la ventana, la misma tabla 36 nos indica un PRN A00F. Los PRN (Parameter Reference Number) están descritos en el estándar SAE J2178-2. En este caso, requiere un PRN A00F, el cual especifica que el byte es simplemente el valor que entrega el A/D encargado de determinar la posición. Para efectos de este trabajo, establecemos un valor 0x00 para ventana abierta hasta abajo, y un byte con su valor máximo 0xFF para el caso contrario de ventana hasta arriba. Los 255 casos posibles intermedios son proporcionales.

Por lo tanto, para enviar el estatus (0xCB) dirigiéndose a ningún nodo en particular (0xFE), donde el bit C es cualquier valor, el ID secundario para posición de ventanas (0x1A) suponiendo que estamos hablando de la ventana del conductor (0x22), y que está cerrada hasta arriba (0xFF para este ejemplo), el string quedaría 18DB CB FE 1A 22 FF

8.31 Windows—(See Table 36.)

Primary ID: Windows
Command ID: \$CA
Status ID: \$CB

TABLE 36—WINDOWS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Open	01	L/R	E	D	8.7	--
Close	02	L/R	E	D	8.7	--
Operation Enable	03	L/R	E	D	8.7	--
Master Controller Open	04	L/R	E	D	8.7	--
Master Controller Close	05	L/R	E	D	8.7	--
Position	1A	L/R	--	x	8.7	A00F
Open Sw. Active	21	R	Y	N	8.7	--
Close Sw. Active	22	R	Y	N	8.7	--
Lockout Sw. Active	23	R	Y	N	8.7	--
Master Controller Open Sw. Active	24	R	Y	N	8.7	--
Master Controller Close Sw. Active	25	R	Y	N	8.7	--

Figura 5.11.3.4 – Ejemplo de mensaje (Fuente: SAE J2178-4)

9.7 Windows—The extended address assignments for windows are in Table 58.

TABLE 58—EXTENDED ADDRESS ASSIGNMENTS FOR WINDOWS

Address	Description
00	ALL
20	All Front
22	Driver Side Front
24	Front Sun Roof
26	Passenger Side Front
28	All Rear
2A	Driver Side Rear
2C	Rear Sun Roof
2E	Passenger Side Rear
34	Rear Windshield

Figura 5.11.3.5 – Ejemplo de mensaje (Fuente: SAE J2178-4)

**TABLE 16—CONVENIENCE PRN ASSIGNMENTS
(PRNS A000 - A7FF)**

PRN	Parameter	Resolution (1 Bit =)	Units of Measure	SLOT # (F-N-#)
A001	Seat Temperature	1/2	Degrees Centigrade	UNM-08-73
A003	Wiper Mode	—	State Encoded	SED-08-1
A004	Wiper Delay	1/4	seconds	UNM-08-51
A006	Washer Fluid Temperature	1	Degrees Centigrade	UNM-08-102
A007	Washer Fluid Pressure	4	kPaG	UNM-08-141
A008	Washer Fluid Level - Percent	100/255	% Full	UNM-08-61
A009	Washer Fluid Level - Volume	1/10	liters	UNM-08-41
A00A	Washer Fluid Capacity	1/10	liters	UNM-08-41
A00C	Mirror Dimming Level	100/255	% Full Dim	UNM-08-61
A00D	Mirror Horizontal Position	1	Raw A/D Counts	UNM-08-101
A00E	Mirror Vertical Position	1	Raw A/D Counts	UNM-08-101
A00F	Window Position	1	Raw A/D Counts	UNM-08-101

Figura 5.11.3.6 – Ejemplo de mensaje (Fuente: SAE J2178-2)

Ejemplo 5. En cualquiera de los casos anteriores donde se reporta un estatus, si la intención es reportarlo a un equipo externo la dirección de nodo es 0xF1 en vez de 0xFE. Por decir el mismo caso anterior de reportar que el Check Engine (MIL) esta prendido, pero hacia un equipo externo, el string queda 18DB 89 F1 13

5.12 Ventajas y desventajas del direccionamiento OBDII *plus*

La presente propuesta es el mejor escenario encontrado buscando respetar lo más posible la intención de diseño del estándar SAE J2178, balanceado contra el requerimiento de este trabajo de realizar un protocolo fácil de usar para mensajes que no son de diagnóstico. Dicho balance además teniendo en cuenta el estado de arte presente de los sistemas embebidos disponibles comercialmente en la actualidad los cuales son bastante más capaces de lo que eran cuando se desarrolló dicho estándar. Entre los compromisos realizados en dicho balance está el sacrificar simplicidad en la estructura de los mensajes, en favor de darle aplicación al trabajo extenso y minucioso ya realizado en el SAE J2178-4. La estructura original J1850, base de este estándar se está forzando en cierto grado para darle sentido sobre una base CAN BUS, en favor de utilizar esta plataforma. La ventaja de CAN BUS sobre cualquiera de los 4 originales OBDII seriales es el grado de simplicidad, precio competitivo, velocidad y confiabilidad superiores que han alcanzado las soluciones CAN BUS a lo largo del tiempo

Página en blanco

6 Implementación de Hardware

Las especificaciones técnicas delinean el comportamiento del circuito a implementar. La manera en que se implemente esta implícitamente abierto al criterio del desarrollador, siempre y cuando cumpla con los requerimientos de la especificación. Esto es, se maneja como plataforma de hardware abierta. Este proyecto de tesis se maneja de manera similar. Hay diversas maneras de implementar una solución para un requerimiento dado. En este trabajo se busca presentar diversas opciones, de las cuales se selecciona alguna para su implementación. Se deja abierto a futuros trabajos el optimizar dicha solución o plantear nuevos, siempre y cuando respetando los requisitos y en especial el objetivo principal de ser estandarizado.

6.1 Esquema General

El proyecto de tesis se divide a grosso modo en tres etapas: Investigación del estado del arte, Desarrollo de los generales del protocolo propuesto, e implementación final.

Es en la tercera etapa, la implementación donde se prueba el sistema y se pone a punto. Esta comprende dos áreas: Hardware y Software

El hardware comprende dos vertientes: El subsistema de comunicación con el tren motriz y el subsistema de comunicación con periféricos de los otros dominios. Estos convergen en una unidad central del dominio de los sistemas auxiliares de seguridad y confort (en inglés “body”).

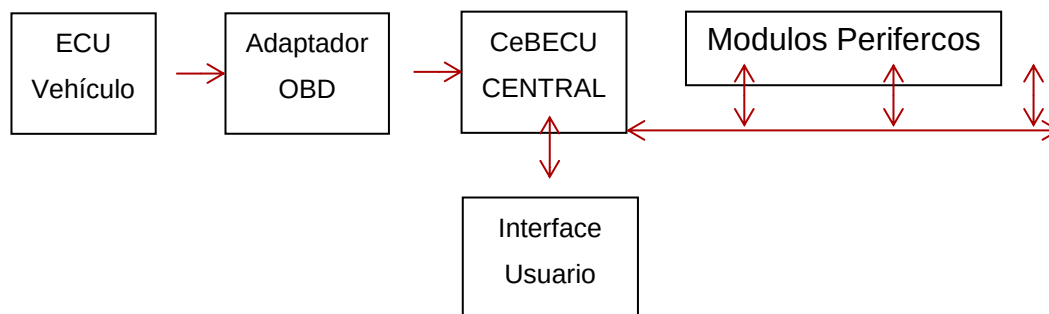


Figura 6.1 – Esquema general de la estructura de Hardware

La parte central donde convergen estas dos vertientes está conformada por la Unidad controladora Central Electrónica del dominio de seguridad y confort (CeBECU, por las siglas en inglés Central Body Electronic Control Unit). La función principal de dicho modulo es el enlace entre los diferentes sistemas. Este actúa como puente de comunicación entre el ECU del vehículo y sus parámetros, los sistemas periféricos a controlar por el presente trabajo, y la interface con el usuario. La CeBECU puede ser un módulo físico independiente o virtual embebido todo o en partes en los diversos módulos físicos.

La estructura del hardware está dividida en diferentes módulos. Siguiendo el orden del flujo de información, el primer componente es el adaptador OBD para comunicación con el ECU del vehículo. Los parámetros del vehículo, el adaptador los comunica a la unidad central. La unidad central a su vez los comunica a la interface de usuario y los diversos módulos periféricos. Los módulos periféricos, estando en el mismo bus se intercomunican entre si y la unidad central. Esta última funciona como controladora del tráfico entre las diversas entidades.

6.2 Adaptador OBD

Para la comunicación con el tren motriz, se utiliza un conector adaptador de OBD (del inglés OBD scanner). Los datos del tren motriz se pueden leer directamente de un vehículo a través del puerto OBD. Entre estos datos esta por decir la velocidad actual del mismo. En la sección 2.30 se evaluaron diversas opciones comerciales.

Para este proyecto se hizo la conversión de un adaptador comercial con interface Bluetooth a interface UART. Para realizar esta tarea, en el adaptador se localizó el tablero Bluetooth estilo HC-05 y se identificaron los pines correspondientes al UART. Estos se localizan a la izquierda del módulo, siendo el pin 1 el TX y el 2 el RX.

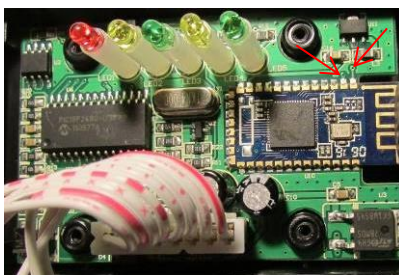


Figura 6.2.1 – Pines TX/RX UART del módulo OBD

El modulo Bluetooth se retiró del tablero para evitar interferencias. A partir de las pistas del PCB se conectan un par de cables hacia un conector. Todo se monta en un PCB protoboard genérico para darle robustez al circuito y facilitar conexiones

En el mismo tablero se incorpora una fuente de poder open frame genérica variable de 1AMP para alimentar al Arduino a partir del voltaje de batería proveniente del puerto OBDII. En el mismo PCB se ubican por debajo pines de conexión que corresponden con los puertos necesarios del Arduino Mega para la comunicación y control de los dispositivos.

Los pines que se encuentran por debajo tienen la misma configuración que un shield genérico utilizado en un Arduino. En caso necesario se puede remover fácilmente del Arduino. Sin embargo, tiene la suficiente sujeción para permanecer en su lugar bajo las vibraciones y movimientos naturales de un vehículo en marcha, sin moverse ni presentar riesgo de corto circuito.

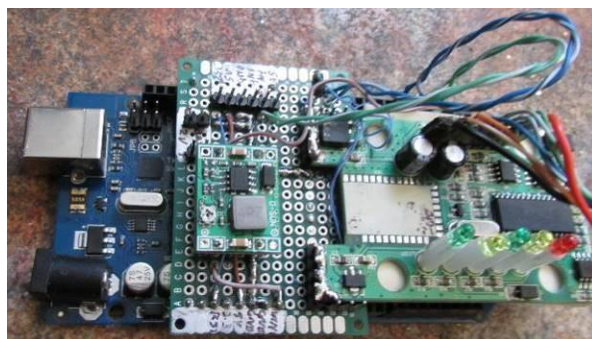


Figura 6.2.2 – Circuito completo

Para propósitos de pruebas básicas de las rutinas OBD, se incorporó un display LCD de dos líneas comercial de la marca Sparkfun. Este shield además incorpora diversos

botones para uso general. La comunicación de dicho display con el Arduino es por medio de un solo bus I2C.

El display se separó del resto del circuito para facilitar su montaje, además de minimizar tableros y peso que estén expuestos a potenciales caídas y accidentes. Dicho tablero se montó sobre un circuito PCB para facilitar las conexiones hacia el tablero principal



Figura 6.2.3 – Display LCD

Todo el conjunto se instaló en un vehículo para pruebas de funcionamiento. El circuito probó ser más robusto que en versiones anteriores ante el manejo del mismo con el vehículo en movimiento.



Figura 6.2.4 – Tablero implementado

6.3 Emulador OBD

Para pruebas de escritorio, se hizo un emulador del adaptador de lectura OBD utilizando un segundo Arduino conectado por comunicación UART al microcontrolador principal. Dicho Arduino se programó para solo responder al PID de la velocidad. Se le conectó un potenciómetro en una entrada analógica para simular la velocidad. Dicha entrada se lee y se escala a un máximo de 255 (máximo valor del PID).

A través del puerto serial se lee constantemente en espera de un código "01 0D", el cual en OBDII significa para el modo 01, PID 0D requerimiento de velocidad. El Arduino entonces contesta "41 0D xx". El protocolo pide que se conteste al primer dígito como 4, seguido de repetir el comando requerido. Las xx es la respuesta de la velocidad en hexadecimal, la cual el Arduino la toma de la lectura análoga del potenciómetro. Adicional se utilizó un display de dos líneas para observar los parámetros. Las rutinas principales del programa del Arduino quedan:

```
// read the value from the sensor:
sensorValue = analogRead(sensorPin);
//convert to max OBD speed
speed = sensorValue/4;
st_speed = String(speed)+" ";
hx_speed = String(speed, HEX)+" ";
if (speed <= 15){
    hx_speed = "0"+hx_speed;
}
lcd.setCursor(14, 1); lcd.print(hx_speed); //print status

//serial communication
if (stringComplete) {
    if (inputString.substring(0,4) == "010D"){
        Serial.println("41 0D " + hx_speed);
        //broadcasts speed
    }
    else {
        Serial.println("? " + inputString);    //broadcasts non valid input
    }
    // clear the string:
    inputString = "";
    stringComplete = false;
}
```



Figura 6.3.1 – Emulador OBD sencillo

6.4 Central Electrónica del dominio de seguridad y confort CeBECU

La parte central de este sistema está conformada por la Unidad controladora Central Electrónica del dominio de seguridad y confort (CeBECU, por las siglas en inglés Central Body Electronic Control Unit). Como se comentaba anteriormente, la función principal de dicho modulo es el enlace entre los diferentes sistemas. Este actúa como puente de comunicación entre el ECU del vehículo y sus parámetros, los sistemas periféricos, y la interface con el usuario.

Adicional a esta comunicación con el tren motriz, el CeBECU es la unidad central para la red local de comunicación de subsistemas a enlazar. Para el presente trabajo, la red esta implementada en esquema Master / multi-esclavos sobre el propio CAN-BUS. Esto es, hay una función Master principal la cual coordina el tránsito de los mensajes dentro de la red. Los esclavos solamente reaccionan y/o contestan cuando el Master les indica hacerlo. De esta manera se logra orden en los mensajes dentro de la red minimizando los conflictos entre burst de datos.

La primera realización de este módulo se realizó de manera virtual embebida en la interface de usuario o HID. Debido a que el prototipo se limitó a solamente el sistema de luces, por simplicidad se permitió al HID absorber las funciones del CeBECU de emitir un comando y solicitar el estado de sus módulos esclavos. El módulo HID cuenta también con salida serial, a la cual a través del UART se conectó un módulo bluetooth HC-05 para monitoreo de todo el tráfico, emisión de comandos y recepción de estados.

De esta manera se crea un puente virtual entre todos los buses de comunicación. Cualquier comando OBD valido presente en un canal se retransmite en otro.

Como implementación futura se recomienda un tablero arduino mega por sus múltiples puertos de comunicación disponibles. De esta manera se puede dividir todo el vehículo en 2 o 3 canales CAN-BUS para recorrer los múltiples módulos del vehículo. En otro puerto se debe conectar los datos del sistema OBDII de manera de traducirlos en estructura OBDIIplus y hacer broadcast a parámetros clave como la velocidad o estado del motor (encendido/apagado/estacionado). Todos los anteriores estados, así como recepción de comandos se pueden desplegar en una pantalla LCD a partir de este módulo.

El hardware se explica de manera más detallada en la sección correspondiente al HID. Así también, el firmware implementado se presenta en la sección correspondiente a los encoders y decoders de OBD

6.5 Red Área Local CAN BUS

La comunicación entre módulos, por requerimiento del proyecto debe realizarse a través de interface CAN-BUS. Como se ha mencionado anteriormente, el CAN-Bus es el estándar de comunicación actual en los vehículos modernos. Entre sus ventajas para la presente propuesta es su inherente manejo de múltiples módulos interconectados entre sí en la misma red. Este es un requisito fundamental en la propuesta, dado que múltiples módulos deben estar intercomunicados compartiendo información de manera simultánea.

Para propósitos de desarrollo de la propuesta, se utilizan módulos tipo shield para Arduino comerciales marca linksprite. Estos están basados en el popular diseño sparkfun de la marca homologa.

El pin out del módulo está diseñado para ser un drop in directo a utilizarse en un tablero arduino UNO. Viene incluido con conectores para hacerse un stack up encima del arduino. Dado que los módulos desarrollados tienen layout propietario basados en arduino nano, fue necesario realizar una conexión alamburada pin a pin. La librerías utilizadas son en base al arduino UNO, por lo que se realizó la conexión siguiendo el pin out del arduino UNO adaptado al Nano.

Los pines utilizados son:

- MOSI – Pin 11 UNO (D11) – Pin 14 nano
- MISO – Pin 12 UNO (D12) – Pin 15 Nano
- SCK – Pin 13 UNO (D13) – Pin 15 Nano
- CS – Pin 10 UNO (D10) – Pin 13 Nano
- INT – Pin 2 UNO (D2) – Pin 5 Nano

La alimentación de poder (power supply) se realiza a través del convertidor DC-DC a bordo de cada tablero, el cual se alimenta del 12V de batería del automóvil y lo convierte a 5VDC.

La resistencia shunt de 120 ohms entre terminales +/- del CAN BUS se removió. Esta terminación se utiliza para adaptar impedancias del bus cuando solamente hay dos módulos conectados. Al haber hasta 5 módulos en este trabajo, se removieron todas menos una para evitar caída de voltaje en el bus.

Estos tableros cuentan además con un conector DB-9, con la intención de conectarse directamente a un OBDII de un automóvil por medio de un cable especial. La intención es facilitar el desarrollo de un sistema OBDII para CAN-BUS de automóvil. Para este trabajo no se utilizó dicho conector, solamente el terminal block azul.

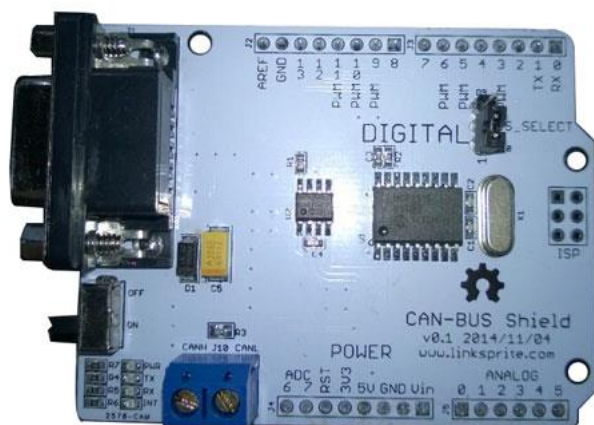


Figura 6.5.1 – Shield CAN-BUS linksprite

6.6 Subsistemas del Vehículo

Como se ha mencionado anteriormente, para este proyecto se requiere acotar el desarrollo a solamente el módulo de las luces. Sin embargo, la propuesta de protocolo en conjunto con las tablas descritas es capaz de manejar todos los sistemas comunes descritos en el capítulo 5.

El módulo de las luces fue seleccionado dada la simplicidad de su hardware, balanceada con la posibilidad de ejercitar una parte bastante representativa del protocolo propuesto. De esta manera puede facilitar el dar un ajuste fino a la propuesta. La idea de los mismos es definir en alto nivel sus requerimientos, y hacer una propuesta de desarrollo

- Los módulos a implementar son:
 - Cuatro módulos idénticos Estos deben contar con un direccionamiento físico que les permita identificar en que esquina del vehículo están. Lo anterior en cumplimiento del objetivo de desarrollar módulos universales.
 - El HID simulando el panel de instrumentos del vehículo. Representa las entradas para conexión con el switch de las luces y la generación de comandos OBDIIplus a partir de dichas entradas.
 - El CeBECU (sistema central), el cual se propone hacer virtual y embebido en las funciones del HID. Como monitoreo de estados, el tablero incluye una serie de leds con los estados de cada luz (encendido/apagado) según corresponda a partir de los comandos generados por el propio tablero, o recibidos por medio de algún puerto de comunicación.
 - Para los sistemas anteriores se pretende en base Arduino, por económico y dado que no requieren mayor capacidad de procesado y velocidad.

6.7 Módulo de Luces

El sistema de las luces esencialmente consta de un sistema remoto direccionado de interruptores de mediana potencia, con sensor de carga en circuito abierto. Repitiendo sus requerimientos de capacidad son

- Al menos 2 puertos de salidas independientes de por lo menos 25W (2 A) c/u para luces de posición o direccionales. Esto son para el lateral y Frontal/trasero
- Al menos 2 puertos de salida independientes de por lo menos 100W (8 A) c/u para luces faro. En el frente estos son Luz baja y luz alta
- Los anteriores deben contar con algún tipo de sensado de corriente el cual pueda ser capaz de detectar una luz no operacional (foco fundido)

6.7.1 Circuito switch de luces

El circuito principal, además del microcontrolador es el de control de las luces. La definición del producto no especifica la manera de desarrollar el mismo, solamente los requerimientos funcionales generales. Es bajo este tenor que se deja abierta al criterio del constructor la manera de realizarlo. Para esto se proponen 3 opciones de circuitos:

Circuito basado en Relay con optoacoplador:

Como la descripción lo indica, el componente interruptor es un relay accionado por el microcontrolador (uP). Entre las ventajas esta su facilidad de implementación, bajo costo, gran capacidad de corriente sin necesidad de disipadores de calor, y aislamiento efectivo de la etapa del uP con la de potencial.

Como desventajas presenta el no soportar una señal PWM y requerir un circuito externo para sensado de corriente. Para esta solución existen algunas aplicaciones comerciales en el mercado, las cuales ya incluyen la interface con el uP a través de un optoacoplador

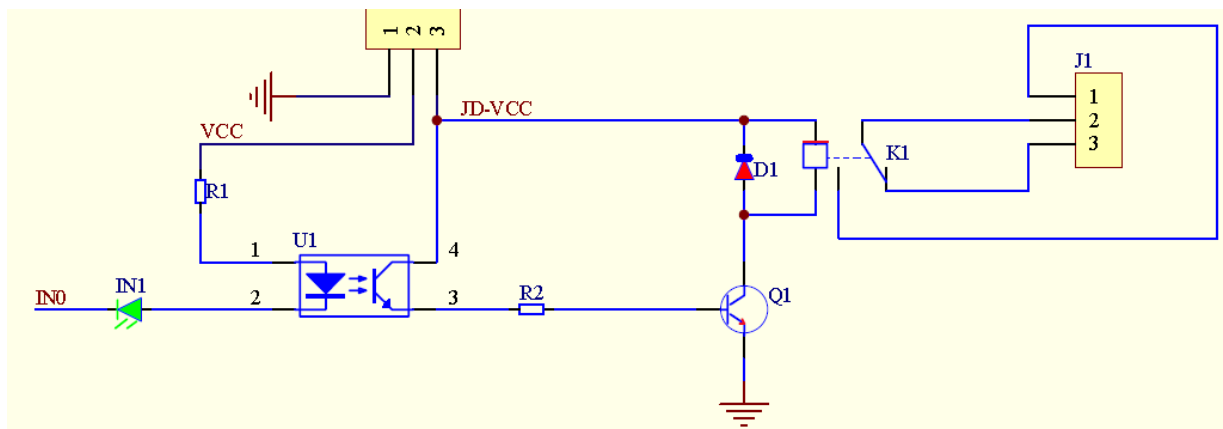


Figura 6.7.1.1 – Circuito implementado con relay

Circuito basado en Mosfet Canal N

Esta solución de estado sólido es una de las más populares entre entusiastas de proyectos con Arduino. Entre sus ventajas está el bajo costo, fácil implementación, soporta PWM, y moderada corriente. Su desventaja principal está en requerir disipador de calor para altas corrientes. El circuito de canal N requiere que la carga está conectada a la alimentación, mientras que el mosfet completa el circuito a tierra conectándola o desconectándola. Esto representa una desventaja en circuitos de automóvil, en los cuales es frecuente que la tierra sea el chasis del vehículo. Por lo tanto, los dispositivos tienen tierra común en el chasis y se les provee voltaje por el lado positivo.

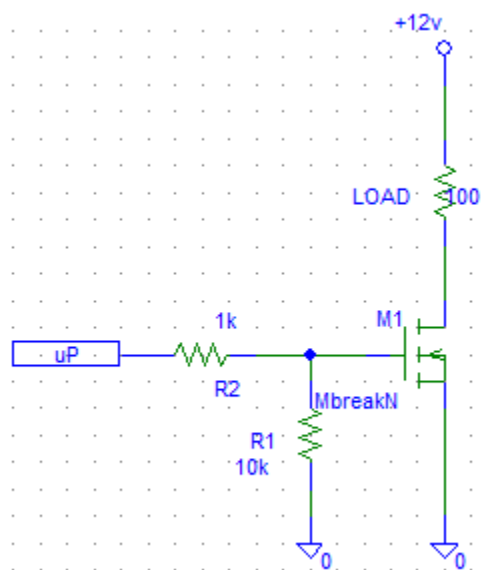


Figura 6.7.1.2 – Circuito implementado con MOSFET N

Circuito basado en Mosfet Canal P

El circuito basado en Mosfet canal P realiza la interrupción de alimentación de la carga por el lado vivo, permitiendo a la carga estar todo momento conectado a tierra. Esta configuración es la que más fácilmente se adapta a los circuitos de los automóviles por su similitud de las configuraciones actuales. Es común que en los circuitos de

automóviles la tierra este siempre conectada a chasis, y sea el positivo el cual se controle. Al igual que el de canal N, este circuito soporta PWM. Sin embargo, es más complejo al requerir un transistor adicional. Esta configuración es un poco más costosa que el canal N por el circuito adicional, además de que los Mosfet canal P suelen ser un poco más caros que sus equivalentes canal N.

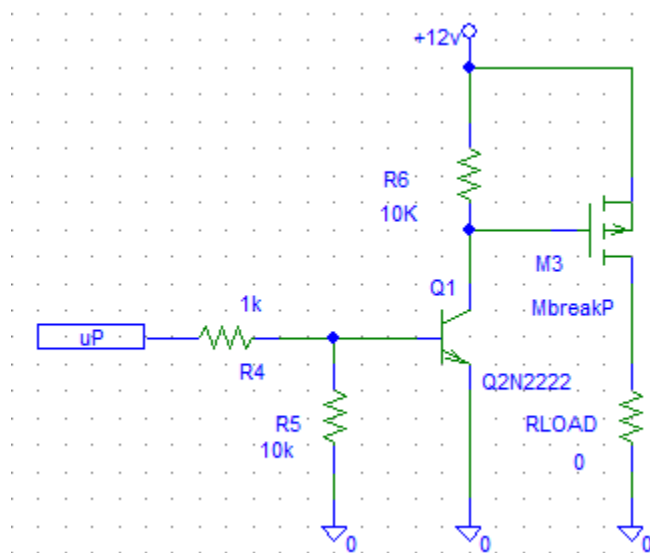


Figura 6.7.1.3 – Circuito implementado con MOSFET P

Circuito basado en Relay activado por MOSFET:

Como la descripción lo indica, el relay es activado por un MOSFET. Este a su vez es controlado directamente por un pin I/O del arduino. Esta configuración combina varias de las ventajas de las propuestas anteriores. Incluye la capacidad de alta corriente de los relays, a la vez de presentar una alta impedancia al pin del arduino. El arduino al tener varios I/O conectados, vuelve crítico el consumo de corriente del integrado en su totalidad. Esto le confiere una ventaja a los drivers MOSFETS.

La desventaja es el no poder manejar un PWM. Dado que de momento está fuera del scope del proyecto el manejar salidas con voltaje variable, no es necesaria la función PWM. Por lo tanto, esta es la opción elegida.

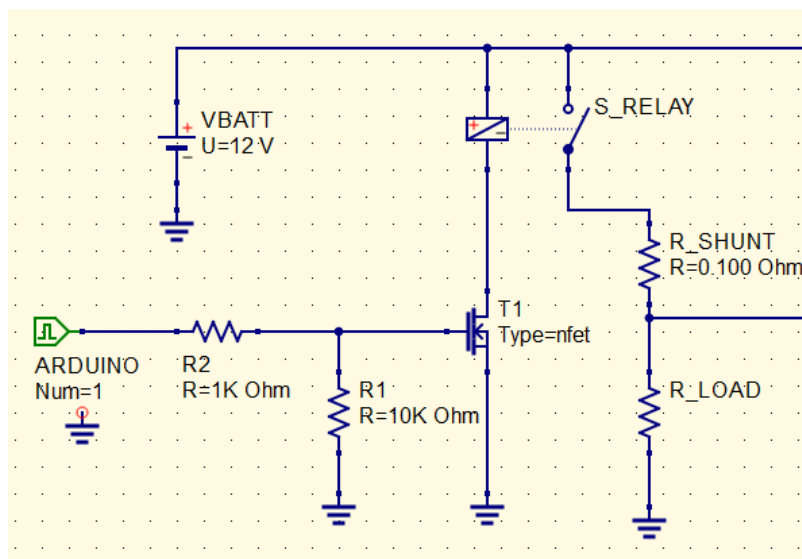


Figura 6.7.1.4 – Circuito con relay / mosfet

6.7.2 Circuito sensado de corriente

Uno de los requerimientos de producto es la capacidad de reportar un malfuncionamiento del circuito. En este caso es un foco fundido o en corto. Una de las maneras de detectar esta condición es por medio de monitorear la corriente de consumo de la carga. Si la corriente es cero, por lo tanto el foco esta fundido. Si la misma es excesiva es indicativo de un corto circuito.

Los focos a utilizar son de 4 niveles de consumo de corriente. Los de posición conocidos como cuartitos son de unos 6 watts o 0.5 Amps. Los direccionales son de unos 25W o 2 Amps. Las luces bajas son de unos 35W o 3Amps. Las luces altas son de unos 50W o 4Amps. Algunas luces pueden llegar hasta los 100W u 8Amps. Pero, para efectos de este trabajo se limita el sensado al rango de 0.5 a 5 Amps.

El sensado de corriente deberá estar a cargo del Arduino, por lo que se revisan primero sus capacidades de ADC. La entrada analógica del Arduino tiene 1024 niveles, de 0 volts a la referencia 5V. Esto es, puede discernir voltajes en escalones de 5mV. Por lo tanto, para ser capaz de discernir un cambio de voltaje, el mínimo recomendado es 10 veces la lectura mínimo. Es decir, 50mV es el mínimo recomendado para sus entradas.

Para medir dicha corriente hay diversos métodos: resistencia shunt, sensor de efecto hall en circuito integrado, bobina sensora de campo magnético, entre otros

Una de las maneras más sencillas y económicas de medir la corriente en un circuito es por medio de una resistencia shunt en serie. A partir de medir la caída de voltaje en dicha resistencia de valor conocido, se puede calcular la corriente que la atraviesa.

Para efectos experimentales del presente trabajo, se construyeron resistencias shunt a partir de un simple alambre. Para esto se hicieron dos resistencias shunt, una para las luces de posición y otra para las luces principales. De las tablas conocidas de la American Wire Gauge, el calibre 24AWG tiene un límite de corriente recomendado de 3.5Amps. Este calibre tiene una resistencia de 0.026Ohms por pie. Por lo que en 4 pies da casi exactamente 0.1 Ohms. Esto da una relación de 100mV/Amp. Las lecturas buscadas para el rango 0.5Amps a 2Amps por lo tanto van a estar en el rango de 50mV a 200mV en el shunt. Para el otro shunt se selecciona el calibre 21. Este calibre tiene aprox la mitad de la resistencia siendo de 0.013Ohms por pie, pero con una capacidad de corriente max de 9Amps. Los mismos 4 pies dan una resistencia de 0.052 Ohms. Esto da una relación de 50mV/Amp. Para el rango de 2 a 5 amps da valores de 100mV a 250mV. Estos shunts aseguran que la potencia disipada en los mismos no sea excesiva, estando aun así en el rango permisible de lecturas.



Figura 6.7.2.1 – Resistencias Shunt

La manera de construirlas es haciendo un rollo pequeño con el alambre. Para evitar que se genere inductancia, primero se dobla por mitad y sobre este se enrolla. Luego se asegura para que no se desbarate. De esta manera, quedan dos bobinas idénticas superpuestas en sentido opuesto, cancelando sus inductancias mutuamente.

El circuito completo para un canal queda de la siguiente manera:

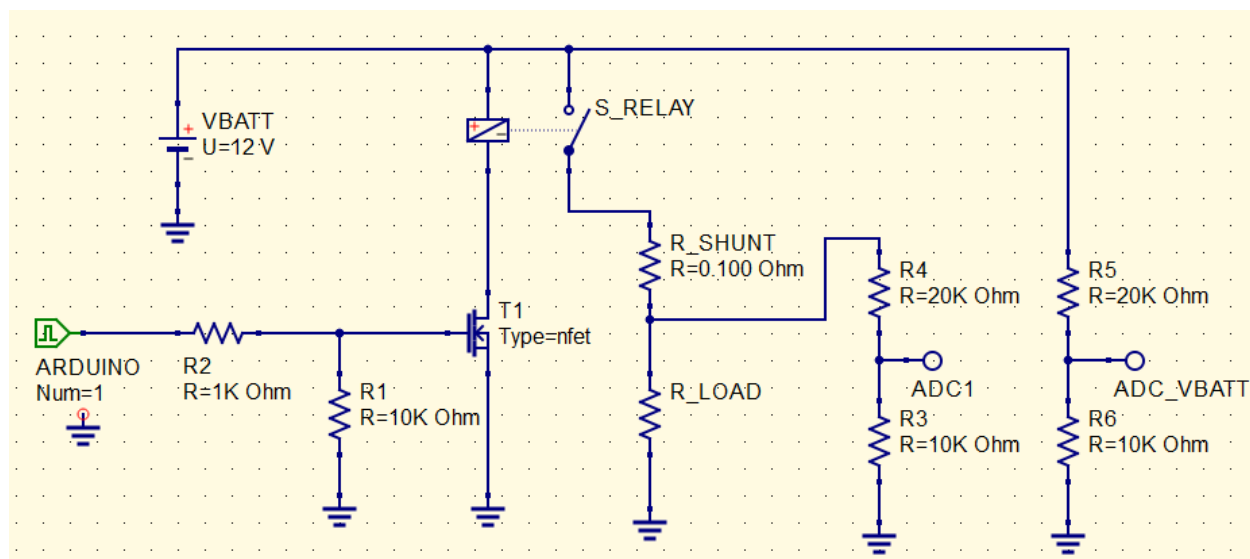


Figura 6.7.2.2 – Circuito completo – Un Canal

Para activar el relay se utiliza un MOSFET comercial canal N, para drenar la mínima corriente posible al Arduino. Las resistencias se sitúan bajo valores comunes, los cuales aseguran la correcta polaridad del MOSFET. El relay es de 12 volts, el cual como se mencionaba es actuado por el MOSFET. Este es alimentado directamente del voltaje del automóvil, el cual en realidad es de 13.2 a 13.8 volts.



Figura 6.7.2.3 – Relays y sus mosfets junto al procesador

Al activarse el relay, conecta el voltaje de batería del auto a la carga por medio del shunt. La carga como se mencionó para este circuito son las luces. La manera de interpretar la corriente a través del shunt es por medio de leer la caída de voltaje entre sus terminales. Tradicionalmente se utiliza un circuito sumador por medio de un opamp. En este caso, se utilizaron los puertos ADC disponibles del Arduino y se realizó el cálculo de caída de voltaje por medio de software. Un punto de medición se determina en el voltaje de entrada VBATT y el otro después de cada shunt. La lectura se realiza por medio de divisores de voltaje a 1/3. Siendo el máximo del Arduino 5V, esta configuración permite una lectura de hasta 15V.

El Arduino posee una entrada para cambiar su referencia de voltaje. Al usar esta directo a batería puede mejorar la resolución de lectura. Sin embargo no se utilizó para dar capacidad de leer el voltaje de batería entrante al módulo. Con respecto a la rutina de software, la especificación de producto no requiere medición de consumo de corriente, solamente presencia de la misma (detección de falla). Por lo tanto, la rutina de software se reduce a solamente comparar ambas lecturas. Si el voltaje después del shunt (VADCx) es menor al de entrada (VBATT) por al menos 10 niveles de ADC, quiere decir que la lámpara esta prendida. Se establece un mínimo de 10 puntos de ADC (sobre un máximo de 1024) solamente para estar por encima del nivel de ruido de 50mV.

6.7.3 Circuito selector de switches

Hay dos conjuntos de entradas adicionales al procesador. Una es para seleccionar la dirección física del módulo, y la otra es una serie de pushbuttons para activación manual de cada relay. La dirección física se selecciona por un dip switch de cuatro posiciones. Ambos circuitos constan de cuatro switches para las funciones descritas, por lo que la circuitería es la misma.

En general, este circuito consta de un divisor de resistencias seleccionable el cual entrega un voltaje diferente por cada switch presionado. Este voltaje es leído en un puerto analógico y dependiendo del valor se infiere el switch activado.

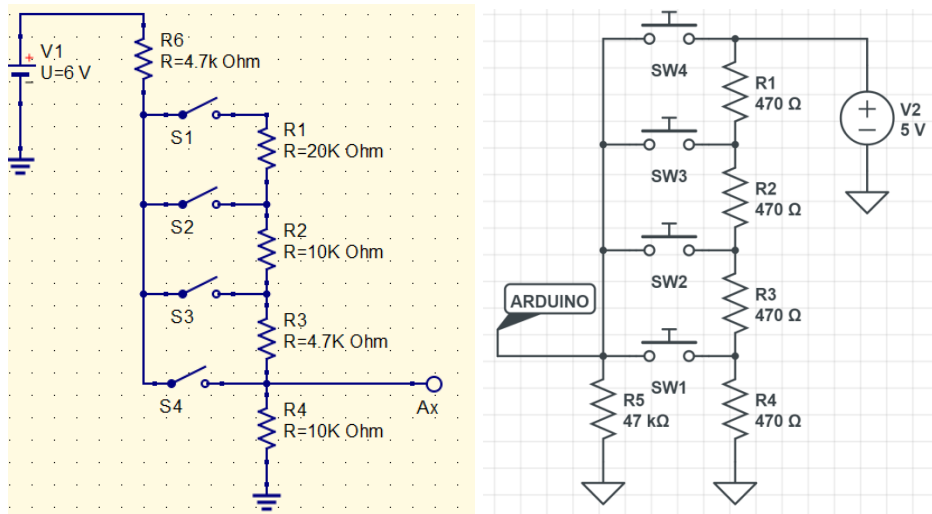


Figura 6.7.3.1.- Switches pushbutton

Se presentan dos propuestas. La primera en la imagen es un divisor con valores selectos, los cuales al activar cada switch entregan voltajes diferenciabiles para cada uno. Su ventaja reside en que al no haber ningún switch activado, no hay consumo de corriente. La desventaja es que no es muy estable y los valores entregados no son fracciones múltiplos del voltaje VCC por lo que la codificación para discernir valores se complica.

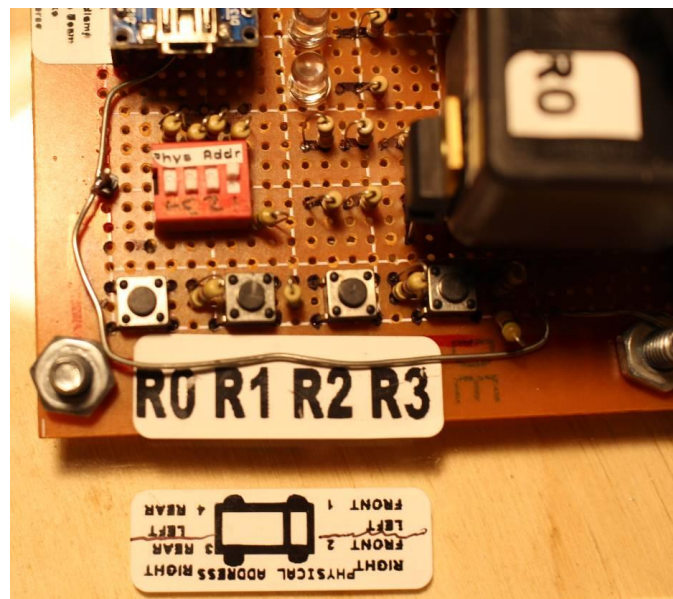


Figura 6.7.3.2 – Switches selectores

La segunda propuesta está basada en un divisor de voltaje fijo con resistencias iguales. El conjunto de switches selecciona el escalón de la escalera correspondiente. A la hilera de switches se le agrega una resistencia de valor alto para evitar que el pin del arduino flote y vea valores no existentes. Este circuito presenta valores de voltaje más estables y proporcionales que la otra propuesta. La desventaja es un consumo de corriente constante. Se seleccionó esta opción por su estabilidad.

6.7.4 Modulo CAN BUS

Para la comunicación entre módulos se le agrego al circuito anterior un shield CAN-BUS comercial. Este se instaló permanentemente en el tablero anterior de manera hardwired.

El modulo se conectó al puerto SPI del arduino nano. Los pines utilizados son D10-CS, D11 – MOSI, D12 – MISO, D13 – SCK, INT – D2

Para contar con un puerto de monitoreo o debug, se dejó habilitado el puerto UART del USB onboard. En paralelo a este puerto se dejó un conector como provisión para conectar un módulo bluetooth HC-05. El pinout de dicho conector corresponde a los pines necesarios para dicho modulo. De esta manera, se puede realizar el monitoreo de manera remota sin necesidad de conexión física.

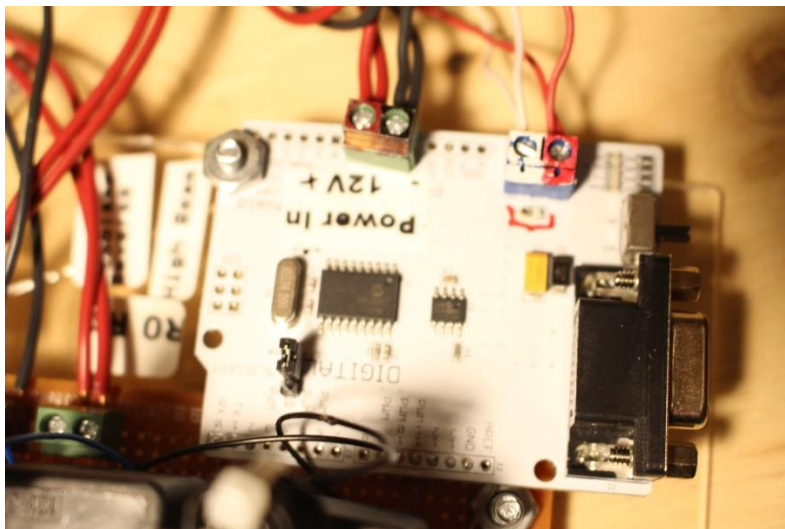


Figura 6.7.4.1 – Modulo CAN-BUS conectado

6.7.5 Diseño del PCB

Una vez definido el circuito a utilizar, se procede a realizar la implementación en un circuito impreso o PCB (del inglés Printed Circuit Board). Para la realización del mismo se utilizó el programa libre disponible PCBWeb Designer versión 2.7.50 (2014) de Silicon Frameworks LLC.

El esquemático se redibujó en dicho programa reemplazando por conectores la entrada de batería, la entrada de activación, la resistencia de carga y las salidas de los sensores. Así también se realizó la librería del símbolo y footprint para un relay comercial de automóvil de 12V/30A. También se agregó un led como indicador piloto de relay encendido. El esquemático redibujado quedó de la siguiente manera

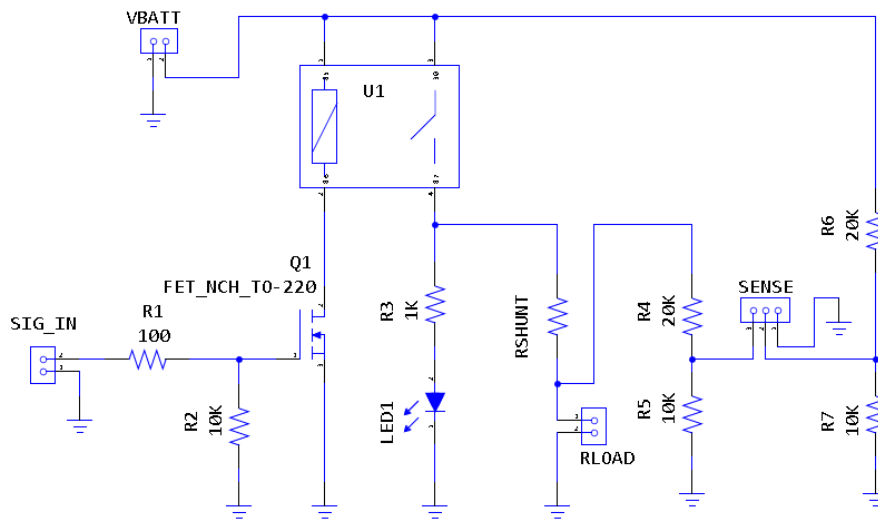


Figura 6.7.5.1 – Circuito completo – Un Canal

Basado en este esquemático y continuando con el uso del programa PCBWeb, se realiza el PCB. Para esta etapa se está solamente construyendo un canal de los 5 requeridos. Por lo tanto, el acomodo de los componentes se realiza en forma lineal, buscando que la integración de varios canales sea lo más sencilla posible. Con esta disposición, los demás canales se colocan uno junto al otro. Los puertos de señal se colocan en un extremo y los de potencia en el otro para separar los diferentes tipos de señales. El circuito impreso queda por lo tanto de la siguiente manera

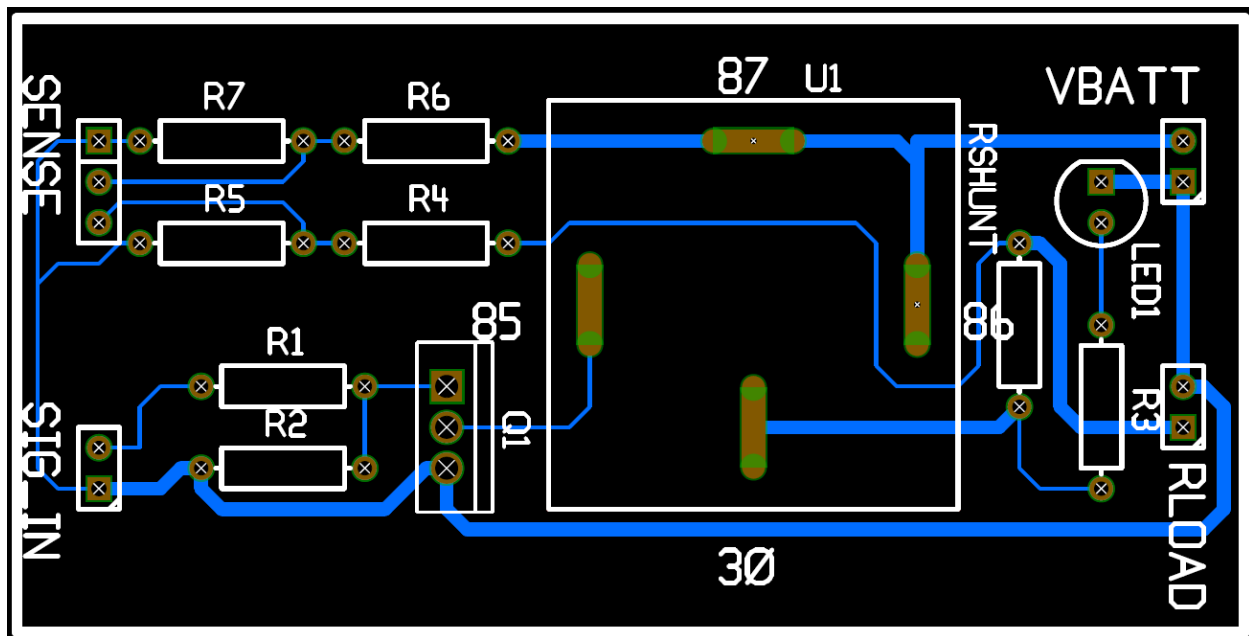


Figura 6.7.5.2. – Circuito Impreso

6.7.6 Implementación en PCB

El diseño anterior del PCB sirve de guía para construir un canal. A partir de este diseño se construye un PCB con tres canales para las luces posteriores. Los tres canales son para luz de freno/direccional, luz de posición y luz de reversa.

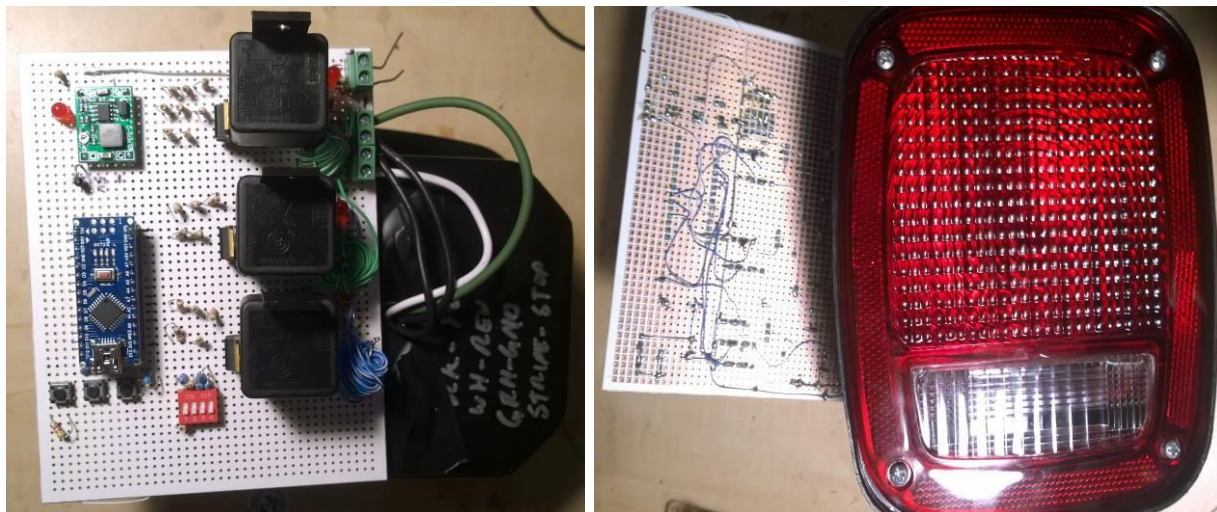


Figura 6.7.6.1.- Primera implementación

La construcción se realiza sobre un tablero perforado comercial de 10 x 15 cm. En el mismo se construyen los tres canales idénticos uno junto al otro. Para el control del mismo se adiciona un Arduino nano. El circuito de control se alimenta de una fuente externa conmutada que reduce el voltaje de batería de 13.5V a 6V para alimentar el Arduino.

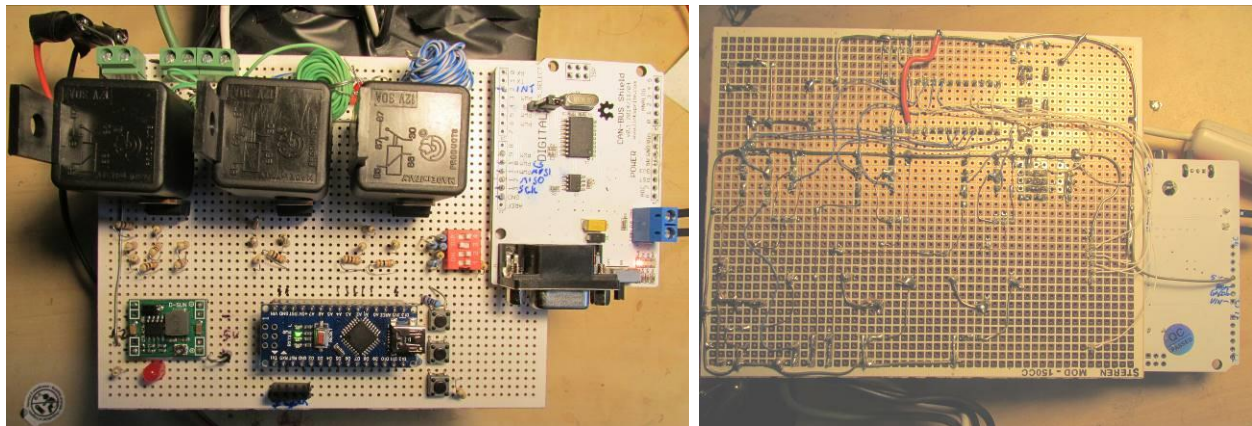


Figura 6.7.6.2.- Primer Prototipo funcional

Como control se agregan pushbutton switches para cada canal. Dado que estos circuitos son pensados para ser genéricos e indistintos de la posición donde se instalen dentro del vehículo, tiene que incluir una manera de decirle al circuito en qué posición se encuentra. Para esto se incorporan unos dip-switch los cuales codifican dicha posición. Todo el tablero se monta por la parte de atrás de una luminaria genérica trasera de vehículo (calavera).

Basado en las lecciones aprendidas del primer prototipo, se construyeron cuatro módulos idénticos con circuitería mejorada. Estos en revisión 2 son los módulos a presentar. Una vez completos se montaron sobre un cuadro de acrílico transparente para soporte físico y protección del alambrado posterior. Adicional se agregan etiquetas indicadoras de los puertos y componentes principales del sistema.

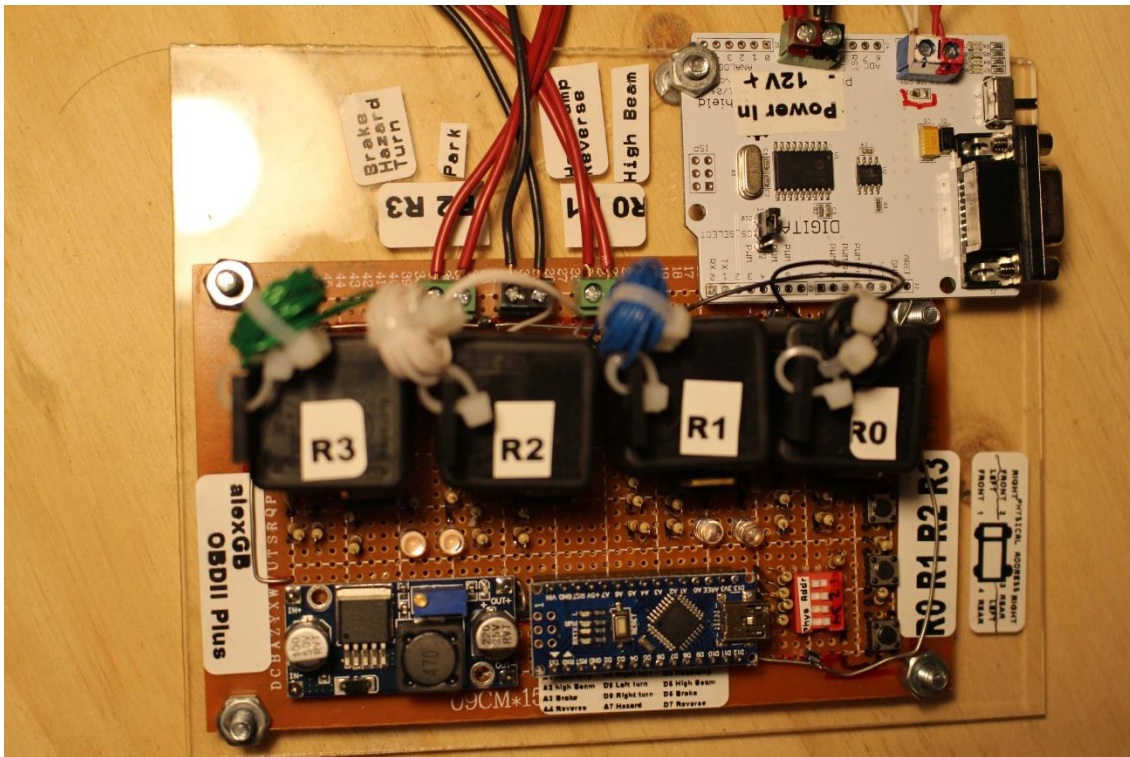


Figura 6.7.6.3.- Versión Final

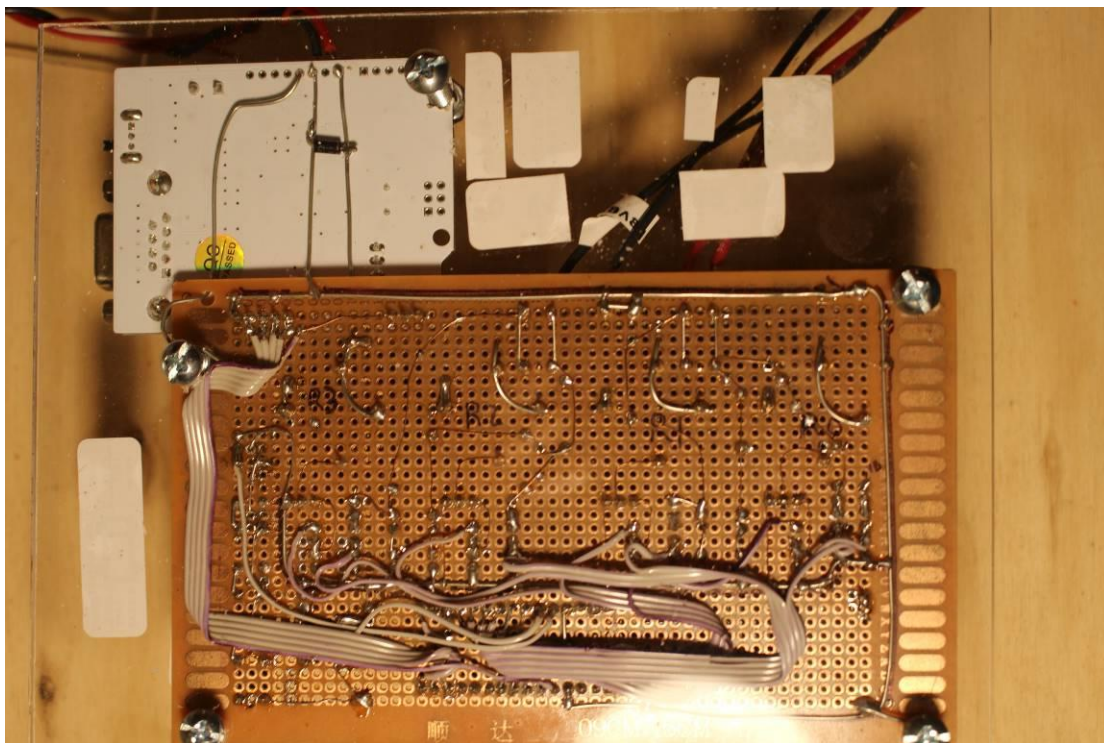


Figura 6.7.6.4.- Versión Final - reverso

6.8 Módulo HID

El tablero HID (del inglés Human Interface Device) es el tablero que proporciona el contacto directo con el usuario. Este sistema representa las palancas de direccionales, el switch de las luces, el pedal del freno y la palanca de la dirección en reversa (que activan las luces correspondientes). Todo el modulo es alimentado por un convertidor DC-DC de voltaje de batería 13.4V a 5V.

6.8.1 Circuito switch selectores

Para simplicidad, se instalaron pushbuttons correspondientes a todas las entradas provenientes de un usuario. Dichas entradas son: Luz de estacionamiento o cuartitos (Park), luces delanteras (headlamps), luces altas (high beam), luz de freno (brake), Luz de reversa (Reverse), Luces intermitentes (hazard), y direccionales izquierda y derecha (left & right turn). Estos pushbuttons se programaron por software con una rutina toggle. Es decir, cada que se pulsa el botón, el software emula que se prende o se apaga el switch.

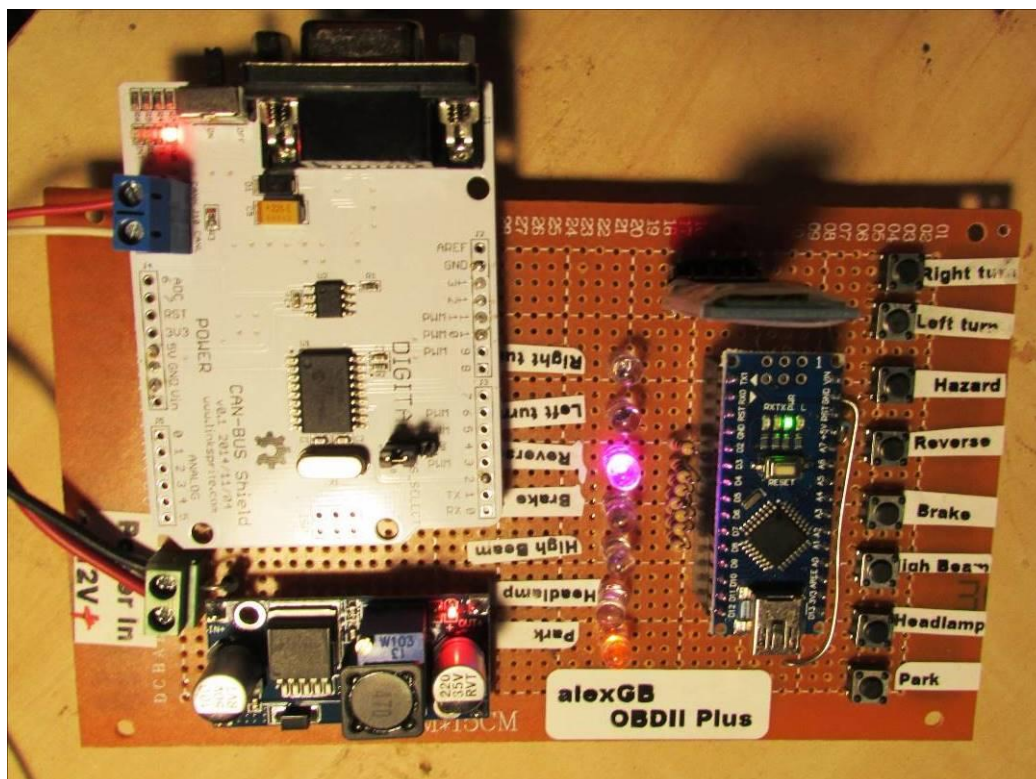


Figura 6.8.1.- Modulo HID

Los pushbutton están conectados a los puertos I/O A0 al A5. Estos se configuraron como puertos digitales. Internamente se habilitaron las resistencias de pull up del uP atmel, por lo que no es necesaria una resistencia externa. Esto permite conectar los pushbutton directos entre tierra y el pin del Arduino. Los pines A6 y A7 no tienen esa facilidad, por lo que se les conecto una resistencia pull up externa.

6.8.2 Circuito LEDS indicadores

Los leds indican el estatus interno de las banderas de cada sistema. Estos se conectaron a los puertos digitales de salida D3 al D9. Se les puso una resistencia de 1K en serie para minimizar el drenado de corriente del uP en el caso de que prendan todos los leds. Los leds no están conectados directamente a los switches de habilitación de estado. Estos responden a las banderas de estado internas en el firmware. En caso de recibir un comando por algún puerto de comunicación, el led correspondiente a dicho comando va a reaccionar encendiendo o apagando según sea el caso.

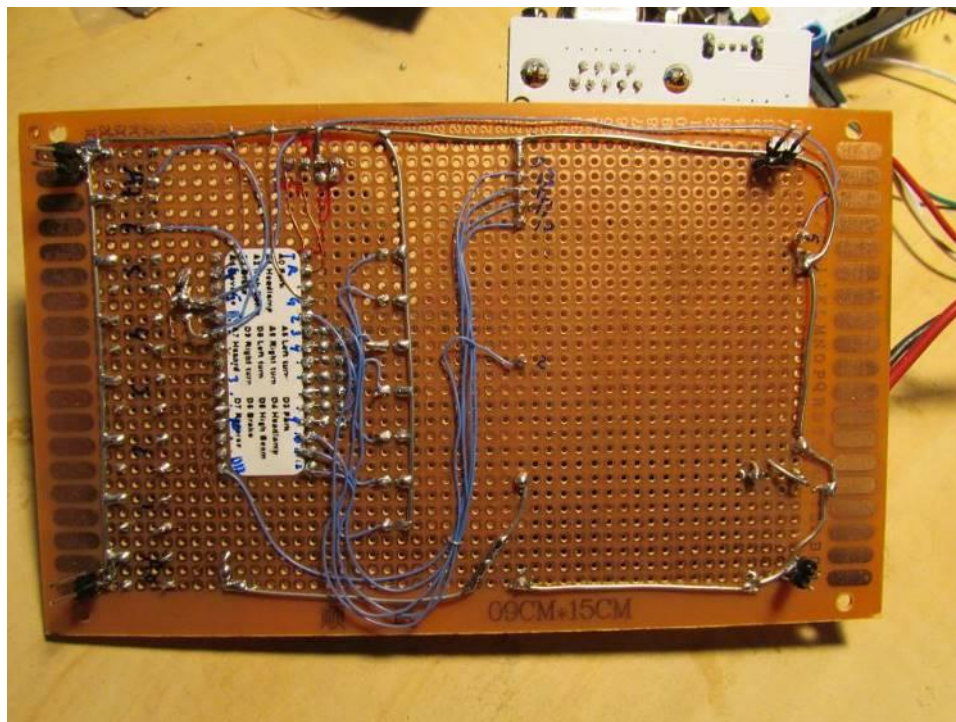


Figura 6.8.2. - Modulo HID

6.8.3 Modulo CAN BUS

De manera similar que el módulo de luces, a este módulo se le incorporo una tarjeta CAN BUS. Este se instaló permanentemente en el tablero anterior de manera hardwired. El CAN BUS es manejado por medio del puerto SPI. El modulo se conectó al puerto SPI del arduino nano. Los pines utilizados son D10-CS, D11 – MOSI, D12 – MISO, D13 – SCK, INT – D2

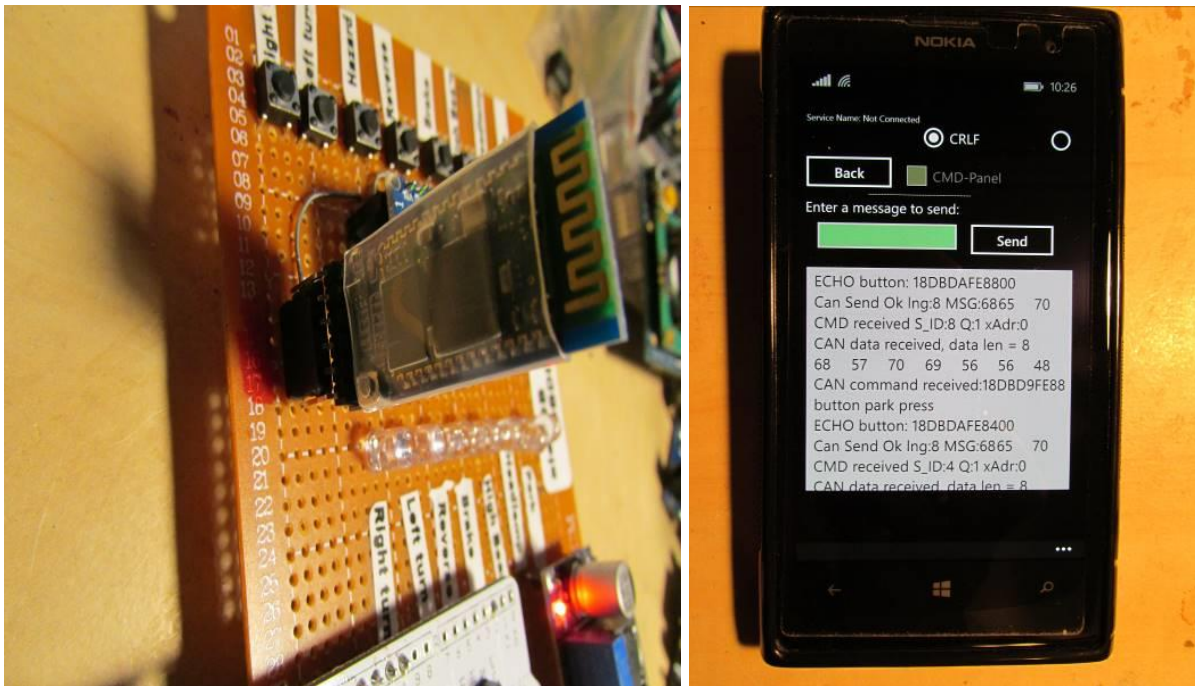


Figura 6.8.3.- Modulo HID - Bluetooth

Para contar con un puerto de monitoreo o debug, se dejó habilitado el puerto UART del USB onboard. En paralelo a este puerto se dejó un conector como provisión para conectar un módulo bluetooth HC-05. El pinout de dicho conector corresponde a los pines necesarios para dicho modulo.

A través de este es posible controlarse de manera remota enviando y recibiendo los comandos. Se le incorporo un puente de comunicación UART-SPI. El comando OBDII plus que se recibe por un puerto se envía por el otro.

6.8.4 Modulo Central Virtual

La parte central de este sistema está conformada por la Unidad controladora Central Electrónica del dominio de seguridad y confort (CeBECU, por las siglas en inglés Central Body Electronic Control Unit). Dos de sus funciones principales son: coordinar el tráfico de comunicación entre módulos, y vigilar el estado general del sistema. Estas funciones básicas se implementaron embebidas en el firmware de este módulo HID. Por lo tanto, el modulo central quedo de manera virtual dentro del HID

El puente entre canales de comunicación se realiza por medio del software. Internamente todo comando recibido por cualesquier canal se hace replay a todos los demás canales. Primeramente se valida el ser un comando OBDIIplus valido. Después dependiendo del origen puede recodificarse para después retransmitirse en todos los canales. Por medio del CAN-BUS se transmite solo la cadena del comando. Por UART, el cual es compartido por USB y bluetooth adicionalmente se envían textos que van mostrando el progreso del programa.

Con respecto a los pushbuttons on board para cada estado, la activación dispara el codificador OBDII correspondiente. El firmware al detectar un botón activado codifica el correspondiente cambio de estado (prender/apagar determinada luz) en un comando OBDII plus. El comando generado se envía por todos los puertos de comunicación.

Este comando internamente se envía también al decodificador OBDII plus del mismo firmware. A partir de este momento, dicho comando codificado internamente se toma de manera igual a un comando recibido externamente. El decodificador lo interpreta y activa/desactiva el correspondiente led. Si se recibe un comando por el serial (ej bluetooth), este comando además de reenviarse por el CAN-BUS, va a dar al mismo decodificador. El decodificador va a realizar la activación/desactivación del led correspondiente, tal como si el comando hubiese venido ahora de los pushbutton.

En otras palabras, el comando que se reciba por cualquier medio (pushbuttons, USB, CAN-BUS, etc) se canaliza a un único comando OBDII interno. Este comando único a su vez es retransmitido a todos los canales de comunicación, incluyendo el decodificador para el estado de las banderas internas representados por los LEDs

6.9 Sistema Completo

Los 4 módulos y el HID se enlazaron con una fuente común de 13.8V y un CAN-BUS compartido. A los módulos se les agregaron luces para simular las reales de un vehículo. En las imágenes se puede apreciar el sistema completo

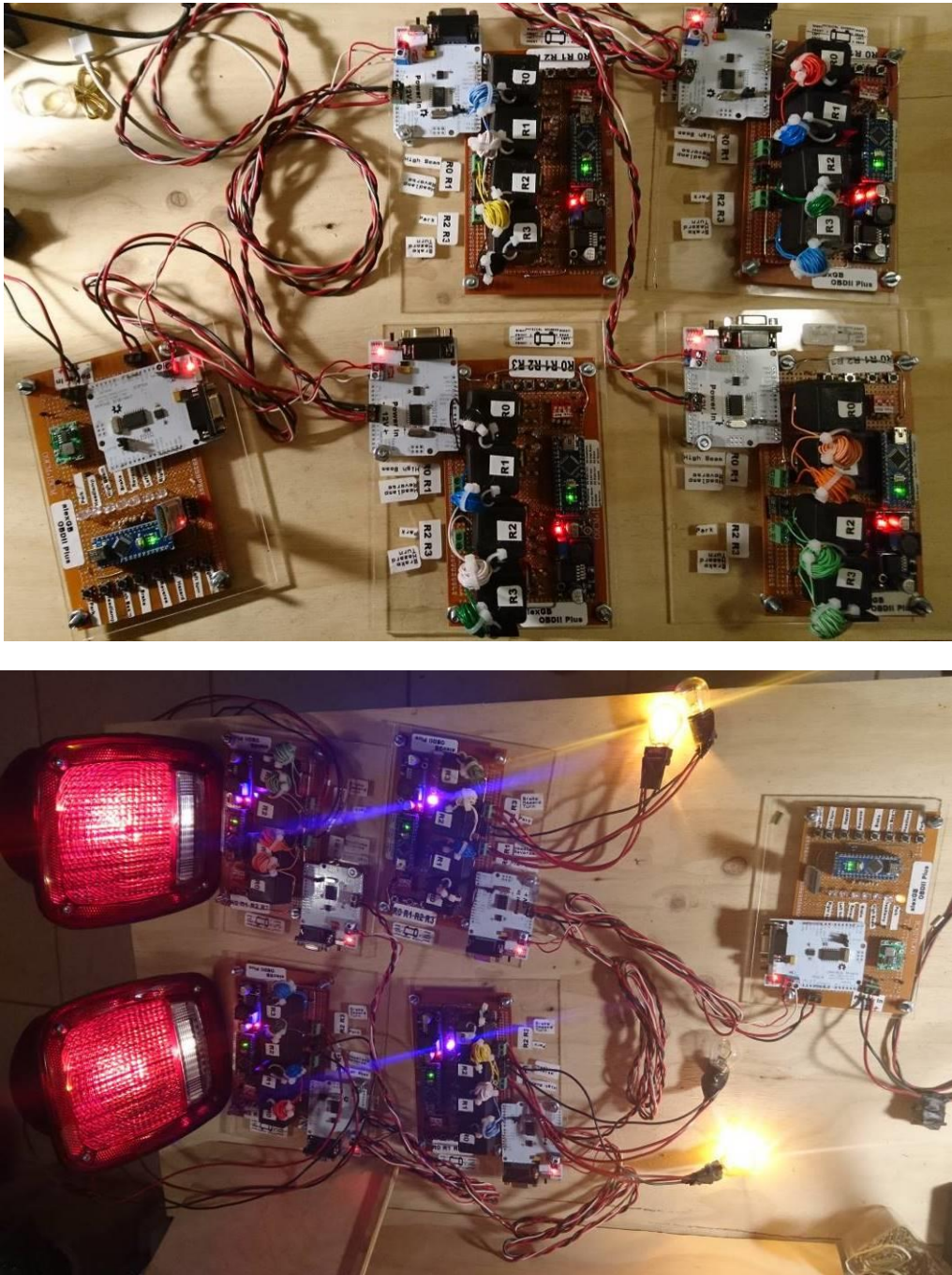


Figura 6.9.1. - Sistema completo

Página en blanco

7 Implementación de Software

El objetivo principal del presente trabajo es el desarrollo de un protocolo de intercomunicación entre módulos. Es entonces que la implementación de software pasa a ser la parte central del desarrollo físico del presente trabajo.

Entre los requisitos de este trabajo es el mantenerlo simple, de manera que cualquier otro usuario con conocimientos de básico a intermedio pueda fácilmente reproducir, adaptar e implementar esta solución en su propio ambiente de trabajo. Es bajo este concepto que se opta por la creación de subrutinas y módulos que sean fáciles de entender y usar.

Siendo este un trabajo mayormente orientado a protocolos OBD, los módulos principales son por lo tanto los que manejan los parámetros del OBD o PIDs. Por lo tanto, se requieren un módulo PID Encoder y uno inverso PID Decoder. En la aplicación más común de lectura de parámetros del vehículo, la ECU del mismo funge como PID Encoder al leer los sensores y codificar sus valores en un formato PID. El escáner de taller funge entonces como PID decoder al decodificar dicho PID para extraer el valor del sensor invocado. La manera más básica de implementación de software de este último es por medio de una función en la cual solamente se le dé a conocer el PID requerido y regrese un valor numérico.

Es entonces que por parte de la implementación de Software, se tendrían que cubrir los siguientes módulos para poder tener una funcionalidad completa:

- Intérprete de PIDs
- Decoder de OBDII Plus
- Encoder de OBDII Plus
- Intérprete de OBDII Plus
- Sistemas de comunicación UART/CAN-BUS/USB/BLUETOOTH
- Supervisor de sistema
- Firmware específico para el Hardware

7.1 Adaptador OBD -

A la hora de aventurarse por caminos inciertos, siempre es una buena recomendación iniciar por terrenos conocidos. En el caso de las aplicaciones OBD, hay diversos desarrollos disponibles en internet de donde tomar ideas. Es por esto que se eligió iniciar realizando un lector de parámetros PID. Es una tecnología conocida que se sabe cómo debe reaccionar de antemano. Con este conocimiento, es más fácil poner a punto la estructura deseada del software.

7.1.1 Intérprete de PIDs

El primer paso es capturar todo el estándar SAE J1979, el cual define los parámetros del vehículo a través del modo 0x01. Los 64 PIDs de dicho modo se definen en texto y a través de un byte del PID respectivo

```

/*****
* Arduino Library for OBD-II PID interpreter
* Library for Encoding / Decoding SAE J1979 Parameter IDs
* (2016) alex Garcia Blanco alex_G_Blanco@hotmail.com
*****/

// #include <Arduino.h>
// Mode 1 PIDs
#define PIDs_Supported_01h20h 0x00
#define TROUBLE_CODE_NUMBER 0x01
#define TROUBLE_CODES_FREEZE 0x02
#define FUEL_SYSTEM_STATUS 0x03
#define PID_ENGINE_LOAD 0x04
#define PID_COOLANT_TEMP 0x05
#define PID_SHORT_TERM_FUEL_TRIM_1 0x06
#define PID_LONG_TERM_FUEL_TRIM_1 0x07
#define PID_SHORT_TERM_FUEL_TRIM_2 0x08
#define PID_LONG_TERM_FUEL_TRIM_2 0x09
#define PID_FUEL_PRESSURE 0x0A
#define PID_INTAKE_MAP 0x0B
#define PID_RPM 0x0C
#define PID_SPEED 0x0D
#define PID_TIMING_ADVANCE 0x0E
#define PID_INTAKE_TEMP 0x0F
.....
#define PID_ENGINE_TORQUE_PERCENTAGE 0x62
#define PID_ENGINE_REF_TORQUE 0x63

```

A través de todo el programa se pueden usar estas referencias en texto para indicar un PID en específico. Internamente el PID seguiría siendo un byte numérico, sin embargo por medio de un nemotécnico es más fácil identificar cual es el PID requerido

La función inversa se realiza por medio de una función switch. A dicha función se le entrega el PID y regresa un string con el nombre

```
//Get a printable name for a PID
String GetPIDName(byte PID){
    String PIDname = "none";
    switch (PID) {
        case PID_RPM: //0x1C
            PIDname = "PID_RPM";
            break;
        case PID_SPEED:
            PIDname = "PID_SPEED";
            break;
```

.....

7.1.2 Decodificador del valor del PID

No todos los valores retornados por el ECU son directamente el valor del sensor invocado. Por decir, la temperatura del motor tiene un desfase de - 40°C. Para un valor retornado de 0x3C o 60 decimal, la temperatura real es de $60 - 40 = 20$ °C. Para resolver esto se crea una función de decodificación.

Para cada PID del SAE J1979 se le crea una instrucción la cual hace el escalamiento adecuado. Dicha función recibe el PID requerido y el valor en hexa, lo convierte a decimal y hace los escalamientos necesarios para retornar el valor real. Utiliza unas subrutinas *hex2uint8* y *hex2uint16* las cuales convierten los valores hexa de 1 o 2 bytes a integer. Los PID no numéricos en esta rutina se dejan solamente comentados para referencia

```

/*****
* PID DECODER
*****/
//Routine to receive PID and data as hexa in ascii thru a pointer, and convert to
scaled integer
int PID_decoder(byte pid, char* data)
{
    int result;
    switch (pid) {
        case PIDs_Supported_01h20h:
            //0x00 List of PIDs supported from 0x01 to 0x20, bitwise coded one bit per PID

        case TROUBLE_CODE_NUMBER:
            //0x01 Number of emmision related trouble codes and MIL status

        case TROUBLE_CODES_FREEZE:
            //0x02 trouble code taht caused freeze frame storage

        case FUEL_SYSTEM_STATUS:
            //0x03 Open/closed loop status of different parts of the fuel system
            //if engine is off and ignition on, all bits are 0

        case PID_ENGINE_LOAD:
            //0x04 Calculated Engine Load
            result = hex2uint8(data) * 100 / 255;
            break;
        case PID_COOLANT_TEMP:
            //0x05 engine coolant temperature in degree celsius
            result = (hex2uint8(data) -40);
            break;
        .....

        case PID_RPM:
            //0x0C Engine crankshaft RPM, 1/4 RPM per bit
            result = hex2uint8(data) / 4;
            break;
        case PID_SPEED:
            //0x0D VEHICLE SPEED
            result = hex2uint8(data);
            break;
        .....

        case PID_ENGINE_REF_TORQUE:
            //0x63 Engine Reference Torque Nm
            result = hex2uint16(data);
            break;
        default:
            result = hex2uint8(data);
    }
    return result;
}
.....

```

7.1.3 Leer PIDs

Dadas las rutinas para decodificar y entender una respuesta OBD, el siguiente paso es preguntar dicha respuesta. Para esto, se implementa una rutina general de lectura de PID, la cual recibe de parámetro el byte PID y regresa un valor numérico con dicha respuesta. La rutina se encarga de construir el comando OBD, enviarlo al adaptador, recibir a la respuesta y decodificarla.

El primer paso es codificar la respuesta. Para esto, se convierte de byte a ASCII y se le agrega 0x01 al inicio y un carriage return al final “\r”. De esta manera, para un PID 0x0C se envía por serial UART la cadena “010C\r”. No se muestran en estos ejemplos, pero se incluyen secciones donde se envía estatus de rutinas y variables a través del puerto serial USB para monitoreo y debugeo. Adicional se utilizan rutinas propietarias write y receive, las cuales se encargan del manejo del serial a bajo nivel. La rutina de escritura de un solo PID de manera condensada es a continuación

```
/*
*****
* Principal routines to read PIDs
*****
*/

//routine to get a single PID. Writes and reads to OBDUART serial port and decodes PID
int readPID(byte PID2read){
    char buff[128]; //buffer for characters received
    char byte2chars[2]; //buffer to convert byte hexa to character
    uint8_t buttons = lcd.readButtons();

    //convert with format ascii hexa to char
    String PIDcmd = bytePID2cmd(PID2read);
    debugSerial.print("readPID Sending ");
    debugSerial.println(PIDcmd);

    //build command
    char Ccmd[8];
    sprintf(Ccmd, "01%02X\r", PID2read);
    debugSerial.print("readPID sprintf ");
    debugSerial.println(Ccmd);
    //send command and receive response
    write(Ccmd);
```

```

        receive (buff, sizeof(buff), TIMEOUT);
        char *answ = strstr(buff, byte2chars);
//find start and end of response
        while (!isAlphaNumeric(*answ)) answ++;
        char *ANSWstart = answ;
        while (!isControl(*answ)) answ++;
        char *ANSWend = answ;
//decode response
        int resultados = PID_decoder(PID2read, ANSWstart + 6);
//return integer with PID decoded result
        return resultados;

```

7.1.4 Display PIDs

El siguiente paso es utilizar esta rutina de lectura de un solo PID. Para esto, se define una lista de comandos, los cuales a través de un loop FOR se van a ir llamando uno por uno y desplegando en el LCD. Igualmente se incluyen instrucciones para despliegue de estados por serial USB y por el display, los cuales por claridad no se muestran. Esta función se repite en loop continuamente. La rutina de manera condensada es como sigue

```

/*****
* Display Routines
*****/

//Routine to read several PIDs and display thru USB-serial and LCD-display
void readPIDs() {
    // list of commands to send
    // Reference of commands sent "010C\r", "010D\r", "0104\r", "0111\r", "0105\r", "010F\r";
    static const byte cmds[] = {PID_RPM, PID_SPEED, PID_ENGINE_LOAD, PID_THROTTLE,
PID_COOLANT_TEMP, PID_INTAKE_TEMP};
    uint8_t buttons = lcd.readButtons();
    //repeat list of commands
    for (byte i = 0; i < sizeof(cmds) / sizeof(cmds[0]); i++) {
//Get PID name
        String PIDname = "Nada";
        PIDname = GetPIDName(cmds[i]);
//read single PID, send byte corresponding to PID and return intege

```

```

        byte cmd2send = cmds[i];
        int valor = readPID(cmd2send);
//update display with PID name
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print(PIDname);
//update display with value
        lcd.setCursor(11,1);
        lcd.print(":");
        lcd.setCursor(12,1);
        lcd.print(valor);

```

De esta manera queda implementada una forma de acceder y desplegar los parámetros actuales del vehículo. Después de intentar diversas combinaciones y formas de código, esta estructura es la que se comportó más estable y con mayores opciones de código reusable. Lo anterior sin perder de vista la simplicidad de implementación tal como lo dictan los requerimientos del proyecto.



Figura 7.1.4.1 – Implementación del código para lectura de PIDs

A partir de esta implementación hubo numerosas lecciones aprendidas de la manera en la cual funciona el esquema OBDII tradicional. Tal como lo dicta la enseñanza de todos los tiempos, la mejor manera de aprender a hacer algo es haciéndolo uno mismo. En este caso, al entrar de lleno en la decodificación, e incluso la codificación del OBDD a través del emulador se logró aterrizar todos los conceptos teóricos aprendidos en el estado de arte.

Si bien esta sección se introduce en el presente capítulo 7 por la estructura lógica del presente trabajo, todo este aprendizaje sirvió de base para el desarrollo de la propuesta en el capítulo 5. En especial relevancia esta la noción de los IDs. En el OBDII tradicional la estructura del mensaje es Header – modo – parameterID – Data Byte/ext address. En el OBDII plus es similar, salvo que la nomenclatura pasa a ser Header – PrimaryID – Secondary ID – Data Byte/ext address. Al ser las estructuras de mensajes prácticamente iguales, el manejo de ambos y en especial la codificación pasa a ser similar.

7.2 Decoder del OBDII plus

La rutina para decodificar el mensaje de OBDII plus empieza con el mensaje mismo. Los puertos de comunicación reciben en su modo nativo el mensaje, y lo pasan en formato string a la rutina.

El primer paso de la rutina es revisar el header 0x18DB. Si es el caso, entonces lo declara un código OBDII plus valido. El presente trabajo se ejercita sobre módulos de luces con dirección ID primaria 0xDA. Si los siguientes caracteres de dirección destino corresponden a este lo toma como un comando valido para el módulo de las luces. Los siguientes son la dirección fuente. La propuesta de protocolo los deja fijos en 0xFE por lo que es opcional verificarlos. Esta implementación simplemente los ignora.

El siguiente byte recibido es el primero de datos. Dentro de estos, el primer bit de datos en el bit Q. El siguiente bit es el C, el cual la propuesta lo establece fijo en 0. Al dejar el siguiente bit C en 0 facilita el leer el mismo. Si el carácter es 8 o mayor (1000) Q es “1”. Si es 0 o menor a 7 corresponde a “0”.

Los siguientes bits 5 a 0 a leer es el ID secundario. Este en combinación con el siguiente byte, la dirección extendida da el mensaje completo. Por medio de una función switch / case se determina el mensaje y se activan las banderas correspondientes según el bit Q (activo / desactivo).

```

/*****
OBD Decoder routine
*****/
void OBD_Decode(String CMD){
  //variables
  boolean valid_OBD = false;
  String Sec_ID;
  String Q_bit;
  boolean Q_value = false;
  String ext_addr;

  //check received commando
  CMD.toUpperCase();
  //Serial.println("ECHO OBD:" + CMD);

  //Check if header is 18DB - OBDII+
  if (CMD.startsWith("18DB")){
    if (CMD.startsWith("DA",4)){
      //Serial.println("received an OBD command for exterior lamps");
      valid_OBD = true;
    }
  }

  //decode CMD
  if (valid_OBD){
    //extract Q bit
    Q_bit = CMD.substring(8,9);
    if (Q_bit == "0"){
      Q_value = false;
    } else {
      Q_value = true;
    }

    // extract secondary ID and extended address
    Sec_ID = CMD.substring(9,10);
    ext_addr = CMD.substring(11,12);
    Serial.println("CMD received S_ID:" + Sec_ID + " Q:" + Q_value+ " xAdr:" + ext_addr );

    //decode secondary ID
    switch (Sec_ID.toInt()){
      case 1:{ //Headlamps
        headlamp = Q_value;
        break;
      }
      case 2:{ //tail, cuartitos de atras
        tail_lamp = Q_value;
        break;
      }
      case 3:{ //brake
        brake_lamp = Q_value;
        break;
      }
      case 4:{ //park, cuartitos
        park_lamp = Q_value;
        break;
      }
      case 5:{ //turn, direccionales
        if(ext_addr == "1" && (physical_addr == 1 || physical_addr == 4)){//left side
          turn_lamp = Q_value;
        }
        if(ext_addr == "7" && (physical_addr == 2 || physical_addr == 3)){//right side

```

```

        turn_lamp = Q_value;
    }
    break;
}
case 6:{ //high beam, luces altas
    highbeam_lamp = Q_value;
    break;
}
case 7:{ //hazard,
    hazard_lamp = Q_value;
    break;
}
case 8:{ //reversa,
    reverse_lamp = Q_value;
    break;
}
}
} //switch sec ID
} //end of if valid OBD
} //end of OBD decode

```

7.3 Encoder del OBDII plus

La codificación de un comando OBD se realiza paso a paso. En este ejemplo de código donde un módulo de luces regresa el estado, el comando se construye agregando secciones a un string. Se inicia con la base de header 0x18DB. Después por medio de una función switch se resuelve el ID secundario para solicitar el estado del relay correspondiente. Igualmente por medio de una función switch y el parámetro global de dirección física se determina la dirección. Por último, todas estas secciones de string se suman en un solo string con el comando OBD completo y se regresa.

```

/*****
OBD Response routine
*****/
String OBD_Response(String CMD){
    //variables
    String Q_bit;
    boolean Q_value = false;
    String Sec_ID;
    String ext_addr;
    boolean estado = false;
    String ok;
    String Ph_addr;
    String OBD_message;

    //check received commando
    CMD.toUpperCase();

    //extract Q bit
    Q_bit = CMD.substring(8,9);
    if (Q_bit == "0"){
        Q_value = false;
    }
}

```

```

    } else {
        Q_value = true;
    }

    // extract secondary ID and extended address
    Sec_ID = CMD.substring(9,10);
    ext_addr = CMD.substring(11,12);

    //decode secondary ID and check relay status
    switch (Sec_ID.toInt()){
        case 1:{ //Headlamps
            estado = relay_status(2);
            break;
        }
        case 2:{ //tail, cuartitos de atras
            estado = relay_status(1);
            break;
        }
        case 3:{ //brake
            estado = relay_status(0);
            break;
        }
        case 4:{ //park, cuartitos
            estado = relay_status(1);
            break;
        }
        case 5:{ //turn, direccionales
            if(ext_addr == "1" && (physical_addr == 1 || physical_addr == 4)){//left side
                estado = relay_status(0);
            }
            if(ext_addr == "7" && (physical_addr == 2 || physical_addr == 3)){//right side
                estado = relay_status(0);
            }
            break;
        }
        case 6:{ //high beam, luces altas
            estado = relay_status(3);
            break;
        }
        case 7:{ //hazard,
            estado = relay_status(0);
            break;
        }
        case 8:{ //reversa,
            estado = relay_status(2);
            break;
        }
    }
} //switch sec ID

//encode the status response
if(estado){
    ok = "8";
} else{
    ok = "0";
}

//encode the response extended address
switch (physical_addr){
    case 1:{ //Left Front
        Ph_addr = "09";
    }

```

```

        break;
    }
    case 2:{ //Right Front
        Ph_addr = "0F";
        break;
    }
    case 3:{ //Left Rear
        Ph_addr = "39";
        break;
    }
    case 4:{ //Right Rear
        Ph_addr = "3F";
        break;
    }
}

//build the response and return
OBD_message = "18DBD9FE" + ok + Sec_ID + Ph_addr;
Serial.println("OBD Encoded: " + OBD_message);
return OBD_message;
}

} //end of OBD response routine

```

7.4 Intérprete OBDII Plus

Las rutinas de encoder / decoder se basan en banderas globales de estado de cada comando. Estos son de manera similar al actuar del módulo supervisor de sistema, pero dedicado al hardware específico. Una vez determinadas las banderas, este módulo de estado de relays correlaciona el estado de las banderas con el estado requerido para cada relay. Por decir, la bandera de estado de la luz de reversa se relaciona con el relay R2. Si uno se activa el otro debe estarlo también y viceversa. Caso más complicado es para direccionales, los cuales comparten el relay R0 con la luz de freno y de emergencia, sin mencionar que dependen de la dirección física para saber si es derecha o izquierda.

```

/*****
Update Relays Status
*****/
void Relay_Update(){
    //depending of the flag, activates the relay

    //Relay R0, solid & flashing, brake has precedence
    if(!brake_lamp && !turn_lamp && !hazard_lamp){
        Relay(0,false);
        RelayFF(0,false);
    }
    if(brake_lamp && !turn_lamp && !hazard_lamp && (physical_addr==3 || physical_addr ==4)){
        Relay(0,true);
        RelayFF(0,false);
    }
}

```

```

    }
    if(!brake_lamp && (turn_lamp || hazard_lamp)){
        RelayFF(0,true);
    }
    if(brake_lamp && (turn_lamp || hazard_lamp)&& (physical_addr==3 || physical_addr ==4)){
        Relay(0,true);
        RelayFF(0,false);
    }

    if(park_lamp){//any physical address
        Relay(1,true);
    }
    if(!park_lamp){
        Relay(1,false);
    }

    if(reverse_lamp && (physical_addr==3 || physical_addr ==4)){
        Relay(2,true);
    }
    if(!reverse_lamp && (physical_addr==3 || physical_addr ==4)){
        Relay(2,false);
    }

    if(headlamp && (physical_addr==1 || physical_addr ==2)){
        Relay(2,true);
    }
    if(!headlamp && (physical_addr==1 || physical_addr ==2)){
        Relay(2,false);
    }

    if(highbeam_lamp && (physical_addr==1 || physical_addr ==2)){
        Relay(3,true);
    }
    if(!highbeam_lamp && (physical_addr==1 || physical_addr ==2)){
        Relay(3,false);
    }
}

```

7.5 Mapa de comandos OBDII plus para luces

El presente trabajo, como se ha comentado se limita a implementar el módulo de luces. Para facilidad del desarrollo y entendimiento del mismo, se incluye a continuación una tabla con los comandos utilizados por dicho modulo.

La tabla contempla los siguientes sistemas, los cuales como se vio en el capítulo 5 son los más comunes encontrados en todos los vehículos:

- Headlamp
- Brake
- Park

- Left Turn
- Right turn
- High Beam
- Hazard
- Reverse

Para cada uno de estos sistemas, se presenta el comando de activación (ON) o desactivación (OFF). La respuesta, denominada status tiene dos posibilidades: al activarse el sistema revisa si el foco esta encendido. En este caso regresa un “good”. Caso contrario regresa un “Bad”. Alternativamente puede contestar simplemente con un estatus de “ON”. Al apagarse, regresa un status de “OFF”

	Command		Return Status	
	ON	OFF	ON	OFF
Headlamp	18DBDAFE8100	18DBDAFE0100	18DBDBFE8100	18DBDBFE0100
Brake	18DBDAFE8300	18DBDAFE0300	18DBDBFE8300	18DBDBFE0300
Park	18DBDAFE8400	18DBDAFE0400	18DBDBFE8400	18DBDBFE0400
Left Turn	18DBDAFE8501	18DBDAFE0501	18DBDBFE8501	18DBDBFE0501
Right turn	18DBDAFE8507	18DBDAFE0507	18DBDBFE8507	18DBDBFE0507
High Beam	18DBDAFE8600	18DBDAFE0600	18DBDBFE8600	18DBDBFE0600
Hazard	18DBDAFE8700	18DBDAFE0700	18DBDBFE8700	18DBDBFE0700
Reverse	18DBDAFE8800	18DBDAFE0800	18DBDBFE8800	18DBDBFE0800

Tabla 7.5.1 Comandos OBDII plus para luces

	Command	Status	
	ask	Good	Bad
Headlamp	18DBD8FE0100	18DBD9FE8100	18DBD9FE8100
Brake	18DBD8FE0300	18DBD9FE8300	18DBD9FE8300
Park	18DBD8FE0400	18DBD9FE8400	18DBD9FE8400
Left Turn	18DBD8FE0501	18DBD9FE8501	18DBD9FE0501
Right turn	18DBD8FE0507	18DBD9FE8507	18DBD9FE0507
High Beam	18DBD8FE0600	18DBD9FE8600	18DBD9FE8600
Hazard	18DBD8FE0700	18DBD9FE8700	18DBD9FE8700
Reverse	18DBD8FE0800	18DBD9FE8800	18DBD9FE8800

Tabla 7.5.2 Respuestas de estatus OBDII plus para luces

7.6 Interface(s) de usuario

Esta sección contempla dos tipos o grupos de interfaces. La primera es una gráfica para despliegue y control de parámetros por medio de una pantalla. La segunda es la interface con todos los actuadores manuales de usuario según vayan requiriendo los subsistemas implementados. Por decir, determinar la posición del switch de luces o de direccionales. Toma los datos a partir del módulo de comunicaciones.

Para propósitos de este trabajo se acotó el sistema a los módulos de luces. El circuito por lo tanto es solamente un Arduino el cual tiene los requeridos switches manuales mapeados en sus entradas I/O. El programa simplemente detecta un cambio de estado en sus entradas y manda el correspondiente código OBDII plus. El programa correspondiente se puede analizar en la sección de hardware específico.

7.7 Puente de comunicación UART / OBD/ CAN BUS/ GUI

Esta sección establece las reglas de control de tráfico entre los sistemas ECU/OBD – Central Comp – Periféricos. El primer paso correspondiente a la lectura de códigos OBD por medio de UART/OBD está dado por el anterior interprete de PIDs

La función principal es la de establecer una comunicación regulada entre todos los sistemas. La manera más práctica es la de fungir como un nodo switch Ethernet. Esto es, el mensaje que se reciba por un puerto lo emite por todos los demás. Todos los mensajes están en formato OBDII y OBDII plus, por lo que el medio físico empleado pasa a segundo término.

Por ejemplo, una aplicación externa en teléfono como el Torque puede enviar un mensaje bluetooth de requerir la velocidad actual. El modulo toma dicho mensaje y lo reenvía por CAN-BUS y por OBD. El ECU del powertrain seria quien contestaría a través de OBD. El puente de comunicación entonces lo reenviaría por CAN-BUS y por bluetooth. La aplicación externa vería transparente la comunicación con el ECU. Al establecer comunicación con CAN-BUS, la aplicación externa también tendría acceso a los módulos OBDII plus a través de este módulo.

Los siguientes códigos son típicos representativos. Cada módulo puede tener versiones ligeramente diferentes, pero la estructura es similar. A cada mensaje recibido por un

canal enviarlo por el otro canal. La única regla es que CAN-BUS solamente puede manejar el mensaje en forma de header + data bytes. Mientras que el UART/serial/USB/bluetooth maneja strings el cual permite mensajes más completos con estatus y mensajes variados

```

/*****
  Communications initialization in setup()
  *****/
// Serial UART for Debugging output.
//Use UART D0-TX D1-RX, Onboard USB or Bluetooth
Serial.begin(38400);
Serial.println("Module Started ");
// init can bus, baudrate: 500k
if (CAN.begin(CAN_500KBPS) == CAN_OK) Serial.print("can init ok!!\r\n");
else Serial.print("Can init fail!!\r\n");
attachInterrupt(0, MCP2515_ISR, FALLING); // start interrupt

/*****
Serial Read
*****/
if (Serial.available()) {
  RCV_command = Serial.readString(); //Global string
  Serial.println("ECHO:" + RCV_command);
  CMD_received = true;
}

/*****
CAN-BUS Read
*****/
//CAN received message
if (Flag_Rcv) // check if get data
{
  Flag_Rcv = 0; // clear flag
  CAN.readMsgBuf(&len, buf); // read data, len: data length, buf: data buf
  Serial.print("CAN data received, data len = "); Serial.println(len);
  //Can receives data bytes, add header manually
  //For this project is fixed 18DB, no need to invoke getCanId(void)
  RCV_command = "18DB";
  //add data bytes
  for (int i = 0; i < len; i++)
  {
    RCV_command = RCV_command + (char)buf[i];
    Serial.print(buf[i]); Serial.print("\t");// print the data
  }
  Serial.println();
  Serial.println("CAN command received:" + RCV_command);
  CMD_received = true;
}

/*****
  update relay status
  *****/
if (CMD_received){

```

```

    OBD_Decode(RCV_command); //update OBD flags
    Relay_Update(); //update relay status depending on flags
    delay(100); //wait for relay and lamp to turn on
    OBD_Response(RCV_command); //return status
    CMD_received = false; //reset flag
}

/*****
interrupt capture for CAN BUS
*****/
void MCP2515_ISR()
{
    Flag_Recv = 1;
}

```

7.8 Supervisor de sistema

Esta sección se encarga de guardar en registro diversas banderas como lo es si el vehículo está apagado, encendido o en movimiento. Este módulo se encarga además del estatus de otros sistemas como luces prendidas o apagadas. Este módulo debe estar en constante trabajo con la comunicación OBD y los demás módulos.

Para conocer el estatus se le incorpora una función de packet sniffer. Esta función monitorea los canales de comunicación entre todos los sistemas y va actualizando las banderas correspondientes. Por decir, si se da la instrucción por parte del módulo HID de prender las luces, el packet sniffer registra que se emitió dicho comando y actualiza la bandera que registra luces prendidas. El módulo de luces, al regresar el estatus de la luz correspondiente registra si está funcionando o esta fundido.

Este mismo modulo se encarga de manejar algunas reglas básicas de acceso y comportamiento entre sistemas. Por decir, al apagar el vehículo desactivar funciones de tablero, asientos eléctricos, etc. Al activar la alarma desactivar vidrios, seguros, y monitorear el estado de las puertas.

Por último, Esta sección selecciona algunos parámetros OBD y del registro supervisor del sistema, y les hace “broadcast” periódico. Por ejemplo, emitir cada 5 segundos si el vehículo esta encendido ($RPM > 0$), velocidad, temperatura, etc. Con esta información, algunos módulos pueden tomar decisiones si son facultados para esto. Por decir, abrir ventanas si hace calor o bloquear los seguros de las puertas si el vehículo esta con la alarma activada.

7.9 Firmware específico para Hardware – Relay/Luces

Esta sección se encarga de desarrollar el código minimalista necesario para cada módulo de los subsistemas. Si bien se listan aquí al último, su desarrollo es en paralelo a todos los anteriores. En esta sección se documentan fragmentos representativos del código necesario para el funcionamiento del hardware implementado de control de luces por relays.

7.9.1 Mapeo de Arduino Nano - Luces

Con la excepción del ECU principal, la gran mayoría de los subsistemas están basados en el Arduino nano. El mapeo de puertos para el mismo, para los módulos presentados en esta sección es como sigue:

SCK	16	D13	D12	15	MISO
X	17	3V3	D11	14	MOSI
X	18	AR	D10	13	CS
VBATT	19	A0	D9	12	R6
VR0	20	A1	D8	11	R5
VR1	21	A2	D7	10	R4
VR2	22	A3	D6	9	R3
VR3	23	A4	D5	8	R2
VR4	24	A5	D4	7	R1
DIP_SW	25	A6	D3	6	R0
PUSHBTN	26	A7	D2	5	INT
+5V	27	+5V	GND	4	GND
X	28	RST	RST	3	X
GND	29	GND	RX0	2	UART RX
VIN	30	VIN	TX0	1	UART TX

Figura 7.9.1.1 – Mapeo pines Arduino nano

Los pines R0 a R6 son para activación de relays. Estos están mapeados a los puertos digitales D3 a D9 correspondientes. En el firmware se mapean según su función usando pinmode, y se activan/desactivan por medio de la función digitalWrite

```
pinMode(R0, OUTPUT);  
digitalWrite(R0, HIGH);
```

El mapeo completo de relays queda como en la tabla 7.9.1.1

Relay	Salida I/O	Entrada Analógica	Función
R0	D3	A1	Brake / Hazard/ Turn
R1	D4	A2	Park (cuartos)
R2	D5	A3	Headlamp / Reverse
R3	D6	A4	Luces Altas

Tabla 7.9.1.1 Comandos OBDII plus para luces

Los puertos analógicos A1 al A5 se utilizan para leer el voltaje de salida de la tarjeta hacia cada canal. Estos son denominados VR0 a VR4. La lectura por lo tanto es después de la resistencia de shunt sensora de corriente. El canal A0 está dedicado a leer el voltaje de batería de entrada. Los canales A1 al A5 están dedicados a leer el voltaje de salida de la tarjeta, después de la resistencia shunt.

Esto para determinar la condición de la carga. Los puertos A6 y A7 correspondientes a Dip_Sw y pushbutton leen el voltaje proveniente del divisor formado por los switches de activación y el dip switch de código.

Los switches y resistencias de dicha red están dispuestos de manera de entregar un voltaje diferente para cada activación, una a la vez. Dichas diferencias son analizadas por medio de un CASE o un IF anidado, para determinar por el nivel del valor leído cual switch fue activado. Para el caso de los 4 switches, los valores de A/D 300, 500, 700 y 900 son suficientes para discernir los casos de los switches. Un valor <300 significa ninguno, $300 < x < 500$ el switch R0, y así sucesivo.

El dipSwitch codifica la dirección física del dispositivo. Recordando la intención es diseñar tableros que puedan funcionar en diferentes posiciones físicas del vehículo, por lo tanto debe contar con una manera de saber en qué posición está. La codificación es simple. En la inicialización (setup) se lee el valor. Solamente un dip switch a la vez puede estar encendido. El mapeo de switch a dirección física es directo, switch 1 con dirección 0xn1, sw 2 con 0xn2, ningún switch con 0xn0 y sucesivo

Los pines CS (D10), MOSI (D11), MISO (D12), SCK (D13) e INT (D2) son para la comunicación con el tablero CAN BUS. Para el manejo de dicho puerto se utiliza la librería estándar CAN-BUS, la cual de manera similar a la librería Serial abre un puerto

de comunicación, solamente que mapeado para los pines del CAN BUS. Para propósitos de debug se deja disponible el puerto UART por medio de RX0 y TX0, el cual usa la librería Arduino estándar Serial.objeto utilizada por el puerto USB.

7.9.2 Activación de relays

Una vez resuelto por medio del encoder / decoder del OBDII Plus la necesidad de activar una salida, se le da la orden al relay correspondiente. Para el manejo de los relays se utilizan diversas rutinas.

La primera es el manejo directo del relay. Se utiliza la función Relay() la cual recibe el número de relay y el estado requerido. Esta rutina activa o desactiva el I/O correspondiente a la acción requerida

```

/*****
Relays handle routine
*****/
void Relay(int Rn, boolean estado){
  //Serial.println("Relay routine, asked " + String(Rn,DEC) + estado);
  switch (Rn){
    case 0:{
      if (estado){
        digitalWrite(R0, HIGH);
      } else{
        digitalWrite(R0, LOW);
      }
      break;
    }
  }

  -----

  case 3:{
    if (estado){
      digitalWrite(R3, HIGH);
    } else{
      digitalWrite(R3, LOW);
    }
    break;//if else
  }//case
} //switch
} //relay routine
```

Caso especial es para las luces intermitentes, las cuales requieren un ciclo de 7 veces prender/apagar en 10 segundos. Este ciclo se determinó por observación de al menos unos 50 vehículos de la mayor diversidad posible de modelos y marcas. Más del 90 % de los casos fueron 7 ciclos en 10 segundos, los demás fueron 6 u 8 ciclos por lo que la media se determinó en 7.

Para los ciclos se utiliza la librería Timer1 desarrollada por la comunidad Arduino. Esta librería se programa con el ciclo mencionado. Al término del timer, se activa una interrupción que llama la subrutina toggle la cual cambia el estado del relay.

```

/*****
Relays flip/flop handle routine
for the turn/hazard flash lights
*****/
void RelayFF(int Rn, boolean estado){
    if (estado == true){
        Timer1.attachInterrupt(toggle);
    }
    else{//deactivate toggle
        Timer1.detachInterrupt();
    }
}

void toggle()
{
    digitalWrite(R0, digitalRead(R0) ^ 1);
}

```

Una rutina especial se implementa en startup, la cual cicla los relays a manera de lamptest. Por el puerto serial regresa valores de estado en cada activación.

```

/*****
    Relay Outputs
*****/
//configure output relay pins
pinMode(R0, OUTPUT);
pinMode(R1, OUTPUT);
pinMode(R2, OUTPUT);
pinMode(R3, OUTPUT);

Serial.println("TEST ON/OFF Relay");
//prueba pines. cycle all on/off as lamp test at startup
for (int i=0; i<=3; i++){
    //Serial.println("TEST ON/OFF Relay R" + String(i,DEC));
    Relay(i, true);
}

```

```

        delay (500);
        boolean estado = relay_status(i);
        Serial.println("TEST ON/OFF Relay R" + String(i,DEC) + " returned lamp status:
" + estado);
        Relay(i, false);
    }

```

A manera de prueba se incorporaron pushbutton correspondientes a cada relay. Estos son leídos por el puerto analógico y dependiendo del nivel se interpreta cual relay es requerido activarse. La rutina se cicla mientras permanezca alto dicho valor, esto es mientras esté presionado el botón. Por el serial se regresan valores de monitoreo

```

/*****
Relays Test routine
*****/
void RelayTest(int VPB){
    //input voltage of pushbutton resistance divider
    int i = 0;
    //first button
    if (VPB > LIM1 && VPB <= LIM2){
        Serial.println("Test R0 ON");
        Relay(0, true);
        VBAT = analogRead(VBAT_pin);
        VR0 = analogRead(VR0_pin);
        while (analogRead(PB_pin) >= LIM1){
            //just wait to release button
            for (i= 0; i<=5; i++) {
                VBAT = (VBAT + analogRead(VBAT_pin)) /2;
                VR0 = (VR0 + analogRead(VR0_pin)) /2;
            }
            VBAT = VBAT + VBAT_cal;
            VR0 = VR0 + VR0_cal;
            Serial.println("VPB=" + String(VPB,DEC) + " VBAT= " + String(VBAT,DEC)
+ " VR0= " + String(VR0,DEC) + " Detection= " + String((VBAT-VR0),DEC));
        }
        Relay(0, false);
    }

    ----

    //4th button
    if (VPB > LIM4){
        Serial.println("Test R3 ON");
        Relay(3, true);
        VBAT = analogRead(VBAT_pin);
        VR3 = analogRead(VR3_pin);
        while (analogRead(PB_pin) >= LIM1){
            //just wait to release button
            for (i= 0; i<=5; i++) {
                VBAT = (VBAT + analogRead(VBAT_pin)) /2;

```

```

        VR3 = (VR3 + analogRead(VR3_pin)) /2;
    }
    VBAT = VBAT + VBAT_cal;
    VR3 = VR3 + VR3_cal;
    Serial.println("VPB=" + String(VPB,DEC) + " VBAT= " + String(VBAT,DEC)
+ " VR0= " + String(VR3,DEC) + " Detection= " + String((VBAT-VR3),DEC));
    }
    Relay(3, false);
}
    Serial.println("End of Relay test");
}
}

```

Además de los relays, al inicio (setup) del programa se lee el valor del divisor de voltaje correspondiente a la dirección física. De manera similar se compara el valor y se determina en una variable global la dirección física asignada al módulo. Esta dirección se utiliza posteriormente en el encoder / decoder OBDII plus para determinar si el comando requerido corresponde al módulo en cuestión o es para el modulo vecino.

```

/*****
Physical address set
*****/
//set analog reference to internal 5V
analogReference(DEFAULT);
//read 10 times to stabilize
for (i= 0; i<=10; i++) {
    analogRead(DipSw_pin);
}
//read analog value 5 times and average to filter noise
VDipSw = analogRead(DipSw_pin);
for (int i= 0; i<=5; i++) {
    VDipSw = (VDipSw + analogRead(DipSw_pin)) /2;
}
Serial.println("Voltage dip switch read " + String(VDipSw,DEC));
//determine dip switch and address
physical_addr = 0;
if (VDipSw <= LIM1){
    physical_addr = 0;
}
if ((VDipSw > LIM1) && (VDipSw<= LIM2)){
    physical_addr = 1;
}
if ((VDipSw > LIM2) && (VDipSw<= LIM3)){
    physical_addr = 2;
}
if ((VDipSw > LIM3) && (VDipSw<= LIM4)){
    physical_addr = 3;
}
}

```



```

    if (VDipSw > LIM4){
        physical_addr = 4;
    }
    Serial.println("Physical address " + String(physical_addr,DEC));

} //end of setup

```

Por último, la rutina principal loop() incluye monitoreo de los pushbutton de activación manual de relays y monitoreo de puertos de comunicación como se mencionó en los puntos correspondientes. Adicional a los anteriores se incorporó un heart bit o mensaje periódico cada aprox 5 segundos para saber que aún está funcionando el modulo

```

/*****
Heart Beat - i'm stil alive!!
*****/
if (heartbeat == 8191){
    Serial.println("heartbeat");
    heartbeat = 0;
}
    heartbeat = heartbeat + 1;
} //end of Main routine - Loop

```

7.9.3 Sensor de luz en funcionamiento

El proyecto contempla que debe de poder determinarse el estado de la carga conectada. Esto es, si el foco esta prendido o fundido. Cuando la bandera de activar la carga, o para este ejemplo prender la luz este activado, se realiza una verificación de las lecturas del puerto análogo. El puerto A0 lee el voltaje de la batería VBATT, o de la entrada al módulo. Los puertos A1 al A5 leen los voltajes de los relays R0 al R4 después del shunt. Este voltaje VBATT, al restarse del voltaje después del shunt da una lectura de diferencial de voltaje a través del shunt, el cual a la vez es proporcional a la corriente. Dado que no es requerimiento el saber exactamente la corriente consumida, sino solamente presencia de la misma, se establece un comparador simple. Se inicia por leer y promediar 5 veces cada canal para filtrar ruidos y variaciones, por medio de un ciclo For. Después al voltaje de batería se le resta el del relay. Cuando esta prendido el relay, si la diferencia es mayor a 5 (nuevamente para tener en cuenta ruidos) se considera que la luz está consumiendo corriente por lo cual

no está fundido. Si la diferencia es menor, se considera que no hay consumo de corriente por lo cual se marca falla. Lo anterior se realiza por medio de la función `relay_status`. Se consulta para determinado relay y regresa un booleano si está activo o no lo está.

La rutina utiliza unas variables especiales de calibración. Estos son valores determinados empíricamente para tener una lectura de 0 corriente con la luz desconectada. Se agregan como offset al valor leído después del shunt

```
/******  
Status Read  
*****/  
boolean relay_status(int Rn){  
    //determines a blown bulbe. Measures battery voltage and  
    //battery after shunt, to determine current consumption  
    int Vshunt;  
    int Vshunt_pin;  
    int Voffset;  
    boolean estado = true;  
  
    //determines analog pin for the query relay  
    switch (Rn){  
        case 0:{  
            Vshunt_pin = VR0_pin;  
            Voffset = VR0_cal;  
            break;  
        }  
        case 1:{  
            Vshunt_pin = VR1_pin;  
            Voffset = VR1_cal;  
            break;  
        }  
        case 2:{  
            Vshunt_pin = VR2_pin;  
            Voffset = VR2_cal;  
            break;  
        }  
        case 3:{  
            Vshunt_pin = VR3_pin;  
            Voffset = VR3_cal;  
            break;  
        }  
    }  
  
    //read analog values, averages to reduce noise  
    VBAT = analogRead(VBAT_pin);  
    Vshunt = (Vshunt + analogRead(Vshunt_pin));  
    for (int i= 0; i<=5; i++) {  
        VBAT = (VBAT + analogRead(VBAT_pin)) /2;
```

```

        Vshunt = (Vshunt + analogRead(Vshunt_pin)) /2;
    }
    VBAT = VBAT + VBAT_cal;
    Vshunt = Vshunt + Voffset;

    //determine current consumption, therefore working bulbe
    if((VBAT - Vshunt) > 3){
        estado = true;
    }
    else {
        estado = false;
    }
    //Serial.println("Rstatus:" + String(Rn,DEC) + " VB:" + String(VBAT,DEC)+
    Voff:"+String(Voffset,DEC) + " Vshunt:"+String(Vshunt,DEC)+" estado:"+estado);
    return estado;
}

```

At initialization section

```

//calibration values, to offset current to 0 with burn lamp
int VBAT_cal = 0;
int VR0_cal = -8;
int VR1_cal = -10;
int VR2_cal = -7;
int VR3_cal = 0;

```

7.10 Firmware específico para Hardware – HID

Al igual que en el módulo de relays de luces, esta sección se encarga de desarrollar el código minimalista necesario para cada módulo de los subsistemas. En esta sección se documentan fragmentos representativos del código necesario para el funcionamiento del hardware implementado que simula la interface de usuario (del inglés HID – Human Interface Device).

7.10.1 Mapeo de Arduino Nano - HID

Con la excepción del ECU principal, la gran mayoría de los subsistemas están basados en el Arduino nano. El mapeo de puertos para el mismo, para los módulos presentados en esta sección es como sigue:

SCK	16	D13	D12	15	MISO
X	17	3V3	D11	14	MOSI
X	18	AR	D10	13	CS
PARK	19	A0	D9	12	RIGHT TURN
HEADLAMP	20	A1	D8	11	LEFT TURN
HIHG BEAM	21	A2	D7	10	REVERSE
BRAKE	22	A3	D6	9	BRAKE
REVERSE	23	A4	D5	8	HIGH BEAM
HAZARD	24	A5	D4	7	HEADLAMO
LEFT TURN	25	A6	D3	6	PARK
RIGHT TURN	26	A7	D2	5	INT
+5V	27	+5V	GND	4	GND
X	28	RST	RST	3	X
GND	29	GND	RX0	2	UART RX
VIN	30	VIN	TX0	1	UART TX

Figura 7.10.1.1 – Mapeo pines Arduino nano

Los pines A0 al A7 son entradas de los diferentes pushbutton. Los puertos analógicos se configuran por software como puertos I/O digitales como cualquier otro pin digital. Adicional se activa el resistor pull up interno de 20K ohm. Con este, se conectan los pushbuttons con una terminal a tierra y la otra directa al Arduino. La lectura digital de los pushbuttons es por lo tanto invertida. Si esta desactivado entonces está conectada al pull up por lo que marca HIGH. Si se presiona el botón se conecta a tierra por lo que pasa a ser LOW. Para invertir dicha lógica, en toda lectura de botones se niega la señal por medio del operador “!”..

```

/*****
Pins Definitions
*****/
//define pins for the current sensing
//Compare Vbatt input and voltage after shunt
int VBAT_pin = A0;
int VR0_pin = A1;
int VR1_pin = A2;
int VR2_pin = A3;
int VR3_pin = A4;
//define pins for siwtch inputs
//use voltage divider and sense levels to determine input
int DipSw_pin = A6;
int PB_pin = A7;
//Relay pins
int R0 = 3;
int R1 = 4;

```

```

int R2 = 5;
int R3 = 6;
...

/*****
    Relay Outputs
    *****/
//configure output relay pins
pinMode(R0, OUTPUT);
pinMode(R1, OUTPUT);
pinMode(R2, OUTPUT);
pinMode(R3, OUTPUT);
/*****

    Configure Pins
    *****/
//configure input pins
pinMode(park_pin, INPUT);
pinMode(headlamp_pin, INPUT);
pinMode(highbeam_pin, INPUT);
pinMode(brake_pin, INPUT);
pinMode(reverse_pin, INPUT);
pinMode(hazard_pin, INPUT);
//pinMode(left_pin, INPUT); //A6 and A7 can not be turn on digital on nano
//pinMode(right_pin, INPUT);

//activate internal pull up resistor 20K
digitalWrite(park_pin, HIGH);
digitalWrite(headlamp_pin, HIGH);
digitalWrite(highbeam_pin, HIGH);
digitalWrite(brake_pin, HIGH);
digitalWrite(reverse_pin, HIGH);
digitalWrite(hazard_pin, HIGH);
//digitalWrite(left_pin, HIGH);
//digitalWrite(right_pin, HIGH);

...
//lectura de un pin
    boolean check_pin1 = !digitalRead(park_pin);

```

Los pines de los leds de salida se mapean a los I/O correspondientes D3 al D9.

```

//configure output relay pins
pinMode(park_led, OUTPUT);
pinMode(headlamp_led, OUTPUT);
pinMode(highbeam_led, OUTPUT);
pinMode(brake_led, OUTPUT);
pinMode(reverse_led, OUTPUT);
pinMode(left_led, OUTPUT);
pinMode(right_led, OUTPUT);
...

```

7.10.2 Banderas de Estado

Internamente, el programa maneja banderas de estado para todas las diversas funciones que maneja. Estos son tanto para los comandos como para los estados

```
/******  
Declaration and initialization of variables  
*****/  
//Activation flags  
boolean brake_lamp = false;  
boolean turn_lamp = false;  
boolean hazard_lamp = false;  
boolean park_lamp = false;  
boolean reverse_lamp = false;  
boolean headlamp = false;  
boolean highbeam_lamp = false;  
boolean tail_lamp = false;  
//Status flags  
boolean brake_lamp_OK = true;  
boolean turn_lamp_OK = true;  
boolean hazard_lamp_OK = true;  
boolean park_lamp_OK = true;  
boolean reverse_lamp_OK = true;  
boolean headlamp_OK = true;  
boolean highbeam_lamp_OK = true;  
boolean tail_lamp_OK = true;
```

A través de estar monitoreando estos estados es como el programa active o desactive funciones, manda comandos o se entera de los diversos estados del Sistema.

7.10.3 Main Loop

La rutina principal loop() se encarga de estar leyendo los puertos de comunicación y estado de los botones. Con respecto a los puertos de comunicación, así como se explicó en secciones anteriores lo que se recibe por uno se emite por el otro.

Dos rutinas son la excepción: la lectura de botones y la activación de leds. La rutina de botones en esencia lee si se activa un botón y genera un comando. Este comando se ingresa al registro global del programa RCV_command como si se hubiese recibido por un puerto de comunicación. Utiliza las subrutinas Read_Buttons() para determinar si se presionó un botón, y cual. Después la toggle_flag() la cual simplemente cambia el estado de la bandera correspondiente.

Posteriormente se genera el comando OBDII plus correspondiente a partir de la bandera activada. Este comando adicionalmente se manda por CAN-BUS a los demás dispositivos. Por último, activa la bandera CMD_received para indicar a las demás rutinas que se “recibió” un comando

```

/*****
read buttons. Determine if one was activated
and create the OBDII Plus command
*****/
activated_button = Read_Buttons();
if (activated_button > 0){
    //Serial.println("button press# "+String(activated_button,DEC));
    toggle_Flag(activated_button);
    RCV_command = OBD_Encode(activated_button);
    Serial.println("ECHO button: " + RCV_command);
    CAN-BUS_send(RCV_command);
    CMD_received = true;
}

```

Una vez generado el comando, prosiguen las rutinas de los puertos de comunicación que reciben y envían comandos. En caso de recibirse un comando por un puerto, en vez de generarse internamente con los botones se actualiza el registro de comando RCV_command correspondiente

El siguiente paso es decodificar dicho comando para activar el led correspondiente. Para esto se utiliza la subrutina OBD_decode() la cual recibe el comando y actualiza las banderas correspondientes. La rutina update_leds() repasa todas las banderas y activa las correspondientes. Después de mandar el comando correspondiente, le solicita el estatus a cada uno de los módulos en orden.

```

/*****
update relay status
*****/
if (CMD_received){
    OBD_Decode(RCV_command); //update OBD flags
    update_leds(); //update leds according to flags
    CMD_received = false; //reset received flag

    //ask for modules status
    Sec_ID = RCV_command.substring(9,10);

    Status_request = "18DBD8FE0" + Sec_ID + "09";
    CAN-BUS_send(Status_request);
    Serial.println("CAN status send:" + Status_request);
    Status_receive = CAN-BUS_receive();
}

```

```

Serial.println("CAN status received:" + Status_receive);
delay(250);

Status_request = "18DBD8FE0" + Sec_ID + "0F";
CAN-BUS_send(Status_request);
Serial.println("CAN status send:" + Status_request);
Status_receive = CAN-BUS_receive();
Serial.println("CAN status received:" + Status_receive);
delay(250);

Status_request = "18DBD8FE0" + Sec_ID + "39";
CAN-BUS_send(Status_request);
Serial.println("CAN status send:" + Status_request);
Status_receive = CAN-BUS_receive();
Serial.println("CAN status received:" + Status_receive);
delay(250);

Status_request = "18DBD8FE0" + Sec_ID + "3F";
CAN-BUS_send(Status_request);
Serial.println("CAN status send:" + Status_request);
Status_receive = CAN-BUS_receive();
Serial.println("CAN status received:" + Status_receive);
delay(250);
}

```

7.10.4 Lectura de botones

La manera de simular las entradas de usuario, como lo es el switch de la direccional derecha, es por medio de pushbuttons. Estos se configuran como detección de flanco de subida por medio de una rutina de software. La rutina guarda el estado anterior por medio de la bandera ej park_last. Si en una vuelta de la rutina principal loop estas banderas son diferentes, quiere decir que hubo una transición en la señal proveniente del pushbuttons. Si la señal del pushbutton es TRUE/positiva quiere decir que la transición fue hacia alto.

La rutina Read_Buttons() regresa el número de botón presionado. Si no hubo ninguno presionado, regresa 0

```

/*****
Check if a button is active
*****/
int Read_Buttons(){
  //returns the analog-in activated + 1, A0 return 1
  //return 0 if no button was activated
  int resultado = 0;
  int VPBL = 0; int VPBR = 0; int i = 0;

  //create a software up transition detection
  //buttons are active on low

```



```

boolean check_pin1 = !digitalRead(park_pin);
if (check_pin1 != park_last){ //detect transition
    if (check_pin1){ //button pressed
        resultado = 1;
        Serial.println ("button park press");
    }
}
park_last = check_pin1; //for the next loop

boolean check_pin2 = !digitalRead(headlamp_pin);
if (check_pin2 != headlamp_last){ //detect transition
    if (check_pin2){ //button pressed
        resultado = 2;
        Serial.println ("button headlamp press");
    }
}
headlamp_last = check_pin2; //for the next loop

boolean check_pin3 = !digitalRead(highbeam_pin);
if (check_pin3 != highbeam_last){ //detect transition
    if (check_pin3){ //button pressed
        resultado = 3;
        Serial.println ("button highbeam press");
    }
}
highbeam_last = check_pin3; //for the next loop

boolean check_pin4 = !digitalRead(brake_pin);
if (check_pin4 != brake_last){ //detect transition
    if (check_pin4){ //button pressed
        resultado = 4;
        Serial.println ("button brake press");
    }
}
brake_last = check_pin4; //for the next loop

boolean check_pin5 = !digitalRead(reverse_pin);
if (check_pin5 != reverse_last){ //detect transition
    if (check_pin5){ //button pressed
        resultado = 5;
        Serial.println ("button reverse press");
    }
}
reverse_last = check_pin5; //for the next loop

boolean check_pin6 = !digitalRead(hazard_pin);
if (check_pin6 != hazard_last){ //detect transition
    if (check_pin6){ //button pressed
        resultado = 6;
        Serial.println ("button hazard press");
    }
}
hazard_last = check_pin6; //for the next loop

//read analog value 5 times and average to filter noise

```

```

VPBL = analogRead(left_pin);
for (i= 0; i<=5; i++) {
    VPBL = (VPBL + analogRead(left_pin)) /2;
}
//Serial.println("left "+String(VPBL,DEC));
boolean check_pin7 = false;
if (VPBL <100){
    check_pin7 = true;
}
if (check_pin7 != left_last){ //detect transition
    if (check_pin7){ //button pressed
        resultado = 7;
        Serial.println ("button left press");
    }
}
left_last = check_pin7; //for the next loop

VPBR = analogRead(right_pin);
for (i= 0; i<=5; i++) {
    VPBR = (VPBR + analogRead(right_pin)) /2;
}
//Serial.println("right "+String(VPBR,DEC));
boolean check_pin8=false;
if (VPBR <100){
    check_pin8 = true;
}
if (check_pin8 != right_last){ //detect transition
    if (check_pin8){ //button pressed
        resultado = 8;
        Serial.println ("button right press");
    }
}
right_last = check_pin8; //for the next loop

return resultado;
} //end Read_Buttons()

```

7.10.5 OBD Encode

La rutina OBD_Encode() recibe el número de botón que se presionó. A partir de esta y utilizando una función switch va construyendo el string del comando OBDII plus correspondiente.

Inicia el string con "18DBDAFE". Este string incluye el header y el ID primario correspondiente al sistema de luces. La dirección de inicio queda en FE como se recomienda.

Una vez seleccionado por el switch cual comando fue invocado, la rutina llama a la bandera correspondiente y activa el bit Q junto con el ID secundario. Este se agrega al string ya formado. Si hay dirección extendida como en el caso de las direccionales, este se agrega también

```

/*****
OBD command Encode routine
*****/
String OBD_Encode(int pin){
    //variables
    boolean valid_OBD = false;
    String Sec_ID;
    String Q_bit;
    boolean Q_value = false;
    String ext_addr;

    //construct the PBD command line step by step
    String CMD = "";

    //header, Target addr "DA" for command lights, fixed "FE" source addr
    CMD = "18DBDAFE";

    //Q bit, Sec ID, ext address
    switch (pin){
        case 1:{
            if (park_lamp){
                CMD = CMD + "8400";
            } else {
                CMD = CMD + "0400";
            }
            break;
        }
        case 2:{
            if (headlamp){
                CMD = CMD + "8100";
            } else {
                CMD = CMD + "0100";
            }
            break;
        }
        case 3:{
            if (highbeam_lamp){
                CMD = CMD + "8600";
            } else {
                CMD = CMD + "0600";
            }
            break;
        }
        case 4:{
            if (brake_lamp){
                CMD = CMD + "8300";
            }
        }
    }
}
```

```

        } else {
            CMD = CMD + "0300";
        }
        break;
    }
    case 5:{
        if (reverse_lamp){
            CMD = CMD + "8800";
        } else {
            CMD = CMD + "0800";
        }
        break;
    }
    case 6:{
        if (hazard_lamp){
            CMD = CMD + "8700";
        } else {
            CMD = CMD + "0700";
        }
        break;
    }
    case 7:{
        if (left_lamp){
            CMD = CMD + "8501";
        } else {
            CMD = CMD + "0501";
        }
        break;
    }
    case 8:{
        if (right_lamp){
            CMD = CMD + "8507";
        } else {
            CMD = CMD + "0507";
        }
        break;
    }
}
} //end switch

return CMD;
} //end OBD encode

```

7.10.6 OBD Decode

La rutina de decodificación es muy similar a la de la rutina utilizada en el módulo de relay de luces. Va descomponiendo el string del comando por secciones. Primero extrae el header y lo valida de ser OBDII plus. Después el secondary ID el cual lo valida también de ser comando de módulo de luces. Enseguida extrae el bit Q para saber si es activación o desactivación. Luego el secondary ID. Con este y una función switch

actualiza la bandera correspondiente al secondary ID con el valor del bit Q (activar o desactivar).

```

/*****
OBD Decoder routine
*****/
void OBD_Decode(String CMD){
    //variables
    boolean valid_OBD = false;
    String Sec_ID;
    String Q_bit;
    boolean Q_value = false;
    String ext_addr;

    //check received commando
    CMD.toUpperCase();
    //Serial.println("ECHO OBD:" + CMD);

    //Check if header is 18DB - OBDII+
    if (CMD.startsWith("18DB")){
        if (CMD.startsWith("DA",4)){
            //Serial.println("received an OBD command for exterior lamps");
            valid_OBD = true;
        }
    }

    //decode CMD
    if (valid_OBD){
        //extract Q bit
        Q_bit = CMD.substring(8,9);
        if (Q_bit == "0"){
            Q_value = false;
        } else {
            Q_value = true;
        }

        // extract secondary ID and extended address
        Sec_ID = CMD.substring(9,10);
        ext_addr = CMD.substring(11,12);
        Serial.println("CMD received S_ID:" + Sec_ID + " Q:" + Q_value+ " xAdr:" +
ext_addr );

        //decode secondary ID
        switch (Sec_ID.toInt()){
            case 1:{ //Headlamps
                headlamp = Q_value;
                break;
            }
            case 2:{ //tail, cuartitos de atras
                tail_lamp = Q_value;
                break;
            }
            case 3:{ //brake

```

```

        brake_lamp = Q_value;
        break;
    }
    case 4:{ //park, cuartitos
        park_lamp = Q_value;
        break;
    }
    case 5:{ //turn, direccionales
        if(ext_addr == "1" && (physical_addr == 1 || physical_addr ==
4)){//left side
            turn_lamp = Q_value;
        }
        if(ext_addr == "7" && (physical_addr == 2 || physical_addr ==
3)){//right side
            turn_lamp = Q_value;
        }
        break;
    }
    case 6:{ //high beam, luces altas
        highbeam_lamp = Q_value;
        break;
    }
    case 7:{ //hazard,
        hazard_lamp = Q_value;
        break;
    }
    case 8:{ //reversa,
        reverse_lamp = Q_value;
        break;
    }
}
} //switch sec ID
} //end of if valid OBD
} //end of OBD decode

```

Página en blanco

8 Validación

Una vez implementado, se procede a la validación. Esta se planea hacer en dos ambientes: simulación y aplicación real con ambiente reducido. Debido a la gran diversidad de sistemas y la dificultad de probar físicamente todos así como su combinación, es por esto que se opta por estas dos situaciones

8.1 Simulación en ambiente virtual

La primera es una validación de escritorio, es decir presentar el problema y la solución (universo de sistemas posibles y su codificación) en papel uno frente a otro y analizar punto a punto dicha solución. El proceso seleccionado para la validación es:

- Seleccionar el comando a validar. Para efectos de este ejercicio se selecciona de manera aleatoria
- Construir un comando OBDII plus tal cual fuera a enviarse por CAN-BUS, utilizando los algoritmos desarrollados en capítulos anteriores.
- Decodificar dicho comando recibido y extraer la acción requerida
- Comparar el comando original con el decodificado.

8.1.1 Algoritmos de codificación y decodificación

La base de la propuesta de estándar se encuentra en la tabla 5.11.2.1, la cual se reproduce a continuación para comodidad en la tabla 8.1.1.1

Tal como se explicaba, el formato del mensaje consta de un header de 29 bits y hasta 3 bytes de datos. La mayoría de los mensajes utilizan solo dos bytes. Para facilidad, diferentes opciones se dejaron fijas seleccionando la mejor opción entre las que presentan los diferentes estándares. La primera parte del header se utiliza fijo en 0x18DB. Es seguido por el ID primario del comando requerido. La dirección fuente se deja fija en 0xFE llamada general. De los datos, el primer bit transmitido es el Q el cual indica activar/desactivar, prender/apagar, etc. El resto de los bits del primer byte es el ID secundario del comando requerido. El siguiente byte transmitido es la dirección extendida. Para el caso de OBDIIplus prácticamente todos los comandos lo utilizan.

HEADER – 29 bits			3 Bytes de datos (1 datos + 2 si se requieren)				
Header	Target Address	Source Address	Bit 7 “Q”	Bit 6 “C”	Bit 5-0	Dirección extendida	Byte de datos
0x18DB	1 Byte 0xNN	0xFE	1/0	0	0x0N	1 Byte 0xNN	1 Byte 0xNN
Header - Fijo	ID Primario por J2178-4	Dirección física – 0xFE llamada a todos los nodos	1- Activo 0-Inactivo	Fijo en 0	ID secundario Por J2178-4	Direccionamiento extendido (Default 0x00 todos)	Dato paramétrico (Si se requiere)

Tabla 8.1.1.1 Estructura del mensaje en OBDII-plus (Fuente: Elaboración propia)

La manera de codificar un comando es ir construyéndolo paso a paso:

- Iniciar con 0x18DB
- Seleccionar el ID primario
- Agregar 0xFE dirección fuente fija
- Codificar el bit Q. Usualmente los siguientes bits del primer byte son “0”, por lo que activar se vuelve un 0x8 y desactivar un 0x0
- Agregar el ID secundario
- Agregar la dirección extendida

Ejemplo Codificación: En la mayoría de los casos, los sistemas del Body son tan diversos y numerosos que requieren un direccionamiento extendido adicional. Este se implementa por medio de agregar un byte al string. Para el caso de prender las direccional derecha (turn right) se utiliza la tabla 41 SAE J2178-4. El ID primario es 0xDA para comando y 0xDB para estado. Esta tabla además llama a la 9.8 (tabla 59) la cual especifica físicamente cual luz deseamos. Por lo tanto, para enviar el comando (0xDA) dirigiéndose a ningún nodo en particular (0xFE) de pasar a estado activo (C=1, 0x8) la direccional (0x05) derecha (0x07) el string quedaría 18DB DA FE 85 07.

8.36 Exterior Lamps—(See Table 41.)

Primary ID: Exterior Lamps
 Command ID: \$DA
 Status ID: \$DB

TABLE 41—EXTERIOR LAMPS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Headlamp On/Off	01	L/R	On	Off	8.8	--
Tail Lamp On/Off	02	L/R	On	Off	8.8	--
Brake Lamp On/Off	03	L/R	On	Off	8.8	--
Park Lamp On/Off	04	L/R	On	Off	8.8	--
Turn Lamp On/Off	05	L/R	On	Off	8.8	--
High Beam Lamp On/Off	06	L/R	On	Off	8.8	--
Hazard Lamp On/Off	07	L/R	On	Off	8.8	--
Reverse Lamp On/Off	08	L/R	On	Off	8.8	--
Fog Lamp On/Off	09	L/R	On	Off	8.8	--
Daytime Running Lamp On/Off	0A	L/R	On	Off	8.8	--
Spot Lamp On/Off	0B	L/R	On	Off	8.8	--
Cargo Lamp On/Off	0C	L/R	On	Off	8.8	--
Cornering Lamp On/Off	0D	L/R	On	Off	8.8	--
Driving Lamp On/Off	0E	L/R	On	Off	8.8	--
Coach Lamp On/Off	0F	L/R	On	Off	8.8	--
Autolamp Delay	10	L/R	E	D	8.8	A014
Flash-to-Pass	11	L/R	E	D	8.8	--
Remote Headlamp On/Off w/Transmitter ID	12	L/R	On	Off	8.8	C001
Remote Headlamp On/Off	13	L/R	On	Off	8.8	--
Headlamp Sw. Active	21	R	Y	N	8.8	--
Right Turn Sw. Active	22	R	Y	N	8.8	--
Park Lamp Sw. Active	24	R	Y	N	8.8	--
Left Turn Sw. Active	25	R	Y	N	8.8	--
High Beam Sw. Active	26	R	Y	N	8.8	--
Hazard Sw. Active	27	R	Y	N	8.8	--
Fog Lamp Sw. Active	28	R	Y	N	8.8	--
Driving Lamp Sw. Active	29	R	Y	N	8.8	--

9.8 External Lamps—The extended address assignments for external lamps are in Table 59.

TABLE 59—EXTENDED ADDRESS ASSIGNMENTS FOR EXTERNAL LAMPS

Address	Description
00	ALL
01	Left Side (Turn Signal)
07	Right Side (Turn Signal)
08	All Front
09	Left Front
0F	Right Front
38	All Rear
39	Left Rear
3C	CHMSL
3F	Right Rear

Figura 8.1.1.2 – Ejemplo de mensaje (Fuente: SAE J2178-4)

Ejemplo Decodificación: Siguiendo el mismo ejemplo se recibe el comando 18DB DA FE 85 07. El primer paso es verificar el header 0x18DB. En este ejemplo es correcto por lo tanto el mensaje recibido es un OBDIIplus. Los siguiente a extraer es el ID primario, 0xDA. La tabla 1 del SAE J2178-4 incluye un listado de todos los Primary IDs como comando y como respuesta. Para el caso de módulos dedicados como lo es la presente propuesta, no es necesario saber todas las combinaciones. Basta saber la función del mismo modulo y cuál es su comando. Para este ejemplo el modulo es de

luces, por lo tanto solamente es necesario verificar que el ID primario sea 0xDA para saber que se están dirigiendo al mismo.

SAE J2178-4 Revised MAR1999

**TABLE 1—PRIMARY ID ASSIGNMENTS BYTE 2 OF
MESSAGE TYPES 0, 1, 2, 8, 9, 10, 11 (CONTINUED)**

Command	Status	Function
C4	C5	Door Locks
C6	C7	External Access
C8	C9	Seat Motion/Control
CA	CB	Windows
..	..	

Figura 8.1.1.3 – Fragmento de la Tabla 1 de ID Primarios (Fuente: SAE J2178-4)

El siguiente byte transmitido es el de la dirección destino, el cual es fijo el 0xFE. Opcionalmente se puede verificar para detección de mensajes erróneos o corrompidos. El primer byte de datos incluye el bit Q. Para decodificar este basta con saber si es igual o mayor a 8 para un “1”, o tener un valor de 0 a 7 para un “0”. La mayoría de los casos este va a ser un “8” o un “0”. Para el ejemplo el bit Q es activado

El siguiente es el ID secundario. En este ejemplo es 0x05 correspondiente por las mismas tablas a las direccionales (turn). La dirección extendida 0x07 indica que es el lado derecho.

Por lo tanto, el comando 18DB DA FE 85 07 decodificado indica prender la direccional derecha.

8.1.2 10 ejemplos aleatorios

Para realizar pruebas de otros comandos, se seleccionan 10 ejemplos al azar de entre las tablas SAE J2178. Para esto, se realiza una tabla en Excel con la función random la cual se extraen los 10 comandos aleatorios. Estos fueron DE, 88, F8, C4, D8, E4, 86, CA, B2, E6. Para el ID secundario y la dirección extendida se elige la primera opción disponible, o alguna que sea de interés representativo. La acción en todo caso es activar el correspondiente

Ejemplo 0xDE: El ID 0xDE corresponde a las luces interiores. Se inicia con el header incluyendo el ID primario seleccionado 18DB DE FE. La tabla SAEJ2178-4

correspondiente a este ID es la Tabla 43. El ID secundario 0x01 corresponde a las luces de cortesía. La dirección extendida se especifica en la tabla 9.9. Por fines ilustrativos se selecciona la luz del conductor con dirección 0x22.

El comando queda 18DB DE FE 81 22

Ejemplo 0x88: El ID 0x88 corresponde a las luces indicadoras del tablero. Se inicia con el header incluyendo el ID primario seleccionado 18DB 88 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 29. El ID secundario corresponde a cada lucecita o indicador en el tablero de instrumentos (36 opciones posibles). Por fines ilustrativos se selecciona apagar la luz del check engine 0x02. No cuenta con dirección extendida (hay un solo tablero), sin embargo por convención OBDIIplus se incluye como 0x00 para estandarizar la longitud de los mensajes.

El comando queda 18DB 88 FE 02 00

Ejemplo 0xF8: El ID 0xF8 corresponde a mensajes para indicar o establecer la hora y/o la fecha actual. Se inicia con el header incluyendo el ID primario seleccionado 18DB F8 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 49. El ID secundario maneja año, mes, día, hora, minuto, segundo, y los anteriores en paquete juntos. No hay dirección extendida, pero si un byte adicional de datos codificado. La tabla llama a los PRN 6011 al 6016 para la manera en codificar cada uno. Estas tablas están en el SAE J2178-2. Para estos PRN el dato se codifica en BCD

El comando para las 12 horas queda 18DB F8 FE 06 12

Ejemplo 0xC4: El ID 0xC4 corresponde a los seguros de las puertas. Se inicia con el header incluyendo el ID primario seleccionado 18DB C4 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 33. El ID secundario 0x01 corresponde a ponerlos o quitarlos. La dirección extendida se especifica en la tabla 9.5. Por fines ilustrativos se selecciona poner el seguro del lado del conductor 0x22.

El comando queda 18DB C4 FE 81 22

Ejemplo 0xD8: El ID 0xD8 corresponde a inquirir sobre el estatus de las luces. Se inicia con el header incluyendo el ID primario seleccionado 18DB D8 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 40. El ID secundario 0x01 corresponde a las luces principales headlamps. La dirección extendida se especifica en la tabla 9.8. Por fines ilustrativos se cambia el ID a respuesta 0xD9, y se da un indicativo de estar mal (0x0) la frontal izquierda (0x09).

El comando queda 18DB D9 FE 01 09

Ejemplo 0xE4: El ID 0xE4 corresponde a monitoreo de las llantas. Se inicia con el header incluyendo el ID primario seleccionado 18DB E4 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 44. El ID secundario 0x11 corresponde a las presión de las llantas. La dirección extendida se especifica en la tabla 9.1. Para la llanta del pasajero, la dirección es 0x17. Este ID necesita incluir un byte de datos. Para la presión requiere el PRN 2851. Este se encuentra especificado en el SAE J2178-2. El PRN2851 indica que la presión se da en KpaG (kilo pascals gauge), codificados de manera UNM-08-141 la cual es de 4 bits sin signo. Para 33 Psi equivalentes a 227 KPa es 0xE3

El comando queda 18DB E4 FE 11 17 E3

Ejemplo 0x86: El ID 0x86 corresponde a posición del switch de encendido. Se inicia con el header incluyendo el ID primario seleccionado 18DB 86 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 28. El ID secundario 0x05 corresponde a la posición del mismo. No hay dirección extendida (un solo switch de encendido).

Para el switch encendido, el bit Q es 1. Con fin ilustrativo se pone la respuesta o estado 0x87.

El comando queda 18DB 87 FE 85 00

Ejemplo 0xCA: El ID 0xCA corresponde a las ventanas. Se inicia con el header incluyendo el ID primario seleccionado 18DB CA FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 36. El ID secundario 0x01 corresponde a la acción de abrir la ventana (Q1 es activado siempre). La dirección extendida se

especifica en la tabla 9.7. Por fines ilustrativos se selecciona abrir la ventana del conductor con dirección 0x22.

El comando queda 18DB CA FE 81 22

Ejemplo 0xB2: El ID 0xB2 corresponde al aire acondicionado. Se inicia con el header incluyendo el ID primario seleccionado 18DB B2 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 30. El ID secundario 0x0A corresponde a la temperatura. La dirección extendida se especifica en la tabla 9.2. Se selecciona todo el frente con 0x20. Este ID necesita incluir un byte de datos. Para la temperatura requiere el PRN 9820. Este se encuentra especificado en el SAE J2178-2. El PRN 9820 indica codificados de manera UNM-08-73 la cual es de 4 bits sin signo. El slot 73 indica que hay que restar 40 al valor sin signo indicado. Para 25 grados queda 0x2D

El comando queda 18DB B2 FE 0A 20 2D

Ejemplo 0xE6: El ID 0xE6 corresponde al desempañante (defrost). Se inicia con el header incluyendo el ID primario seleccionado 18DB E6 FE. La tabla SAEJ2178-4 correspondiente a este ID es la Tabla 45. El ID secundario 0x01 corresponde a la activación o desactivación del mismo. La dirección extendida se especifica en la tabla 9.3. Por fines ilustrativos se selecciona el desempañante de atrás 0x34.

El comando queda 18DB E6 FE 81 34

8.2 Aplicación en ambiente real

Las pruebas anteriores de escritorio funcionaron para probar el protocolo en alto nivel. Los algoritmos probaron ser eficientes en poder codificar la intención de quien emite el comando OBDII plus, así como la capacidad de entender dicho comando por parte del receptor.

Es turno de ver en funcionamiento real dicho protocolo. Como se comentó en la etapa de implementación, se seleccionó el sistema de luces para realizar dicho ejercicio. Este sistema presenta las ventajas de ser sencillo pero a la vez ser representativo de los aspectos del protocolo como son ID primario y secundario, dirección extendida y estatus de retorno. Además, para efectos de validación es visualmente intuitivo cual es

el comando y la acción requerida. Los comandos son solamente prender o apagar luces, los cuales es fácil verificar si el comando fue efectivo al ver si se prende o se apaga la luz correspondiente. Algunos selectos se permitieron mantener encendido un estado previo para probar comandos simultáneos. Es el caso de los cuartitos con luz headlamp, o direccionales con luz de freno.

Los módulos son luces, por lo cual todos los comandos son para este sistema. El ejercicio consta de emitir un comando, ya sea por un canal de comunicación o utilizando el encoder del HID (presionando un botón). Después se verifica que se realizó la acción requerida. Si hay respuesta se lee la misma y se verifica su coherencia. Esto se repite para cada uno de los 8 comandos codificados en el sistema

8.2.1 Prueba de luz de cuartitos (Park)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz de estacionamiento, el ID secundario es 0x04. La dirección extendida es para todos con 0x00. Con este comando, se espera que solamente enciendan el relay R1 de todos los módulos. El comando OBDII plus correspondiente es

18DB DA FE 84 00

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```
button park press
ECHO button: 18DBDAFE8400
Can Send Ok lng:8 MSG:68 65 70 69 56 52 48 48 0
CMD received S_ID:4 Q:1 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 57 0
CAN status send:18DBD8FE0409
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 70 0
CAN status send:18DBD8FE040F
CAN status received:18DBD9FE8409
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 57 0
CAN status send:18DBD8FE0439
CAN status received:18DBD9FE840F
```

```

Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 70 0
CAN status send:18DBD8FE043F
CAN status received:18DBD9FE8439
CAN data received, data len = 8
68 57 70 69 56 52 51 70
CAN command received:18DBD9FE843F

```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

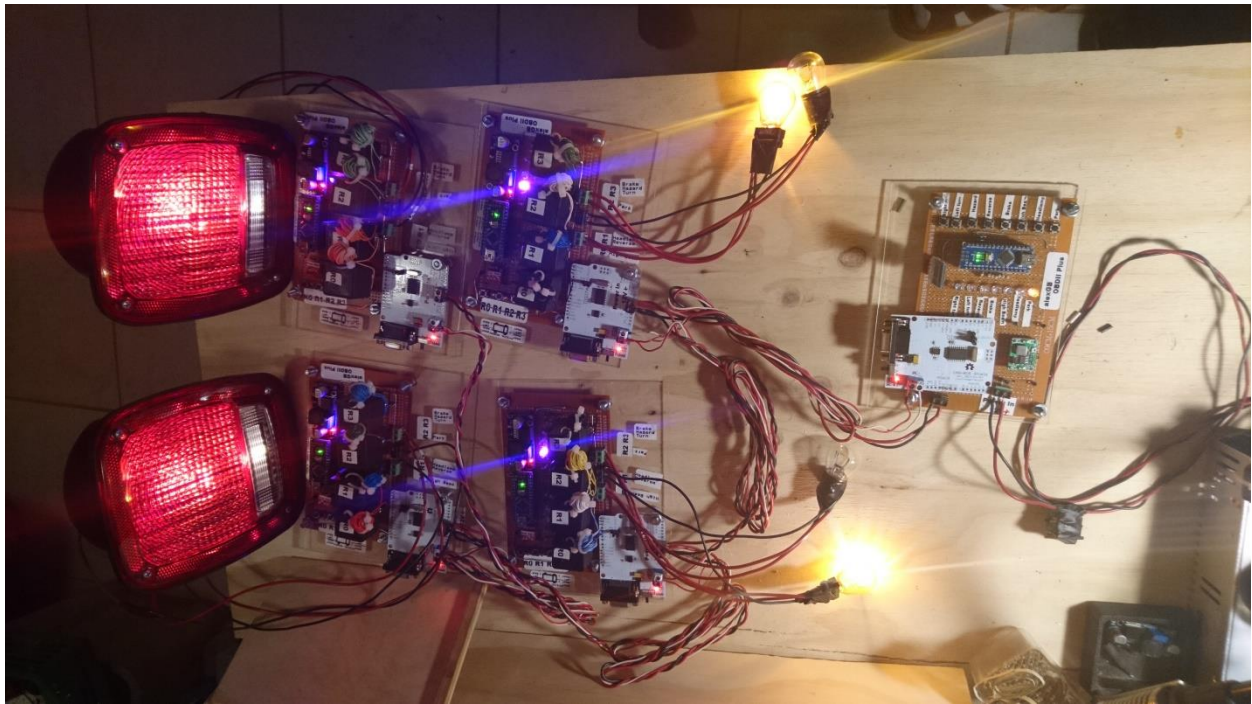


Figura 8.2.1.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 04 00

La captura de respuesta del módulo HID es la siguiente:


```

button park press
ECHO button: 18DBDAFE0400
Can Send Ok lng:8 MSG:68 65 70 69 48 52 48 48 0
CMD received S_ID:4 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 57 0
CAN status send:18DBD8FE0409
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 70 0
CAN status send:18DBD8FE040F
CAN status received:18DBD9FE8409
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 57 0
CAN status send:18DBD8FE0439
CAN status received:18DBD9FE840F
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 70 0
CAN status send:18DBD8FE043F
CAN status received:18DBD9FE8439
CAN data received, data len = 8
68 57 70 69 56 52 51 70
CAN command received:18DBD9FE843F

```

La reacción del módulo es apagar la luz correspondiente.

8.2.2 Prueba de luz de luces principales (headlamps)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz de headlamp, el ID secundario es 0x01. La dirección extendida es para todos con 0x00. Con este comando, se espera que solamente enciendan el relay R1 de los módulos delanteros. El comando OBDII plus correspondiente es

18DB DA FE 81 00

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```

button headlamp press
ECHO button: 18DBDAFE8100
Can Send Ok lng:8 MSG:68 65 70 69 56 49 48 48 0
CMD received S_ID:1 Q:1 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 49 48 57 0
CAN status send:18DBD8FE0109
CAN status received:

```

```

Can Send Ok lng:8 MSG:68 56 70 69 48 49 48 70 0
CAN status send:18DBD8FE010F
CAN status received:18DBD9FE8109
Can Send Ok lng:8 MSG:68 56 70 69 48 49 51 57 0
CAN status send:18DBD8FE0139
CAN status received:18DBD9FE810F
Can Send Ok lng:8 MSG:68 56 70 69 48 49 51 70 0
CAN status send:18DBD8FE013F
CAN status received:18DBD9FE8139
CAN data received, data len = 8
68 57 70 69 56 49 51 70
CAN command received:18DBD9FE813F

```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

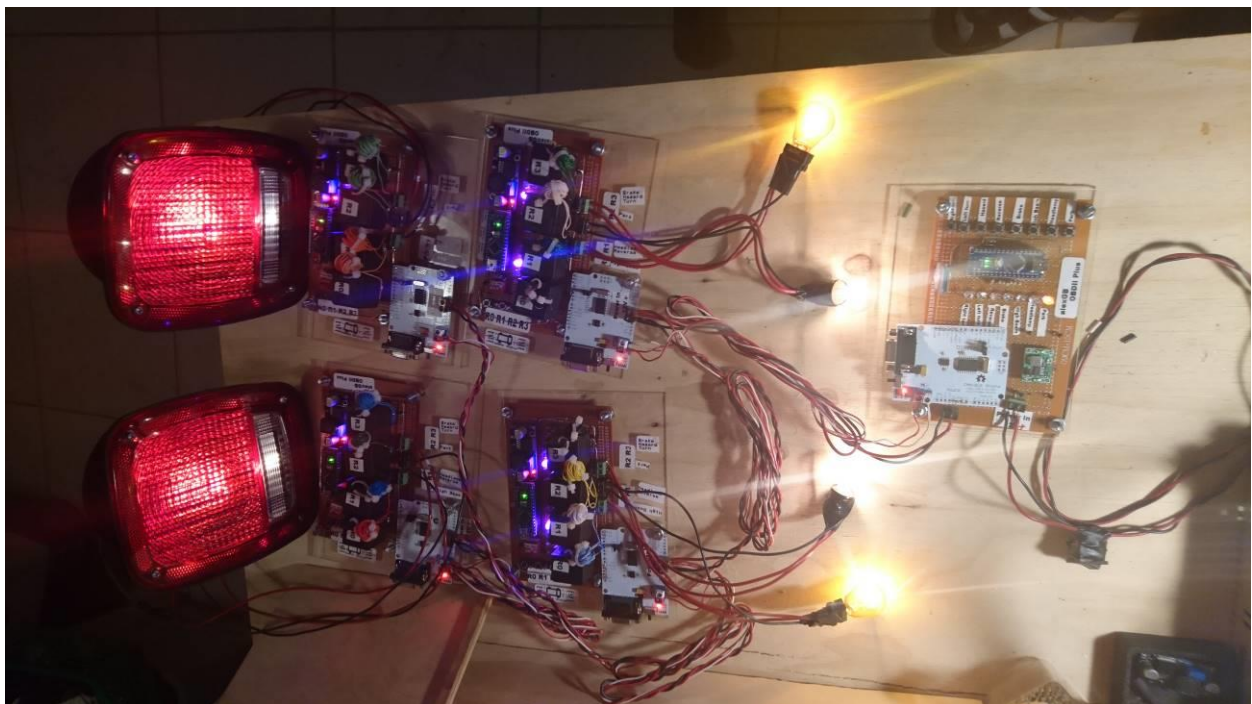


Figura 8.2.2.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 01 00

La captura de respuesta del módulo HID es la siguiente:

```
button headlamp press
ECHO button: 18DBDAFE0100
Can Send Ok lng:8 MSG:68 65 70 69 48 49 48 48 0
CMD received S_ID:1 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 49 48 57 0
CAN status send:18DBD8FE0109
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 49 48 70 0
CAN status send:18DBD8FE010F
CAN status received:18DBD9FE8109
Can Send Ok lng:8 MSG:68 56 70 69 48 49 51 57 0
CAN status send:18DBD8FE0139
CAN status received:18DBD9FE810F
Can Send Ok lng:8 MSG:68 56 70 69 48 49 51 70 0
CAN status send:18DBD8FE013F
CAN status received:18DBD9FE8139
CAN data received, data len = 8
68 57 70 69 56 49 51 70
CAN command received:18DBD9FE813F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.3 Prueba de luz de luces altas (high beam)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz alta, el ID secundario es 0x06. La dirección extendida es para todos con 0x00. Con este comando, se espera que solamente enciendan el relay R3 de los módulos delanteros. El comando OBDII plus correspondiente es

18DB DA FE 86 00

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```
button highbeam press
ECHO button: 18DBDAFE8600
Can Send Ok lng:8 MSG:68 65 70 69 56 54 48 48 0
CMD received S_ID:6 Q:1 xAdr:0
```

```

Can Send Ok lng:8 MSG:68 56 70 69 48 54 48 57 0
CAN status send:18DBD8FE0609
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 54 48 70 0
CAN status send:18DBD8FE060F
CAN status received:18DBD9FE8609
Can Send Ok lng:8 MSG:68 56 70 69 48 54 51 57 0
CAN status send:18DBD8FE0639
CAN status received:18DBD9FE860F
Can Send Ok lng:8 MSG:68 56 70 69 48 54 51 70 0
CAN status send:18DBD8FE063F
CAN status received:18DBD9FE8639
CAN data received, data len = 8
68 57 70 69 56 54 51 70
CAN command received:18DBD9FE863F

```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

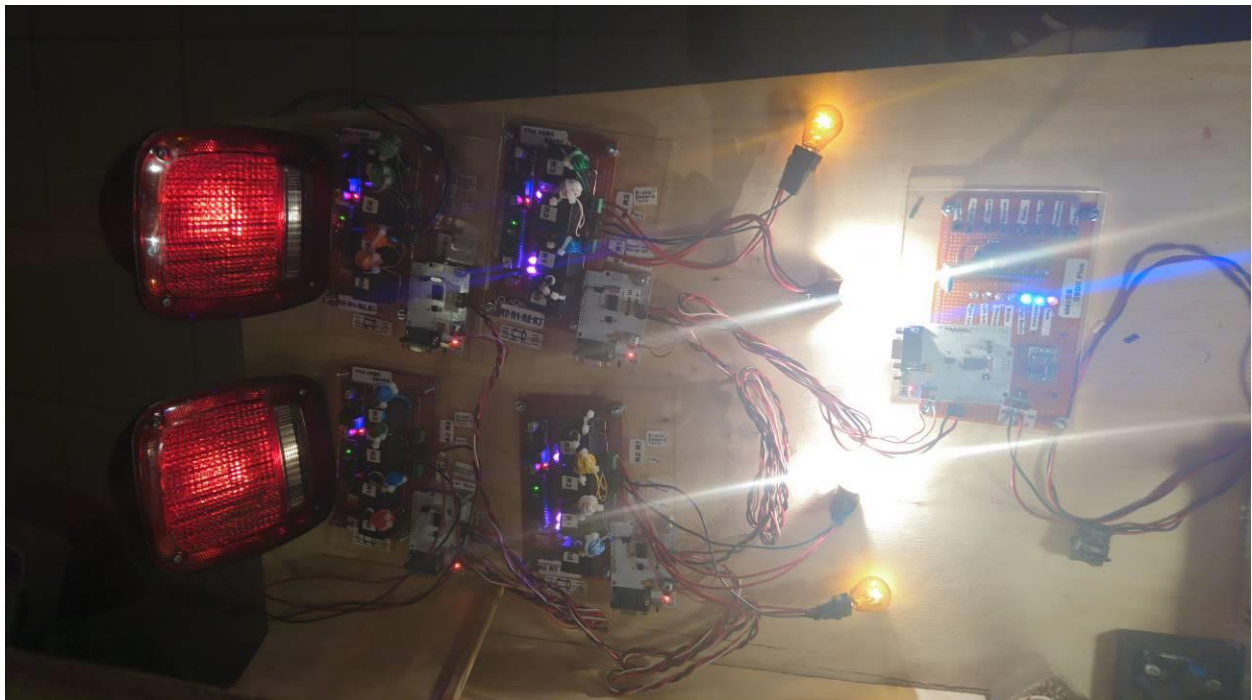


Figura 8.2.3.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 06 00

La captura de respuesta del módulo HID es la siguiente:

```
button highbeam press
ECHO button: 18DBDAFE0600
Can Send Ok lng:8 MSG:68 65 70 69 48 54 48 48 0
CMD received S_ID:6 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 54 48 57 0
CAN status send:18DBD8FE0609
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 54 48 70 0
CAN status send:18DBD8FE060F
CAN status received:18DBD9FE8609
Can Send Ok lng:8 MSG:68 56 70 69 48 54 51 57 0
CAN status send:18DBD8FE0639
CAN status received:18DBD9FE860F
Can Send Ok lng:8 MSG:68 56 70 69 48 54 51 70 0
CAN status send:18DBD8FE063F
CAN status received:18DBD9FE8639
CAN data received, data len = 8
68 57 70 69 56 54 51 70
CAN command received:18DBD9FE863F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.4 Prueba de luz de freno (Brake)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz de freno, el ID secundario es 0x03. La dirección extendida es para todos con 0x00. Con este comando, se espera que solamente enciendan el relay R0 de los módulos traseros. El comando OBDII plus correspondiente es

18DB DA FE 83 00

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:


```

button brake press
ECHO button: 18DBDAFE8300
Can Send Ok lng:8 MSG:68 65 70 69 56 51 48 48 0
CMD received S_ID:3 Q:1 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 51 48 57 0
CAN status send:18DBD8FE0309
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 51 48 70 0
CAN status send:18DBD8FE030F
CAN status received:18DBD9FE8309
Can Send Ok lng:8 MSG:68 56 70 69 48 51 51 57 0
CAN status send:18DBD8FE0339
CAN status received:18DBD9FE830F
Can Send Ok lng:8 MSG:68 56 70 69 48 51 51 70 0
CAN status send:18DBD8FE033F
CAN status received:18DBD9FE8339
CAN data received, data len = 8
68 57 70 69 56 51 51 70
CAN command received:18DBD9FE833F

```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

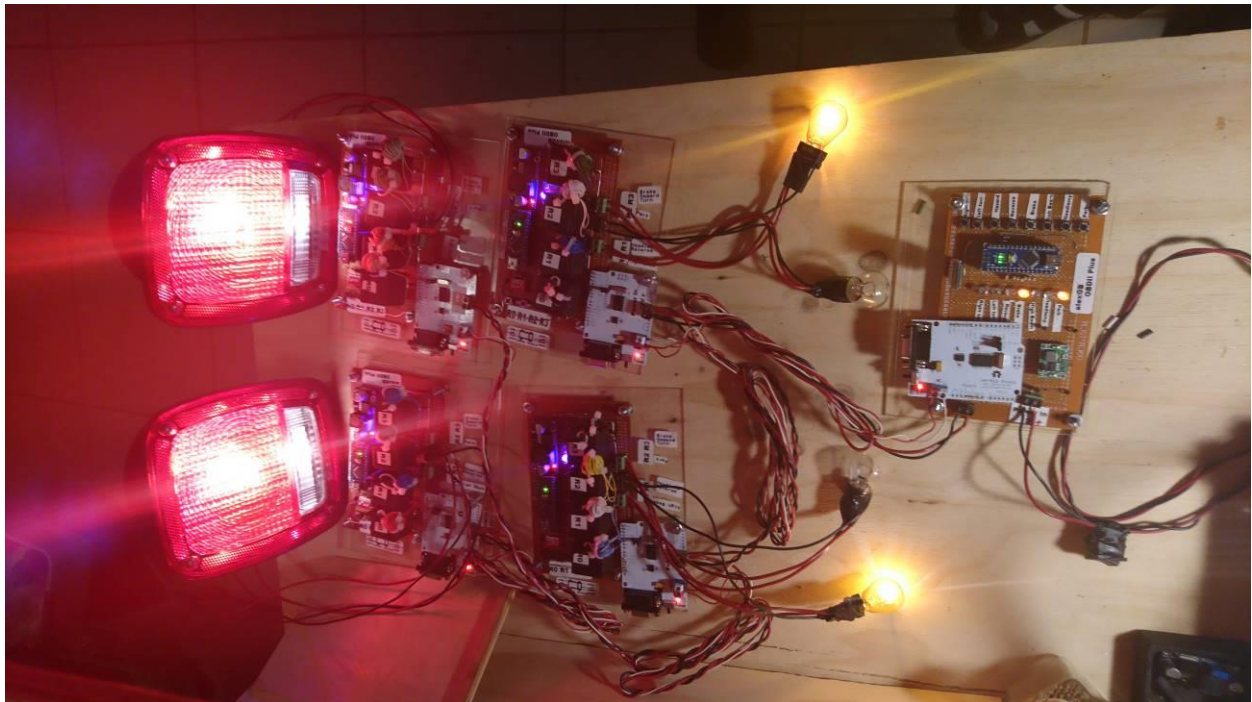


Figura 8.2.4.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 03 00

La captura de respuesta del módulo HID es la siguiente:

```
button brake press
ECHO button: 18DBDAFE0300
Can Send Ok lng:8 MSG:68 65 70 69 48 51 48 48 0
CMD received S_ID:3 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 51 48 57 0
CAN status send:18DBD8FE0309
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 51 48 70 0
CAN status send:18DBD8FE030F
CAN status received:18DBD9FE8309
Can Send Ok lng:8 MSG:68 56 70 69 48 51 51 57 0
CAN status send:18DBD8FE0339
CAN status received:18DBD9FE830F
Can Send Ok lng:8 MSG:68 56 70 69 48 51 51 70 0
CAN status send:18DBD8FE033F
CAN status received:18DBD9FE8339
CAN data received, data len = 8
68 57 70 69 56 51 51 70
CAN command received:18DBD9FE833F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.5 Prueba de luz de Reversa

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz de reversa, el ID secundario es 0x08. La dirección extendida es para todos con 0x00. Con este comando, se espera que solamente enciendan el relay R2 de los módulos traseros. El comando OBDII plus correspondiente es

18DB DA FE 88 00

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```

button reverse press
ECHO button: 18DBDAFE8800
Can Send Ok lng:8 MSG:68 65 70 69 56 56 48 48 0
CMD received S_ID:8 Q:1 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 56 48 57 0
CAN status send:18DBD8FE0809
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 56 48 70 0
CAN status send:18DBD8FE080F
CAN status received:18DBD9FE8809
Can Send Ok lng:8 MSG:68 56 70 69 48 56 51 57 0
CAN status send:18DBD8FE0839
CAN status received:18DBD9FE880F
Can Send Ok lng:8 MSG:68 56 70 69 48 56 51 70 0
CAN status send:18DBD8FE083F
CAN status received:18DBD9FE8839
CAN data received, data len = 8
68 57 70 69 48 56 51 70
CAN command received:18DBD9FE083F

```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

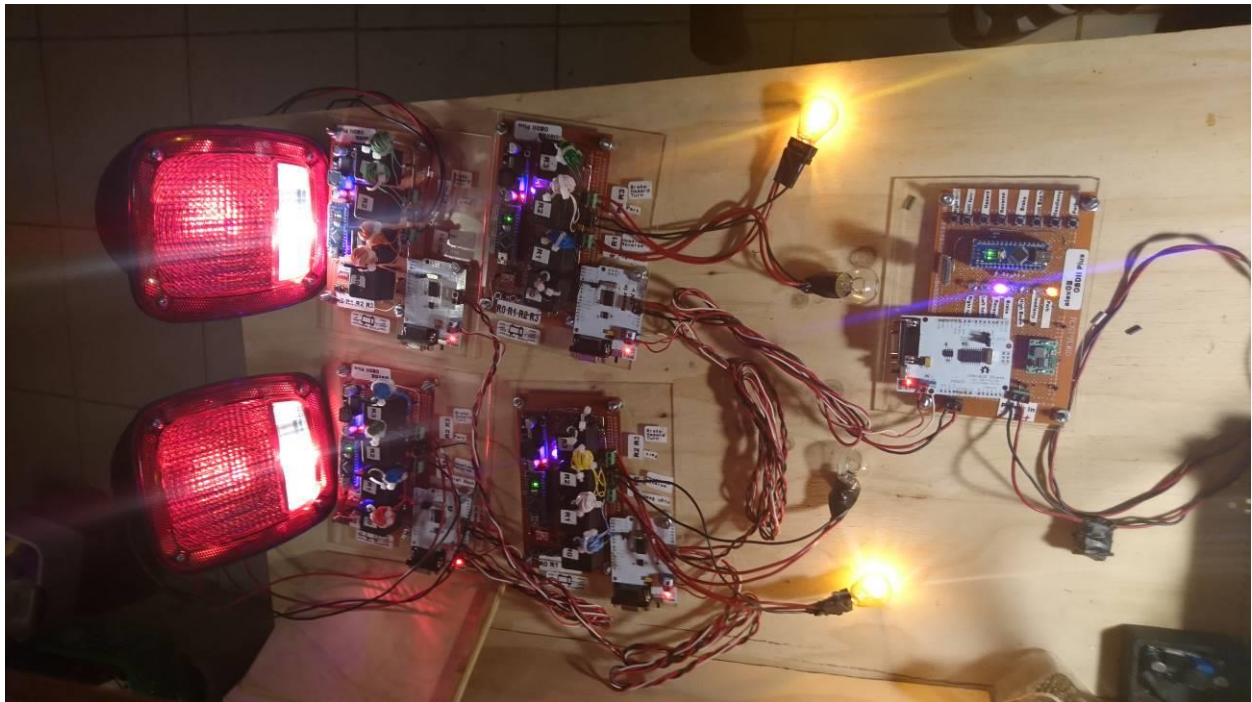


Figura 8.2.5.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 08 00

La captura de respuesta del módulo HID es la siguiente:

```
button reverse press
ECHO button: 18DBDAFE0800
Can Send Ok lng:8 MSG:68 65 70 69 48 56 48 48 0
CMD received S_ID:8 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 56 48 57 0
CAN status send:18DBD8FE0809
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 56 48 70 0
CAN status send:18DBD8FE080F
CAN status received:18DBD9FE8809
Can Send Ok lng:8 MSG:68 56 70 69 48 56 51 57 0
CAN status send:18DBD8FE0839
CAN status received:18DBD9FE880F
Can Send Ok lng:8 MSG:68 56 70 69 48 56 51 70 0
CAN status send:18DBD8FE083F
CAN status received:18DBD9FE8839
CAN data received, data len = 8
68 57 70 69 56 56 51 70
CAN command received:18DBD9FE883F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.6 Prueba de luces intermitentes (hazard)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz de hazard, el ID secundario es 0x07. La dirección extendida es para todos con 0x00. Con este comando, se espera que solamente enciendan el relay R0 de todos los módulos con un timer. El comando OBDII plus correspondiente es

18DB DA FE 87 00

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```

button hazard press
ECHO button: 18DBDAFE8700
Can Send Ok lng:8 MSG:68 65 70 69 56 55 48 48 0
CMD received S_ID:7 Q:1 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 55 48 57 0
CAN status send:18DBD8FE0709
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 55 48 70 0
CAN status send:18DBD8FE070F
CAN status received:18DBD9FE8709
Can Send Ok lng:8 MSG:68 56 70 69 48 55 51 57 0
CAN status send:18DBD8FE0739
CAN status received:18DBD9FE870F
Can Send Ok lng:8 MSG:68 56 70 69 48 55 51 70 0
CAN status send:18DBD8FE073F
CAN status received:18DBD9FE8739
CAN data received, data len = 8
68 57 70 69 56 55 51 70
CAN command received:18DBD9FE873F

```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

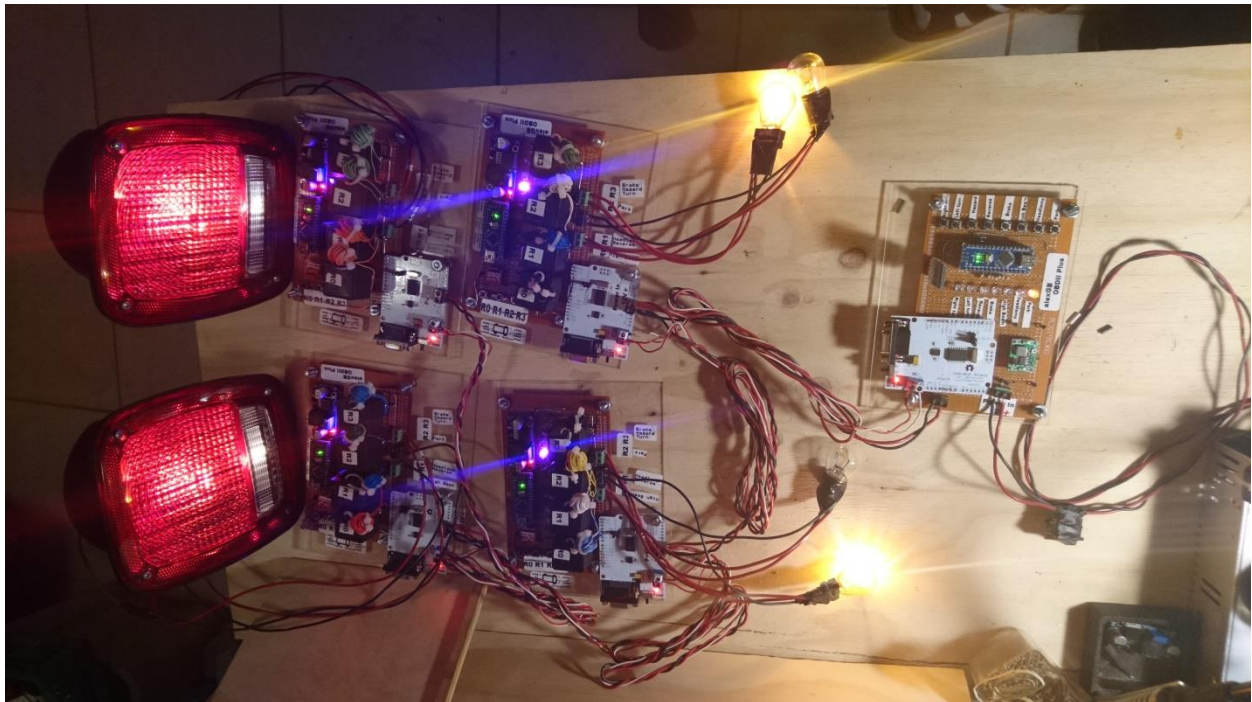


Figura 8.2.6.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 07 00

La captura de respuesta del módulo HID es la siguiente:

```
button hazard press
ECHO button: 18DBDAFE0700
Can Send Ok lng:8 MSG:68 65 70 69 48 55 48 48 0
CMD received S_ID:7 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 55 48 57 0
CAN status send:18DBD8FE0709
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 55 48 70 0
CAN status send:18DBD8FE070F
CAN status received:18DBD9FE8709
Can Send Ok lng:8 MSG:68 56 70 69 48 55 51 57 0
CAN status send:18DBD8FE0739
CAN status received:18DBD9FE870F
Can Send Ok lng:8 MSG:68 56 70 69 48 55 51 70 0
CAN status send:18DBD8FE073F
CAN status received:18DBD9FE8739
CAN data received, data len = 8
68 57 70 69 56 55 51 70
CAN command received:18DBD9FE873F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.7 Prueba de luces direccional izquierda (turn left)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz direccional izquierda, el ID secundario es 0x05. La dirección extendida es para todos los de la izquierda con 0x01. Con este comando, se espera que solamente enciendan el relay R0 de los módulos izquierdos con un timer. El comando OBDII plus correspondiente es

18DB DA FE 85 01

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```
button left press
ECHO button: 18DBDAFE8501
Can Send Ok lng:8 MSG:68 65 70 69 56 53 48 49 0
CMD received S_ID:5 Q:1 xAdr:1
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 57 0
CAN status send:18DBD8FE0509
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 70 0
CAN status send:18DBD8FE050F
CAN status received:18DBD9FE0509
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 57 0
CAN status send:18DBD8FE0539
CAN status received:18DBD9FE050F
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 70 0
CAN status send:18DBD8FE053F
CAN status received:18DBD9FE0539
CAN data received, data len = 8
68 57 70 69 48 53 51 70
CAN command received:18DBD9FE053F
```

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

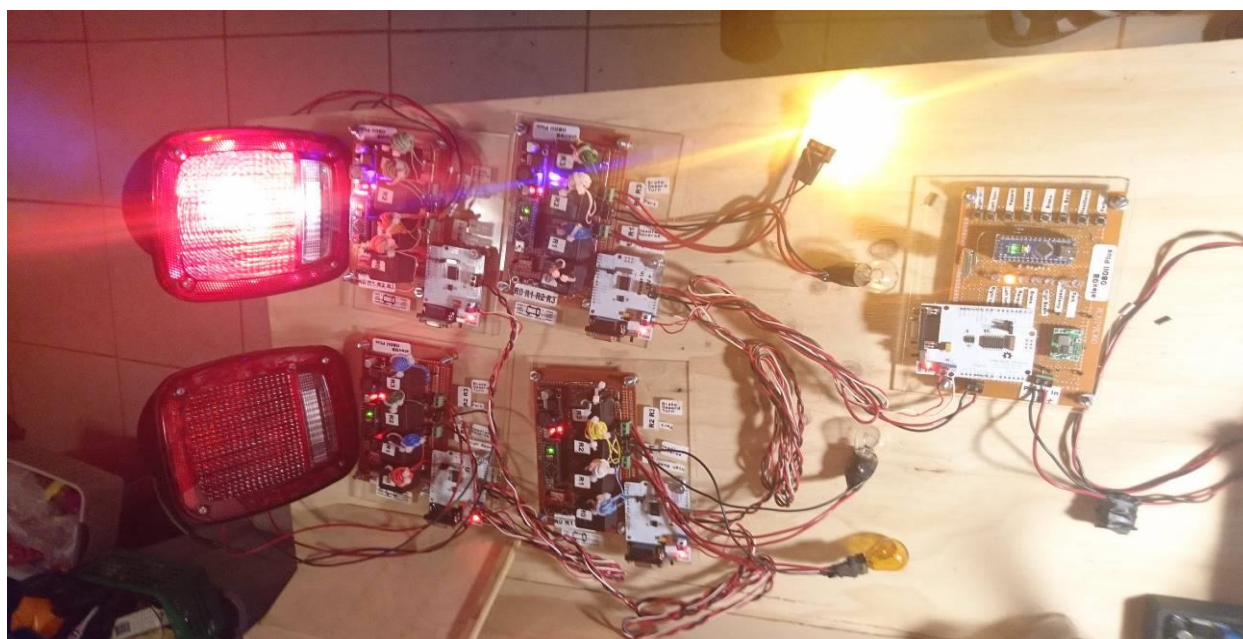


Figura 8.2.7.1 – Respuesta de los módulos ante el comando enviado

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 05 01

La captura de respuesta del módulo HID es la siguiente:

```
button left press
ECHO button: 18DBDAFE0501
Can Send Ok lng:8 MSG:68 65 70 69 48 53 48 49 0
CMD received S_ID:5 Q:0 xAdr:1
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 57 0
CAN status send:18DBD8FE0509
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 70 0
CAN status send:18DBD8FE050F
CAN status received:18DBD9FE0509
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 57 0
CAN status send:18DBD8FE0539
CAN status received:18DBD9FE050F
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 70 0
CAN status send:18DBD8FE053F
CAN status received:18DBD9FE0539
CAN data received, data len = 8
68 57 70 69 48 53 51 70
CAN command received:18DBD9FE053F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.8 Prueba de luces direccional derecha (turn right)

En el protocolo OBDIIplus, El sistema de luces corresponde con el ID primario 0xDA para comando y 0xDB para responder a dicho comando. El estado de las luces, su comando para pregunta es 0xD8 y su respuesta es 0xD9. En específico para la luz direccional derecha, el ID secundario es 0x05. La dirección extendida es para todos los de la derecha con 0x07. Con este comando, se espera que solamente enciendan el relay R0 de los módulos izquierdos con un timer. El comando OBDII plus correspondiente es

18DB DA FE 85 07

El comando OBDIIplus para encendido se envía a través de pulsar el correspondiente botón en el módulo HID. La captura de respuesta del módulo HID es la siguiente:

```
button right press
ECHO button: 18DBDAFE8507
Can Send Ok lng:8 MSG:68 65 70 69 56 53 48 55 0
CMD received S_ID:5 Q:1 xAdr:7
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 57 0
CAN status send:18DBD8FE0509
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 70 0
CAN status send:18DBD8FE050F
CAN status received:18DBD9FE0509
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 57 0
CAN status send:18DBD8FE0539
CAN status received:18DBD9FE050F
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 70 0
CAN status send:18DBD8FE053F
CAN status received:18DBD9FE0539
CAN data received, data len = 8
68 57 70 69 48 53 51 70
CAN command received:18DBD9FE053F
```

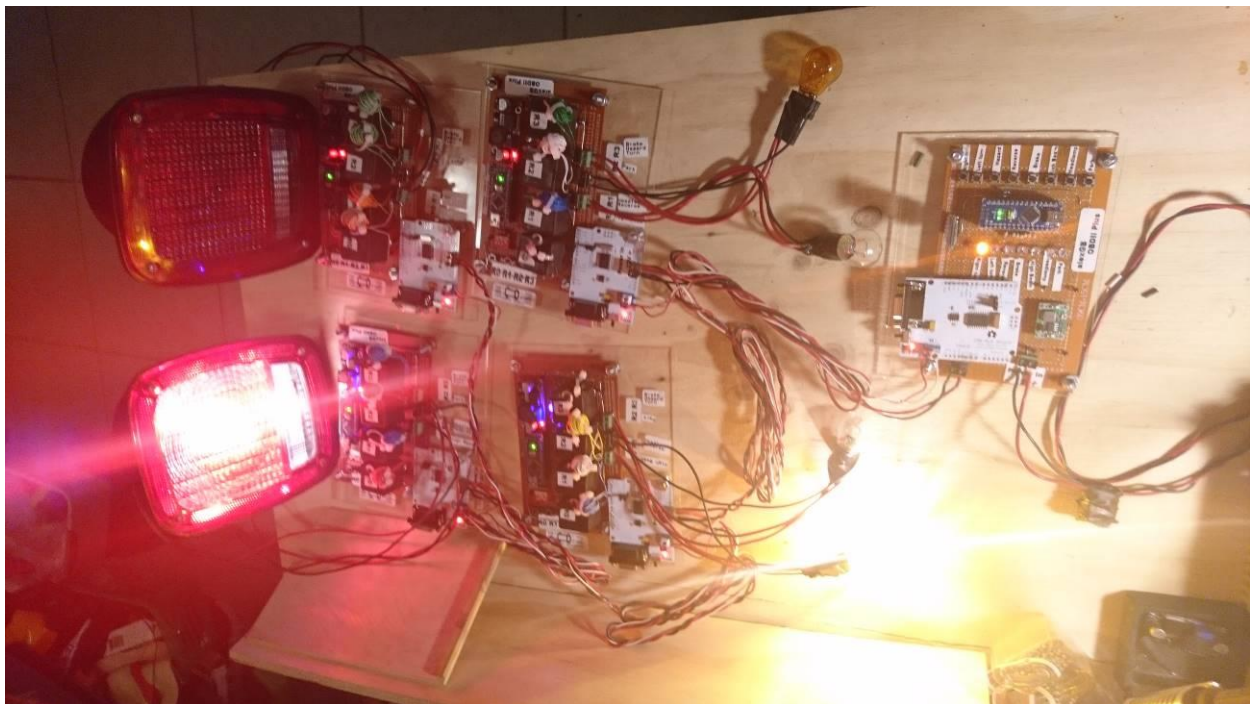


Figura 8.2.8.1 – Respuesta de los módulos ante el comando enviado

En la captura de monitoreo de señal del módulo HID, se aprecia el comando enviado. Acto seguido el módulo HID envía a cada módulo un requerimiento de estado. En el log se aprecia la respuesta de cada uno de los cuatro módulos indicando el estado de sus luces. En la imagen se aprecia la acción efectuada por los módulos correspondientes al comando enviado.

Posteriormente se envía el comando OBDIIplus correspondiente para apagar el anteriormente emitido.

18DB DA FE 05 07

La captura de respuesta del módulo HID es la siguiente:

```
button right press
ECHO button: 18DBDAFE0507
Can Send Ok lng:8 MSG:68 65 70 69 48 53 48 55 0
CMD received S_ID:5 Q:0 xAdr:7
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 57 0
CAN status send:18DBD8FE0509
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 53 48 70 0
CAN status send:18DBD8FE050F
CAN status received:18DBD9FE0509
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 57 0
CAN status send:18DBD8FE0539
CAN status received:18DBD9FE050F
Can Send Ok lng:8 MSG:68 56 70 69 48 53 51 70 0
CAN status send:18DBD8FE053F
CAN status received:18DBD9FE0539
CAN data received, data len = 8
68 57 70 69 48 53 51 70
CAN command received:18DBD9FE053F
```

La reacción del módulo es apagar la luz correspondiente.

8.2.9 Prueba de monitoreo de estados

Para simular una falla de foco fundido, se desconecta el cable de un foco y se envía el comando de activación. Esto se realiza en el módulo 1 (dirección 0x09) con la luz de cuartitos. Cuando este bien el foco, el comando enviado y recibido fue:

```

button park press
ECHO button: 18DBDAFE8400
...
CAN status send:18DBD8FE0409
...
CAN status received:18DBD9FE8409

```

El mensaje retornado indica que para el estado del módulo de luces 0xD9 en los cuartitos 0x4 del módulo uno delantero izquierdo 0x09, el estado es Q=1 (0x8), todo bien.

Con el mismo foco desconectado, la comunicación fue

```

button park press
ECHO button: 18DBDAFE8400
...
CAN status send:18DBD8FE0409
...
CAN status received:18DBD9FE0409

```

El mensaje es similar, salvo que el bit Q = 0 (0x0) indicando un problema con dicho foco.

8.2.10 Prueba de puente de comunicación

Para probar el puente interno de comunicación, se le envía un comando externo por medio del serial. El módulo HID contesta con ECHO el comando recibido, y posteriormente los maneja de manera similar a un comando generado internamente y enviado por CAN-BUS. La comunicación completa del comando enviado y recibido fue:

```

Module Started
Enter setting mode success
set rate success!!
Enter Normal Mode Success!!
can init ok!!
TEST ON/OFF lamps
ECHO:18DBDAFE8400

Can Send Ok lng:8 MSG:68 65 70 69 56 52 48 48 0
CMD received S_ID:4 Q:1 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 57 0
CAN status send:18DBD8FE0409
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 70 0

```



```

CAN status send:18DBD8FE040F
CAN status received:18DBD9FE8409
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 57 0
CAN status send:18DBD8FE0439
CAN status received:18DBD9FE840F
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 70 0
CAN status send:18DBD8FE043F
CAN status received:18DBD9FE8439
CAN data received, data len = 8
68 57 70 69 56 52 51 70
CAN command received:18DBD9FE843F
heartbeat
ECHO:18DBDAFE0400

Can Send Ok lng:8 MSG:68 65 70 69 48 52 48 48 0
CMD received S_ID:4 Q:0 xAdr:0
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 57 0
CAN status send:18DBD8FE0409
CAN status received:
Can Send Ok lng:8 MSG:68 56 70 69 48 52 48 70 0
CAN status send:18DBD8FE040F
CAN status received:18DBD9FE8409
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 57 0
CAN status send:18DBD8FE0439
CAN status received:18DBD9FE840F
Can Send Ok lng:8 MSG:68 56 70 69 48 52 51 70 0
CAN status send:18DBD8FE043F
CAN status received:18DBD9FE8439
CAN data received, data len = 8
68 57 70 69 56 52 51 70
CAN command received:18DBD9FE843F
heartbeat

```

Con esto se comprueba la funcionalidad del módulo puente de comunicación, y su habilidad de manejar comandos a partir de otros medios.

9 Análisis e interpretación de resultados

El marco teórico planteado en el capítulo 2 sienta las bases para el entendimiento del sistema OBDII actual, orientado a una mejor comprensión del OBDIIplus propuesto. Es en este capítulo donde se desglosa los diferentes sistemas actuales y se logra un entendimiento a fondo del OBDII. A partir de este entendimiento, se selecciona de entre las múltiples opciones OBDII las más adecuadas para el OBDIIplus. Derivado de este conocimiento es que se plantean los requerimientos de comunicación, requisitos mínimos de funcionalidad, la estructura del mensaje y algunos códigos fijos. Así se pudo plantear de una manera más objetiva la propuesta.

En el capítulo 4 se definió lo que es el requerimiento del producto. Repitiendo para este trabajo se define como:

“Sistema automotriz de plataforma abierta para manejo de sub-sistemas no críticos pertenecientes al dominio de seguridad y confort (body), con un protocolo y plataforma de hardware estandarizados y funcionales para la mayoría de vehículos independientemente de la marca, de fácil implementación en vehículos nuevos o ya existentes, con enfoque al bajo costo por medio de alto volumen derivado de la estandarización y plataforma abierta. “

A partir del requerimiento, el perfil de usuarios potenciales y las expectativas de los potenciales clientes, se desarrollaron los siguientes objetivos generales para el sistema como un producto terminado:

- Facilidad de desarrollo e instalación. No requerir herramientas especializadas
- Poder adaptarse fácilmente a cualquier vehículo
- Bajo costo
- Proveer funcionalidades adicionales a las soluciones actuales.

De los anteriores, se desarrolló un Documento de Requerimiento de Producto. Este documento expande a detalle los requerimientos necesarios para el proyecto planteado. Especifica cuestiones como la interacción con otros sistemas, plataformas de hardware y software, y requerimientos mínimos de funcionalidades a cumplir.

Todos los anteriores puntos le dieron forma a la propuesta de protocolo OBDIIplus planteada en el capítulo 5. Es en este punto donde se presenta la propuesta a detalle junto con sus reglas y ejemplos.

El plan original de la presente propuesta contemplaba el desarrollar tablas para todos los sistemas enlistados en el capítulo 4. Durante la investigación del estado del arte, se encontró un estándar con un listado de tablas de sistemas no de emisiones. El estándar SAE J2178 en su parte primera define los requisitos para el header de los mensajes OBDII. Esta sección se sigue utilizando de referencia a la fecha. Al continuar investigando, se encontraron las secciones 3 y 4. Dichas secciones proponen un sistema de direccionamiento para sistemas no de emisiones basados en la estructura OBD anterior a 1996 (OBD uno). En una búsqueda de aplicaciones, no se encontró aplicación o literatura alguna que las utilice o las mencione. Después de un profundo análisis a dicha aplicación propuesta, se encontró en extremo difícil de implementar, en mayor parte por estar basado en sistemas OBD primitivos no adecuados para manejar estos sistemas. Sin embargo, la sección 4 contiene más de treinta tablas con los sistemas no de emisiones que contemplaban cubrir con dicho sistema. Estas tablas fueron rescatadas por el presente trabajo y utilizadas bajo la referencia de ser las tablas del estándar SAE J2178-4

Una vez definidas las reglas del protocolo propuesto se ponen a prueba para afinar detalles. Los capítulos 6 y 7 son relativos a la construcción física de un prototipo para tales fines. Respectivamente los capítulos dan los detalles del hardware y el firmware. Se eligió el sistema de luces como representativo para este fin. Dicho sistema cumple por el lado del hardware con ser sencillo de implementar. Sin embargo, por parte del protocolo y el firmware estos presentan la complejidad suficiente para poder ayudar a afinar los detalles del protocolo.

El hardware se hizo basado en los requerimientos técnicos de producto del capítulo 5 para el manejo de las luces. Siguiendo estos lineamientos se construyeron 4 módulos idénticos a los cuales se les agregó direccionamiento físico por medio de switches. Al

ser idénticos, simplemente con asignarles una dirección física diferente los módulos reaccionaban acorde a su nueva localización. Los módulos cuentan con los circuitos necesarios para los sistemas de luces presentes en la vasta mayoría de los vehículos, tales como luces frontales, cuartos, frenos y reversa. Estos sistemas se seleccionaron en el capítulo 4. Al ser módulos idénticos y contar con los circuitos necesarios para las luces comunes, se cumple satisfactoriamente con el objetivo de ser módulos universales.

El capítulo 7 donde se desarrolló el firmware presentó múltiples oportunidades para reforzar y afinar detalles en el protocolo propuesto. Entre estos se encuentra afinar los bytes propuestos como fijos, cuantos bytes para la longitud fija de los mensajes, cambiar la estructura al tipo maestro-multiesclavo para direccionamiento a múltiples módulos, afinar la estructura de las respuestas de estados, entre otros menores. En sí, durante este capítulo se desarrolló una sinergia de mejora continua entre el firmware y el protocolo. En definitiva el firmware cumple con los requisitos de la versión final del protocolo.

El capítulo 8 presento la oportunidad de realizar las pruebas al sistema completo. En este capítulo se realizaron dos tipos de validaciones: en ambiente virtual y ambiente real.

Con respecto al ambiente virtual, se realizaron diez ejercicios con situaciones tomadas al azar. Cada ejercicio presentó una situación en la cual se tenía que codificar un comando OBDII plus. Estas situaciones presentaron diversos desafíos de activar sistemas o reportar estados por todo el vehículo. En cada caso, el protocolo propuesto fue capaz de codificar un comando o respuesta OBDIIplus valido. A partir de esta muestra representativa se infiere que el protocolo es capaz de codificar comandos y estados para los diversos sistemas que componen los vehículos modernos como se describe en el capítulo 4.

La segunda validación se realizó con los módulos construidos. El sistema tiene 8 estados: Cuartos, luz delantera, luz alta, freno, Reversa, intermitentes, direccional izquierda y direccional derecha. Cada estado se activó usando el módulo HID. El módulo HID tiene un pushbutton para activar y desactivar cada uno de los módulos mencionados. Internamente genera un comando OBDIIplus al activar y/o desactivar

cada estado. Dicho comando es enviado a través del CAN-BUS al resto de los módulos. Adicionalmente a este método, se le envió un comando directo al módulo HID a través del puerto USB/UART. Dicho comando OBDIIplus recibido, el modulo lo reenvió a través del CAN-BUS. De esta manera se probó cada estado, así como combinaciones entre los mismos. En todos los casos, los módulos fueron capaces de codificar y decodificar adecuadamente cada estado requerido. Los módulos respondieron adecuadamente a la acción requerida en todos los casos.

En resumen, de los resultados de las pruebas realizadas se puede inferir que el protocolo propuesto OBDIIplus es capaz de manejar la mayoría de los sistemas no-de-emisiones del vehículo de manera sencilla.

10 Conclusiones

El protocolo fue puesto a prueba en ambientes virtuales y reales. En ambientes reales se cubrieron diversos sistemas del vehículo del tipo no-de-emisiones. En ambiente real se construyeron módulos físicos representativos de un sistema, las luces en este caso. En ambos casos se ejercitó el protocolo propuesto bajo diversas condiciones. En todos los casos fue posible codificar y decodificar los comandos OBDIIplus requeridos. Adicionalmente de ser funcional, el protocolo probó con los objetivos generales de ser fácil de utilizar y de posibilitar módulos universales.

Como sumario, fue posible cumplir con los objetivos principales del proyecto:

- *Desarrollo de un protocolo a partir del actual OBDII.* El OBDIIplus toma las bases del actual OBDII en cuanto a la estructura de sus mensajes en modalidad CAN-BUS (29bits). Una vez definida la estructura, se proponen algunos valores fijos siguiendo los estándares OBDII, para simplificar el propuesto.
- *Fácil de entender e implementar.* A partir de un profundo análisis de los sistemas actuales OBDII, se seleccionaron las estructuras y referencias más adecuadas para el manejo de los subsistemas del vehículo no relacionadas a emisiones. Durante la implementación se experimentaron diversas opciones. A partir de estas se refinó la propuesta para hacerla más sencilla y fácil de implementar.
- *Facilidad de desarrollo e instalación.* La plataforma Arduino seleccionada es una de las más populares con gran soporte. El modelo Arduino nano basado en el microcontrolador Atmel atmega328 seleccionado probó ser suficiente para los requerimientos del proyecto. El presente trabajo se desarrolló a nivel prototipo, con sus limitaciones propias de un modelo solamente de trabajo. Aplicaciones industriales o comerciales tendrán que incluir funciones adicionales como protección ESD, condiciones especiales en el firmware, entre otras. Aun así, se conserva la esencia de solamente requerir un nivel intermedio para su desarrollo. Con respecto a su instalación, solamente es necesario conectar voltaje y las luces al módulo, e interconectar el CAN bus entre ellos. No requiere mayor complejidad

- *Cubrir la mayoría de los sistemas no-de-emisiones del vehículo.* A partir de las tablas del estándar SAE J2178-4, utilizadas por la presente propuesta de protocolo, se pueden cubrir todos los sistemas seleccionados en el capítulo 4 y varios adicionales.
- *Poder adaptarse fácilmente a cualquier vehículo.* Al cubrir el punto de soportar la mayoría de los sistemas presentes en vehículos actuales y anteriores hasta cierto punto, queda cubierto el poder soportar la mayoría de los vehículos. La facilidad de su instalación ayuda a cumplir este objetivo.
- *Bajo costo.* Un sistema comercial en alto volumen similar al prototipo, el cual cubre las funciones más básicas puede tener un costo inferior a los \$20USD. Agregando funciones adicionales y/o componentes de mejor calidad el costo puede elevarse hasta unos \$50USD. El costo comparado con la facilidad de instalación, ahorro de cableado (material y labor), y potencial facilidad de encontrar refacciones lo vuelve un costo altamente competitivo.
- *Proveer funcionalidades adicionales a las soluciones actuales.* El protocolo en si proporciona varias funciones adicionales. La más básica y directa es el reporte de estatus del sistema (ejemplo: foco fundido). Lo siguiente son funcionalidades extendidas, por decir el poder direccionarse a un solo modulo en específico (ejemplo, prender cada foco uno por uno). Todas las anteriores son cubiertas por el OBDIIplus de manera natural, que no están cubiertas por sistemas tradicionales alambrados. Adicional a esto, los módulos de hardware pueden incluir salidas adicionales como luces extra, o funciones extra como dimmer de luces o sensores en los vidrios de las ventanas.

Por todos los anteriores puntos, se puede ver que el protocolo OBDIIplus cumple con el objetivo de ser una buena propuesta fácil de implementar para manejar la mayoría los sistemas del vehículo no relacionados con emisiones.

Los sistemas OBD actuales no cuentan con un sistema único entre compañías automotrices, además de ser información propietaria (ETI, 2016). Este resultado fortalece la propuesta ante la necesidad de un sistema estandarizado único.

La propuesta se ha probado utilizando módulos prototipo, los cuales manejan luces reales de vehículos. Esta es una pequeña parte de todos los posibles sistemas. Las luces y su estado son solo dos de las 45 tablas del SAE J2178-4. Sin mencionar las posibles aplicaciones de usuario final basados en este mismo. Sin embargo, se eligió el sistema de luces para el desarrollo por su sencillez de aplicación y la posibilidad de ejercitar todos los aspectos del protocolo. Utilizando estos módulos fue posible simplificar y poner a punto la propuesta de codificación y decodificación de comandos.

Varios otros casos se tomaron como ejemplo siguiendo los pasos OBDII plus. Se hizo a mano codificación y decodificación para probar en escritorio la propuesta. Todos los casos ejercitados probaron la facilidad de implementación y uso del protocolo.

Con respecto al estándar SAE J2178 el cual se tomó como base, como se comentó en la sección 5.6 además de la definición del header contiene varias tablas que manejan los sistemas no de diagnóstico. Sin embargo, no se encontró referencia alguna que lo mencione o haga uso de estas. La misma SAE lo declaró fuera de uso en 2012. Durante el desarrollo de este trabajo se encontraron dichas tablas por casualidad. Después de su análisis, se encontró que su implementación fue basada en el sistema OBD de los 90's por lo que se volvió extremo complicado al forzar un sistema adelantado a su tiempo en una plataforma aún no madura. Esta puede ser la razón por la cual no tuvo aceptación en la industria. Fue decisión el rescatarlas y darles una implementación practica dentro del marco de este trabajo, el cual contempla entre otros la facilidad de uso y el usar tecnología actual.

La presente propuesta es el mejor escenario encontrado buscando respetar tanto como sea posible la intención de diseño original del J2178, sin utilizar la codificación complicada del mismo. A la par, buscando el cumplimiento de las recomendaciones de los más de 30 estándares americanos y europeos que conforman el OBD, sin contar sus secciones.

Esta propuesta equilibra el requisito de mensajes para sistemas vehiculares no-de-emisiones, con un protocolo fácil de usar. Pero, también teniendo en cuenta el estado actual de los sistemas embebidos disponibles comercialmente ahora que son más poderosos y capaces que hace una década.

11 Contribuciones originales

El popular protocolo automotriz OBDII en uso hoy en día actualmente se enfoca solamente al tren motriz y control de emisiones. La principal contribución de este trabajo es el desarrollo de una plataforma de protocolo de interconexión y control de periféricos comunes en los autos de hoy en día. Dicha plataforma está orientada de manera de ser una extensión de la mencionada OBDII.

Esta plataforma es inherentemente abierta, u open source, de manera de atraer el interés de más desarrolladores a todos niveles que impulsen la plataforma. Los fabricantes automotrices utilizan protocolos propietarios para la interacción entre sus sistemas. Estos no son disponibles al público en general por lo que su desarrollo e implementación está sujeto al sistema de licencias y costos de cada fabricante. Sin mencionar que cada uno maneja el suyo propio. La contribución original de este trabajo está en ser un sistema abierto al desarrollo independiente de la marca o modelo del vehículo. El protocolo contempla los sistemas más comunes que manejan la vasta mayoría de los vehículos. De esta manera, se pretende impulsar el desarrollo público de estos sistemas. Esto conlleva entre otros beneficios la libre competencia y desarrollo comunitario. Tal como se ha visto en otras iniciativas de plataforma libre, estos beneficios propician la creación de sistemas cada vez más competentes a costos competitivos. De esta manera, se pueden llegar a lograr como ejemplo módulos universales para puertas, luces, asientos, control de a/c entre otros que se puedan comercializar en cualquier tienda. Dicho módulo se podría adquirir con la confianza de que funcionaría para el propio vehículo.

12 Evidencia de la productividad académica resultado del proyecto de tesis

12.1 Coloquio UPAEP 2015

Asistencia y presentación como ponente en el Coloquio de Posgrado UPAEP 2015 – Puebla, Puebla. Tema “Mini-liderazgo ocasional para el líder día a día”

** Constancia original extraviada en correo

Re: Artículo Coloquio 2015

YESICA MAYETT MORENO <yesica.mayett@upaep.mx>

Mon 5/25/2015 5:52 PM

To: alex G. Blanco <alex_g_blanco@hotmail.com>;

C: rubi.salazar@upaep.mx <rubi.salazar@upaep.mx>; agarcia@rainbird.com <agarcia@rainbird.com>;

Estimado Alex:

Por este medio nos es grato saludarte e informarte que el Comité Evaluador de la mesa de trabajo de Dirección de Organizaciones y Finanzas ha leído cuidadosamente tu trabajo, por lo que considera valiosa la presentación de tus avances de investigación en la modalidad de "Ponencia".

Para ser publicado en las Memorias del Coloquio le pedimos atender lo siguiente:

Para la exposición del día 26 de junio les pedimos preparen su exposición de máximo 15 minutos (10 a 12 diapositivas en Power Point) y envíarla por correo electrónico (yesica.mayett@upaep.mx) a más tardar el 22 de junio.

El día de su ponencia será el viernes 26 de junio. El horario se lo comunicaremos el lunes 22 de junio.

Sin más por el momento quedamos a sus órdenes y te felicitamos por tu esfuerzo:

Atte.

COMITÉ EVALUADOR

El 8 de mayo de 2015, 0:17, alex G. Blanco <alex_g_blanco@hotmail.com> escribió:

Buen día

Favor de encontrar anexo un artículo respecto a un modelo de liderazgo. Favor de considerarlo para incluirse en el próximo Coloquio 2015.

Con toda confianza, comentarios respecto al mismo son bienvenidos.

Muchas gracias y quedo en espera de sus comentarios

Saludos, alex García Blanco

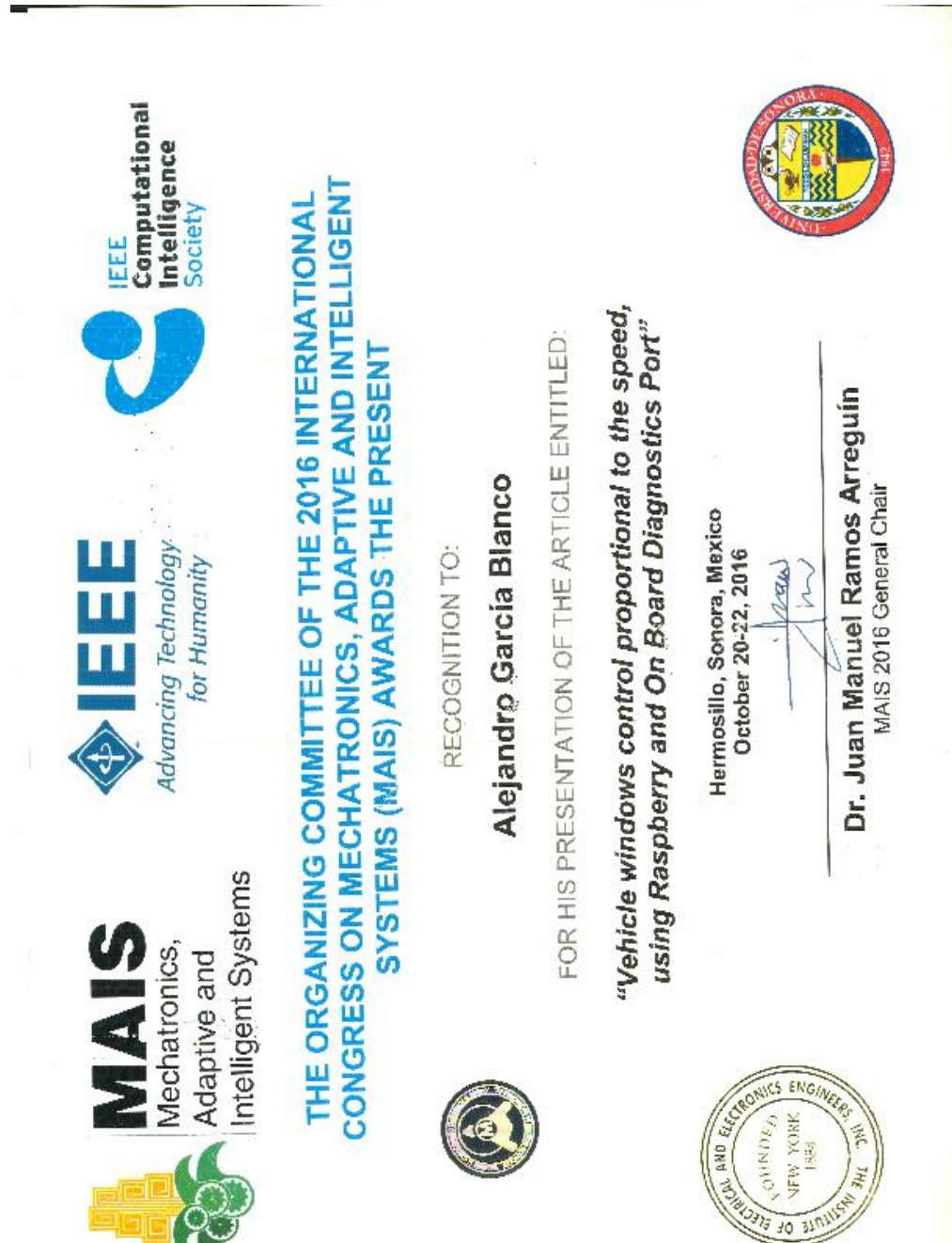
12.2 Coloquio UPAEP 2016

Asistencia y presentación como ponente en el Coloquio de Posgrado UPAEP 2016, Puebla, Puebla. Presentación de tema “Los sistemas automotrices y una Propuesta de OBDII plus”



12.3 Coloquio IEEE/MAIS 2016

Asistencia al Congreso de la Asociación Mexicana de Mecatrónica, simultánea con IEEE Conference on Mechatronics, Adaptive and Intelligent Systems. Presentación del artículo "Vehicle Windows control proportional to the speed, using Raspberry and On Board Diagnostics Port" ante IEEE (ISBN: 978-607-9394-07-3). Hermosillo, Sonora 2016



12.4 Wikilibros – El OBDII Completo

Elaboración de un libro online en la plataforma Wikimedia – Wikilibros con el estado del arte del trabajo. El libro está enfocado a las bases del sistema OBDII

https://es.wikibooks.org/wiki/El_OBDII_Completo



The screenshot shows a web browser window displaying the Wikilibros page for 'El OBDII Completo'. The browser's address bar shows the URL https://es.wikibooks.org/wiki/El_OBDII_Completo. The page features the Wikilibros logo on the left, which includes the text 'WIKILIBROS' and 'Libros libres para un mundo libre'. Below the logo is a sidebar with various navigation links such as 'Portada', 'Categorías de libros', 'Cambios recientes', 'Páginas nuevas', 'Libro aleatorio', 'Ayuda', 'Donaciones', 'Comunidad', 'Café', 'Portal de la comunidad', 'Subproyectos', 'Recetario', 'Wikichicos', 'Imprimir/exportar', 'Crear una colección', 'Descargar como PDF', 'Versión para imprimir', 'Herramientas', 'Lo que enlaza aquí', 'Cambios relacionados', 'Subir archivo', and 'Páginas especiales'. The main content area has a title 'El OBDII Completo' and a subtitle 'El OBDII Completo'. Below the title is a section for 'Desarrollo' and a 'Presentación' section with an '[editar]' link. The 'Presentación' section contains two paragraphs of text. The first paragraph discusses the advancement of technology and the use of electronic devices in cars, mentioning the incorporation of sensors and the development of a computer to centralize information. The second paragraph discusses the development of the OBD (On Board Diagnostics) protocol in the 1990s, which standardized communication in cars through different companies, allowing for the transmission of critical status messages.

El OBDII Completo

Desarrollo: 

Presentación [\[editar\]](#)

La tecnología al ir avanzando ha permitido hacer más complejos y confiables los diversos dispositivos que se usan en la actualidad. En los automóviles, en la década de los '90 es cuando se señala como la era donde la electrónica llegó a estos. Iniciando con la incorporación cada vez más común de sensores eléctricos y en diversos lugares, hasta el desarrollo de una computadora que centralice toda esa información y coordine los actuadores correspondientes. Dicha computadora abrió la posibilidad de hacer más dinámicos e inteligentes los sistemas automotrices, permitiendo por ejemplo incrementar la eficiencia de combustible a la vez de reducir emisiones tóxicas.

A inicios de esta década de los 90's surgió el primer protocolo de comunicación, OBD (On Board Diagnostics) el cual estandarizó la manera de comunicarse en los autos a través de las diferentes compañías. Estos mensajes eran básicos y solamente de estados críticos del tren motriz (Roberts, 2009)^[1].

12.5 Artículo – Revista Ciencia y Desarrollo

Elaboración de artículo “Entendiendo el OBDII” para la revista Ciencia y Desarrollo (Conacyt). En proceso de publicación.

RE: Revisión recibida

Margarita Alura Guzmán Gómora <malurag@conacyt.mx>

Mon 9/4/2017, 1:43 PM

To: alex G. Blanco <alex_g_blanco@hotmail.com>;

Gracias, Maestro. Seguiremos en contacto.

Saludos cordiales.

De: alex G. Blanco [mailto:alex_g_blanco@hotmail.com]

Enviado el: sábado, 2 de septiembre de 2017 16:07

Para: Margarita Alura Guzmán Gómora <malurag@conacyt.mx>

Asunto: Re: Dictamen sobre texto enviado a CyD

Buen día

Muchas gracias por la oportunidad de publicar con ustedes. Adjunto esta la revisión del artículo con las enmiendas sugeridas. Cualquier sugerencia adicional o enmienda con toda confianza haganme saber

Gracias y saludos

alex García Blanco

From: Margarita Alura Guzmán Gómora <malurag@conacyt.mx>

Sent: Friday, August 18, 2017 11:54 AM

To: alex_g_blanco@hotmail.com

Subject: Dictamen sobre texto enviado a CyD



Maestro Blanco, informo a usted que su texto enviado a esta redacción ya sido revisado y su publicación condicionada a la aplicación de algunas enmiendas que le presento en el documento anexo, con el fin de cumplir con el perfil de la revista.

Considere usted la pertinencia de aplicarlas y, en tal caso, por favor envíenos su nueva versión para su publicación.

Saludos cordiales.

Margarita A. Guzmán Gómora

Ciencia y Desarrollo

Coordinadora editorial

Insurgentes Sur 1582, 3er piso, Norte

Col. Crédito Constructor

B. Juárez 03940, D. F.

12.6 Artículo – IEEE Capitulo Latinoamérica

Elaboración de artículo para la IEEE Latinoamérica. En proceso de aceptación.

 editor-in-chief-r9@ieee.org a través de pea.usp.br 18 may.   

 inglés > español [Traducir mensaje](#) [Desactivar para: inglés](#)

Los revisores recomendaron modificaciones en su artículo..
Por favor, acrescente una página al artículo conteniendo los comentarios de los revisores y sus respuestas.
Responda a los comentarios de los revisores en los casos en que fueron atendidos, o en los casos em que usted no concuerde o merezca alguna explicación.
Después de hacer las modificaciones reenvíe su artículo con las modificaciones a través del site.

Tiempo máximo para el envío del artículo es de 01 (un) mes.
Si necesita más tiempo, por favor póngase en contacto con nosotros.

Atentamente,
Editor jefe **IEEE-R9**.

18/05/2017 Modify.

Dear Author
This paper has clearly in language used, it presents a good state of the art review, however it is too long.
I consider that it could take more space describing the proposed prototype, and highlight its real life or potential applications.
An introducción and the like the one presented is very appreciated, however it needs to be followed by a detailed description of the system or prototype developed. It needs to highlight the contribution of the paper to the subject. At this point it seems more like a technology review than a R&D paper.
Best regards

13 Referencias

13.1 Estándares SAE sobre OBD-II

- J1113 – Procedimientos de medición de susceptibilidad electromagnética de componentes en el vehículo
- J1211 – Procedimientos y recomendaciones para diseño de equipo electrónico
- J1213 – Glosario de términos para redes vehiculares
- J1547 – Procedimientos de medición de susceptibilidad electromagnética de componentes de inyección de modo común
- J1699 – Pruebas de cumplimiento y métodos de prueba para SAE J1850
- J1850 – Define el protocolo serial de datos. Hay dos variantes 10.4 kbit/s (un cable, VPW) y 41.6 kbit/s (dos cables, PWM). Mayormente usado por fabricantes USA, PCI (Chrysler, 10.4 kbit/s), Class 2 (GM, 10.4 kbit/s), y SCP (Ford, 41.6 kbit/s)
- J1879 – Calificación general y criterios de aceptación de producción para circuitos integrados en aplicaciones automotrices
- J1930-DA Comunicación entre el vehículo y equipo externo para diagnóstico de emisiones. Guía de términos, definiciones, abreviaciones y acrónimos; DA es un anexo listado en formato digital. Técnicamente equivalente al ISO15031-2
- J1939 - comunicación en vehículos de carga pesada
- J1962 – Define el conector físico para la interface OBD-II
- J1978 – Define estándares de operación mínimos para herramientas de escaneo OBD-II
- J1979 – Define estándares para servicios o modos de PIDs del \$01 al \$0A
- J1979-DA Modos de Prueba. DA es un anexo listado en formato digital. Técnicamente equivalente al ISO15031-5
- J2008 – Organización recomendada para la información de servicio del vehículo
- J2012 – Define códigos de problema estándar y definiciones (del inglés DTC diagnostic Trouble Codes). Técnicamente equivalente al ISO15031-6 y en partes al ISO27145-2. El estándar con –DA es un anexo con el listado en formato digital

- J2178 Mensajes de comunicación de datos por red clase B en formato serial para sistemas automotrices
 - J2178-1 – Define estándar para formatos de encabezado de mensaje de red y asignación de dirección física
 - J2178-2 – Da definiciones de parámetros de datos
 - J2178-3 – Define estándares para IDs de cuadros de mensajes de red para encabezados de un solo byte (del inglés frame IDs for single-byte forms of headers)
 - J2178-4 – Define estándar para mensajes de red con encabezados de tres bytes
- J2186 – Seguridad de Datos. Este estándar establece una práctica uniforme para proteger los módulos computarizados del vehículo contra accesos no autorizados a través del conector de datos del vehículo. Los módulos en cuestión son los que tienen alguna memoria cuyo contenido puede ser accedido o alterado a través del conector de datos.
- J2190 – Define modos de prueba extendidos, cubriendo modos de prueba relacionados con emisiones y otros no relacionados. Este estándar estaba intencionado a complementar SAE J1979, sin embargo fue cancelado en 2008 por poco uso. Algunos estándares cubren partes del mismo
- J2205 – Protocolos expandidos de diagnóstico para herramientas de escaneo
- J2284-3 – Define la capa física y de enlace de datos del CAN de 500 kbit/s
- J2300 – Procedimientos de cumplimiento de pruebas de herramientas de diagnósticos
- J2411 – Define el protocolo GMLAN (CAN de un solo cable) usado en los vehículos GM nuevos. Frecuentemente accesible en el conector OBD como pin 1 en los autos nuevos GM
- J2534 – Prácticas Recomendadas para reprogramación de vehículos

13.2 Estándares ISO

ISO 9141: Sistemas de diagnóstico, ISO 1989

- Parte 1: Requerimientos para el intercambio de información digital
- Parte 2: Requerimientos CARB (California Air Resources Board) para el intercambio de información digital. Sistemas de diagnóstico para intercambio de información serial asíncrono, similar al RS232 y al ISO15031-5
- Parte 3: Verificación de la comunicación entre el vehículo y la herramienta de escaneo OBD

ISO 11898: Controller area network (CAN). ISO, 2003.

- Parte 1: Señalización en la capa física y de enlace de datos
- Parte 2: Unidad de medio de acceso de alta velocidad
- Parte 3: Interface dependiente del medio, tolerante a errores, de baja velocidad
- Parte 4: comunicación disparada por tiempo

ISO 14229: Servicios de diagnóstico unificado (del inglés UDS Unified Diagnostic Services) en vehículos terrestres. ISO, 2013

- Parte 1: Especificaciones y Requerimientos
- Parte 2: Servicios de capa de sesión. Se relaciona para OBD con los protocolos ISO 15765-4 DoCAN e ISO 14230-4 DoK-Line
- Parte 3: UDS en implementaciones sobre CAN (UDSonCAN)
- Parte 4: UDS en implementaciones en FlexRay (UDSonFR)
- Parte 5: UDS en implementaciones sobre internet (UDSonIP)
- Parte 6: UDS en implementaciones de línea K (UDSonK-Line)
- Parte 7: UDS en implementaciones sobre Redes Locales (UDSonLAN)

ISO 14230: sistemas de diagnóstico, protocolo de palabras clave, ISO, 2013.

- Parte 1: Capa física
- Parte 2: Capa de enlace de datos utilizando línea K (DoK-Line)
- Parte 3: Capa de aplicación KWP2000

- Parte 4: Requerimientos para sistemas relacionados con emisiones. DoK-Line diagnóstico sobre línea K o KWP2000, similar al ISO9141-2

ISO 15031: Comunicación entre vehículos y equipo externo para diagnósticos relacionados a emisiones, ISO, 2010.

- Parte 1: Información general y definición de casos de uso
- Parte 2: Guía en el uso de términos, definiciones, abreviaciones y acrónimos. Técnicamente equivalente al SAE J1930-DA
- Parte 3: conector de diagnóstico y circuitos eléctricos relacionados, especificaciones y usos
- Parte 4: Equipo de prueba externo
- Parte 5: Servicios de diagnóstico relacionados a emisiones. Modos de prueba. Técnicamente similar al SAE J1850.
- Parte 6: Definiciones de códigos de diagnóstico (del inglés DTC diagnostic Trouble Codes). Técnicamente equivalente al SAE J2012-DA
- Parte 7: Seguridad del enlace de datos

ISO 15765: Diagnósticos en sistemas Controller Area Networks (del inglés DoCAN Diagnostic communication over Controller Area Network) ISO 2004

- Parte 1: Información General
- Parte 2: Servicios de capa de red (del inglés Transport protocol and network layer services)
- Parte 3: Implementación de servicios unificados de diagnóstico (UDS on CAN)
- Parte 4: Requerimientos para sistemas relacionados con emisiones DoCAN diagnóstico sobre CAN. Modos o servicios de prueba del \$01 al \$0A

ISO 27145: Implementación del diagnóstico armonizado mundial buscando una sola regulación (del inglés World Wide Harmonized On board Diagnostics WWH-OBD). ISO 2012

- Parte 1: Información General y definición de casos de uso
- Parte 2: Diccionario de Datos (códigos) Común
- Parte 3: Diccionario de mensajes comunes
- Parte 4: Conexión entre el vehículo y equipos de prueba
- Parte 5: cumplimiento de aplicaciones del vehículo y de sistemas que lean códigos
- Parte 6: Equipo de prueba externo (requerimientos)

13.3 Referencias bibliográficas

Actron. (2006). *How's Your Car Running? Check Your Smartphone*. Business Wire (English).

Alberich, T. (1993). *La crisis de los movimientos sociales y el asociacionismo de los años noventa*. Documentación social, 90, 101-114.

AUSTIN, M. (2014). *Saturday Mechanic: CHECK PLEASE*. Popular Mechanics, 191(4), 100-101.

Verantwortlich i.S.d. § 6 MDSStV & 6 TDG. (2015). *AUTOSAR Classic Platform*. 17 febrero 2016, de AUTOSAR development partnership Sitio web: <https://www.autosar.org/>

Breeves, S.. (2013). *Accessing Advanced Ford PIDs With a Scan Tool*. Mayo 30, 2016, de Taurus Car club of America Sitio web: <http://www.taurusclub.com/forum/124-taurus-sable-general-discussion/182569-accessing-advanced-ford-pids-scan-tool.html>

Cagan, M. (2005). *How to write a good PRD*, silicon Valley product Group. USA

Carley, L. (2008). *Tech Update: Scan Tools & Code Readers*. Fleet Equipment, 20-22.

Cassias, I., & Kun, A. L. (2007). *Vehicle telematics: A literature review*. Univ. New Hampshire, Durham, NH, USA, ECE. P, 54.

CiA. (2016). *History of CAN technology*. 17 febrero 2016, de CiA - CAN in Automation Association of Austria Sitio web: <http://www.can-cia.org/can-knowledge/can/can-history/>

CHILDERS, B. (2013). *BE A MECHANIC with ANDROID and LINUX!*. Linux Journal, (234), 64-72.

DEC. (2000). *Automotive diagnosis tools*. Argentina: DEC Automotive.

ETI. (2015). *ETI Brochure*. Mayo 30, 2016, de The Equipment and Tools Institute Sitio web: <http://www.etools.org/resources/Documents/Marketing/ETI%20Brochure%202015.pdf>

G. Leen & D.Heffernan. *Expanding automotive electronic systems*. IEEE Computer, 35(1), January 2002, pp 88-93

Gary.(2015). *OBDSim*. 1 junio 2015, de icculus.org Sitio web: <http://icculus.org/obdgpslogger/obdsim.html>

Gary. (2009). *Free Software OBD II Simulator redux* . 1 junio 2015, de vBulletin Solutions Sitio web: <http://www.mp3car.com/engine-management-obd-ii-engine-diagnostics-etc/137428-free-software-obd-ii-simulator-redux.html>

González, P. (2008). *Electrónica del Automóvil OBDII*. Barcelona, España: Escuela Técnica Superior de Ingeniería Industrial de Barcelona.

Green, J. (2014). *The OBD2 port*. Automotive Industries, 193(4), 6-8.

Henderson, Bob; Haynes, John. (2006). *The Haynes OBD-II & Electronic Engine management systems manual*. California, USA: Haynes Publishing Group.

Investor's Business, D. (2013, March 14). *High-tech upgrade for old cars*. Investors Business Daily. p. A02.

López, J.B. (2008). *PROGRAMA PARA SISTEMA DE DIAGNÓSTICO DE VEHÍCULOS (OBDII) Y COMPROBADOR DE ACTUADORES QUE FUNCIONEN CON SEÑALES (PWM)*. Elche, España: ESCUELA POLITÉCNICA SUPERIOR DE ELCHE.

Kiencke, U., Dais, S., & Litschel, M. (1986). *Automotive serial controller area network* (No. 860391). SAE Technical Paper.

MOORE, A. (2013). *10 Modes of OBDII*. Motor Age, 132(4), 40-48.

Navet, N. & Simonot-Lion.. (2009). *Automotive embedded systems handbook*. Estados Unidos de America: Taylor & Francis Group.

Posada, F & Bandivadekar, A . (2015). *Global overview of on-board diagnostic (OBD) systems for heavy-duty vehicles*. The International council on Clean air transportation, on line.

PR, N. (2015, January 5). *Car Connection 2.0 Brings OEM Style Telematics to the Aftermarket*. PR Newswire US.

Real Academia Española. (Edición del Tricentenario). *Diccionario de la Lengua Española*. 2016, de Asociación de Academias de la Lengua Española Sitio web: <http://dle.rae.es/>

Roberts, C. (2009, January). *OBDII is coming; are you ready?*. Fleet Equipment. p. 12.

Viladrich, M. (2004). *Las principales aportaciones a la teoría de la regulación medioambiental. Los últimos cuarenta años*. Economía agraria y recursos naturales, (8), 41-62.

Warnecke, A.. (2016). *El taller del futuro: comunicación directa con el auto*. Marzo 25, 2016, de Dpa News Sitio web: <http://prodigy.msn.com/es-mx/autos/noticias/el-taller-del-futuro-comunicaci%c3%b3n-directa-con-el-auto/ar-BBqQbi8?li=AAggsml>

Wiki creative commons. (2016). *OBD-II PIDs*. Marzo 28, 2016, de The Wikimedia Foundation Sitio web: https://en.wikipedia.org/wiki/OBD-II_PID

Wiki creative commons. (2016). *PIDs PBDII*. Marzo 28, 2016, de Copro Sitio web: http://copro.com.ar/PIDs_OBD-II.html

WOJDYLA, B. (2013). *A Cloud for Every Car*. Popular Mechanics, 190(7), 108-110.

WONG, B. (2013, September 5). *Dongles Watch Your Driving*. Electronic Design. p. 13.

14 Anexos

14.1 Tablas del SAE j2178-4

Las siguientes páginas incluyen las tablas de las secciones 8 y 9 del estándar SAE J2178-4 revisión 1999. Estas tablas se utilizan para determinar el ID primario, ID secundario, significado físico del bit Q y el direccionamiento extendido. En casos excepcionales un byte de datos adicionales

Table 34 - External Access
 Table 35 - Seat Motion/Control
 Table 36 - Windows
 Table 37 - Steering Column
 Table 38 - Seat Switches
 Table 39 - Restraints
 Table 40 - Exterior Lamps Outage
 Table 41 - Exterior Lamps
 Table 42 - Interior Lamps Outage
 Table 43 - Interior Lamps
 Table 44 - Tires
 Table 45 - Defrost
 Table 46 - Displays
 Table 47 - Exterior Environment
 Table 48 - Interior Environment
 Table 49 - Time/Date
 Table 50 - Vehicle Identification
 Table 51 - Network Control

8.1 Engine Torque—(See Table 6.)

Primary ID: Engine Torque
 Command ID: \$08
 Status ID: \$09

TABLE 6—ENGINE TORQUE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Crankshaft Torque - Percent	01	L / R	—	x	—	1020
Crankshaft Torque - Absolute	02	L / R	—	x	—	1019
Maximum Crankshaft Torque	10	R	—	x	—	1033

8.2 Engine Air Intake—(See Table 7.)

Primary ID: Engine Air Intake
 Command ID: \$0A
 Status ID: \$0B

TABLE 7—ENGINE AIR INTAKE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Boost On/Off	01	L / R	On	Off	--	--
Boost Value	02	R	--	x	--	1021
Mass Air Flow Rate	08	R	--	x	--	0010
Manifold Absolute Pressure	11	R	--	x	--	000B
Intake Air Temperature	20	R	--	x	--	000F

8.3 Throttle—(See Table 8.)

Primary ID: Throttle
 Command ID: \$12
 Status ID: \$13

TABLE 8—THROTTLE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Sensor 1 Position	01	R	--	x	--	0011
Sensor 2 Position	02	R	--	x	--	1035
Sensor 3 Position	03	R	--	x	--	1036
Throttle Kicker	10	L / R	E	D	--	--
Pedal Position	11	R	--	x	--	1034

8.4 Air Conditioning Clutch—(See Table 9.)

Primary ID: Air Conditioning Clutch
 Command ID: \$14
 Status ID: \$15

TABLE 9—AIR CONDITIONING CLUTCH MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Enable	01	L / R	E	D	--	--
Enable Pending	02	L / R	Y	N	--	--
Load	03	R	--	x	--	100E
Required	04	L / R	Y	N	--	--

8.5 Engine RPM—(See Table 10.)

Primary ID: Engine RPM
 Command ID: \$1A
 Status ID: \$1B

TABLE 10—ENGINE RPM MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Low Resolution	01	R	--	x	--	1022
High Resolution	02	R	--	x	--	000C
Idle Speed	20	L / R	E	D	--	1023

8.6 Wheels—(See Table 11.)

Primary ID: Wheels
 Command ID: \$24
 Status ID: \$25

TABLE 11—WHEELS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Wheel Speed - Low Resolution	01	R	--	x	8.1	2801
Wheel Slip	02	R	--	x	8.1	2809
Wheel Load	04	R	--	x	8.1	2839
Wheel Speed - High Resolution	11	R	--	x	8.1	2802

8.7 Vehicle Speed—(See Table 12.)

Primary ID: Vehicle Speed
 Command ID: \$28
 Status ID: \$29

TABLE 12—VEHICLE SPEED MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Low Resolution	01	R	--	x	--	000D
High Resolution	02	R	--	x	--	6001

8.8 Traction Control—(See Table 13.)

Primary ID: Traction Control
 Command ID: \$2A
 Status ID: \$2B

TABLE 13—TRACTION CONTROL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Engine Traction Active	01	R	Y	N	--	--
Brake Traction Active	02	R	Y	N	--	--
System On/Off	04	L / R	On	Off	--	--
System Active	05	R	Y	N	--	--
System Faulted	0A	R	Y	N	--	--
System Enable Sw. Active	20	R	Y	N	--	--

8.9 Brakes—(See Table 14.)

Primary ID: Brakes
 Command ID: \$32
 Status ID: \$33

TABLE 14—BRAKES MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
ABS Active	03	R	Y	N	--	--
ABS System On / Off	04	L / R	On	Off	--	--
Fluid Life Reset	09	L / R	R	-R	--	--
System Faulted	0A	R	Y	N	--	--
Fluid Temperature	10	R	--	x	--	281A
Supply Pump Fluid Pressure	11	R	--	x	--	2819
Fluid Level - Percent	12	R	--	x	--	2841
Fluid Level - Volume	13	R	--	x	--	2842
Fluid Remaining Life	14	R	--	x	--	2843
Fluid Capacity	16	R	--	x	--	2844
Parking Brake Sw. Active	20	R	Y	N	--	--
Torque Converter Clutch - Brake Sw. Active	21	R	Y	N	--	--
Brake Lamp - Brake Sw. Active	22	R	Y	N	--	--
Fluid Life Reset Sw. Active	29	R	Y	N	--	--

8.10 Steering/Steering Wheel—(See Table 15.)

Primary ID: Steering/Steering Wheel
 Command ID: \$34
 Status ID: \$35

TABLE 15—STEERING/STEERING WHEEL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Fluid Life Reset	09	L / R	R	-R	--	--
Fluid Temperature	10	R	--	x	--	3005
Fluid Pressure	11	R	--	x	--	3006
Fluid Level - Percent	12	R	--	x	--	3007
Fluid Level - Volume	13	R	--	x	--	3008
Fluid Remaining Life	14	R	--	x	--	3009
Fluid Capacity	16	R	--	x	--	300B
Steering Wheel Angle	18	R	--	x	--	3001
Steering Wheel Rate	19	R	--	x	--	300C
Steering Wheel Torque	1A	R	--	x	--	300D
Wheel Steer Angle	20	R	--	x	8.1	300E
Fluid Life Reset Sw. Active	29	R	Y	N	--	--

8.11 Transmission/Transaxle/PRNDL—(See Table 16.)

Primary ID: Transmission/Transaxle/PRNDL
 Command ID: \$3A
 Status ID: \$3B

TABLE 16—TRANSMISSION/PRNDL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Torque Converter Lock(ed)	01	L/R	Y	N	--	--
Clutch Enable	02	L/R	E	D	--	--
Actual Gear Position w/ Shift In Progress	03	R	Y	N	--	180E
Range Selected (PRNDL position)	04	R	--	x	--	1809
Transfer Case (4WD)	05	R	--	x	--	180A
Commanded Gear	06	L/R	--	x	--	180D
Range Actual (PRNDL sense at transmission)	07	R	--	x	--	1806
Transmission Kickdown	08	L/R	Y	N	--	--
Fluid Life Reset	09	L/R	R	--R	--	--
Fluid Temperature	10	R	--	x	--	180B
Fluid Pressure	11	R	--	x	--	180C
Fluid Level - Percent	12	R	--	x	--	1801
Fluid Level - Volume	13	R	--	x	--	1802
Fluid Remaining Life	14	R	--	x	--	1804
Fluid Capacity	16	R	--	x	--	1803
Park/Neutral Sw. Active	20	R	Y	N	--	--
Fluid Life Reset Sw. Active	29	R	Y	N	--	--

8.12 Engine Sensors - Other—(See Table 17.)

Primary ID: Engine Sensors - Other
 Command ID: \$46
 Status ID: \$47

TABLE 17—ENGINE SENSORS—OTHER MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Converter Bank 1, Sensor 1 Heater	01	L/R	On	Off	--	--
Converter Bank 1, Sensor 2 Heater	02	L/R	On	Off	--	--
Converter Bank 2, Sensor 1 Heater	03	L/R	On	Off	--	--
Converter Bank 2, Sensor 2 Heater	04	L/R	On	Off	--	--
Converter Bank 1, Sensor 1 Temperature	11	R	--	x	--	1037
Converter Bank 1, Sensor 2 Temperature	12	R	--	x	--	1038
Converter Bank 2, Sensor 1 Temperature	13	R	--	x	--	1039
Converter Bank 2, Sensor 2 Temperature	14	R	--	x	--	103A

8.13 Engine Coolant—(See Table 18.)

Primary ID: Engine Coolant
 Command ID: \$48
 Status ID: \$49

TABLE 18—ENGINE COOLANT MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Fan 1 Speed	01	L/R	On	Off	--	102A
Fan 2 Speed	02	L/R	On	Off	--	103B
Fan 1 Clutch	06	L/R	E	D	--	--
Fan 2 Clutch	07	L/R	E	D	--	--
Fluid Life Reset	09	L/R	R	--R	--	--
Fluid Temperature	10	R	--	x	--	0005
Fluid Pressure	11	R	--	x	--	1029
Fluid Level - Percent	12	R	--	x	--	1026
Fluid Level - Volume	13	R	--	x	--	1027
Fluid Remaining Life	14	R	--	x	--	103D
Fluid Capacity	16	R	--	x	--	1028
Fluid Life Reset Sw. Active	29	R	Y	N	--	--
Fluid Temperature High	30	R	Y	N	--	--
Low Coolant Level	32	R	Y	N	--	--

8.14 Engine Oil—(See Table 19.)

Primary ID: Engine Oil
 Command ID: \$4A
 Status ID: \$4B

TABLE 19—ENGINE OIL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Fluid Life Reset	09	L/R	R	--R	--	--
Fluid Temperature	10	R	--	x	--	102B
Fluid Pressure	11	R	--	x	--	102F
Fluid Level - Percent	12	R	--	x	--	102C
Fluid Level - Volume	13	R	--	x	--	102D
Fluid Remaining Life	14	R	--	x	--	1030
Oil Viscosity	15	R	--	x	--	103F
Fluid Capacity	16	R	--	x	--	102E
Fluid Life Reset Sw. Active	29	R	Y	N	--	--
Fluid Temperature High	30	R	Y	N	--	--
Low Oil Level	32	R	Y	N	--	--

8.15 Engine Systems - Other—(See Table 20.)

Primary ID: Engine Systems - Other
 Command ID: \$52
 Status ID: \$53

TABLE 20—ENGINE SYSTEMS—OTHER MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Engine Redline - Low Resolution	01	R	--	x	--	1048
Number Engine Cylinders	02	R	--	x	--	1040
Number Valves/Cylinder	03	R	--	x	--	1041
Engine Running	04	R	Y	N	--	--
Engine Displacement	05	R	--	x	--	1043
Engine Redline - High Resolution	10	R	--	x	--	1049
Engine Accelerating	20	R	Y	N	--	--

8.16 Suspension—(See Table 21.)

Primary ID: Suspension
 Command ID: \$58
 Status ID: \$59

TABLE 21—SUSPENSION MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Ride Settling	01	L/R	--	x	--	3804
Fluid Life Reset	09	L/R	R	--R	--	--
System Faulted	0A	R	Y	N	--	--
Fluid Temperature	10	R	--	x	--	3805
Fluid Pressure	11	R	--	x	--	3806
Fluid Level - Percent	12	R	--	x	--	3807
Fluid Level - Volume	13	R	--	x	--	3808
Fluid Remaining Life	14	R	--	x	--	3809
Fluid Capacity	16	R	--	x	--	380A
Fluid Life Reset Sw. Active	29	R	Y	N	--	--

8.17 Vehicle Speed Control—(See Table 22.)

Primary ID: Vehicle Speed Control
 Command ID: \$62
 Status ID: \$63

TABLE 22—VEHICLE SPEED CONTROL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
System On/Off	01	L/R	On	Off	--	--
Set Speed	02	L/R	E	D	--	A015
Coast Mode	03	L/R	E	D	--	--
Resume Mode	04	L/R	E	D	--	--
Accelerate Mode	05	L/R	E	D	--	--
Speed Control Active	06	R	Y	N	--	--
On/Off Sw. Active	21	R	Y	N	--	--
Set Speed Sw. Active	22	R	Y	N	--	--
Coast Sw. Active	23	R	Y	N	--	--
Resume Sw. Active	24	R	Y	N	--	--
Accelerate Sw. Active	25	R	Y	N	--	--

8.18 Electric Vehicle Energy Transfer System

Primary ID: Electric Vehicle Energy Transfer System
 Command ID: \$70
 Status ID: \$71

TABLE 23—ELECTRIC VEHICLE ENERGY TRANSFER SYSTEM

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
ETS Sleep	01	L/R/M	Y	N	—	—
Delay Timer Expired	02	L/R/M	Y	N	—	—
App Comm State	03	L/R/M	—	x	—	C825
EVSE Configuration	04	L/R/M	—	—	—	C808
Transfer Type	05	L/R/M	—	—	—	C81E
Voltage Mode Control	06	L/R/M	—	—	—	C823
Pulse Mode Enabled	07	L/R/M	Y	N	—	—
Vehicle Ready	08	L/R/M	—	—	—	C821
EVSE Ready	09	L/R/M	—	—	—	C809
Coupling Proximity Detected	0A	L/R/M	Y	N	—	—
Power Out of Range	0B	L/R/M	—	—	—	C817
Sw DC Present	0C	L/R/M	Y	N	—	—
Sw AC Present	0D	L/R/M	Y	N	—	—
Transfer Ready	0E	L/R/M	Y	N	—	—
Pulse Period	0F	L/R/M	—	—	—	C81A
Power Level	10	L/R/M	—	—	—	C816
Max Transfer Power	16	L/R/M	—	—	—	C813
Conversion Power Range	17	L/R/M	—	—	—	C803
Requested Stage Index	18	L/R/M	—	—	—	C81B
Stage Power Range	19	L/R/M	—	—	—	C81D
LMS Preference Toggle	1A	L/R/M	Y	N	—	—
LMS Preference Override	1B	L/R/M	Y	N	—	—
Base Charging Complete	1C	L/R/M	Y	N	—	—

TABLE 23—ELECTRIC VEHICLE ENERGY TRANSFER SYSTEM (CONTINUED)

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Battery Design Capacity	1D	L/R/M	—	—	—	C800
Battery SOC	1E	L/R/M	—	—	—	C801
Conversion Load	1F	L/R/M	—	—	—	C802
Current Limit	20	L/R/M	—	—	—	C804
Current Limit Mandate	21	L/R/M	—	—	—	C805
Delay Timer Count	22	L/R/M	—	—	—	C806
Delay Timer Enable	23	L/R/M	Y	N	—	—
Delay Timer Period	24	L/R/M	—	—	—	C807
EVSE Location	25	L/R/M	INDOOR	OUTDOOR	—	—
LMS Current Limit Mandate	26	L/R/M	—	—	—	C80A
LMS Current Limit Preference	27	L/R/M	—	—	—	C80B
LMS Power Limit Mandate	28	L/R/M	—	—	—	C80C
LMS Power Limit Preference	29	L/R/M	—	—	—	C80D
Max Power Level	2A	L/R/M	—	—	—	C80F
Max Power Level Mandate	2B	L/R/M	—	—	—	C810
Recovery Timeout	2C	L/R/M	Y	N	—	—
Stage Power Limited	2D	L/R/M	Y	N	—	—
Transfer Type Valid	2E	L/R/M	Y	N	—	—
Usage Mode	2F	L/R/M	—	—	—	C81F
Usage Mode Time	30	L/R/M	—	—	—	C820
Vent Fault	31	L/R/M	Y	N	—	—
Vent Required	32	L/R/M	Y	N	—	—
App Service Request Enabled	33	L/R/M	—	—	—	C827
App Service Request	34	L/R/M	—	—	—	C826
Transfer Type Preference	35	L/R/M	—	—	—	C82C
Vent Confirmed	36	L/R/M	Y	N	—	—
Vent Request	37	L/R/M	Y	N	—	—

8.19 Charging System—(See Table 24.)

Primary ID: Charging System
 Command ID: \$72
 Status ID: \$73

TABLE 24—CHARGING SYSTEM MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Charging Voltage	01	R	—	x	—	6035
Charging Current	03	R	—	x	—	6037
Battery Voltage	09	R	—	x	—	600A
Battery Current	0A	R	—	x	—	6038
Battery Temperature	10	R	—	x	—	600B
Charging System Faulted	21	R	Y	N	—	—

8.20 Electrical Energy Management—(See Table 25.)

Primary ID: Electrical Energy Management
 Command ID: \$74
 Status ID: \$75

TABLE 25—ELECTRICAL ENERGY MANAGEMENT MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Delayed Accessory	01	L/R	E	D	--	--
Short Term Storage	02	L/R	E	D	--	--
Long Term Storage	03	L/R	E	D	--	--

8.21 Odometer—(See Table 26.)

Primary ID: Odometer
 Command ID: \$7A
 Status ID: \$7B

TABLE 26—ODOMETER MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Vehicle Odometer	01	R	--	x	--	6004
Trip Reset	03	L/R	R	--R	--	--
Trip Odometer	04	R	--	x	--	6039
Trip Reset Sw. Active	20	R	Y	N	--	--

8.22 Fuel System—(See Table 27.)

Primary ID: Fuel System
 Command ID: \$82
 Status ID: \$83

TABLE 27—FUEL SYSTEM MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Average Fuel Economy	01	R	--	x	--	6020
Instant. Fuel Economy	05	R	--	x	--	603A
Average Fuel Economy Reset	09	L/R	R	--R	--	--
Fuel Used - Percent	0B	R	--	x	--	603B
Fuel Used - Volume	0C	R	--	x	--	603C
Fuel Used Reset	0F	L/R	R	--R	--	--
Fuel Temperature	10	R	--	x	--	1044
Fuel Pressure	11	R	--	x	--	000A
Fuel Level - Percent	12	R	--	x	--	6005
Fuel Level - Volume	13	R	--	x	--	6006
Fuel Capacity	16	R	--	x	--	6007
Fuel Pump Low Speed	18	L/R	E	D	--	--
Fuel Pump High Speed	19	L/R	E	D	--	--
Fuel Economy Reset Sw. Active	29	R	Y	N	--	--
Fuel Used Reset Sw. Active	2F	R	Y	N	--	--
Low Fuel Level	32	R	Y	N	--	--

8.23 Ignition Switch/Starter—(See Table 28.)

Primary ID: Ignition Switch / Starter
 Command ID: \$86
 Status ID: \$87

TABLE 28—IGNITION SWITCH/STARTER MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Ignition Switch Position	04	R	--	x	--	1047
Key-In-Ignition	05	R	Y	N	--	--

8.24 Tell Tales—(See Table 29.)

Primary ID: Tell Tales
 Command ID: \$88
 Status ID: \$89

TABLE 29—TELL TALES MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Seatbelt	01	L/R	On	Off	--	--
Service Engine Soon	02	L/R	On	Off	--	--
Check Engine (MIL)	03	L/R	On	Off	--	--
High Beam Indicator	04	L/R	On	Off	--	--
Left Turn Indicator	05	L/R	On	Off	--	--
Right Turn Indicator	06	L/R	On	Off	--	--
Airbag	07	L/R	On	Off	--	--
Anti-Lock Brake System Failed	08	L/R	On	Off	--	--
Traction Control System Failed	09	L/R	On	Off	--	--
Security	0A	L/R	On	Off	--	--
Low Fuel	0B	L/R	On	Off	--	--
Low Coolant	0C	L/R	On	Off	--	--
Low Oil	0D	L/R	On	Off	--	--
Low Voltage	0E	L/R	On	Off	--	--
Upshift	0F	L/R	On	Off	--	--
Low Washer Fluid	10	L/R	On	Off	--	--
Traction Control Active	11	L/R	On	Off	--	--
Alternator Failure	12	L/R	On	Off	--	--
Low Brake Fluid	13	L/R	On	Off	--	--
Overdrive	14	L/R	On	Off	--	--
Traction Control Disabled	15	L/R	On	Off	--	--
Convertible Latch Warning	21	L/R	On	Off	--	--
Super Lock System Warning	22	L/R	On	Off	--	--
Catalyst Overtemperature	23	L/R	On	Off	--	--
Vehicle Speed Control	24	L/R	On	Off	--	--

8.25 Climate Control (HVAC)—(See Table 30.)

Primary ID: Climate Control (HVAC)
 Command ID: \$B2
 Status ID: \$B3

TABLE 30—CLIMATE CONTROL (HVAC) MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Blower Fan Speed	02	L/R	--	x	--	--
Multi-Zone Mode	06	L/R	E	D	--	--
Low Refrigerant	07	R	Y	N	--	--
Fluid Life Reset	09	L/R	R	--R	--	--
HVAC Set Temperature	0A	L/R	--	x	8.2	9820
		M	Inc	Dec		--
High Side Fluid Temperature	10	R	--	x	--	9808
High Side Fluid Pressure	11	R	--	x	--	9813
Fluid Charge - Percent	12	R	--	x	--	980B
Fluid Charge - Weight	13	R	--	x	--	980C
Fluid Remaining Life	14	R	--	x	--	980D
Fluid Capacity - Weight	16	R	--	x	--	980E
Low Side Fluid Temperature	20	R	--	x	--	9809
Low Side Fluid Pressure	21	R	--	x	--	980A
Fan Increment Speed Sw. Active	22	R	Y	N	--	--
Fan Decrement Speed Sw. Active	23	R	Y	N	--	--
Multi-Zone Mode Sw. Active	26	R	Y	N	--	--
Fluid Life Reset Sw. Active	29	R	Y	N	--	--
Increment Temp Sw. Active	2A	R	Y	N	--	--
Decrement Temp Sw. Active	2B	R	Y	N	--	--

8.26 Window Wiper/Washer—(See Table 31.)

Primary ID: Window Wiper/Washer
 Command ID: \$B8
 Status ID: \$B9

TABLE 31—WINDOW WIPER/WASHER MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Wiper Mode	01	L/R	--	x	8.3	A003
Washer On/Off	02	L/R	On	Off	8.3	--
Wiper Delay	03	L/R	E	D	8.3	A004
Wiper Pulse	04	L/R	E	D	8.3	--
Fluid Temperature	10	R	--	x	8.3	A006
Fluid Pressure	11	R	--	x	8.3	A007
Fluid Level - Percent	12	R	--	x	8.3	A008
Fluid Level - Volume	13	R	--	x	8.3	A009
Fluid Capacity	16	R	--	x	8.3	A00A
Washer On/Off Sw. Active	22	R	Y	N	8.3	--
Wiper Pulse Sw. Active	24	R	Y	N	8.3	--

8.27 Mirrors—(See Table 32.)

Primary ID: Mirrors
 Command ID: \$C2
 Status ID: \$C3

TABLE 32—MIRRORS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Up Motion	01	L/R	E	D	8.4	--
Down Motion	02	L/R	E	D	8.4	--
Right Motion	03	L/R	E	D	8.4	--
Left Motion	04	L/R	E	D	8.4	--
Retract Motion	05	L/R	E	D	8.4	--
Extend Motion	06	L/R	E	D	8.4	--
Horizontal Position	10	L/R	--	x	8.4	A00D
Vertical Position	11	L/R	--	x	8.4	A00E
Heater On/Off	17	L/R	On	Off	8.4	--
Nighttime On/Off	18	R	On	Off	8.4	--
Dimming Level	19	L/R	--	x	8.4	A00C
Up Sw. Active	21	R	Y	N	8.4	--
Down Sw. Active	22	R	Y	N	8.4	--
Right Sw. Active	23	R	Y	N	8.4	--
Left Sw. Active	24	R	Y	N	8.4	--
Retract Sw. Active	25	R	Y	N	8.4	--
Extend Sw. Active	26	R	Y	N	8.4	--
Heater Sw. Active	27	R	Y	N	8.4	--

8.28 Door Locks—(See Table 33.)

Primary ID: Door Locks
 Command ID: \$C4
 Status ID: \$C5

TABLE 33—DOOR LOCKS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Lock	01	L/R	L	U	8.5	--
Unlock Enable	02	L/R	E	D	8.5	--
Lock Cylinder Secure	03	R	Y	N	8.5	--
Key-In-Lock Cylinder	04	R	Y	N	8.5	--
Master Controller Lock	05	L/R	L	N	8.5	--
Lock Cylinder State	06	R	--	x	8.5	A010
Super/Double Lock	07	L/R	L	U	8.5	--
Remote Lock w/ Transmitter ID	08	L/R	L	U	8.5	C001
Remote Lock	09	L/R	L	U	8.5	--
Lock Sw. Active	20	R	Y	N	8.5	--
Unlock Sw. Active	21	R	Y	N	8.5	--
Unlock Enable Sw. Active	22	R	Y	N	8.5	--
Master Controller Lock Sw. Active	25	R	Y	N	8.5	--
Master Controller Unlock Sw. Active	26	R	Y	N	8.5	--

8.29 External Access—(See Table 34.)

Primary ID: External Access
 Command ID: \$C6
 Status ID: \$C7

TABLE 34—EXTERNAL ACCESS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Open	01	L/R	Y	N	8.5	--
Close	02	L/R	Y	N	8.5	--
Remote Open/Close w/ Transmitter ID	11	L/R	O	C	8.5	C001
Remote Open/Close	12	L/R	O	C	8.5	--
Ajar Sw. Active	21	R	Y	N	8.5	--
Door Handle Sw. Active	22	R	Y	N	8.5	--
Door Jamb Sw. Active	23	R	Y	N	8.5	--

8.30 Seat Motion/Control—(See Table 35.)

Primary ID: Seat Motion / Control
 Command ID: \$C8
 Status ID: \$C9

TABLE 35—SEAT MOTION/CONTROL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Horiz. Forward Motion	01	L/R	E	D	8.6	--
Horiz. Rearward Motion	02	L/R	E	D	8.6	--
Vert. Up Motion	03	L/R	E	D	8.6	--
Vert. Down Motion	04	L/R	E	D	8.6	--
Front Vert. Up Motion	05	L/R	E	D	8.6	--
Front Vert. Down Motion	06	L/R	E	D	8.6	--
Rear Vert. Up Motion	07	L/R	E	D	8.6	--
Rear Vert. Down Motion	08	L/R	E	D	8.6	--
Recline Forward Motion	09	L/R	E	D	8.6	--
Recline Rearward Motion	0A	L/R	E	D	8.6	--
Lumbar Up Motion	0B	L/R	E	D	8.6	--
Lumbar Down Motion	0C	L/R	E	D	8.6	--
Lumbar In Motion	0D	L/R	E	D	8.6	--
Lumbar Out Motion	0E	L/R	E	D	8.6	--
Headrest Up Motion	0F	L/R	E	D	8.6	--
Headrest Down Motion	10	L/R	E	D	8.6	--
Heater On/Off	11	L/R	On	Off	8.6	--
Heater Temperature	12	L/R	--	x	8.6	A001
		M	Inc	Dec		
Occupied	13	R	Y	x	8.6	--
Upper Bolster In Motion	14	L/R	E	D	8.6	--
Upper Bolster Out Motion	15	L/R	E	D	8.6	--
Lower Bolster In Motion	16	L/R	E	D	8.6	--
Lower Bolster Out Motion	17	L/R	E	D	8.6	--

8.31 Windows—(See Table 36.)

Primary ID: Windows
 Command ID: \$CA
 Status ID: \$CB

TABLE 36—WINDOWS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Open	01	L/R	E	D	8.7	--
Close	02	L/R	E	D	8.7	--
Operation Enable	03	L/R	E	D	8.7	--
Master Controller Open	04	L/R	E	D	8.7	--
Master Controller Close	05	L/R	E	D	8.7	--
Position	1A	L/R	--	x	8.7	A00F
Open Sw. Active	21	R	Y	N	8.7	--
Close Sw. Active	22	R	Y	N	8.7	--
Lockout Sw. Active	23	R	Y	N	8.7	--
Master Controller Open Sw. Active	24	R	Y	N	8.7	--
Master Controller Close Sw. Active	25	R	Y	N	8.7	--

8.32 Steering Column—(See Table 37.)

Primary ID: Steering Column
 Command ID: \$CC
 Status ID: \$CD

TABLE 37—STEERING COLUMN MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Up Motion	01	L/R	E	D	--	--
Down Motion	02	L/R	E	D	--	--
In Motion	03	L/R	E	D	--	--
Out Motion	04	L/R	E	D	--	--
Vertical Position	1A	L/R	--	x	--	A012
Horizontal Position	1B	L/R	--	x	--	A011
Up Sw. Active	21	R	Y	N	--	--
Down Sw. Active	22	R	Y	N	--	--
In Sw. Active	23	R	Y	N	--	--
Out Sw. Active	24	R	Y	N	--	--

8.33 Seat Switches—(See Table 38.)

Primary ID: Seat Switches
 Command ID: \$D0
 Status ID: \$D1

TABLE 38—SEAT SWITCHES MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Horz. Forward Sw. Active	01	R	Y	N	8.6	--
Horz. Rearward Sw. Active	02	R	Y	N	8.6	--
Vert. Up Sw. Active	03	R	Y	N	8.6	--
Vert. Down Sw. Active	04	R	Y	N	8.6	--
Front Vert. Up Sw. Active	05	R	Y	N	8.6	--
Front Vert. Down Sw. Active	06	R	Y	N	8.6	--
Rear Vert. Up Sw. Active	07	R	Y	N	8.6	--
Rear Vert. Down Sw. Active	08	R	Y	N	8.6	--
Recline Forward Sw. Active	09	R	Y	N	8.6	--
Recline Rearward Sw. Active	0A	R	Y	N	8.6	--
Lumbar Up Sw. Active	0B	R	Y	N	8.6	--
Lumbar Down Sw. Active	0C	R	Y	N	8.6	--
Lumbar In Sw. Active	0D	R	Y	N	8.6	--
Lumbar Out Sw. Active	0E	R	Y	N	8.6	--
Headrest Up Sw. Active	0F	R	Y	N	8.6	--
Headrest Down Sw. Active	10	R	Y	N	8.6	--
Heater On/Off Sw. Active	11	R	Y	N	8.6	--
Increment Temp. Sw. Active	12	R	Y	N	8.6	--
Decrement Temp. Sw. Active	13	R	Y	N	8.6	--
Upper Bolster In Sw. Active	14	R	Y	N	8.6	--
Upper Bolster Out Sw. Active	15	R	Y	N	8.6	--
Lower Bolster In Sw. Active	16	R	Y	N	8.6	--
Lower Bolster Out Sw. Active	17	R	Y	N	8.6	--

8.34 Restraints—(See Table 39.)

Primary ID: Restraints
 Command ID: \$D2
 Status ID: \$D3

TABLE 39—RESTRAINTS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Passive Restraint Engaged	01	L/R	Y	N	8.6	--
Passive Restraint Retracted	02	L/R	Y	N	8.6	--
Passive Restraint Attached	03	R	Y	N	8.6	--
Seatbelt Attached	04	R	Y	N	8.6	--
Shoulder Adjustment Up Motion	05	L/R	E	D	8.6	--
Shoulder Adjustment Down Motion	06	L/R	E	D	8.6	--
Air Bag Deployed	07	R	Y	N	8.6	--
Shoulder Position	1A	L/R	--	x	8.6	5801
Shoulder Adjustment Up Sw. Active	25	R	Y	N	8.6	--
Shoulder Adjustment Down Sw. Active	26	R	Y	N	8.6	--

8.35 Exterior Lamps Outage—(See Table 40.)

Primary ID: Exterior Lamps Outage
 Command ID: \$D8
 Status ID: \$D9

TABLE 40—EXTERIOR LAMPS OUTAGE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Headlamp OK	01	R	Y	N	8.8	—
Tail Lamp OK	02	R	Y	N	8.8	—
Brake Lamp OK	03	R	Y	N	8.8	—
Park Lamp OK	04	R	Y	N	8.8	—
Turn Lamp OK	05	R	Y	N	8.8	—
High Beam Lamp OK	06	R	Y	N	8.8	—
Hazard Lamp OK	07	R	Y	N	8.8	—
Reverse Lamp OK	08	R	Y	N	8.8	—
Fog Lamp OK	09	R	Y	N	8.8	—
Daytime Running Lamp OK	0A	R	Y	N	8.8	—
Spot Lamp OK	0B	R	Y	N	8.8	—
Cargo Lamp OK	0C	R	Y	N	8.8	—
Cornering Lamp OK	0D	R	Y	N	8.8	—
Driving Lamp OK	0E	R	Y	N	8.8	—
Coach Lamp OK	0F	R	Y	N	8.8	—

8.36 Exterior Lamps—(See Table 41.)

Primary ID: Exterior Lamps
 Command ID: \$DA
 Status ID: \$DB

TABLE 41—EXTERIOR LAMPS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Headlamp On/Off	01	L/R	On	Off	8.8	--
Tail Lamp On/Off	02	L/R	On	Off	8.8	--
Brake Lamp On/Off	03	L/R	On	Off	8.8	--
Park Lamp On/Off	04	L/R	On	Off	8.8	--
Turn Lamp On/Off	05	L/R	On	Off	8.8	--
High Beam Lamp On/Off	06	L/R	On	Off	8.8	--
Hazard Lamp On/Off	07	L/R	On	Off	8.8	--
Reverse Lamp On/Off	08	L/R	On	Off	8.8	--
Fog Lamp On/Off	09	L/R	On	Off	8.8	--
Daytime Running Lamp On/Off	0A	L/R	On	Off	8.8	--
Spot Lamp On/Off	0B	L/R	On	Off	8.8	--
Cargo Lamp On/Off	0C	L/R	On	Off	8.8	--
Cornering Lamp On/Off	0D	L/R	On	Off	8.8	--
Driving Lamp On/Off	0E	L/R	On	Off	8.8	--
Coach Lamp On/Off	0F	L/R	On	Off	8.8	--
Autolamp Delay	10	L/R	E	D	8.8	A014
Flash-to-Pass	11	L/R	E	D	8.8	--
Remote Headlamp On/Off w/Transmitter ID	12	L/R	On	Off	8.8	C001
Remote Headlamp On/Off	13	L/R	On	Off	8.8	--
Headlamp Sw. Active	21	R	Y	N	8.8	--
Right Turn Sw. Active	22	R	Y	N	8.8	--
Park Lamp Sw. Active	24	R	Y	N	8.8	--
Left Turn Sw. Active	25	R	Y	N	8.8	--
High Beam Sw. Active	26	R	Y	N	8.8	--
Hazard Sw. Active	27	R	Y	N	8.8	--
Fog Lamp Sw. Active	28	R	Y	N	8.8	--
Driving Lamp Sw. Active	29	R	Y	N	8.8	--

8.37 Interior Lamps Outage—(See Table 42.)

Primary ID: Interior Lamps Outage
 Command ID: \$DC
 Status ID: \$DD

TABLE 42—INTERIOR LAMPS OUTAGE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Courtesy Lamp OK	01	R	Y	N	8.9	--
Dome Lamp OK	02	R	Y	N	8.9	--
Puddle Lamp OK	03	R	Y	N	8.9	--
Vanity Mirror Lamp OK	04	R	Y	N	8.9	--
Opera Lamp OK	05	R	Y	N	8.9	--
Reading Lamp OK	06	R	Y	N	8.9	--
Hood Lamp OK	07	R	Y	N	8.9	--
Trunk Lamp OK	08	R	Y	N	8.9	--
Glove Box Lamp OK	09	R	Y	N	8.9	--

8.38 Interior Lamps—(See Table 43.)

Primary ID: Interior Lamps
 Command ID: \$DE
 Status ID: \$DF

TABLE 43—INTERIOR LAMPS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Courtesy Lamp On/Off	01	L/R	On	Off	8.9	--
Dome Lamp On/Off	02	L/R	On	Off	8.9	--
Puddle Lamp On/Off	03	L/R	On	Off	8.9	--
Vanity Mirror Lamp On/Off	04	L/R	On	Off	8.9	--
Opera Lamp On/Off	05	L/R	On	Off	8.9	--
Reading Lamp On/Off	06	L/R	On	Off	8.9	--
Hood Lamp On/Off	07	L/R	On	Off	8.9	--
Trunk Lamp On/Off	08	L/R	On	Off	8.9	--
Glove Box Lamp On/Off	09	L/R	On	Off	8.9	--
Illuminated Entry	10	L/R	E	D	--	--
Display Brightness & External Lamps On/Off	11	L/R	On	Off	--	602B
Courtesy Lamp Sw. Active	21	R	Y	N	8.9	--
Dome Lamp Sw. Active	22	R	Y	N	8.9	--
Puddle Lamp Sw. Active	23	R	Y	N	8.9	--
Vanity Mirror Sw. Active	24	R	Y	N	8.9	--
Opera Lamp Sw. Active	25	R	Y	N	8.9	--
Reading Lamp Sw. Active	26	R	Y	N	8.9	--
Hood Lamp Sw. Active	27	R	Y	N	8.9	--
Trunk Lamp Sw. Active	28	R	Y	N	8.9	--
Glove Box Lamp Sw. Active	29	R	Y	N	8.9	--

8.39 Tires—(See Table 44.)

Primary ID: Tires
 Command ID: \$E4
 Status ID: \$E5

TABLE 44—TIRES MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Temperature	10	R	--	x	8.1	2849
Pressure	11	R	--	x	8.1	2851
Wear Level	14	R	--	x	8.1	2861
Spare Present	15	R	Y	N	--	--

8.40 Defrost—(See Table 45.)

Primary ID: Defrost
 Command ID: \$E6
 Status ID: \$E7

TABLE 45—DEFROST MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Defrost On/Off	01	L/R	On	Off	8.3	--
Defrost Pending	02	L/R	Y	N	8.3	--
Defrost Switch	21	R	Y	N	8.3	--

8.41 Displays—(See Table 46.)

Primary ID: Displays
 Command ID: \$EA
 Status ID: \$EB

TABLE 46—DISPLAYS MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Metric Display	08	L/R	Y	N	--	--

8.42 Exterior Environment—(See Table 47.)

Primary ID: Exterior Environment
 Command ID: \$F2
 Status ID: \$F3

TABLE 47—EXTERIOR ENVIRONMENT MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Outside Air Temperature	10	R	--	x	--	602E
Barometric Pressure	11	R	--	x	--	1025
Sun Load	13	R	--	x	8.3	9817
Photo Cell Dark	15	R	Y	N	8.3	--

8.43 Interior Environment—(See Table 48.)

Primary ID: Interior Environment
 Command ID: \$F4
 Status ID: \$F5

TABLE 48—INTERIOR ENVIRONMENT MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Air Filtration On/Off	02	L/R	On	Off	8.2	--
Interior Air Temperature	10	L/R	--	x	8.2	9830
Humidity Level	12	R	--	x	--	9815
Air Filter Remaining Life	14	R	--	x	--	9816

8.44 Time/Date—(See Table 49.)

Primary ID: Time / Date
 Command ID: \$F8
 Status ID: \$F9

TABLE 49—TIME/DATE MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Alarm Time	03	R	--	x	--	6047
		L	--	x	--	6047
Time of Day	05	R	--	x	--	6016
		L	--	x	--	6016
Hour of Day	06	R	--	x	--	6017
		L	--	x	--	6017
		M	Inc	Dec	--	--
Minute of Hour	07	R	--	x	--	6018
		L	--	x	--	6018
		M	Inc	Dec	--	--
Second of Minute	08	R	--	x	--	6019
		L	--	x	--	6019
		M	Inc	Dec	--	--
Date	10	R	--	x	--	600D
		L	--	x	--	600D
Day of Week	11	R	--	x	--	6013
		L	--	x	--	6013
		M	Inc	Dec	--	--
Day of Month	12	R	--	x	--	6015
		L	--	x	--	6015
		M	Inc	Dec	--	--
Month of Year	13	R	--	x	--	6011
		L	--	x	--	6011
		M	Inc	Dec	--	--
Year	14	R	--	x	--	600E
		L	--	x	--	600E
		M	Inc	Dec	--	--
Set Time Sw.Active	25	R	Y	N	--	--

8.45 Vehicle Identification—(See Table 50.)

Primary ID: Vehicle Identification
 Command ID: \$FA
 Status ID: \$FB

TABLE 50—VEHICLE IDENTIFICATION MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
VIN Packet 1 (First digit)	01	R	--	x	--	E021
VIN Packet 2 (digits 2 – 5)	02	R	--	x	--	E022
VIN Packet 3 (digits 6 – 9)	03	R	--	x	--	E023
VIN Packet 4 (digits 10 – 13)	04	R	--	x	--	E024
VIN Packet 5 (digits 14 – 17)	05	R	--	x	--	E025
VIN Packet 6 (reserved for future)	06	R	--	x	--	E026
VIN Packet 7 (reserved for future)	07	R	--	x	--	E027

8.46 Network Control—(See Table 51.)

Primary ID: Network Control
 Command ID: \$FE
 Status ID: \$FF

TABLE 51—NETWORK CONTROL MESSAGES

Secondary ID Name	Sec ID	Msg Op	Q1	Q0	Ext Addr	PRN
Bus Wake-up	02	L / R	Y	N	--	--
Node Alive	03	R	Y	N	--	--
Node Sleep	04	L / R	Y	N	--	--

9. Extended Address Assignments—The following tables contain the extended address assignments categorized by function. These table are referenced by paragraph number in the message tables on the previous pages.

9.1 Brakes, Tires, and Wheels—The extended address assignments for brakes, tires, and wheels are in Table 52.

TABLE 52—EXTENDED ADDRESS ASSIGNMENTS FOR BRAKES, TIRES, AND WHEELS

Address	Description
00	ALL
10	All Front
11	Left Front
17	Right Front
30	All Rear
31	Left Rear
37	Right Rear
3C	Spare Tire

9.2 HVAC Zones—The extended address assignments for HVAC zones are in Table 53.

TABLE 53—EXTENDED ADDRESS ASSIGNMENTS FOR HVAC ZONES

Address	Description
00	ALL
20	All Front
22	Driver Side Front
26	Passenger Side Front
28	All Rear
2A	Driver Side Rear
2E	Passenger Side Rear

9.3 Window Wiper/Washer, Defrost, and Photocell—The extended address assignments for window wiper/washer, defrost, and photocell are in Table 54.

TABLE 54—EXTENDED ADDRESS ASSIGNMENTS FOR WINDOW WIPER/WASHER, DEFROST, AND PHOTOCCELL

Address	Description
00	All
1C	Front
34	Rear

9.4 Mirrors—The extended address assignments for mirrors are in Table 55.

TABLE 55—EXTENDED ADDRESS ASSIGNMENTS FOR MIRRORS

Address	Description
00	ALL
1A	Driver Side
1C	Rear View
1E	Passenger Side

9.5 Doors and Door Locks—The extended address assignments for doors and door locks are in Table 56.

TABLE 56—EXTENDED ADDRESS ASSIGNMENTS FOR DOORS AND DOOR LOCKS

Address	Description
00	ALL
14	Hood
1E	Passenger Side Glove Box
20	All Front
22	Driver Side Front
24	Convertible Top
26	Passenger Side Front
28	All Rear
2A	Driver Side Rear
2E	Passenger Side Rear
31	Left Side Fuel Door
34	Trunk
37	Right Side Fuel Door
3C	Only or Rear Center Fuel Door

9.6 Seats and Restraints—The extended address assignments for seats and restraints are in Table 57.

TABLE 57—EXTENDED ADDRESS ASSIGNMENTS FOR SEATS AND RESTRAINTS

Address	Description
00	ALL
20	All Front
22	Driver Side Front
24	Front Center
26	Passenger Side Front
28	All Rear
2A	Driver Side Rear
2C	Rear Center
2E	Passenger Side Rear
30	All Rear - Rear (Van)
32	Driver Side Rear - Rear (Van)
36	Passenger Side Rear - Rear (Van)

9.7 Windows—The extended address assignments for windows are in Table 58.

TABLE 58—EXTENDED ADDRESS ASSIGNMENTS FOR WINDOWS

Address	Description
00	ALL
20	All Front
22	Driver Side Front
24	Front Sun Roof
26	Passenger Side Front
28	All Rear
2A	Driver Side Rear
2C	Rear Sun Roof
2E	Passenger Side Rear
34	Rear Windshield

9.8 External Lamps—The extended address assignments for external lamps are in Table 59.

TABLE 59—EXTENDED ADDRESS ASSIGNMENTS FOR EXTERNAL LAMPS

Address	Description
00	ALL
01	Left Side (Turn Signal)
07	Right Side (Turn Signal)
08	All Front
09	Left Front
0F	Right Front
38	All Rear
39	Left Rear
3C	CHMSL
3F	Right Rear

9.9 Internal Lamps—The extended address assignments for internal lamps are in Table 60.

TABLE 60—EXTENDED ADDRESS ASSIGNMENTS FOR INTERNAL LAMPS

Address	Description
00	ALL
20	All Front
22	Driver Side Front
26	Passenger Side Front
28	All Rear
2A	Driver Side Rear
2C	Dome Lamp
2E	Passenger Side Rear

10. Notes

10.1 Marginal Indicia—The change bar (|) located in the left margin is for the convenience of the user in locating areas where technical revisions have been made to the previous issue of the report. An (R) symbol to the left of the document title indicates a complete revision of the report.

PREPARED BY THE SAE VEHICLE NETWORK FOR MULTIPLEX AND
DATA COMMUNICATION STANDARDS COMMITTEE