ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

ESCUELA DE INGENIERÍA AUTOMOTRIZ

"SOFTWARE Y HARDWARE PARA MONITOREAR PARÁMETROS DE MOVILIDAD Y CONSUMO DE COMBUSTIBLE EN VEHÍCULOS OBD2"

MONTERO QUIROZ WILSON EDMUNDO

ABRIL GUERRERO JOSÉ EDUARDO

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO AUTOMOTRIZ

RIOBAMBA – ECUADOR

2012

ESPOCH

Facultad de Mecánica

CERTIFICADO DE APROBACIÓN DE TESIS

Noviembre, 19 de 2012

Yo recomiendo que la Tesis preparada por:

WILSON EDMUNDO MONTERO QUIROZ

Titulada:

"SOFTWARE Y HARDWARE PARA MONITOREAR PARÁMETROS DE MOVILIDAD Y CONSUMO DE COMBUSTIBLE EN VEHÍCULOS OBD2"

Sea aceptada como parcial complementación de los requerimientos para el título de:

INGENIERO AUTOMOTRIZ

Ing. Geovanny Novillo A.
DECANO DE LA FAC. DE MECÁNICA

Nosotros coincidimos con esta recomendación:

Doctor Mario Audelo
DIRECTOR DE TESIS

Ing. Romel Insuasti
ASESOR DE TESIS

ESPOCH

Facultad de Mecánica

CERTIFICADO DE APROBACIÓN DE TESIS

Noviembre, 19 de 2012

Yo recomiendo que la Tesis preparada por:

JOSÉ EDUARDO ABRIL GUERRERO

Titulada:

"SOFTWARE Y HARDWARE PARA MONITOREAR PARÁMETROS DE MOVILIDAD Y CONSUMO DE COMBUSTIBLE EN VEHÍCULOS OBD2"

Sea aceptada como parcial complementación de los requerimientos para el título de:

INGENIERO AUTOMOTRIZ

Ing. Geovanny Novillo A.
DECANO DE LA FAC. DE MECÁNICA

Nosotros coincidimos con esta recomendación:

Doctor Mario Audelo
DIRECTOR DE TESIS

Ing. Romel Insuasti
ASESOR DE TESIS

ESPOCH

Facultad de Mecánica

CERTIFICADO DE EXAMINACIÓN DE TESIS

NOMBRE DEL ESTUDIANTE: WILSON EDMUNDO MONTERO QUIROZ

TÍTULO DE LA TESIS: "**SOFTWARE Y HARDWARE PARA MONITOREAR PARÁMETROS DE MOVILIDAD Y CONSUMO DE COMBUSTIBLE EN VEHÍCULOS OBD2"**

Fecha de Examinación: 19 de Noviembre de 2012

RESULTADO DE LA EXAMINACIÓN:

COMITÉ DE EXAMINACIÓN	APRUEBA	NO APRUEBA	FIRMA
Ing. Ángel Tierra (PRESIDENTE TRIB. DEFENSA)			
Doctor Mario Audelo (DIRECTOR DE TESIS)			
Ing. Romel Insuasti (ASESOR)			

* Más que un voto de no aprobación es razón suficiente para la falla total.

RECOMENDACIONES:

El Presidente del Tribunal certifica que las condiciones de la defensa se han cumplido.

f) Presidente del Tribunal

ESPOCH

Facultad de Mecánica

CERTIFICADO DE EXAMINACIÓN DE TESIS

NOMBRE DEL ESTUDIANTE: JOSÉ EDUARDO ABRIL GUERRERO

TÍTULO DE LA TESIS: "**SOFTWARE Y HARDWARE PARA MONITOREAR PARÁMETROS DE MOVILIDAD Y CONSUMO DE COMBUSTIBLE EN VEHÍCULOS OBD2"**

Fecha de Examinación: 19 de Noviembre de 2012

RESULTADO DE LA EXAMINACIÓN:

COMITÉ DE EXAMINACIÓN	APRUEBA	NO APRUEBA	FIRMA
Ing. Ángel Tierra (PRESIDENTE TRIB. DEFENSA)			
Doctor Mario Audelo (DIRECTOR DE TESIS)			
Ing. Romel Insuasti (ASESOR)			

* Más que un voto de no aprobación es razón suficiente para la falla total.

RECOMENDACIONES:

El Presidente del Tribunal certifica que las condiciones de la defensa se han cumplido.

f) Presidente del Tribunal

DERECHOS DE AUTORÍA

El trabajo de grado que presentamos, es original y basado en el proceso de investigación y/o adaptación tecnológica establecido en la Facultad de Mecánica de la Escuela Superior Politécnica de Chimborazo. En tal virtud, los fundamentos teóricos - científicos y los resultados son de exclusiva responsabilidad de los autores. El patrimonio intelectual le pertenece a la Escuela Superior Politécnica de Chimborazo.

Wilson Montero Quiroz

José Eduardo Abril

DEDICATORIA

El presente trabajo lo dedico a mis queridos padres Cecilia y Rilo quienes con su apoyo no solo económico sino también moral me dieron el aliento necesario para poder culminar con éxito esta etapa de mi vida.

A todas aquellas personas que estuvieron incondicionalmente apoyándome en momentos difíciles quienes con su apoyo y amistad supieron darme la fuerza para continuar adelante y no rendirme bajo ninguna circunstancia.

José Abril.

Dedico el presente trabajo, al esfuerzo incansable de mi madre Débora Quiroz García; que realiza día a día para sacarme adelante y convertirme en una mejor persona. Y a todos mis familiares y amigos que supieron brindarme su apoyo en aquellos momentos difíciles.

Wilson Montero.

AGRADECIMIENTO

Un sincero agradecimiento a la Escuela Superior Politécnica de Chimborazo en especial a la escuela de Ingeniería Automotriz por habernos brindado la oportunidad de formarnos en sus instalaciones y así poder obtener una profesión para poder ser personas útiles para la sociedad.

A nuestros profesores quienes con sus conocimientos contribuyeron en nuestra formación académica.

Además agradecemos a nuestro asesor y director de tesis, en especial al Dr. Mario Audelo ya que con su apoyo incondicional se hizo posible la culminación de este proyecto.

José Abril Wilson Montero

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ABREVIACIONES

ASCII	American Standard CodeforInformationInterchange
CAN	Controller Area Network
CMD	Command
CMOS	Complementary Metal Oxide Semi-Conductor
CRC	CyclicRedundancyCheck
CVT	ConventionColumn (Dato incluido para detección de errores)
DTC	Diagnostic Trouble Code
ECM	Engine Control Module
ECU	Electronic Control Unit
EEPROM	Electrical Erasable-Programmable Read-Only Memory
ERR	Error Detection
GDI	Graphics Device Interface
GUID	Globally Unique Identifier
HEX	Hexadecimal
HID	Human Interface Device
I2C	Inter-Integrated Circuit
ISO	International Standards Organization
LCD	Liquid-Crystal Display
LED	light-emitting diode
LSB	Least Significant Bit
MAF	Mass Air Flow
MSB	Most Significant Bit
MSSP	Master Synchronous Serial Port
MUA	Movimiento Uniformemente Acelerado
O/I	Out/In
OBD	On-Board Diagnostic
OBD2	On-Board Diagnostic 2 Generation
P	Power

PCI	Protocol Control Information
PCM	Powertrain Control Module
PID	Parameter ID
PID	Product ID
PROM	Programmable Read-Only Memory
PWM	Pulse-Width Modulation (Ford motors)
RAM	Random-Access Memory
RSP	In-frame response
SAE	Society of Automotive Engineers
SCI	Serial Communications Interface
SID	Service ID
SQL	Structured Query Language
ST	Schmitt Trigger
TCM	Transmission Control Module
TID	Test ID
TTL	Transistor Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
USART	Universal Synchronous-Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VID	Vendor ID
VPW	Variable Pulse Width (General motors)
VSS	Vehicle Speed Sensor

SIMBOLOGÍA

a	Aceleración
A	Amperios
F	Fuerza
m	Masa
v	Velocidad
v_o	Velocidad inicial
v_f	Velocidad final
t	Tiempo
x_f	Posición en función del tiempo
x_o	Posición inicial
ξ	Número perteciente a un intervalo $[a, b]$
δ	Densidad
$\dot{m}$	Flujo másico
s	Segundos
g	Gramos
V	Voltaje
Hz	Hercios
Vol	Volumen
Ω	Ohmios
Km	Kilómetros
h	Horas

LISTA DE ANEXOS

RESUMEN

El diseño y construcción de un dispositivo capaz de almacenar información acerca del consumo y recorrido de un automóvil, está basado en la necesidad de monitorear el comportamiento del vehículo en tiempo real.

La selección de los componentes necesarios se ha realizado con énfasis en la economía y accesibilidad, es por ello que se ha elegido una en que la mayoría de automóviles hoy en día utilizan, como es computadoras, las cuales están guiadas por normas, las mismas que son la SAE J1979 y la ISO 15031-5. Otra característica es que éstas pueden entregar datos de los sensores en tiempo real. Para este proyecto únicamente hemos usado los datos de los sensores MAF y VSS, pero dejamos la pista para posteriores proyectos donde se necesite utilizar datos del automóvil.

Como el acceso las computadoras del vehículo se realiza por un conector normalizado, la conexión entre el dispositivo y el vehículo se lleva a cabo únicamente por este medio, eliminando posibles conexiones adicionales al cableado del automotor, como el cambio de conector de una marca a otra.

Una vez conectado el dispositivo, cuando el motor se ponga en marcha, los cálculos de kilometraje recorrido, consumo de combustible, fecha y hora tanto de encendido como de apagado se realizarán automáticamente, guardándose en la memoria interna del dispositivo al final de cada ciclo de funcionamiento. Quedando tan solo necesaria la intervención humana, al momento de descargar los datos hacia el computador, para su posterior análisis.

CAPÍTULO I

1. GENERALIDADES

1.1 Introducción

En los últimos años los fabricantes de autos han optado por estandarizar los sistemas de diagnóstico a bordo (OBD), recopilándose toda la información necesaria como: control de emisiones, codificación de las averías de los componentes del sistema, información del vehículo, entre los más importantes.

Por lo tanto esta información queda accesible a quien lo considere necesario; el uso de la misma, cuando y el tiempo necesario para satisfacer las necesidades de quien manipule esta información.

Es ahí cuando surge la aplicación del presente proyecto, que centra en solicitar la información disponible en la computadora de todos los vehículos que cumplen con las normas SAE J1979, ISO 15031-5, las mismas que cumplen los requerimientos internacionales máximos de polución. Para mediante la información del sensor MAF y VSS, determinar el consumo de combustible y el recorrido realizado por un determinado automóvil, y a través del IC DS1307 adquirir el lapso de tiempo en el que se generó los valores antes mencionados.

Hay que destacar que este proyecto no reduce el rendimiento del vehículo, ya que, este tiene varios módulos que funcionan simultáneamente y durante toda la vida útil del mismo. Y al conectar el presente módulo se convertiría en uno más, como se podrá observar en el desarrollo de este trabajo.

1.2 Antecedentes

En los últimos meses se ha venido hablando mucho sobre el mal uso de los automóviles públicos, tanto de uso en horarios no permitidos, como de corrupción en el combustible. Hasta ahora la única forma de llevar un control del vehículo era simplemente basarse en catálogos del fabricante para realizar cálculos aproximados, que no solo requieren molestias tanto para el conductor, sino como también para quien requiera realizar un análisis del tema.

1.3 Justificaciones

1.3.1 *Justificación técnica.* El sistema de control propuesto, proveerá de las funciones necesarias, para que con el simple hecho de abrir el software se tenga el control total, sin contar que es sumamente confiable debido a que el encargado de realizar el control sería la única persona que tendría acceso a los datos extraídos del automóvil evitándose así su posible manipulación.

1.3.2 *Justificación ecológica.* Al mejorar las condiciones de manejo, la reducción el consumo de combustible es evidente y por ende también sucederá lo mismo con las emisiones de gases contaminantes hacia el medio ambiente.

Sin contar que se podrá notar a tiempo si hay un excesivo consumo a causa de factores mecánicos, pudiéndose corregir inmediatamente.

1.4 Objetivos

1.4.1 *General.* Elaborar el software y hardware necesario para el análisis de movilidad y consumo de combustible.

1.4.2 *Específicos*

Investigar la escritura y lectura de datos, a través de las distintas interfaces propias de OBD2

Diseñar el software y hardware, necesario para tratar información OBD2 mediante un computador

Crear una interfaz amable con el usuario, que permita visualizar y tratar los datos de forma clara y sencilla

Desarrollar conocimientos más profundos en sistemas OBD2, estableciendo precedentes para futuros proyectos relacionados

CAPÍTULO II

2. MARCO TEÓRICO

2.1 On-BoardDiagnostic (OBD)[1]

En un mundo donde apunta a la normalización, este sistema de regulaciones requerido en automóviles de pasajeros, carga liviana y mediana, soporta un sistema de comunicación con el computador a bordo; establecido en su fabricación, de manera que pueda diagnosticar información; mediante la utilización de un equipo. Pudiendo esta información ser utilizada de acuerdo a una necesidad propia.

Ahora, el objetivo de este breve resumen es especificar los servicios de diagnóstico y funcionabilidad direccionada a la petición/respuesta de los mensajes requerida para ser soportado por un motor de vehículo como también del equipo externo destinado para el propósito de llevar a cabo la transacción de datos de los diferentes sensores pertenecientes al vehículo. Estos mensajes usados por un equipo de diagnóstico externo deben tener un formato normalizado para poder recuperar la información de un vehículo OBD y por lo tanto los requerimientos se encuentran en **SAE J1979.**

Pero el OBD como tal, es la norma SAE J1979 que fue originalmente desarrollada para encontrar los requerimientos OBD U.S. para los vehículos de 1996 en adelante. Hoy en día se la hizo internacional creando la ISO 15031-5 basada en SAE J1979, la cual combina los requerimientos U.S. con los OBD europeos que apareció para modelos del 2000 o superiores, llamándose entonces OBD 2 por la adición de PIDs. Pero el formato de los mensajes se ha conservado el que apareció en 1996.

2.1.1 *Requerimientos especificados en SAE J1979, ISO 15031-5*

a. Formato de los mensajes para petición y respuesta.
b. Tiempo entre el mensaje de petición del equipo externo y el mensaje de respuesta por parte del vehículo, y entre esos mensajes y posterior a esos mensajes de petición.
c. Comportamiento del vehículo y el equipo externo si el dato no está disponible.
d. Un grupo de servicios de diagnóstico, con contenido correspondiente entre mensajes de petición y respuesta, para satisfacer las regulaciones OBD.

2.1.2 *Partes básicas de un paquete de datos o mensaje de petición*

1. 3 bytes de Cabecera (1 byte de prioridad/Tipo, 2 bytes de dirección(excepto CAN que puede ser 11 o 29 bits))
2. 1 Byte del tipo/identificador de servicio(Desde 0x01 al 0x09)
3. 1 byte de Tipo/PID soportado dentro del servicio
4. 1 byte de verificación CRC (Es diferente para cada interface)

2.1.3 *Formato del mensaje de diagnóstico en forma general*

Tabla 1. Formato de mensajes para ISO 9141-2, ISO 14230-4, SAE J1850

Bytes de cabecera			**Bytes de datos**								
Prioridad (Hex)	**Dirección del objetivo (Hex)**	**Dirección de la fuente (Hex)**	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**ERR**	**Resp**
Petición de diagnóstico a 10.4 Kbit/s: SAE J1850 e ISO 9141-2											
68	6A	F1	& bytes máximo							Si	No
Respuesta de diagnóstico a 10.4 Kbit/s: SAE J1850 e ISO 9141-2											
48	6B	Dirección	7 bytes máximo							Si	No
Petición de diagnóstico a 10.4 Kbit/s (ISO 14230-4)											
11LL LLLL[1]b	33	F1	7 bytes máximo							Si	No
Respuesta de diagnóstico a 10.4 Kbit/s (ISO 14230-4)											
10LL LLLLb	F1	Dirección	& bytes máximo							Si	No
Petición de diagnóstico a 41.6 Kbit/s (SAE J1850)											
61	6A	F1	7 bytes máximo							Si	Si
Respuesta de diagnóstico a 41,6 Kbit/s (SAE J1850)											
41	6B	Dirección	7 bytes máximo							Si	Si

Fuente: SAE J1979 REV APR2002

[1]LL LLLL=Longitud de bytes de datos; RSP= Respuesta en marco de datos; ERR= Detección de error

Tabla 2. Formato de mensajes para ISO 15765-4 CAN

Bytes de cabecera	**Campo para el marco de datos CAN**							
Identificador CAN (11 0 29 bits)	#1	#2	#3	#4	#5	#6	#7	#8

Fuente: SAE J1979 REV APR2002

Ejemplo: 0x68, 0x6A, 0xF0, 0x01, 0x0D, CRC

Muestra como solicitar la velocidad actual del vehículo con interface J1850 VPW o J1850 PWM

Tabla 3. Ejemplo de solicitud de datos

Byte	**Valor**	**Descripción**
1	0x68	Prioridad en este caso normal
2	0x6A	Dirección de la ECU
3	0xF0	Dirección de la ECU que solicita(En este caso el equipo)[2]
4	0x01	Servicio (En este caso Data values)
5	0x0D	PID correspondiente a Velocidad del vehículo. Ver anexo B
6	CRC	Byte que resulta al codificar todos los bytes anteriores mediante código, que provee la norma correspondiente. Este asegura que los datos lleguen correctamente. (cyclicredundancycheck)

Fuente: SAE J1979 REV APR2002

2.1.4 *Posición del grupo de bits.* El siguiente Figura indica la relación que existe entre el orden de llegada y la correspondencia de los bytes, Por lo que el primero en llegar será el más significativo y el último el menos significativo.

[2] La dirección para un equipo externo va desde 0xF0 hasta 0xFF

Tabla 4. Posición de un bit dentro de un byte

MSB[3]							**LSB**[4]
7	6	5	4	3	2	1	0

Fuente: SAE J1979 REV APR2002

2.1.5 *Tipos de servicios*

Tabla 5. Posibles SID soportados por un vehículo

Servicio	**Valor (Hexadecimal)**
Petición de datos actuales de diagnóstico pertenecientes al motor y transmisión.	0x01
Petición de datos congelados pertenecientes al motor y transmisión.	0x02
Petición de códigos de falla	0x03
Borrar/resetear la información de diagnóstico.	0x04
Petición de resultados monitoreo de los sensores de oxigeno	0x05
Petición de resultados de monitores a bordo, para sistemas específicos de monitoreo.	0x06
Petición de códigos de fallas detectados y que son relacionados con el sistema de emisiones	0x07
Petición de control de sistema a bordo, prueba de un componente	0x08
Petición de información del vehículo	0x09

Fuente: SAE J1979 REV APR2002

[3]MSB (most significant bit, bit mas significativo);
[4]LSB (least significant bit, bit menossignificativo)

2.1.6 *Formato de datos*

Tabla 6. Formato de los datos a ser mostrados

Dato	**Servicio**	**Formato**
Device ID : Dirección de la fuente de la respuesta	Todos	ISO 9141-2: Hexadecimal (00 a FF) ISO 14230-4: Hexadecimal (00 a FF) SAE J1850: Hexadecimal (00 a FF) ISO 15765-4: Hexadecimal (11 bit o 29 bit Identificador CAN)
Parameter ID (PID)	0x01 & 0x02	Hexadecimal (00 a FF) Ver anexo B
FrameNumber	0x02	Decimal (0 a 255)
Data values	0x01 & 0x02	Ver Anexo B
DTCs Códigos de fallas	0x03 & 0x07	“P”, “B”, “C” o “U”, más 4 caracteres hexadecimal y definición de datos
Test ID	0x05, 0x06 & 0x08	Hexadecimal (00 a FF)
Test value and test limits	0x05	Unidades para Test Ids menores que $80 Decimal (0 a 255) para Test IDs superiores a $80
Test value and test limits	0x06	Decimal (0 a 65535)
Component ID	0x06	Hexadecimal (00 a FF)
Optional data bytes	0x08	4 bytes
Vehicleinformationtype	0x09	Hexadecimal (00 a 7F)
Vehicleinformation data	0x09	ASCII para los tipos $02 y $04; Hexadecimal para el $06 y decimal para el $08

Fuente: SAE J1979 REV APR2002

2.2 Interfaces OBD2

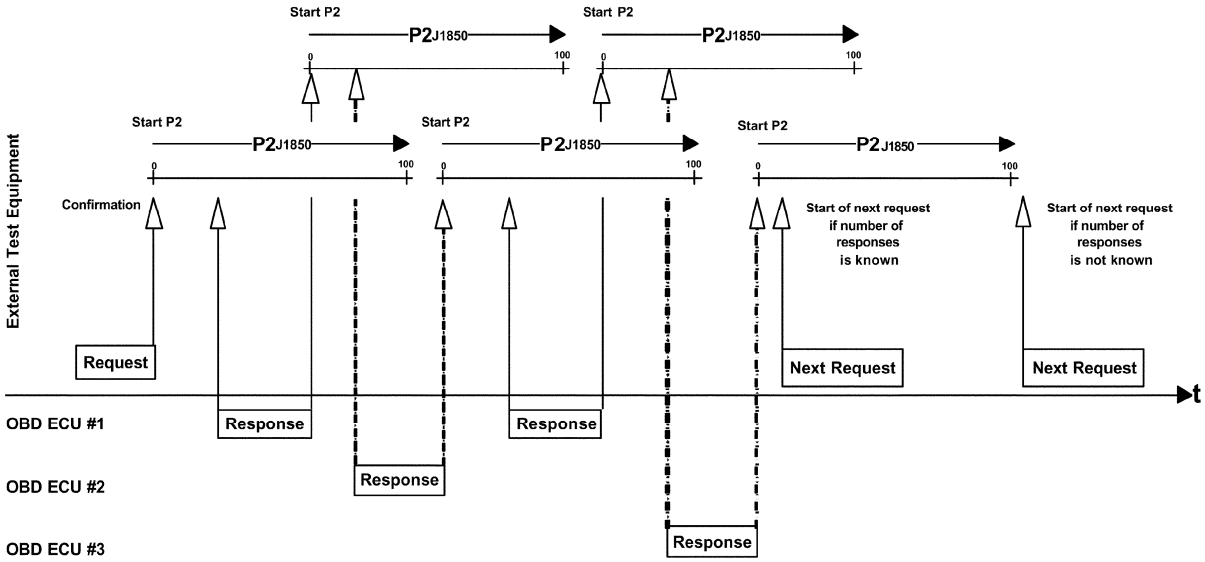
2.2.1 *SAE J1850 VPW y SAE J1850 PWM.* Este sistema a bordo responderá dentro de $P2_{J1850}$ ya sea a una petición o a un mensaje previo respuesta. Es posible que se de múltiples respuestas de un mensaje de petición. Esto tardara tanto tiempo como sea necesario para que todas las ECUs puedan acceder a la línea de datos y transmitan su(s) mensaje(s) de respuesta. Pero no debe pasar lo expuesto a continuación:

Tabla 7. Tiempos de respuesta de una ECU SAE J1850

Parámetro	**Valor mínimo**	**Valor máximo**	**Descripción**
$P2_{J1850}$	0	100	Tiempo entre un mensaje de petición de un equipo externo y una transmisión satisfactoria del mensaje(s) de respuesta por parte de la(s) ECU(s). Cada OBD ECU intentara enviar su mensaje de respuesta (o al menos el primero de múltiples mensajes de respuesta) dentro de $P2_{J1850}$ después de un mensaje de petición correctamente recibido. Los posteriores mensajes también serán transmitidos dentro de $P2_{J1850}$ previo al mensaje de múltiples respuestas.

Fuente: SAE J1979 REV APR2002

Figura1. Perspectiva de la aplicación de parámetros de tiempo en SAE J1850



Fuente: SAE J1979 REV APR2002

2.2.2 *ISO 9141-2.* El estrato de requerimiento de tiempo es especificado de forma extendida, en el documento de su mismo nombre, pero a continuación se muestra un breve resumen.

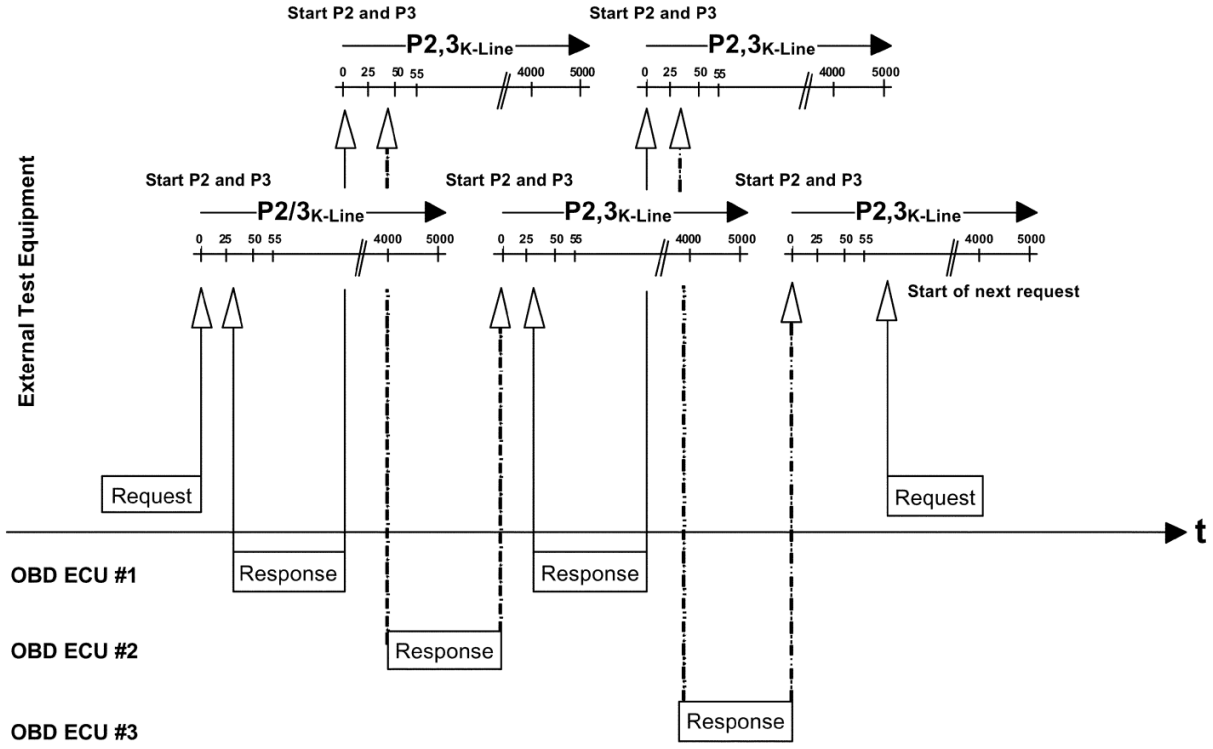
Tabla 8. Tiempo de respuesta de una ECU ISO 9141-2 e ISO 14230-4

Parámetro	**Valor mínimo (ms)**	**Valor máximo (ms)**	**Descripción**
$P2_{K\text{-}Line}$ Bytes clave: 0x08 0x08 Una o más ECUs	25	50	Tiempo entre mensaje de petición desde equipo externo y transmisión satisfactoria de mensaje(s) de respuesta de la ECU(s). Cada ECU OBD empezara enviando su mensaje de respuesta dentro de $P2_{K\text{-}Line}$ después del mensaje de petición correctamente recibido. La respuesta posterior también será transmitida dentro de $P2_{K\text{-}Line}$previo al mensaje respuesta de múltiples

			respuestas.
$P2_{K\text{-}Line}$ Bytes clave: 0x94 0x94 Solo una ECU	0	50	Tiempo entre mensaje de petición desde equipo externo y transmisión satisfactoria de mensaje(s) de respuesta de la ECU(s). La ECU OBD empezará enviando su mensaje de respuesta dentro de $P2_{K\text{-}Line}$ después del mensaje de petición correctamente recibido. La respuesta posterior también será transmitida dentro de $P2_{K\text{-}Line}$ previo al mensaje respuesta de múltiples respuestas.
$P3_{K\text{-}Line}$	55	5000	Tiempo entre el fin de un mensaje de respuesta transmitido correctamente y el inicio de un mensaje de petición por parte de un nuevo equipo externo. El equipo externo puede enviar un nuevo mensaje de petición si todos los mensajes de respuesta relacionados previamente enviados han sido recibidos y si el tiempo minimo$P3_{K\text{-}Line}$ ha expirado.

Fuente: SAE J1979 REV APR2002

Figura2. Perspectiva de la aplicación de parámetros de tiempo en ISO 9141-2



Fuente: SAE J1979 REV APR2002

2.2.3 *Mensaje negativo en SAE J1850 e ISO 9141-2.* Se entenderá como mensaje negativo, si la(s) ECU(s) no responde dentro del tiempo mencionado anteriormente. Por lo que el equipo externo deberá tomar en cuenta para evitar cualquier error.

Habrá este mensaje de rechazo si el mensaje de petición no es funcional. Estas interfaces evitan mensajes de respuesta desde todas las ECUs que no soportan un servicio o un dato específico.

2.2.4 *ISO 14230-4*

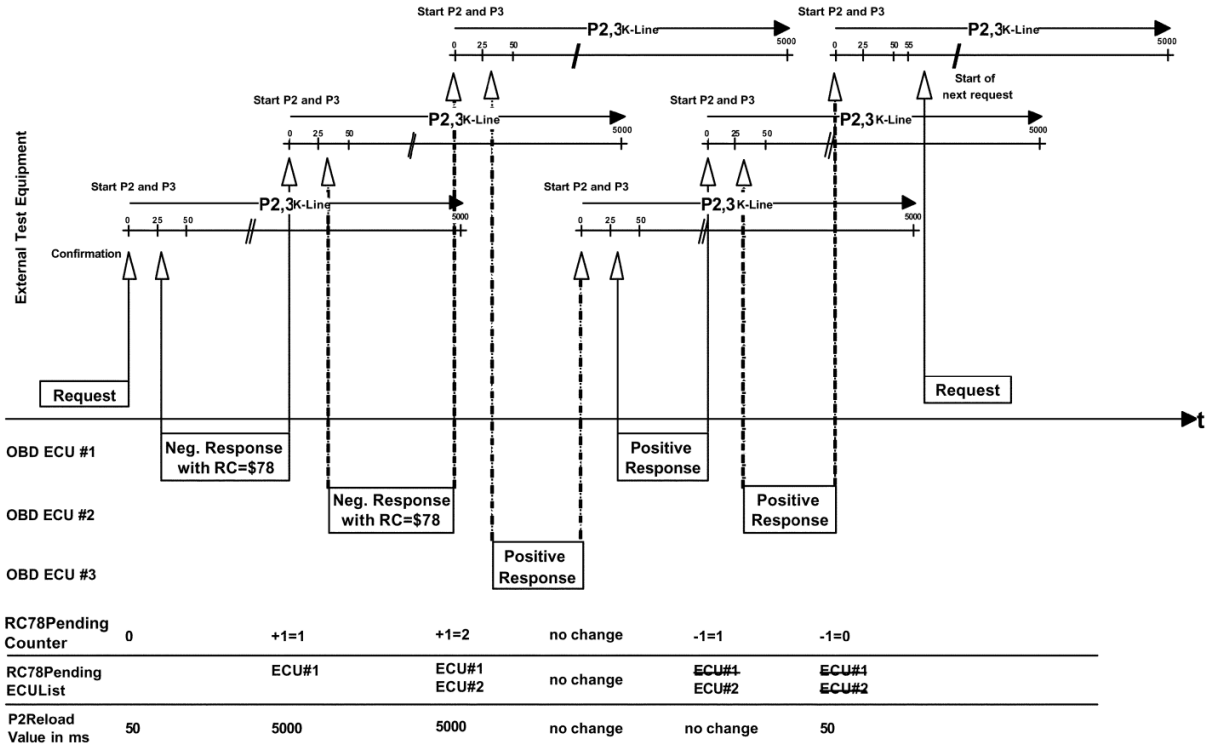
Tabla 9. Tiempo de respuesta de una ECU ISO 14230-4

Parámetro	**Valor mínimo (ms)**	**Valor máximo (ms)**	**Descripción**
$P2_{K\text{-}Line}$	25	50	Tiempo entre mensaje de petición desde

			equipo externo y transmisión satisfactoria de mensaje(s) de respuesta de la ECU(s). Cada ECU OBD empezará enviando su mensaje de respuesta dentro de $P2_{K\text{-}Line}$ después del mensaje de petición correctamente recibido. La respuesta posterior también será transmitida dentro de $P2_{K\text{-}Line}$ previo al mensaje respuesta de múltiples respuestas.
$P3_{K\text{-}Line}$	55	5000	Tiempo entre el fin de un mensaje de respuesta transmitido correctamente y el inicio de un mensaje de petición por parte de un nuevo equipo externo. El equipo externo puede enviar un nuevo mensaje de petición si todos los mensajes de respuesta relacionados previamente enviados han sido recibidos y si el tiempo minimo$P3_{K\text{-}Line}$ ha expirado.

Fuente: SAE J1979 REV APR2002

Figura3. Perspectiva de la aplicación de parámetros de tiempo en ISO 14230-4



Fuente: SAE J1979 REV APR2002

2.2.5 *ISO 15765-4—Diagnostics on Controller Area Network (CAN).* Todas las ECUs del sistema OBD responderán dentro de $P2_{CAN}$. La siguiente tabla especifica los valores de tiempo, aplicables en ISO 15765-4.

Tabla 10. Tiempo de respuesta de una ECU ISO 1576-4

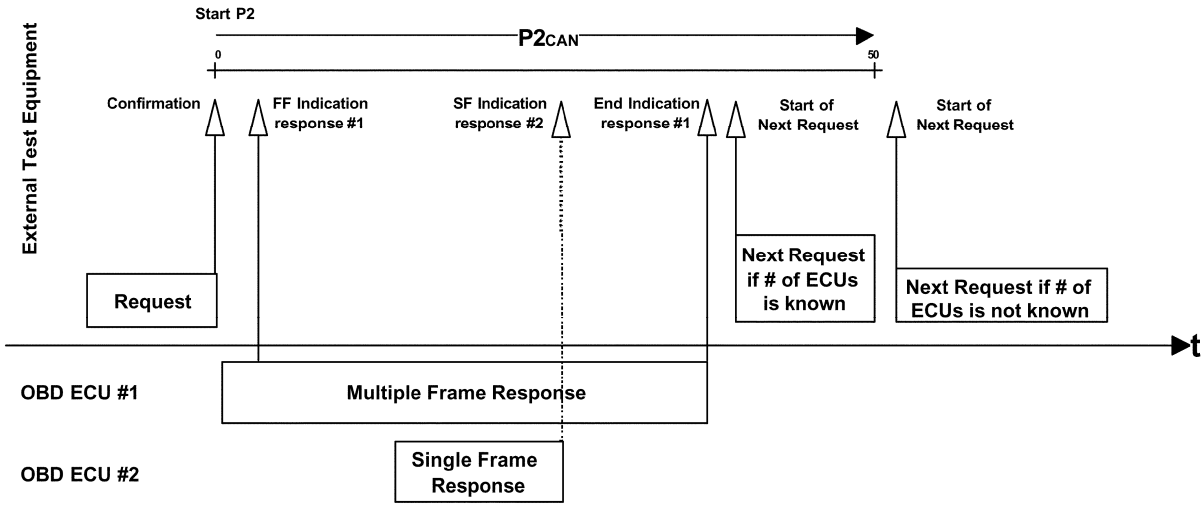
Parámetro	**Valor mínimo (ms)**	**Valor máximo (ms)**	**Descripción**
$P2_{CAN}$	0	50	Tiempo entre mensaje de petición desde equipo externo y la recepción de todo el mensaje no segmentado o del primer paquete de un mensaje segmentado. Cada ECU OBD empezará a enviar su mensaje dentro de

			$P2_{CAN}$después de que el mensaje ha sido recibido correctamente.
$P2_{CAN}$	0	5000	Tiempo entre la recepción satisfactoria de un mensaje negativo con código 0x78 y el próximo mensaje de respuesta (positivo o negativo).

Fuente: SAE J1979 REV APR2002

El equipo externo puede enviar un nuevo mensaje de petición inmediatamente después de que este determine que todas las ECU(s) relacionadas con el mensaje hayan respondido, y si el equipo no sabe si ya ha recibido todos los mensajes de respuesta, este deberá esperar el tiempo máximo después del último mensaje de petición o del último mensaje de respuesta.

Figura4. Perspectiva de la aplicación de parámetros de tiempo en ISO 15765-4 CAN



Fuente: SAE J1979 REV APR2002

En *ISO 15765-4* se puede presentar las siguientes situaciones:

- Los IDs OBD soportados por la ECM[5],no serán soportados por TCM[6], PCM[7]u otros módulos (los módulos que no soportan no harán nada).

[5] Engine Control Module
[6] Transmission Control Module

- Los IDs OBD soportados por la TCM, PCM u otros módulos, no serán soportados por la ECM(se recibe el mensaje de respuesta del Módulo xx, la ECM no envía mensaje negativo)
- Puede existir varios módulos incluyendo la ECM que soporten el mismo ID (Siempre sucede con el SID 0x01, PID 0x00)
- Ninguna de las ECUs soporta el ID, en este caso la ECM responderá con un mensaje negativo (la ECM es la única que responde con mensajes negativos).

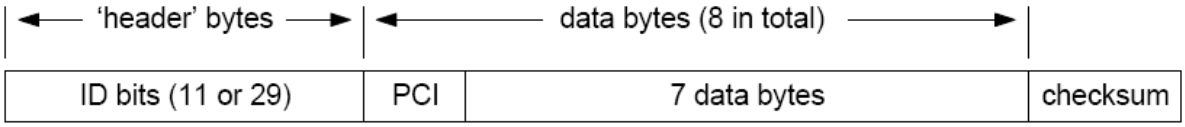
Algo muy importante que hay que tener en cuenta es que en *ISO 15765-4* el equipo externo es un módulo más, y que todos los demás módulos del vehículo siempre reciben el mensaje enviado por el equipo externo, así como los mensajes enviados por otro modulo.

2.2.5.1 *Tipos de ISO 15765-4 (CAN)*

Aquí podemos encontrar dos variantes:

- CAN 11 Bits ID en cabecera (header)
- CAN 29 Bits ID en cabecera (header)

Figura5. Estructura de un mensaje de datos en CAN



Fuente: SAE J1979 REV APR2002

[7] Powertrain Control Module

2.2.6 *Formato de un mensaje para respuesta negativa, ISO 14230-4, ISO 15765-4.* Habrá un mensaje de respuesta cada mensaje de petición bien positivo o negativo. Con el fin de evitar la comunicación innecesaria para la(s) ECU(s) que no soportan una petición funcional PID, TID, o INFTYPE es permitido no enviar un mensaje negativo si otra ECU enviará una respuesta positiva. El formato y posibles códigos de respuestas negativas son especificados a continuación.

Tabla 11. Formato de respuesta negativa para ISO 14230-4, ISO 15765-4

Byte	**Nombre del parámetro**	**CVT**	**Valor Hex**	**Mnemónico**
#1	Identificador de respuesta negativa	M	7F	SIDNR
#2	Identificador de petición de servicio	M	XX	SIDRQ
#3	Código de respuesta	M	XX	RC_

Fuente: SAE J1979 REV APR2002

2.2.6.1 *Descripción del código de servicio negativo*

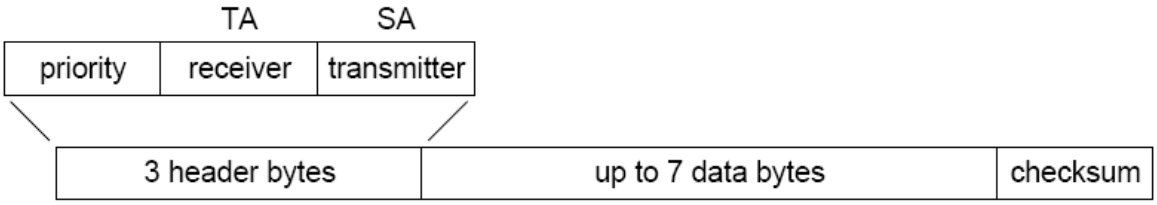
Tabla 12. Posibles códigos de error

Suportado por ISO	**Valor Hex**	**Descripción**	**Mnemónico**
14230-4	10	Rechazo general (no especifica razón)	GR
14230-4	11	Servicio no soportado	SNS
14230-4	12	Sub-función no soportada-formato invalido	SFNSIF
14230-4 15765-4	21	Ocupado-Repita la petición más tarde	BRR
14230-4 15765-4	22	Condición no correcta o error en la secuencia de petición	CNCORSE
14230-4 15765-4	78	Petición recibida correctamente, pero respuesta pendiente	RCR-RP

Fuente: SAE J1979 REV APR2002

2.2.7 *Diferencias entre mensajes SAE 1850, ISO 9141-2, ISO 14230-4 e ISO 15765-4.* Los siguientes mensajes SAE 1850, ISO 9141-2, ISO 14230-4 tienen el mismo formato de cabecera (header).

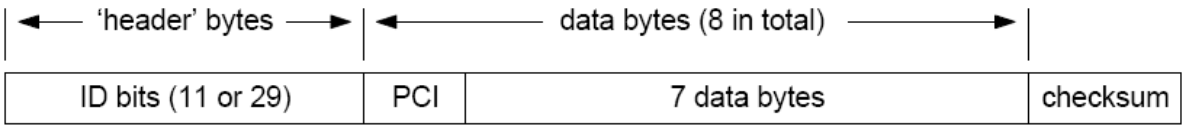
Figura6. Estructura de la cabecera en SAE J1850, ISO 9141-2 e ISO 142304



Fuente: SAE J1979 REV APR2002

Mientras que en ISO 15765-4 CAN

Figura7. Estructura de la cabecera en ISO 15765-4 CAN



Fuente: SAE J1979 REV APR2002

Tabla 13. Descripción de una cabecera CAN 29 bits

Cabecera (header) CAN 29 bits			
5 Bits VV	**XX**	**YY**	**ZZ**
Prioridad (generalmente 0x12)	Describe tipo de mensaje (toma valores de 0xDB o 0xDA)	Dirección del objetivo(receptor)	Dirección del transmisor

Fuente: SAE J1979 REV APR2002

Tabla 14. Descripción de una cabecera CAN 11 bits

Cabecera (header) CAN 11 bits

3 Bits YY	ZZ
Dirección del objetivo(receptor)	Dirección del transmisor

Fuente: SAE J1979 REV APR2002

Como podemos observar todos tienen en común los datos de SID y PID que se transmite desde el equipo externo, a continuación podemos observar la petición de un mismo PID en los dos diferentes formatos de mensajes encontrados en OBD2.

2.2.7.1 *Petición*

Tabla 15. Ejemplo de petición de valores del MAF para SAE J1850, ISO 9141-2, ISO 14230-4

Petición del valor del sensor MAF SAE J1850, ISO 9141-2, ISO 14230-4					
Bytes de cabecera			**Bytes de datos**		
Prioridad (Hex)	**Dirección del objetivo (Hex)**	**Dirección de la fuente (Hex)**	**SID[8]**	**PID[9]**	**Checksum**
Petición de diagnóstico a 10.4 Kbit/s: SAE J1850 e ISO 9141-2					
68	6B	F1	O1	10	XX
Petición de diagnóstico a 10.4 Kbit/s (ISO 14230-4)					
E8	48	F1	01	10	XX
Petición de diagnóstico a 41.6 Kbit/s (SAE J1850)					
61	6A	F1	01	10	XX

Fuente: SAE J1979 REV APR2002

Tabla 16. Ejemplo de petición de valores del MAF para ISO 15764-4 CAN 11 bits y 29 bits

Petición del valor del sensor MAF ISO 15765-4 CAN 11 Bits y 29 Bits			
Cabecera (Header)	**PCI[10]**	**SID**	**PID**

[8] Service ID (veranexo A)
[9] Parameter ID
[10] Protocol Control Information

Petición de diagnóstico ISO 15765-4 CAN 11 Bits			
7E0	XX	01	10
Petición de diagnóstico ISO 15765-4 CAN 29 Bits			
6 0EAFF F9	XX	01	10

Fuente: SAE J1979 REV APR2002

2.2.7.2 *Respuesta*

Tabla 17. Valores del sensor MAF para SAE J1850, ISO 9141-2, ISO 14230-4

Respuesta del valor del sensor MAF SAE J1850, ISO 9141-2, ISO 14230-4							
Bytes de cabecera			**Bytes de datos**				
Prioridad (Hex)	**Dirección del objetivo (Hex)**	**Dirección de la fuente (Hex)**	**SID + 40**	**PID**	**Byte Alto**	**Byte bajo**	**Checksum**
Respuesta de diagnóstico a 10.4 Kbit/s: SAE J1850 e ISO 9141-2							
48	F1	6B	41	10	0E	23	XX
Respuesta de diagnóstico a 10.4 Kbit/s (ISO 14230-4)							
A8	F1	F1	41	10	0E	23	XX
Respuesta de diagnóstico a 41.6 Kbit/s (SAE J1850)							
41	F1	6A	41	10	0E	23	XX

Fuente: SAE J1979 REV APR2002

Tabla 18. Valores del sensor MAF para ISO 15765-4 CAN 11 bits, 29 bits

ISO 15765-4 CAN 11 Bits y 29 Bits					
Cabecera (Header)	**PCI**[11]	**SID + 40**	**PID**	**Byte Alto**	**Byte bajo**
Respuesta de diagnóstico ISO 15765-4 CAN 11 Bits					
7E8	XX	41	10	0E	23

[11]Protocol Control Information

Respuesta de diagnóstico ISO 15765-4 CAN 29 Bits					
7 0EBFE F9	XX	41	10	0E	23

Fuente: SAE J1979 REV APR2002

2.3 USB-Universal Serial Bus [2]

2.3.1 *Introducción.* Fue creado en los años 90 por una asociación de empresas, con la idea de mejorar las técnicas plug-and-play, es decir, permitir a los dispositivos conectarse y desconectarse sin necesidad de reinicio, configurándose automáticamente al ser conectados; además pudiendo ser dotado de energía eléctrica para los dispositivos conectados.
Este bus tiene una estructura de árbol y se pueden conectar dispositivos en cadena, pudiendo conectar hasta 127 dispositivos permitiendo la transferencia síncrona y asíncrona.

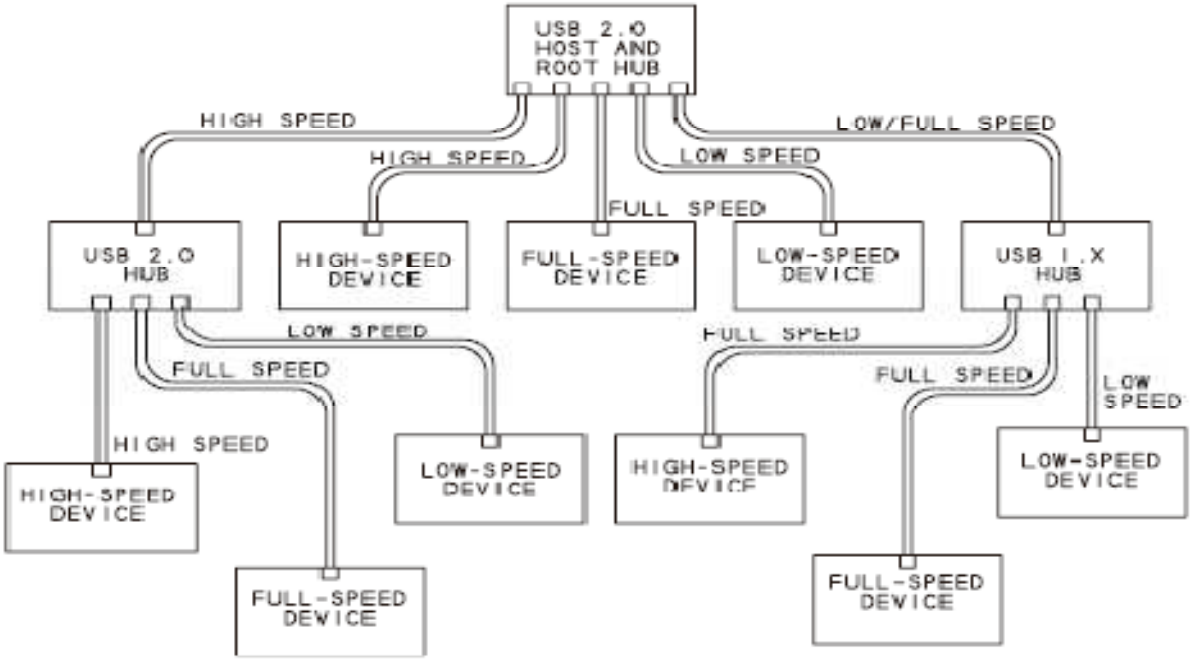
Se pueden clasificar según su velocidad de transferencia de datos (desde Kilobytes hasta Megabytes):

- Baja Velocidad (1.0) usados para los Dispositivos de Interfaz Human a (HID) como ratones, teclados, etc.
- Velocidad Completa (1.1)
- Alta Velocidad (2.0) para conexiones a internet, memorias, dispositivos, etc.
- SúperSpeed (3.0) con velocidades de hasta 5 Gbps, para discos duros externos, pen drives, etc.

Físicamente, los datos del USBse transmiten por un par trenzado (D- y D+) además de la masa y alimentación (+5V). Los conectores están sujetos al estándar (tipo A, tipo B).

2.3.2 *USB 2.0 [3]*. Este tipo de USB ha ganado mucha popularidad en las PCs debido a su gran velocidad y compatibilidad con los dispositivos de baja, completa y alta velocidad. Este tipo lo podemos encontrar a partir de Windows XP SP2 como reemplazo a USB 1.1. Es decir en un dispositivo USB 2.0 pueden funcionar en cualquiera de los tres modos (1.0, 1.1, 2.0).

Figura8. Distribución de los diferentes HUBS USB



Fuente: USB complete, 4ta edición

2.3.2.1 *Componentes del bus USB 2.0*

- La comunicación USB requiere un HOST que soporte uno o más dispositivos USB.
- Un lugar en la computadora donde se conectará el dispositivo, llamado HUB.
- Conectores que pueden ser de Tipo A o Tipo B.
- Cable.
- Dispositivo con el hardware y software, necesarios para la comunicación entre él y la computadora.

2.3.3 *Desarrollo de un dispositivo USB [4].* En el diseño de un dispositivo USB para PCs involucra tanto conseguir un dispositivo con el hardware como proveer del software necesario para manejar adecuadamente el hardware.

2.3.3.1 *Componentes. Un dispositivo USB necesita lo siguiente:*

- Un circuito integrado con interface USB y un CPU o cualquier hardware inteligente para manejar el controlador.
- Código de programa, que acarre los datos del dispositivo al computador o viceversa.

El host que comunica el PC con el dispositivo necesita lo siguiente:

- Hardware que controle el host y software (incluido con el sistema operativo).
- Driver del dispositivo (Archivos *Autorun.inf, WdfCoInstaller01009.dll y WUDFUpdate_0109.dll*), el cual es un software que habilita la comunicación entre las aplicaciones y el dispositivo.
- Software o aplicación que permite al usuario comunicarse con el dispositivo. Para ratones, teclados, memorias, discos duros y demás, no es necesario generar este software.

2.3.3.2 *Herramientas para el desarrollo*

- Un ensamblador o compilador para crear el firmware[12] del dispositivo (Compilador usado en este proyecto: CCS PIC C Compiller PCWHD 4.110).
- Un programador de hardware que guarda nuestro firmware (binario resultado del compilador) en el dispositivo.
- Un compilador para escribir y depurar el software del host, el cual puede incluir una combinación de driver del dispositivo, driver filtro[13], código de la aplicación (Compilador usado en este proyecto: Visual C Sharp 2008.)

[12] Código que corre dentro del dispositivo (binario)
[13] Comunica el host con la aplicación

2.3.3.3 *Tipos de transferencias soportados por USB 2.0*

Par enviar o recibir datos, el host inicia una transferencia USB, cada transferencia usa un formato definido para enviar los datos, direccionar la información, detección de errores, estado y control de la información.

*Transferencia de control*Se utiliza para la enumeración[14] del dispositivo, siempre empieza con una transacción que contiene una petición. Dependiendo de la petición, para completar la transferencia, el host y el dispositivo puede intercambiar datos y estado de la información. Cada transferencia de control tiene al menos una transacción que envía información en cada dirección.

Un ejemplo de una transferencia de control fallida es cuando al conectar un dispositivo, nos aparece un mensaje en la barra de información de Windows, que nos dice, ***Dispositivo no reconocido.***

*Transferencia masiva*Este tipo de transferencia es usada para transferir datos cuando el tiempo no es crítico. Una transferencia masiva puede enviar y recibir grandes cantidades de datos sin presencia de errores de transmisión, por ejemplo impresoras, escáneres, leer y escribir en un dispositivo. Es así que este tipo de transferencia de datos es la más rápida y está disponible solo en dispositivos USB 2.0.

*Transferencia asíncrona*Es usada para transferencia de datos en tiempo real donde se necesita transferir a velocidad constante o dentro de un límite, ocasionalmente se pueden dar errores. Un ejemplo de este tipo es la transmisión de voz y música en tiempo real, ya que si fuera archivo de música en una memoria se podría transferir como masiva en el menor tiempo posible.

2.3.4 *Chips con hardware USB*

Tabla 19. Integrados con controlador USB

Estos integrados son compatibles con la popular arquitectura de una micro controlador

Familia	**Fabricante**	**Chip**	**Velocidad Del Bus**
ARM	ATMEL	AT91SAM7S	Full
		AT91SAM9R64	High
	NXP	LPC292x, LPC214x	Full

[14] Transferencia de datos que se realiza cuando se conecta el dispositivo

	Semiconductors		
ATMEL AVR	ATMEL	AT90USBx	Low/full
Microchip PIC18	Microchip Technology	PIC18F2455/2550/4455/4550 PIC18(L)Fxk50	Low/full
STMicroelectroniks ST7	STMicroelectronics	ST7260 ST7265x ST7268x	Low Full Full/High

Fuente: USB complete, 4ta edición

2.3.5 *PID (Product ID) y VID (Vendor ID) [5].* Estos datos son muy importantes y deben ser grabados tanto en el driver (archivo *.inf) que debe incluirse (Detalle Componentes) como en el descriptor del dispositivo. Ya que deberán coincidir en el momento de la enumeración para que la instalación del dispositivo sea satisfactoria.

A continuación podemos observar un fragmento del archivo Autorun.inf donde se debe colocar los valores del PID, VID. Los valores de xxxx&yyyy deben colocarse en Hexadecimal excepto para Windows ME donde se debe poner en decimal.

USB\Vid_xxxx&Pid_yyyy

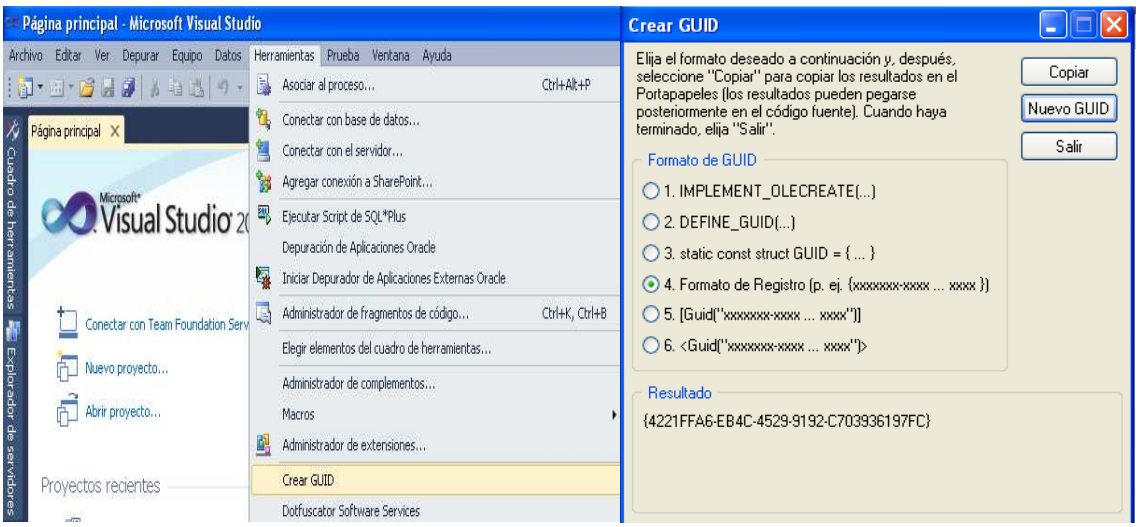
Hay que tener mucho cuidado con VID, ya que este nos provee el vendedor del chip, y si quisiéramos poner un número propio debemos pagar la patente en usb.org ya que si fabricaríamos un dispositivo en serie con un VID no legal, la organización USB podría levantar una situación legal contra la empresa que fabrica el aparato.

Ventajosamente Microchip nos provee un VID con la compra de uno de esos chips y es el valor de **0x 04D8**, Mientras que el valor del PID debemos elegir uno que no interfiera con algún producto existente en el mercado.

2.3.6 *GUID.* Una vez elegido nuestro PID, el próximo dato a proporcionar al archivo autorun.inf es la interface de dispositivo GUID, este valor debe ser único para cada dispositivo. Esto permitirá que se pueda trabajar con varios dispositivos ya que no podría haber ambigüedad, ya que si conectaríamos dos dispositivos con igual identificación el host no sabría con cual dispositivo comunicarse, provocando un error fatal.

Una GUID se lo puede obtener mediante cualquier Visual Studio como podemos ver en las siguientes imágenes.

Figura9. Proceso de obtención de una GUID



Fuente: Autores

2.3.7 *Archivo autorun.inf [6].* Es un archivo que contiene la información para la configuración del dispositivo, y el archivo le dice al sistema que driver o drivers contiene la información necesaria para la instalación del dispositivo en el registro de Windows los archivos están en %directorio de Windows%\inf[15]. El sistema operativo copia un archivo *.inf del dispositivo en este directorio, para que la próxima vez que se conecte el dispositivo no sea necesario proveer el archivo otra vez.

El siguiente es el código ejemplo que contiene el archivo ***autorun.inf:***

```
; ================ Version section ================

[Version]
```

[15] Este directorio por lo general está oculto

Signature = "$Windows NT$"

Class = %ClassName%

ClassGuid = {12E030E7-8912-400B-AC2F-6EF88641571F}

Provider = %ProviderName%

DriverVer = 10/09/2011,1.0.0.0

……

……

[PicWinUSB]

%DeviceDesc% =WinUSB_Install, USB\VID_04D8&PID_0012

[PicWinUSB.NTamd64]

%DeviceDesc% =WinUSB_Install, USB\VID_04D8&PID_0012

[ClassInstall32]

Addreg=PicWinUSBClassReg86

……

……

AddReg = Dev_AddReg

[Dev_AddReg]

HKR,,DeviceInterfaceGUIDs,0x00010000,"{85810410-1C54-49F1-89C0-8451C493D367}"

[WinUSB_Install.CoInstallers]

AddReg = CoInstallers_AddReg

2.3.8 *Descriptor [7]*. Es la parte que complementa la configuración y se tiene que proveer cuando se programa el dispositivo, y esta debe corresponder con el archive *.inf. Para mayor información se puede ver en los archivos adjuntos al Software PIC, llamado **usb_desc_scope.h**, ya que es muy extenso como para agregar su contenido en esta sección, se puede ver a continuación únicamente el lugar donde se cambia los valores de PID y VID.

```
charconst USB_DEVICE_DESC[] ={
    USB_DESC_DEVICE_LEN,        //the length of this report
0x01,           //constant DEVICE (0x01)
0x10,0x01,      //usb version in bcd 2.0
    0x00,           //class code (if 0, interface defines class.  FF is vendor defined)
    0x00,           //subclass code
    0x00,           //protocol code
USB_MAX_EP0_PACKET_LENGTH,  //max packet size for endpoint 0.(SLOW SPEED SPECIFIES 8)
0xD8,0x04,      //vendor id (0x04D8 is Microchip)
    0x12,0x00,      //product id 0x0012
    0x00,0x01,      //device release number
    0x01,           //index of string description of manufacturer.
    0x02,           //index of string descriptor of the product
    0x00,           //index of string descriptor of serial number
    USB_NUM_CONFIGURATIONS  //number of possible configurations
};
```

2.4 Funciones PIC C Compiller para PICs Microchip [8]

2.4.1 *Funciones para el manejo de interface USB*

Tabla 20. Funciones disponibles para el manejo de USB

Funciones relevantes	
usb_init()	Inicializa el hardware USB. Entonces esperará en un bucle infinito, hasta que el USB periférico sea conectado al bus (No quiere decir que necesariamente haya sido enumerada). Habilita y usa la interrupción USB.
usb_init_cs()	Igual que usb_init(), pero no espera a que el dispositivo USB sea conectado al bus. Esto es usado si el dispositivo no es autoalimentado y puede operar sin una conexión USB.
usb_task()	Si se usa un PIN sensible a la conexión, y el usb_init_cs() para inicializar, entonces se deberá llamar periódicamente esta función para mantener vigilada la conexión con el bus. Cuando la PIC sea conectada al bus esta función prepara el USB periférico. Mientras que cuando la PIC sea desconectada esta reseteara la memoria USB y el estado periférico. Habilitará y usara la interrupción USB.
Nota: En su aplicación deberá ser definido USB_CON_SENSE_PIN al pin establecido para este fin.	
usb_detach()	Remueve la PIC del bus. Sera llamada automáticamente por usb_task() si la conexión es perdida, pero también puede ser llamada manualmente por el usuario.
usb_attach()	Enlaza el PIC al bus. Será llamada automáticamente

	por usb_task() si la conexión ha sido realizada, pero puede ser llamada manualmente por el usuario.
usb_attached()	Si se usa un Pin para sentir la conexión ((USB_CON_SENSE_PIN), retorna TRUE si ese pin está en alto (PIC conectada al bus). Caso contrario siempre retornara FALSE.
usb_enumerated()	Retorna TRUE si el dispositivo ha sido enumerado por el PC. Eso quiere decir que está en operación normal y tú puedes enviar/recibir paquetes de datos.
usb_put_packet (endpoint, data, len, tgl)	Pone el paquete de datos en el endpoint del buffer especificado. Retorna TRUE si el proceso fue satisfactorio, FALSE si el buffer está todavía lleno con el último paquete.
usb_puts (endpoint, data, len, timeout)	Envía data al endpoint especificado. La funciónusb_puts() difiere de usb_put_packet() en que este enviara múltiples paquetes[16] de datos, si data no cabe dentro de un paquete (si data tiene más de 64 bytes).
usb_kbhit(endpoint)	Retorna TRUE si el endpoint especificado tiene datos en su buffer de recepción.
usb_get_packet(endpoint, ptr, max)	Lee hasta la cantidad max de bytes desde el buffer del endpoint especificado y lo guarda en el putero ptr.
usb_gets(endpoint, ptr,max, timeout)	Lee un mensaje desde el endpoint especificado. La diferencia entre usb_get_packet() y usb_gets() es que usb_gets() esperara hasta que el mensaje haya sido recibido completamente, el mensaje puede contener más de un paquete[13]. Retorna el número de bytes recibidos.

[16] Un paquete no puede tener más de 64 bytes

Fuente: Ayuda CCS PIC C COMPILER

2.5 Funciones para el manejo de interface UART

2.5.1 *Configuración del dispositivo SCI*

Sintaxis: #USE RS232 (opciones)

Tabla 21. Opciones posibles para la configuración de la interface UART

Todas las opciones deben estar separadas por comas[17].	
BAUD=x	Establece la velocidad de transmisión a *x* (opcional cuando se trabajara a una velocidad determinada). Para cambiar la velocidad en tiempo de ejecución use: *setup_uart(9600);*
XMIT=pin	Define el pin de transmisión
RCV=pin	Define el pin de recepción
FORCE_SW	Genera software necesario para la transición incluso cuando los pines UART han sido.
PARITY=X	Donde *x* es N, E o O
BITS =X	Donde *x* va de 5-9 (5 y 7 puede no ser usado con el SCI)
STREAM=id	Asocia a un identificador de flujo de datos con este puerto RS232. El identificar puede entonces ser usado en funciones como fputc, por ejemplo.

Fuente: Ayuda CCS PIC C COMPILER

Ejemplo de uso:

#use RS232 (baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

[17]*:Las opciones mostradas son unas de las más importantes*

2.5.2 *Funciones relevantes*

Tabla 22. Funcionesdisponibles para el manejo de interface UART

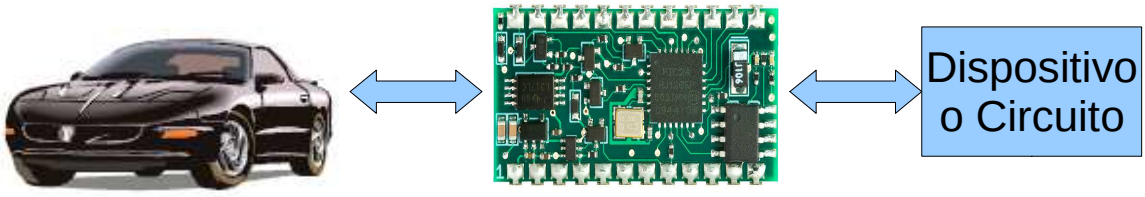
Estas funciones y directivas pueden ser usadas para configurar y usar las funcionalidades de RS232 I/O	
getc() o getch() *getchar() o fgetc()*	Consigue un carácter en el pin de recepción (de un stream específico en el caso de fgetc, entrada estándar por defecto). Use KBHIT para revisar si el carácter está disponible.
gets() orfgets()	Consigue una cadena de caracteres sobre el pin de recepción (de un stream específico en el caso de fgets, entrada estándar por defecto).
putc() o putchar() o fputc()	Pone un carácter sobre el pin de transmisión (de un stream específico en el caso de fputc, salida estándar por defecto).
puts() ofputs()	Pone una cadena de caracteres sobre el pin de recepción (de un stream específico en el caso de fputs, salida estándar por defecto).
printf() o fprintf()	Genera una cadena de caracteres con un formato específico y lo pone en el pin de transmisión (de un stream específico en el caso de fprint, salida estándar por defecto).
kbhit()	Retorne TRUE cuando un caracterestá disponible en el buffer de recepción cuando el PIC tiene Hardware RS232 o cuando el primer es enviado en el pin de recepción en caso de software RS232. Revísese continuamente antes de usar getc o similar.
setup_uart(baud,[stream]) o setup_uart_speed(baud,[stream])	Usado para cambiar la velocidad de transmisión del hardware UART en tiempo de ejecución. Especificar el stream es opcional.

Fuente: Ayuda CCS PIC C COMPILER

2.6 Módulo para interface obd2-UART (Micro OBD200) [9]

Este módulo es una rápida y sencilla forma de añadir un soporte OBD a cualquier proyecto en aplicaciones automotrices. Este es basado en los comandos del popular ELM327, soporta velocidades UART de hasta 10 Mbps.

Figura10. Empleo del módulo OBD200



Fuente: Autores

2.6.1 *Protocolos OBD soportados*

- SAE J1850 PWM
- SAE J1850 VPW
- ISO 9141
- ISO 14230 (KWP 2000)
- ISO 15765 (CAN)
- ISO 11898
- SAE J1939

2.6.2 *Características del Micro OBD200*

Este módulo a diferencia del vehículo recibe/envia los datos en ASCII y no en binario puro como el vehículo, y los comandos enviados hacia el módulo, deben terminar en 0x0D o retorno del carro para que sean válidos mientras que los datos recibidos desde este, deben terminar con “>” (0x3E); que indica que es el fin del mensaje. Por lo que se debe siempre esperar este valor para enviar otro comando y no colapsar la transmision.

Entonces si enviamos un comando para la solicitud del valor del MAF hacia el vehículo, este se debe enviar al OBD200 en la siguiente forma:

Comando MAF (sin la cabecera y el checksum) ”0110\r”

Esto en binario seria {0x30,0x31,0x31,0x30,0x0D}, lo cual es muy diferente al número binario puro que recibe el vehiculo {0x01,0x10}. Por lo tanto se debe transformar siempre de ASCII a birario puro y viceversa, para cualquier cálculo respectivo.

Con las diferencias antes mencionadas que de alguna forma es algo incomodo para hacer cálculos y tratar los datos, este módulo ofrece ciertas ventajas a la hora de comunicarse con el vehículo ya que este se puede o no configurar los bytes de cabecera (header) , pero para cuando lo configuramos esto se lo hace una sola vez, ya que el módulo lo hara por nosotros cada vez que enviamos datos hacia el vehículo. Por ejemplo esto quedaría asi:

Si nos comunicaramos directamente con el vehiculo estos bytes tendriamos que enviar lo siguiente.

Tabla 23. Ejemplo de comunicación sin el módulo OBD200

Bytes de cabecera			**Bytes de datos**		**Checksum**
68	6B	F1	O1	10	XX

Fuente: SAE REV APR2002

Pero con el módulo Micro OBD200 cuando nos vayamos a comunicar podríamos configurar la cabecera con “AT SH 686BF1\r”, entonces la próxima vez que necesitamos solicitar el valor del MAF solo enviaríamos “0110\r”.

2.6.3 *Tipos de comandos en OBD200*

Existen 3 tipos de comandos que el Micro OBD200 reconoce:

- Comandos AT
- Comandos ST
- Comandos OBD

2.6.3.1 *Comandos AT*

Muchos parámetros dentro del Micro OBD200 pueden ser ajustados con la finalidad de modificar su comportamiento y el comportamiento con la transmisión con el vehículo. Por ejemplo apagar los caracteres que este envía como eco, ajustar valores de tiempo, cambiar los valores de los bytes de cabecera, etc.

Para este fin los comandos deben empezar con los caracteres ASCII “AT”, para mayor información acerca de estos comandos sírvase revisar el anexo C.

2.6.3.2 *Comandos ST*

Estos comandos son designados para proveer una funcionalidad extendida a la que ofrecen los Comandos AT, y deben empezar con las letras “ST”. Los dos tipos están disponibles simultáneamente.

Para mayor información acerca de estos comandos sírvase revisar el anexo C.

2.6.3.3 *Comandos OBD*

Si los bytes que son enviados al OBD200 no empiezan con las letras “AT” o “ST”, este asume que es un comando para el vehículo. Cada ASCII bytes son testeados para asegurarse que son dígitos hexadecimal válidos. Entonces este combinará los demás bytes que se deben enviar al vehículo.

Luego de que pase el tiempo necesario para que el vehículo responda, este lo enviará para que le demos el uso, que nosotros consideremos necesario.

Si se necesita más información sírvase observar los anexos A y B, solo recuerde que se debe enviar en ASCII.

2.7 Programación mediante Visual Express C Sharp Edition 2008 [10]

2.7.1 *Lenguaje C#.* La sintaxis de C# cuenta con menos de 90 palabras clave; es sencilla y fácil de aprender. La sintaxis de C# ofrece funciones eficaces tales como tipos de valores que aceptan valores NULL, enumeraciones, delegados, métodos anónimos y acceso directo a memoria, que no se encuentran en Java. C# también admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteradores, que permiten a los implementadores de clases de colección definir comportamientos de iteración personalizados que el código de cliente puede utilizar fácilmente.

2.7.2 *Formulario Windows Forms.* Los formularios Windows Forms son la tecnología que se utiliza en Visual C# para crear aplicaciones basadas en Windows que se ejecutan en .NET Framework. Cuando crea un proyecto de aplicación para Windows, está creando una aplicación basada en formularios Windows Forms. Utilizará el Diseñador de Windows Forms para crear la interfaz de usuario y tendrá acceso a otras funciones de diseño y tiempo de ejecución:

- Implementación ClickOnce
- Compatibilidad enriquecida de bases de datos con el control DataGrisView
- Barras de herramientas y otros elementos de interfaz de usuario que pueden tener el aspecto y comportamiento de Microsoft® Windows® XP, Microsoft Office o Microsoft Internet Explorer.

En Visual C#, la forma más rápida y cómoda de crear la interfaz del usuario (UI) es hacerlo visualmente, con el Diseñador de Windows Forms y el Cuadro de herramientas. Hay tres pasos básicos para crear todas las interfaces de usuario:

- Agregar los controles a la superficie de diseño.
- Establecer las propiedades iniciales de los controles.
- Escribir los controladores para los eventos especificados.

2.7.3 *Herramientas de Visual C#.* A continuación se detallan las herramientas y ventanas más importantes de Visual C#. Las ventanas de la mayoría de estas herramientas se pueden abrir desde el menú Ver.

- El Editor de código, para escribir código fuente.
- El compilador de C#, para convertir el código fuente de C# en un programa ejecutable.
- El depurador de Visual Studio, para probar el programa.
- El Cuadro de herramientas y el Diseñador, para desarrollar rápidamente interfaces de usuario con el mouse.
- El Explorador de soluciones, para ver y administrar archivos de proyecto y configuraciones.
- El Diseñador de proyectos, para configurar opciones del compilador, rutas de implementación, recursos, etc.
- La Vista de clases, para desplazarse por el código fuente según los tipos, no los archivos.
- La Ventana Propiedades, para configurar propiedades y eventos en los controles de la interfaz de usuario.
- El Examinador de objetos, para ver los métodos y clases disponibles en las bibliotecas de vínculos dinámicos, incluidos los ensamblados de .NET Framework y los objetos COM.
- Document Explorer, para explorar y buscar la documentación del producto en su equipo local y en Internet.

2.7.4 *Herramientas GDI*

2.7.4.1 *Clases y Estructuras del namspaceSystem.Drawing.* El namespaceSystem.Drawing proporciona acceso a la funcionalidad básica del GDI+.

Clases

Brush: Proporciona funcionalidad para rellenar una determinada región o figura con el color, textura, etc. que se indique.

Font: Esta clase permite trabajar y definir las características: tipo, tamaño, color, estilo, etc. de letra en el objeto graphics.

Pen: Sirve para especificar el ancho y el estilo de la línea de dibujo o de la figura Geométrica.

Graphics: Encapsula las Superficies de dibujo de los formularios. Antes de dibujar cualquier objeto Figura: un punto, una elipse, un rectángulo etc. Es necesario crear u obtener el objeto Graphics de la superficie.

Si se desea utilizar un evento, como por ejemplo si se quiere dibujar al pulsar un botón se puede obtener el objeto Graphics invocando al método CreateGraphics() del formulario. En este caso es muy importante invocar al método Dispose() al final del método, para liberar esos recursos, ya que el GarbageCollector no lo hará por defecto en este caso.

```
Private void formulafio_Click(objet sender, EventArgs e)
{
        Graphics g = this.CreateGraphics();
        //código
        g.Dispose();    //Muy importante liberar recursos
}
```

Estructuras

Point: Esta estructura tiene dos propiedades que representan las coordenadas X e Y del punto a partir del origen.

El constructor más usual es:

Public point (int x, int y);

Point F: Es una estructura muy similar a Point, except o que X e Y son de tipo *float*.

Rectangle: Esta estructura permite construir objetor que representan una región rectangular. Como sucede en la estructura Point, también se define la estructura RectangleF, que es idéntica a esta pero que sus coordenadas son del tipo *float*.

Los constructores de esta clase son:

```
Public Rectangle (Point esquinaSupIzda, Size tamaño);
```

Public Rectangle (int x, inty, intancho, int alto);

Métodos

Una vez que se tiene la referencia del objeto Graphics se puede invocar cualquier miembro de la clase Graphics para dibujar objetos.

DrawEllipse: Dibuja una elipse
DrawLine: Dibuja una línea recta.
DrawRectangle: Dibuja un rectángulo.
FillEllipse: Rellena el interior de una elipse.
FillRectangle: Rellena el interior de un rectángulo con un determinado Brush.
DrawString: Escribe texto

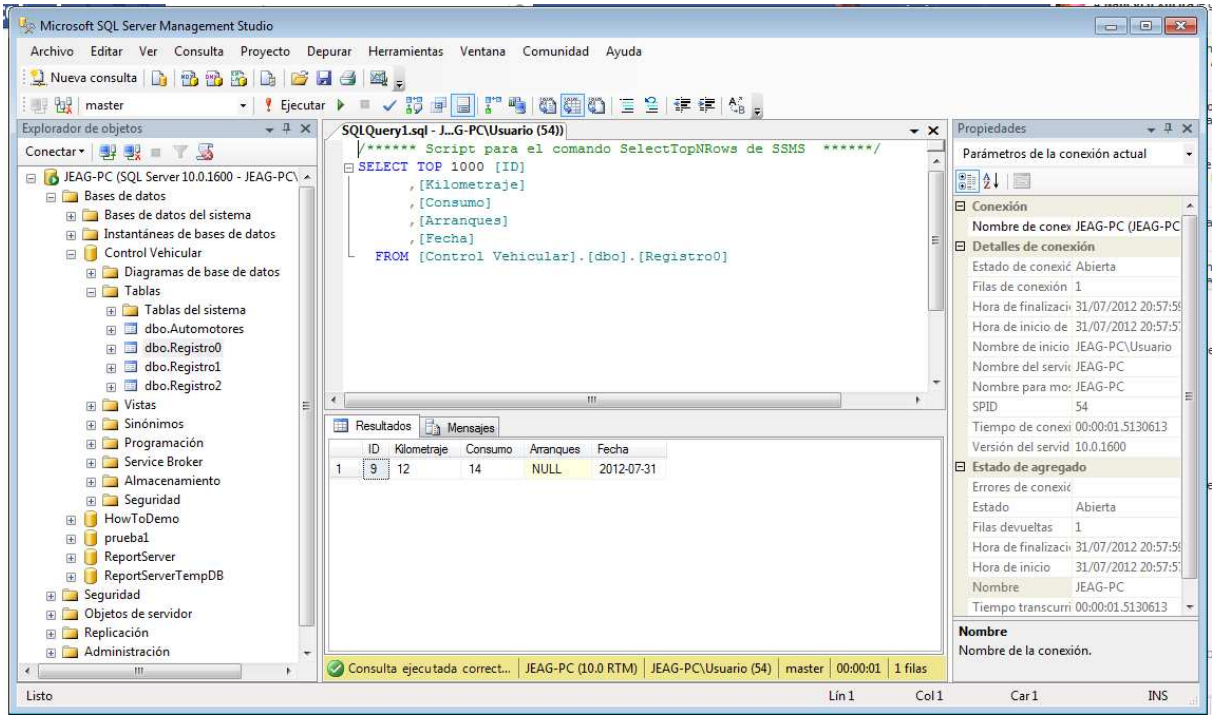
2.7.5 *Conexión a base de datos SQL*

2.7.5.1 *Microsoft SQL Server.* Microsoft SQL Server es un sistema de base de datos relacional de gestión desarrollado por Microsoft. Como una base de datos, es un producto de software cuya principal función es de almacenar y recuperar datos según lo solicitado por otras aplicaciones de software, ya sea los que en el mismo equipo o los que se ejecuta en otro ordenador a través de una red (incluido Internet).

2.7.5.2 *SQL Management Studio 2008.* SQL Server Management Studio es una aplicación de software lanzada por primera vez con el servidor de Microsoft SQL Server 2005 que se utiliza para configurar, gestionar y administrar los componentes de Microsoft SQL Server.

Una característica central de SQL Server Management Studio es el explorador de objetos, que permite al usuario navegar, seleccionar y actuar en cualquiera de los objetos en el servidor.

Figura11. Pantalla de Microsoft SQL Server Management Studio



Fuente: Autores

2.7.5.3 *System.Data.SqlClientNamespace.* El espacio de nombres System.Data.SqlClient es el proveedor de datos .NET Framework para SQL Server.

El proveedor de datos .NET Framework para SQL Server describe una colección de clases que se utiliza para obtener acceso a una base de datos de SQL Server en el espacio administrado.Al utilizar SqlDataAdapter, se puede rellenar un objeto DataSet residente en memoria, que sirve para consultar y actualizar la base de datos.

2.7.5.4 *Clases*

SqlDataAdapter: Representa un conjunto de comandos de datos y una conexión de base de datos que se utilizan para rellenar un DataSet y actualizar una base de datos de SQL Server.

El método fill sobrecargado rellena un objeto **DataSet** o un objeto **DataTable**.

El método update(DataTable) Llama a las instrucciones INSERT, UPDATE o DELETE respectivas para cada fila insertada, actualizada o eliminada en la tabla DataTable especificada.

SqlCommandBuilder: Genera automáticamente comandos de tabla única que se utilizan para conciliar los cambios realizados en un objeto DataSet con la base de datos de SQL Server asociada. Esta clase no puede heredarse.

DataTable: Representa una tabla de datos en memoria.

La propiedad Rows del DataTable obtiene la colección de filas que pertenecen a esta tabla.

El método NewRow del DataTable crea un nuevo DataRow con el mismo esquema de la tabla.

El método AcceptChanges del DataTableconfirma todos los cambios realizados en esta tabla desde que se ha cargado o desde la última vez que se ha llamado a AcceptChanges.

DataRow: Es uno de los componentes principales de un DataTable. Se utiliza el objeto DataRow y sus propiedades y métodos para recuperar, evaluar, insertar, eliminar y actualizar los valores de DataTable.

CAPÍTULO III

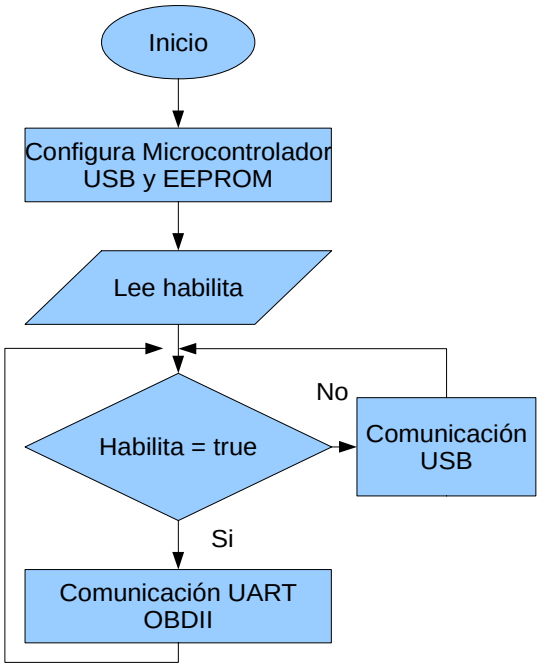
3. DISEÑO

3.1 Software

3.1.1 *Programación del firmware*

3.1.1.1 *Diagrama de flujo de programa principal.* Al proveer de energía eléctrica suficiente al micro controlador PIC, este empezará con las instrucciones correspondientes, a la configuración del dispositivo, USB y memoria EEPROM 24FC512. Para posteriormente ponerse en un bucle donde se pregunta el estado de la variable habilita, la misma que es cambiada de estado por la interrupción INT2, que se produce al cambiar de estado el vehículo. Es ahí que, sí el vehículo está apagado este atenderá funciones USB si son requeridas y sí el vehículo está encendido, el PIC se dedicara a realizar los cálculos de consumo y recorrido únicamente. Hay que destacar que la velocidad de procesamiento de las instrucciones es de 48 MHz.

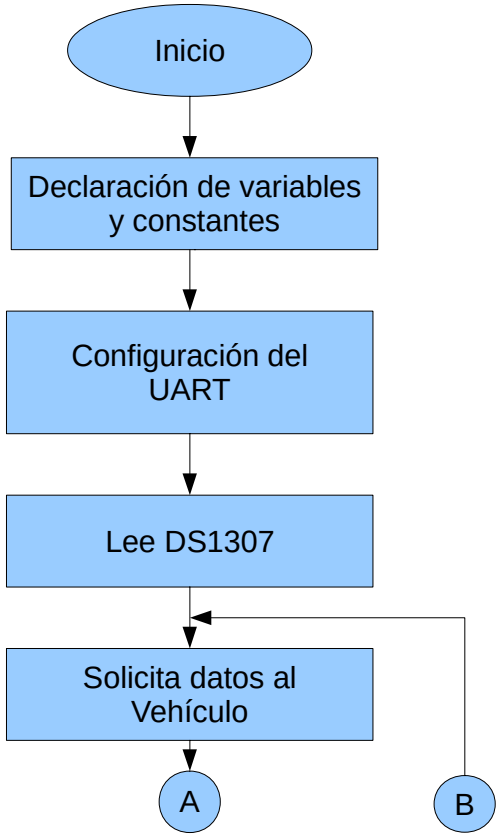
Figura12. Diagrama de flujo del programa principal del micro-controlador

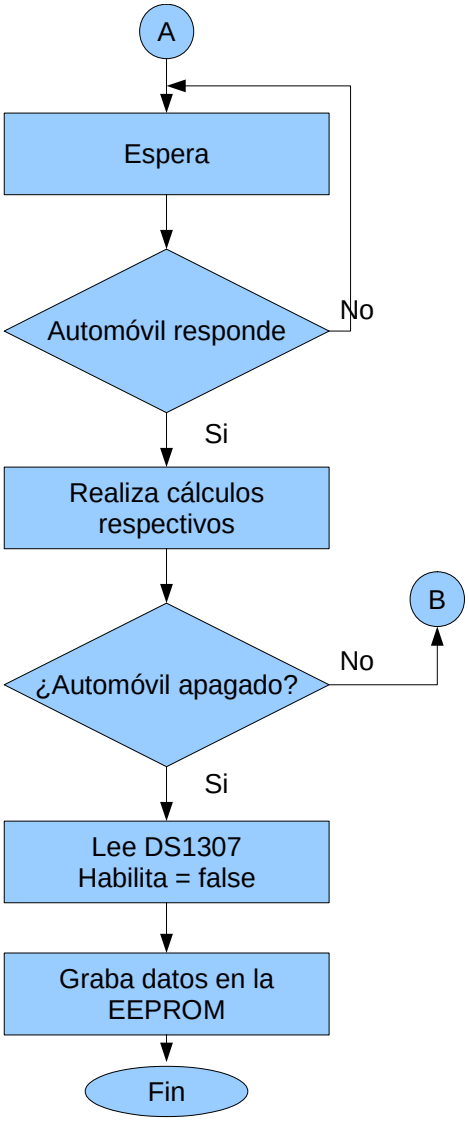


Fuente: Autores

3.1.1.2 *Diagrama de flujo de la función comunicación UART-OBD2.* Esta función es la encargada, una vez que se prende el vehículo de realizar las configuraciones necesarias de la transmisión I2C, UART, Micro OBD200. Las mismas que son necesarias para leer datos de fecha de encendido y apagado, comunicarse con el vehículo, realizar los cálculos respectivos para calcular el consumo y recorrido. Y al final de todo, cuando se apague el vehículo, guardar todos los datos en la memoria EEPROM. Para cuando se solicite enviar hacia la base de datos de la PC que complementa el proyecto.

Figura13. Diagrama de flujo de la función comunicación UART-OBDII

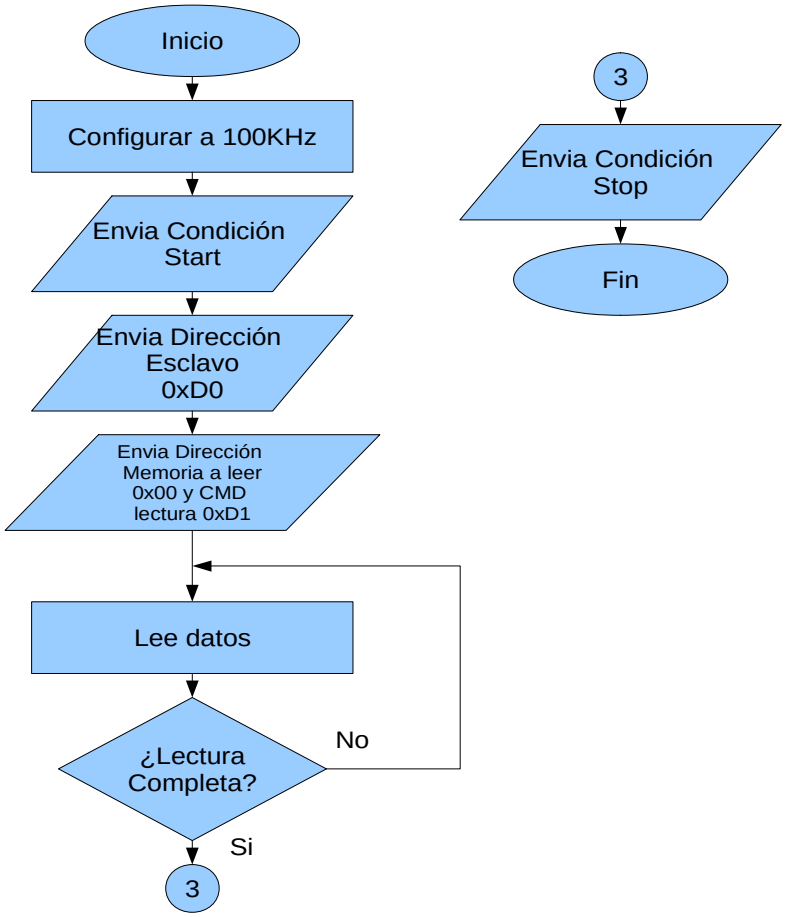




Fuente: Autores

3.1.1.3 *Diagrama de flujo de la función Lee IC DS1307 Reloj-Calendario.* Esta función se encarga de generar la secuencia necesaria para la lectura de fecha y hora desde el DS1307, así como de reconfigurar la velocidad I2C necesaria para una lectura y escritura satisfactoria, la misma que es de 100 KHz.

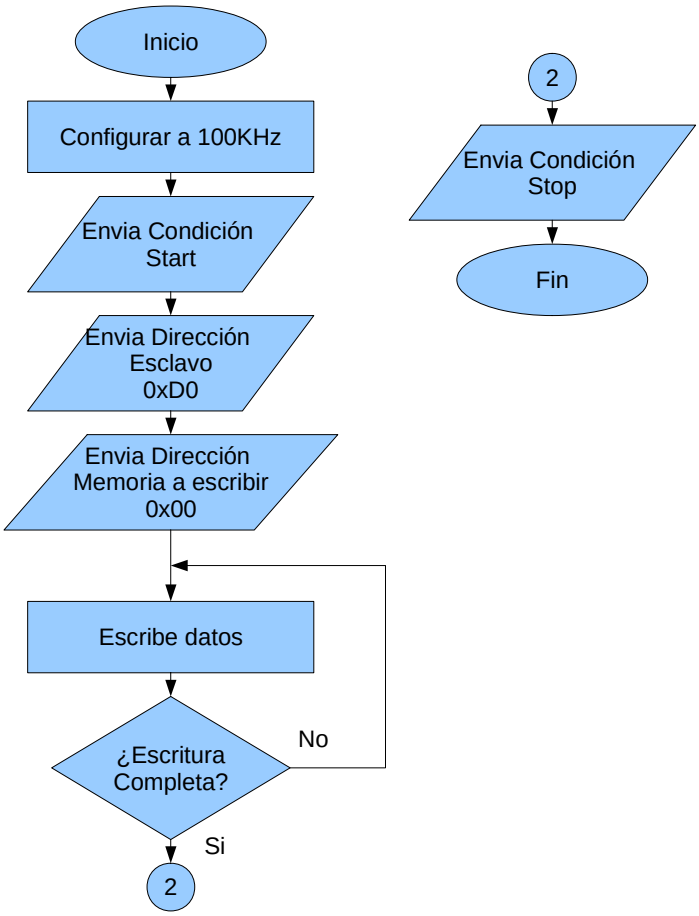
Figura14. Diagrama de flujo de la función que lee IC DS1307



Fuente: Autores

3.1.1.4 *Diagrama de flujo de la función actualiza IC DS1307 Reloj-Calendario.* Esta función se encarga de generar la secuencia necesaria para la escritura, tanto de hora como fecha, así como de activar el DS1307, así como de reconfigurar la velocidad I2C necesaria para una lectura y escritura satisfactoria, la misma que es de 100 KHz.

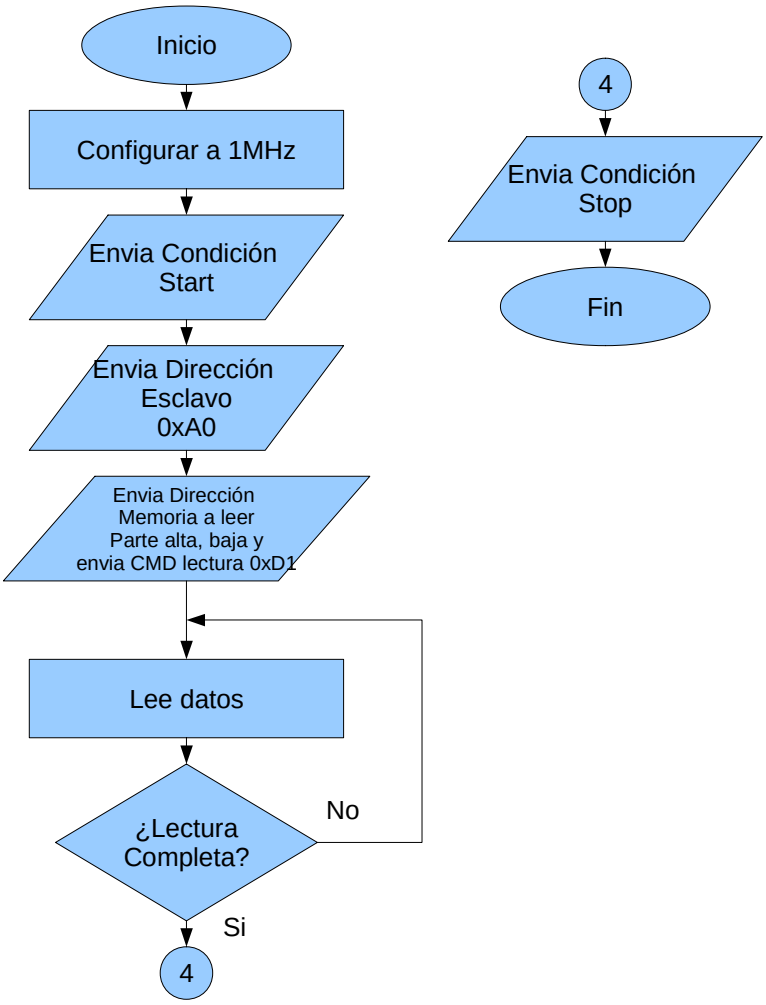
Figura15. Diagrama de flujo de la función que actualiza IC DS1307



Fuente: Autores

3.1.1.5 *Diagrama de flujo de la función Lee datos de la EEPROM.*Esta función se encarga de generar la secuencia necesaria para la lectura de la EEPROM 24FC512, y ponerla en un puntero de memoria, que luego será enviado hacia la PC por la interface USB, así como de reconfigurar la velocidad I2C necesaria de 1 MHz.

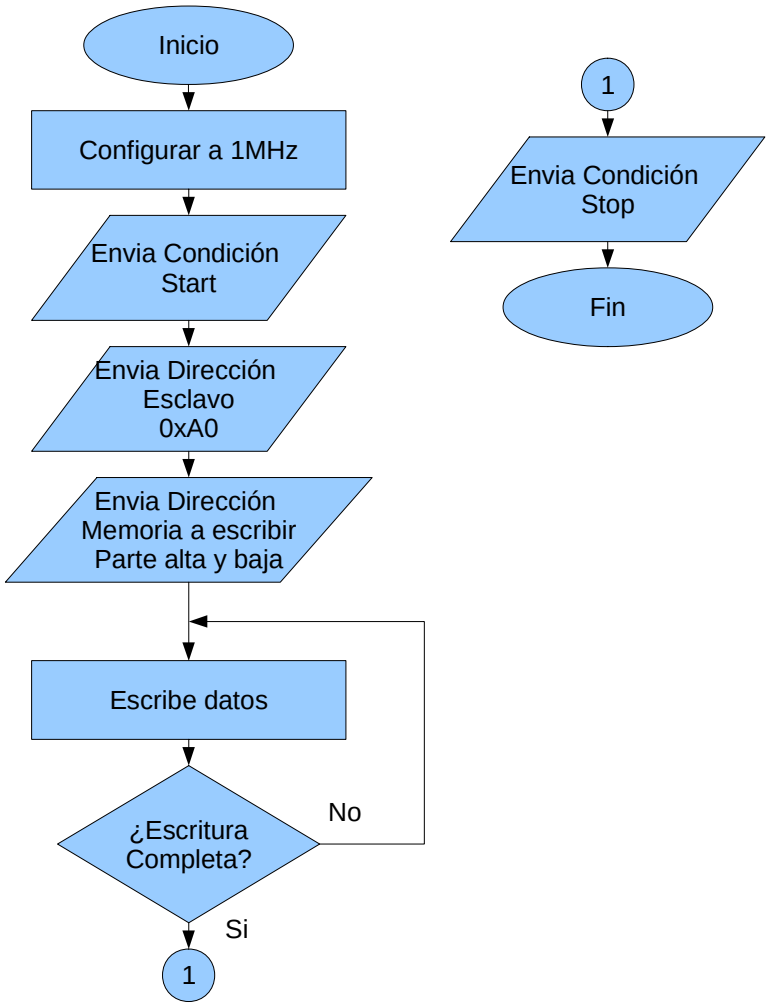
Figura16. Diagrama de flujo de la función lee datos de la EEPROM



Fuente: Autores

3.1.1.6 *Diagrama de flujo de la función escribe datos de la EEPROM.* Esta función se encarga de generar la secuencia necesaria para la escritura de la EEPROM 24FC512, tomando en cuenta que la memoria necesita 5ms para la grabación y que el paquete máximo para escritura secuencial es de 128 bytes, así como de reconfigurar la velocidad I2C necesaria de 1 MHz.

Figura17. Diagrama de flujo de la función escribe datos de la EEPROM

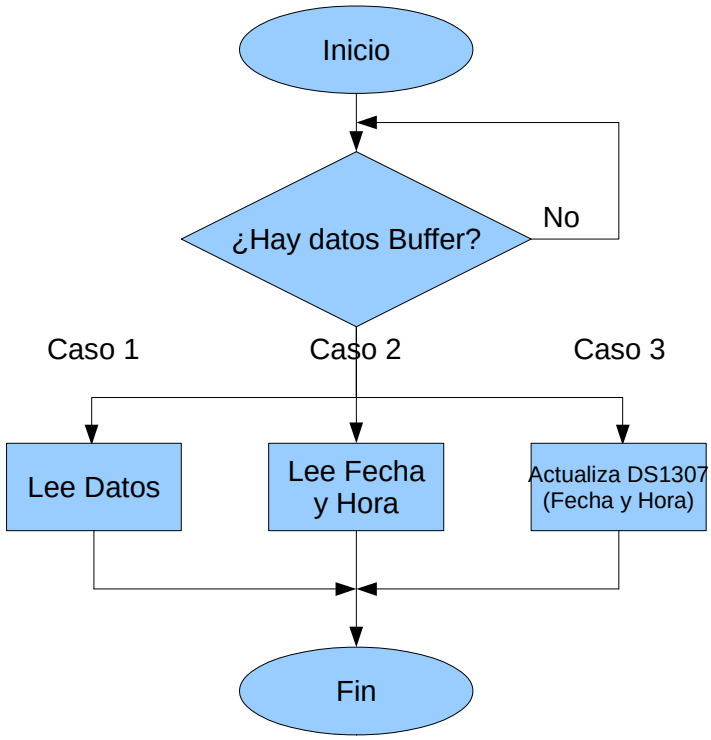


Fuente: Autores

3.1.1.7 *Diagrama de flujo de la función comunicación USB.*Esta función estará únicamente accesible cuando el vehículo se encuentra apagado con la finalidad de que el micro controlador esté disponible y le dedique todo su tiempo a recibir órdenes desde el computador. Los cuales pueden ser los tres casos mostrados en el siguiente diagrama. Todas estas transmisiones se realizan por transferencia masiva[18].

[18]Bulktransfers

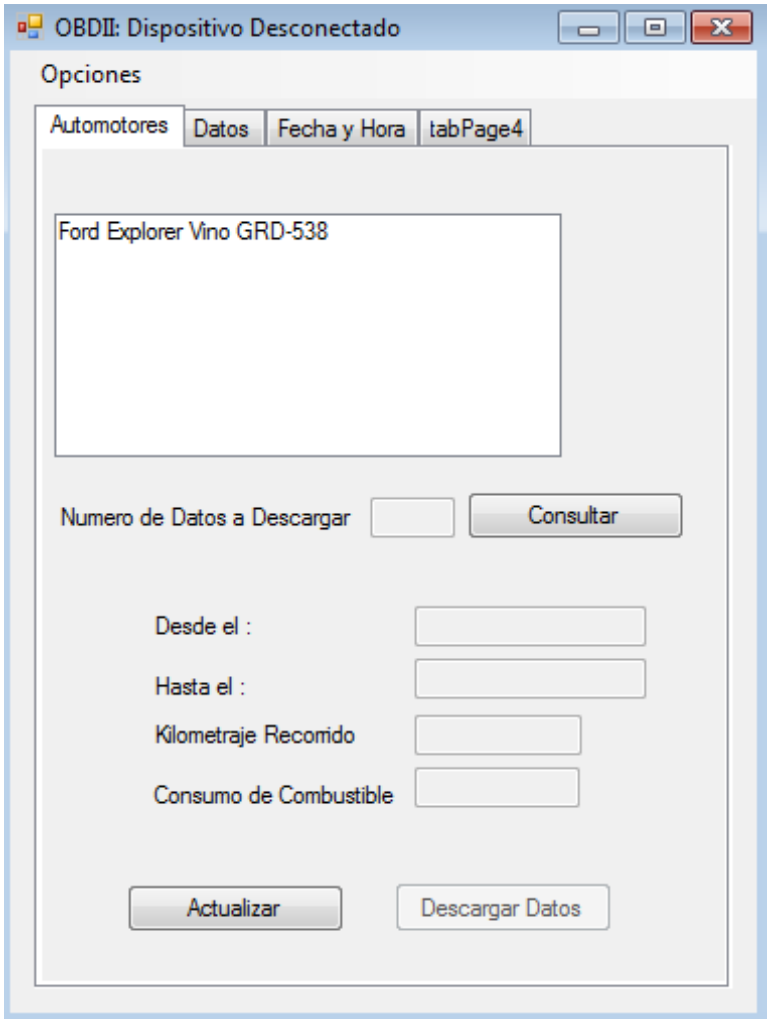
Figura18. Diagrama de flujo de la función comunicación USB



Fuente: Autores

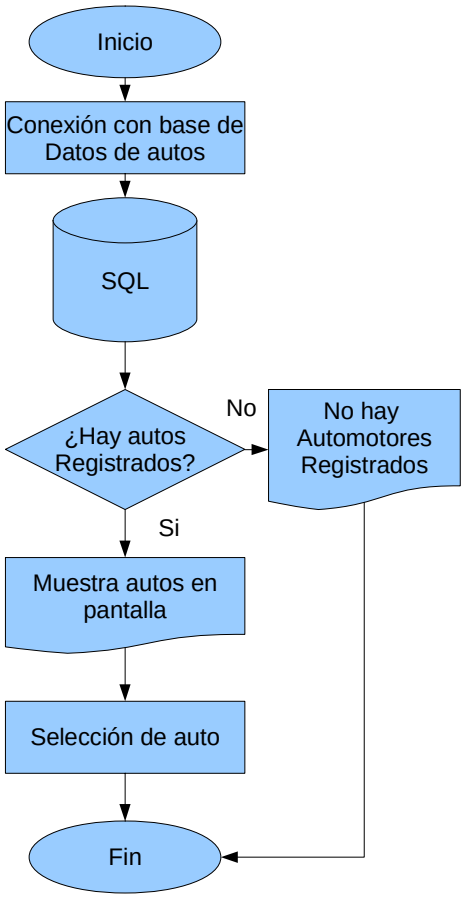
3.1.1.8 *Programación del software de la aplicación de Windows.* Al cargar el programa se realiza la conexión con la base de datos de SQL server en la tabla “Automotores”, si existe algún registro previo estos datos se mostraran en la pantalla.

Figura19. Pantalla principal de la aplicación de Windows



Fuente: Autores

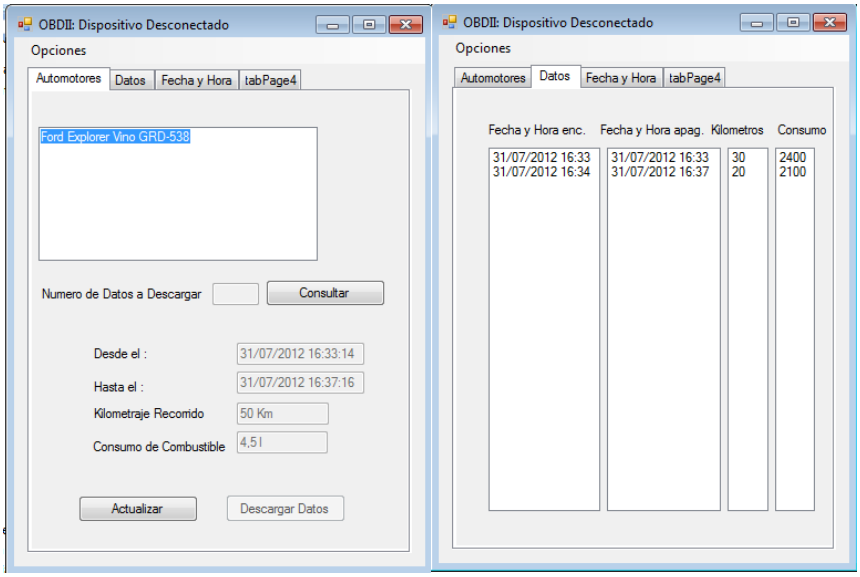
Figura20. Diagrama de flujo principal de la aplicación de Windows



Fuente: Autores

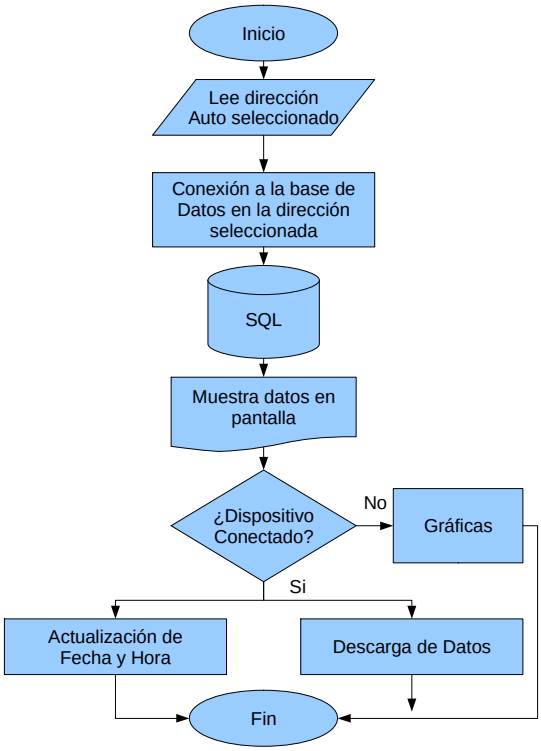
Al seleccionar uno de los autos mostrados se registra una variable correspondiente a dicho auto con la cual se realiza nuevamente la conexión al SQL server pero esta vez a la tabla perteneciente a la variable seleccionada, mostrándose en la pantalla los datos registrados.

Figura21. Pantalla de datos de la aplicación de windows



Fuente: Autores

Figura22. Diagrama de flujo de la función selección de auto

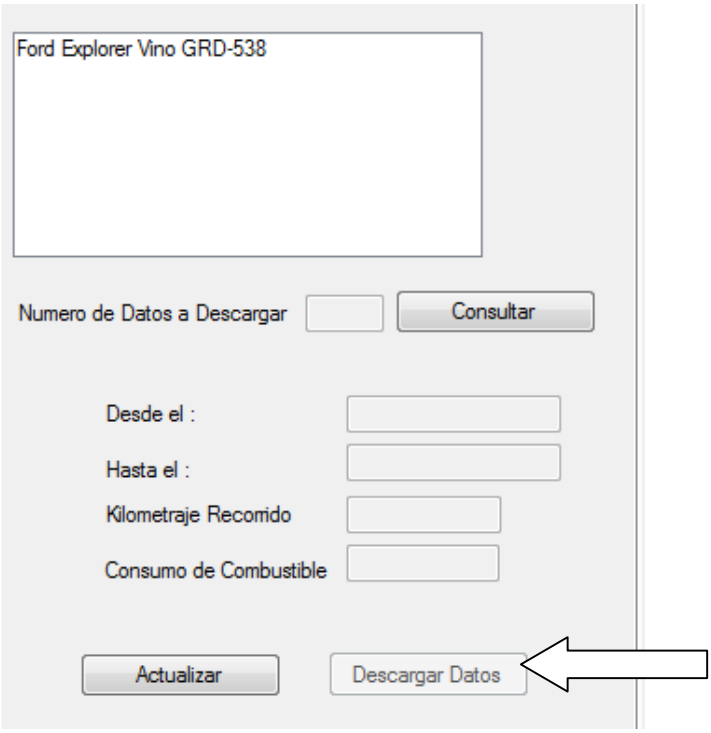


Fuente: Autores

Si el dispositivo se encuentra conectado se puede elegir entre las opciones de descargar datos o actualizar la fecha y hora del dispositivo.

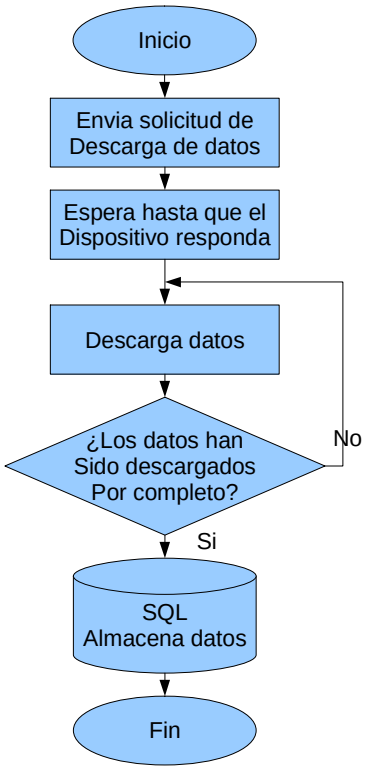
La opción de descargar datos permite guardar en el SQL server los datos almacenados en la memoria del dispositivo, para ello se envía la solicitud de descarga de datos mediante el comando de control 0x07, 0x07, una vez hecho esto se habilita la conexión a la base de datos en la tabla correspondiente al auto seleccionado y se guardan en la columna respectiva los datos de: recorrido, consumo, fecha y hora de encendido, fecha y hora de apagado.

Figura23. Botón de descarga de datos



Fuente: Autores

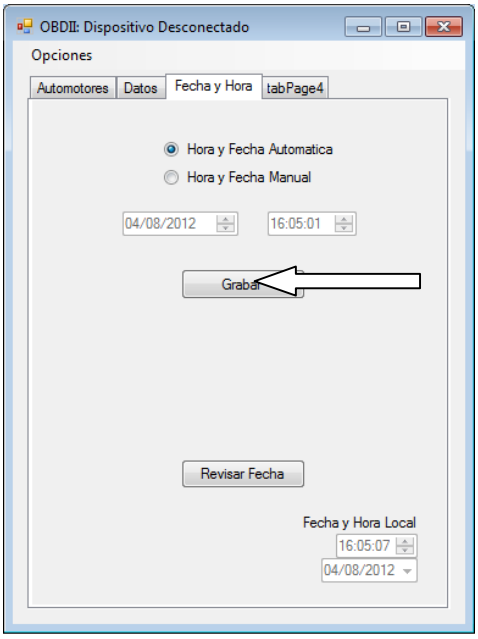
Figura24. Diagrama de flujo de la función descarga de datos



Fuente: Autores

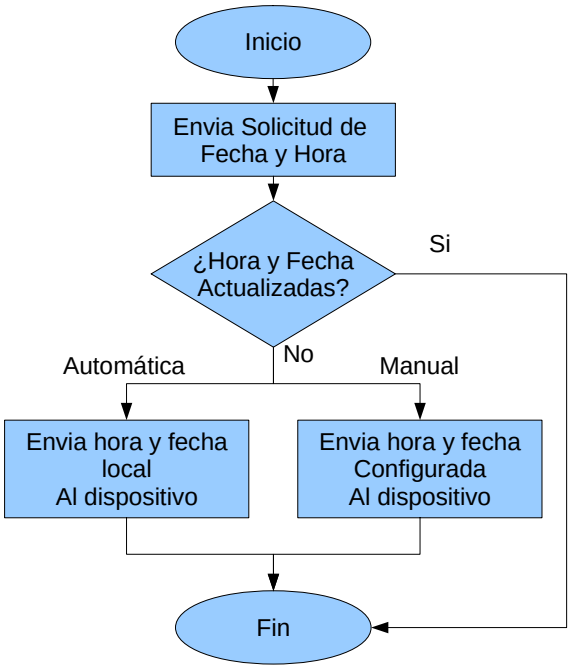
Actualizar la fecha y hora permite fijar la hora del componente DS1307 tanto automática como manualmente, para ello se envía la solicitud de actualizar fecha y hora mediante el comando de control 0x4, 0x4.

Figura25. Botón de actualización de fecha y hora



Fuente: Autores

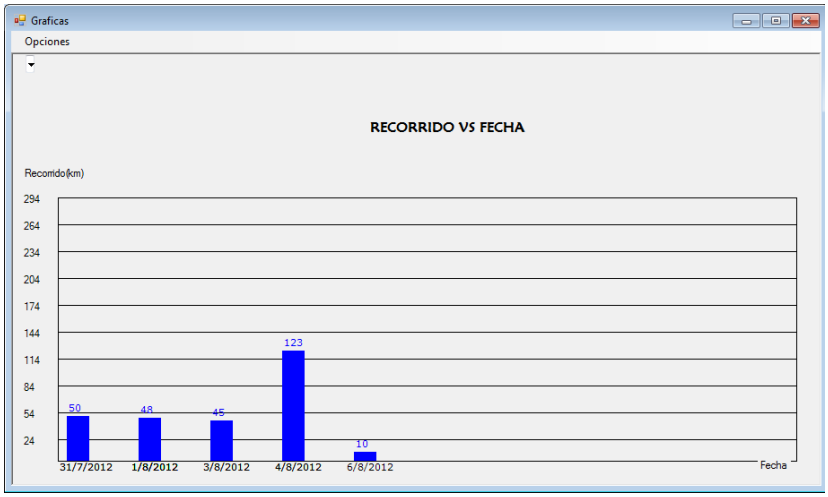
Figura26. Diagrama de flujo de la función Actualización de fecha y hora



Fuente: Autores

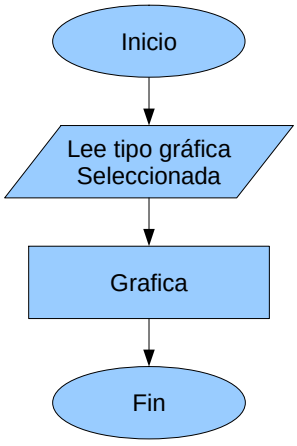
Cuando la aplicación esta desconectada del dispositivo se puede elegir la opción de gráficas, luego de seleccionar un automotor se abre la conexión a la base de datos en la tabla que corresponde a dicho auto; se puede elegir entre varios tipos de Figuras los cuales muestran registros tanto por fecha como por hora.

Figura27. Pantalla de gráficas de la aplicación de windows



Fuente: Autores

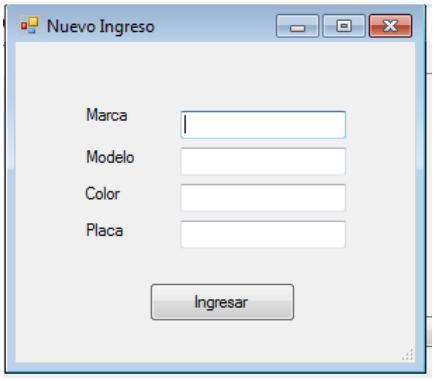
Figura28. Diagrama de flujo de la función Gráficas



Fuente: Autores

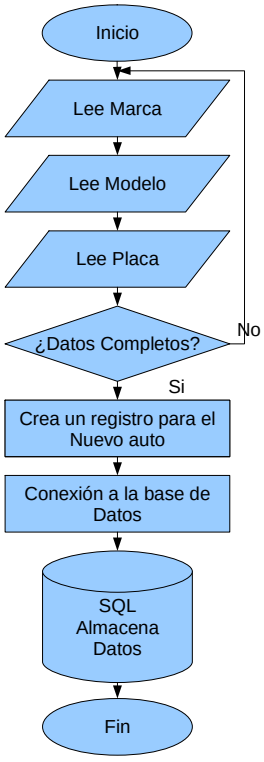
Para ingresar un nuevo registro, se ingresa todos los datos del automotor, una vez hecho esto se abre la conexión al SQL server en la tabla “Automotores” y se crea una nueva fila con los datos del nuevo auto, luego se actualiza la base de datos y se cierra la conexión.

Figura29. Pantalla de nuevo ingreso de automotor de la aplicación de windows



Fuente: Autores

Figura30. Diagrama de flujo de la función nuevo ingreso auto



Fuente: Autores

Para conocer cómo se maneja el software consultar el manual de usuario ANEXO D

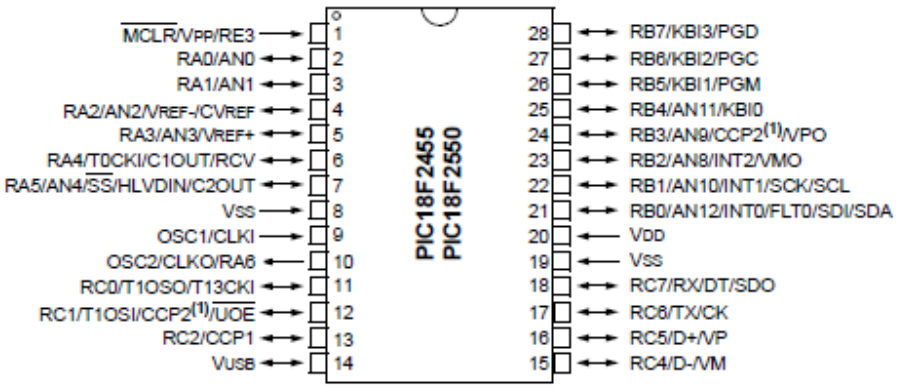
3.2 Hardware

3.2.1 *Selección de componentes*

3.2.1.1 *PIC 18F2550.*Se ha seleccionado este elemento ya que proporciona todas las características necesarias para la elaboración del presente proyecto tales como:

- Soporte USB.
- Un PLL de alta precisión que ayuda a conseguir los 48MHz necesarios para el USB 2.0.
- Incorpora un módulo USART.
- Un módulo Master Synchronous Serial Port (MSSP) que incorpora un módulo I2C™ que puede trabajar en modo maestro y esclavo.
- Arquitectura optimizada para compilador C.
- Retención de firmware en la EEPROM mayor de 40 años.
- Rango de voltaje de operación de 2,0V a 5,5V.
- Dos niveles de prioridad para las interrupciones.
- Cuatro módulos Timer (Timer0 a Timer3).

Figura31. PIC 18F2550



Fuente: Datasheet PIC 18F2550

Tabla 24. Características del PIC 18f2550

Device	ProgramMemory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	#Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I2C™			
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3

	Numero Pin	Tipo Pin	Tipo Buffer	Descripción
MCLR/Vpp/RE3	1			Master Clear (entrada) o voltaje de programación (entrada).
MCLR		I	ST	Master Clear (Reset) entrada. Este pin resetea al dispositivo en nivel bajo.
Vpp		P		Entrada de voltaje de programación.
RE3		I	ST	Entrada Digital.
OSC1/CLKI	9			Cristal oscilador o entrada de reloj externo.
		I	Análogo	Entrada de cristal oscilador o entrada de reloj externo
		I	Análogo	Entrada de reloj externo. Siempre asociado con la función del pin OSC1
OSC2/CLKO/RA6	10			Salida de cristal oscilador o reloj
OSC2		O	--	Salida de cristal oscilador. Conecta al cristal o resonador en modo de cristal oscilador
CLKO		O	--	Pin OSC2 salidas CLOK que tiene ¼ de la

RA6		I/O	TTL	frecuencia del OSC1. Pin de propósito general I/O
				PORTA es un puerto I/O bidireccional
RA0/AN0	2			
RA0		I/O	TTL	Digital I/O
AN0		I	Análo	Entrada análoga 1
RA1/AN1	3		go	
RA1		I/O		Digital I/O
AN1		I	TTL	Entrada análoga I/O
RA2/AN2/V_{REF}/CV_{REF}	4		Análo	
		I/O	go	Digital I/O
RA2		I		Entrada análoga 2
AN2		I	TTL	Entrada de voltaje de referencia (bajo) A/D
V_{REF}		O	Análo	Salida de comparador de referencia
CV_{REF}	5		go	análogo
RA3/AN3/V_{REF+}		I/O	Análo	
RA3		I	go	Digital I/O
AN3		I	Análo	Entrada análoga 3
V_{REF+}	6		go	Entrada de voltaje de referencia (alto) A/D
RA4/T0CKI/C1OUT		I/O		
/RCV		I	TTL	Digital I/O
RA4		O	Análo	Entrada de reloj externo Timer0
T0CKI		I	go	Salida comparador 1
C1OUT	7		Análo	Entrada de USB externo transceiver RCV
RCV			go	
RA5/AN4/SS/HLVD		I/O		
IN/		I	ST	Digital I/O
C2OUT		I	ST	Entrada análoga 4
RA5		I	--	Entrada del esclavo SPI
AN4		O	TTL	Entrada de detección de voltaje alto/bajo

SS HLVDIN C2OUT RA6	--	--	 TTL Análogo TTL Análogo -- --	Salida comparador 2 Ver el pin OCSC2/CLKO/RA6
RB0/AND12/INT0/FLT0/ SDI/SDA RB0 AND12 INT0 FLT0 SDI SDA RB1/AN10/INT1/SCK/ SCL RB1 AN10 INT1 SCK SCL RB2/AN8/INT2/VM	21 22 23 	 I/O I I I I/O I/O I I I/O I/O I/O	 TTL Análogo ST ST ST ST TTL Análogo ST ST 	PORTB es un puerto bidireccional I/O. El PORTB puede ser programado por software las resistencias internas pull-ups Digital I/O Entrada análoga 12 Interruptor externo 0 Entrada PWM (Modulo CCP1). Entrada de datos SPI I2C™ data I/O Digital I/O Entrada análoga 10 Interruptor externo 1 Synchronous serial clock I/O para el modo SPI Synchronous serial clock I/O para el modo I2C

O		I		
RB2		I	TTL	Digital I/O
AN8		O	Análo	Entrada análoga 8
INT2	24		go	Interruptor externo 2
VMO		I/O	ST	Salida USB transceiver VMO externo
RB3/AN9/CCP2/VP		I	--	
O		I/O		Digital I/O
RB3			TTL	Entrada análoga 9
AN9		O	Análo	Entrada captura 2/Salida comparador
CCP2	25		go	2/Salida PWM 2
		I/O	ST	Salida USB transceiver VPO externo.
VPO		I		
RB4/AN11/KBI0		I	--	Digital I/O
RB4	26			Entrada análoga 11
AN11		I/O	TTL	Interrupt-on-change pin
KBI0		I	Análo	
RB5/KBI1/PGM		I/O	go	Digital I/O
RB5			TTL	Interrupt-on-change pin
KBI1	27			Pin habilitado en voltaje bajo para
PGM		I/O	TTL	programación de ICSP
		I	TTL	
RB6/KBI2/PGC		I/O	ST	Digital I/O
RB6				Interrupt-on-change pin
KBI2	28			In-CircuitDebugger y pin de reloj en la
PGC		I/O	TTL	programación ICSP
		I	TTL	
RB7/KBI3/PGD		I/O	ST	Digita I/O
RB7				Interrupt-on-change pin
KBI3				In-Circuit-Debugger y pin de datos en la
PGD			TTL	programación ICSP

			TTL ST	
				PORTC es un puerto I/O bidireccional
RC0/T1OSO/T13CK	11			
I		I/O	ST	Digital I/O
RC0		O	--	Salida del oscilador Timer1
T1OSO		I	ST	Entrada de reloj externo Timer1/Timer3
T13CKI	12			
RC1/T1OSI/CCP2/U		I/O	ST	Digital I/O
OE		I	CMO	Entrada del oscilador Timer1
RC1		I/O	S	Entrada captura 2/salida comparador
T1OSI			ST	2/salida PWM2
CCP2		--		Salida USB transceiver OE externo
	13		--	
UOE		I/O		Digital I/O
RC2/CCP1		I/O	ST	Entrada captura 1/salida comparador
RC2	15		ST	1/salida PWM1
CCP1		I		Entrada digital
RC4/D-/VM		I/O	TTL	Línea negativa diferencial USB
RC4		I	--	(Entrada/Salida)
D-	16		TTL	Entrada USB transceiver VP externo
VM		I		
RC5/D+/VP		I/O	TTL	Entrada digital
RC5		O	--	Línea positiva diferencial USB
D+	17		TTL	(Entrada/Salida
VP		I/O		Entrada USB transceiver VP externo
RC6/TX/CK		O	ST	
RC6		I/O	--	Digital I/O
TX	18		ST	EUSART asynchronous transmit
CK		I/O		EUSART synchronous clock (ver RX/DT)

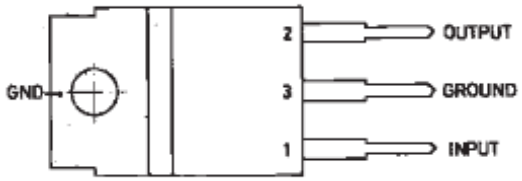
RC7/RX/DT/SDO RC7 RX DT SDO		I I/O O	ST ST ST --	 Digital I/O EUSART asynchronous receive EUSART synchronous data (ver TX/CK) Salida de datos SPI
RE3	--	--	--	Ver pin MCLR/Vpp/RE3
V_{USB}	14	O	--	Regulador de voltaje interno para USB de 3.3V
V_{SS}	8, 19	P	--	Referencia de tierra para pines lógicos y I/O
V_{DD}	20	P	--	Suministro positivo para pines lógicos y I/O

Fuente: Datasheet PIC 18F2550

TTL = Entrada compatible con TTL
CMOS = Entrada o salida compatible con CMOS
ST = Entrada Schmitt Trigger con niveles CMOS
O = Salida
I = Entrada
P = Fuente de energía

3.2.1.2 *Regulador de voltaje positivo L7805.* Este tipo de regulador emplea una limitación de corriente interna, apagado termal y un área segura de protección haciéndolo esencialmente indestructible, su voltaje de salida es de 5V los cuales son los necesarios para el correcto funcionamiento del circuito.

Figura32. Regulador L7805

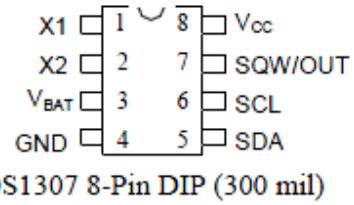


Fuente: Datasheet L7805

3.2.1.3 *Reloj Calendario DS1307.* Este dispositivo es el encargado de proporcionar la fecha y hora que será empleada para poder guardar los registros de consumo y recorrido en tiempo real, posee las siguientes características:

- El reloj cuenta segundos, minutos, horas, día de la semana, día del mes, mes y año con una compensación de año bisiesto,válido hasta el 2100.
- 56 bytes de RAM no volátil para almacenamiento de datos.
- 2 pines para interface serial.
- Consume menos de 500 nA en modo de batería de respaldo.

Figura33. Reloj calendario DS1307



Fuente: Datasheet DS1307

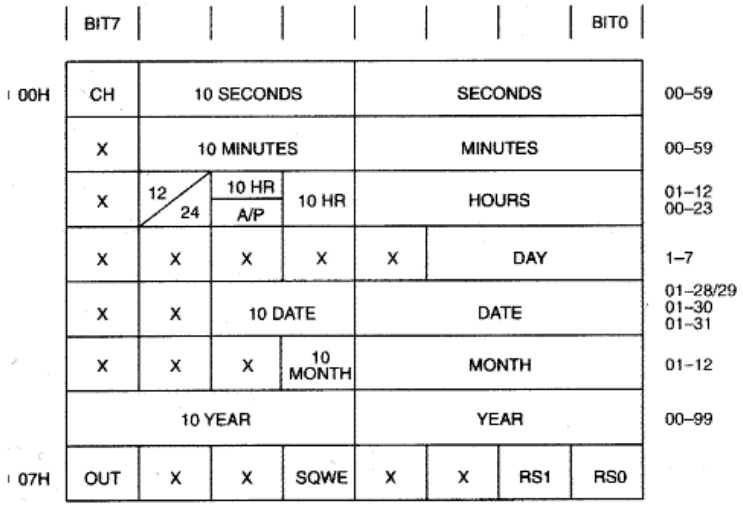
Tabla 25. Descripción de los pines del reloj calendario DS1307

Nombre Pin	**Numero Pin**	**Descripción**
V_{CC}	8	Suministro primario de energía

X1, X2	1, 2	Conexión de cristal de 32 768 KHz
V_{BAT}	3	Entrada de batería +3V
GND	4	Tierra
SDA	5	Dato serial
SCL	6	Reloj serial
SQW/OUT	7	Onda cuadrada/salida Driver

Fuente: Datasheet DS1307

Figura34. Registros que guardan el tiempo en el DS1307



Fuente: Datasheet DS1307

3.2.1.4 *Memoria 24FC512.* Esta es una memoria PROM de 64k x 8 (512 Kbit) que se puede borrar eléctricamente de forma serial cuyo rango de operación va de 1,8V a 5,5V, puede ser empleada para aplicaciones de baja energía tales como comunicaciones personales y adquisición de datos.

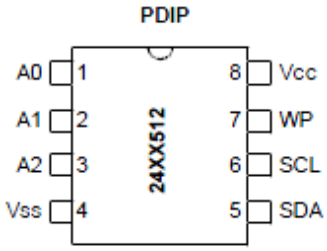
La razón principal para la selección de este elemento es una alta velocidad de transmisión de datos (1 MHz), además posee las siguientes características:

- Tecnología CMOS de baja energía
    - Corriente máxima de escritura 5mA en 5,5V

- Corriente máxima de lectura 400µA en 5,5V
- Corriente de espera 100nA, típico en 5,5V

- 2 pines de bus de interface serial, compatible con I2C™
- Se pueden enlazar hasta 8 dispositivos.
- Tiempo de ciclo de escritura máximo 5ms
- 1 000 000 de ciclos escritura/lectura
- Protección de descarga electrostática > 4000V
- Retención de datos >200 años

Figura35. Memoria 24FC512



Fuente: Datasheet 24FC512

Tabla 26. Descripción de los pines de la memoria 24FC512

Nombre Pin	**Número Pin**	**Descripción**
A0	1	Pin para dirección configurada por el usuario
A1	2	Pin para dirección configurada por el usuario
A2	3	Pin para dirección configurada por el usuario
V_{SS}	4	Tierra
SDA	5	Dato serial
SCL	6	Reloj serial
WP	7	Entrada de protección de escritura

V_{CC}	8	+2,5V a 5,5V

Fuente: Datasheet 24FC512

3.2.1.5 *Modulo Micro OBD200*

Tabla 27. Descripción de los pines del módulo OBD200

Pin #	**Nombre del Pin**	**Tipo de Pin**	**Descripción del Pin**
1	+5V	P	Suministro de Positivo 5V
2	GND	P	Negativo para lógica digital (J1962 pin 5)
3	N/A	--	No conectada
4	VBAT	P	Suministro de batería 12V (J1962 pin 16)
5	L-Line	O	ISO9141/ISO14230 salida L-Line (J1962 pin 15)
6	K-Line	I/O	ISO9141/ISO14230 Bidireccional L-Line (J1962 pin 7)
7	____ Sleep	I	Entrada de control externo del modo ahorro de energía (sleep). Cuando está habilitado por software, pone el módulo en estado de bajo consumo. La polaridad puede ser configurada en firmware (bajo por defecto). Una resistencia pull-up para 3.3V es activo por defecto, pero puede ser desactivado en firmware. Deje libre el pin si no lo usa.
8	$\overline{\text{LP\_OUT}}$	O	Salida activa en bajo para interruptor externo en estado bajo de energía. Este pin tiene una resistencia pull-up a +3.3V. Déjelo desconectado si no lo usa.
9	UART_RX	I	Entrada del receptor de UART, Compatible con +3.3 y +5V lógico.
10	UART_TX	O	Salida de transmisión de UART.

			Drenadorabierto-requiere una resistencia pull-up a +3.3V o +5V (4 mA máx.). La resistencia de pull-up depende de la velocidad UART y la longitud de trazo; el valor típico es de 4.7 KΩ
11	CAN_L	I/O	CAN bajo línea bidireccional (J1962 pin 14).
12	CAN_L	I/O	CAN alto línea bidireccional (J1962 pin 6).
13	_____ _____ OBD_TX_LED/RST_NVM	I/O	Activo en bajo para la salida de LED de actividad en la transmisión OBD (4 mA máx.)/ Entrada activa en bajo para resetear NVM a valores de fábrica. Drenador abierto-pull up a +3.3V interna. Déjelo desconectado si no lo usa.
14	_____ ___ OBD_RX_LED / INT	O	Activo en bajo para la salida de LED de actividad en la recepción OBD (4 mA máx.). Drenador abierto por defecto (requiere pull-up a +3.3V o +5V cuando es configurado como interrupción); puede ser configurado como salida digital +3.3V
15	_____ UART_TX_LED	O	Salida activa en bajo para LED de actividad en transmisión UART (4 mAmax.). El voltaje en el ánodo del LED no debe exceder +3.3V.
16	UART_TX_LED	O	Salida activa en bajo para LED de actividad en recepción UART (4 mAmax.). El voltaje en el ánodo del LED no debe exceder +3.3V.
17	+3.3V_OUT	P	Suministro de voltaje positivo de 3.3 voltios. Máxima corriente disponible de 85 mA.

18	AVDD	P	Suministro de voltaje analógico. Debe ser conectado a +3.3V_OUT o a una referencia externa de voltaje (entre +3.0 y +3.6V).
19	AVSS	P	Referencia de tierra para la entrada analógica ANALOG_IN. Tiene un punto de conexión a GND. Debería ser conectada a una referencia externa de tierra o si lo prefiere déjelo desconectado.
20	ANALOG_IN	A	Entrada para la medida de voltaje análogo (+3.3V máx.). Por defecto, calibrado para un divisor de voltaje de 62KΩ/10KΩ conectado a VBAT. Conecte a AVSS si no lo usa.
21	NC	--	No conecte
22	_____ RESET	I	Entrada activa en bajo para resetear el módulo. Internamente tiene una resistencia pull-up a +3.3V. Déjelo desconectado si no lo usa.
23	J1850_BUS-	I/O	Línea bidireccional SAE J1850 BUS- (J1962 pin 10).
24	J1850_BUS+	I/O	Línea bidireccional SAE J1850 BUS- (J1962 pin 2)

Fuente: Datasheet MICRO OBD200

Nota: P Principal

A Entrada analógica

I Entrada digital

I/O Entrada/Salida digital

O Salida digital

$\overline{\text{XXX}}$ El pin es negado

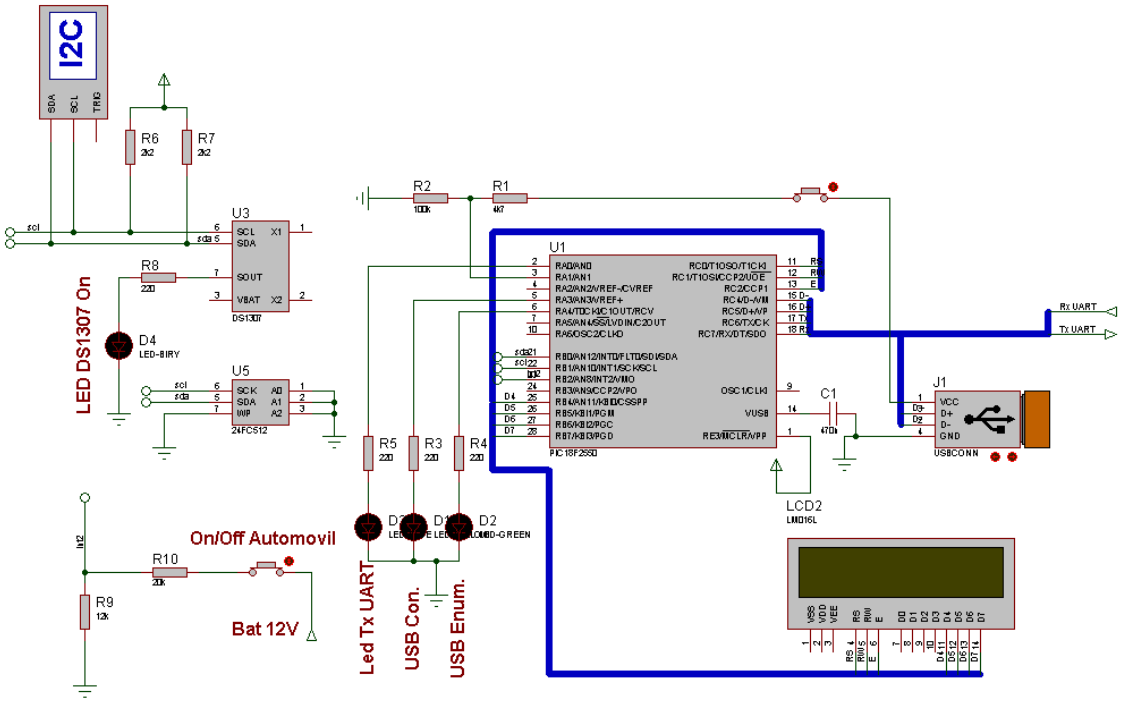
3.2.2 *Diseño del sistema.* Cada uno de los componentes mencionados en la sección anterior, necesitan de componentes pasivos para garantizar los valores de voltaje y capacitancia adecuada. Por lo que a continuación se muestra el ensamble de todos los componentes correspondiente al circuito asociado con:

- PIC 18F2550
- I2C DS1307 Reloj-Calendario
- I2C 24FC512 Memoria EEPROM externa
- LCD[19]
- Interface USB 2.0
- Conexión UART

Hay que destacar que las conexiones tales como, UART, I2C, USB; fueron realizadas en los pies del micro controlador designados para dicho fin.

[19]LCD fue implementado únicamente para fase de pruebas, con el fin de observar los datos en tiempo real.

Figura36. Diagrama del circuito experimental

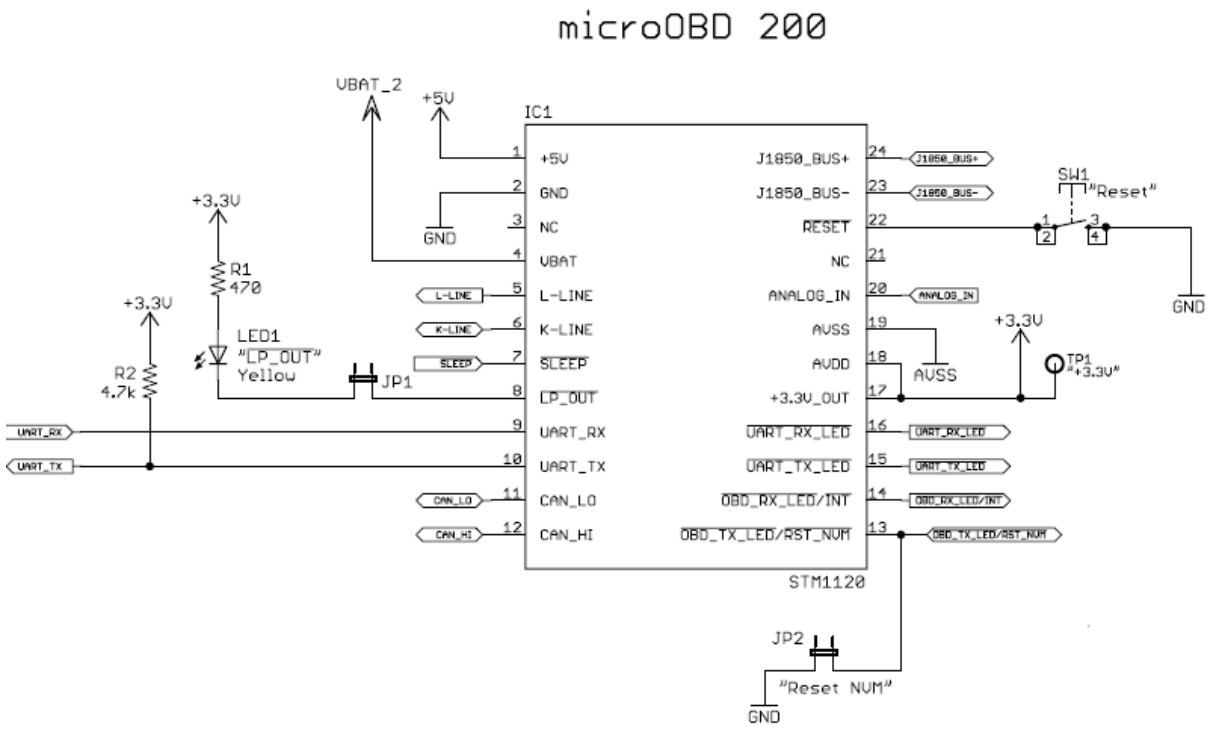


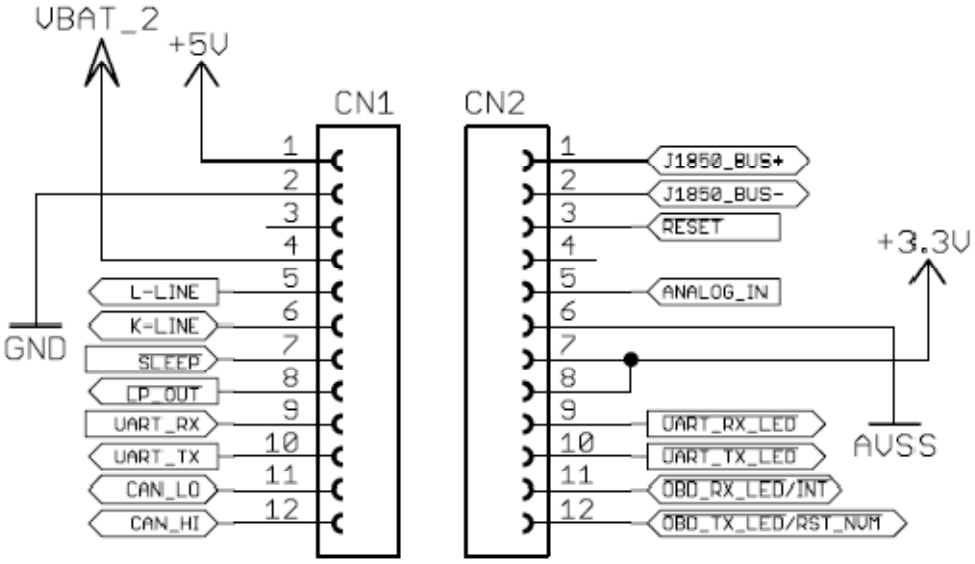
Fuente: Autores

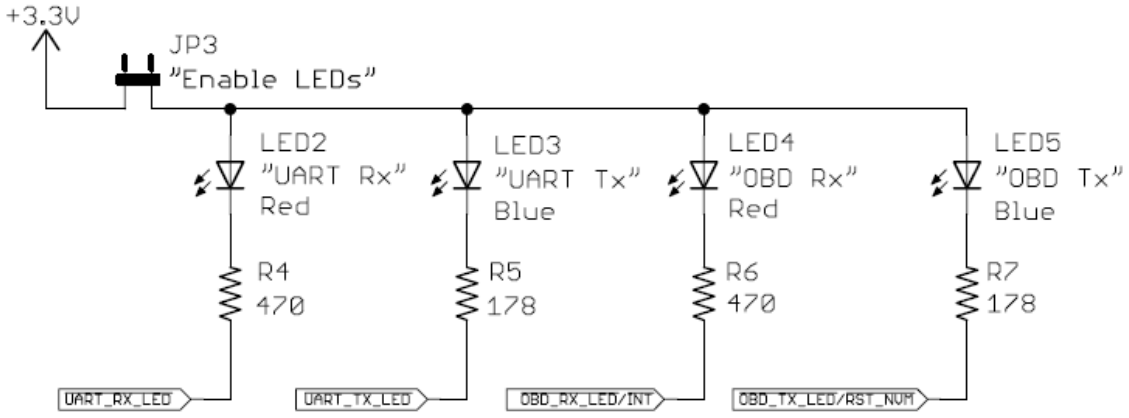
3.2.2.1 *Circuito para Micro OBD200*

El siguiente circuito es otorgado por el fabricante (OBD tools)

Figura37.MicroOBD 200

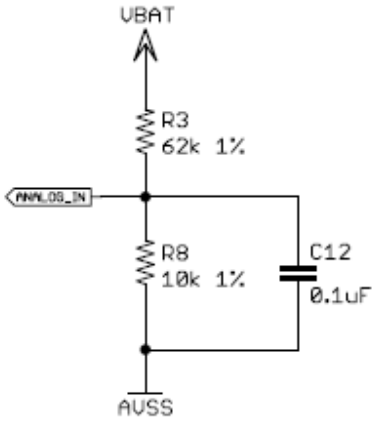






Fuente: Datasheet MICRO OBD200

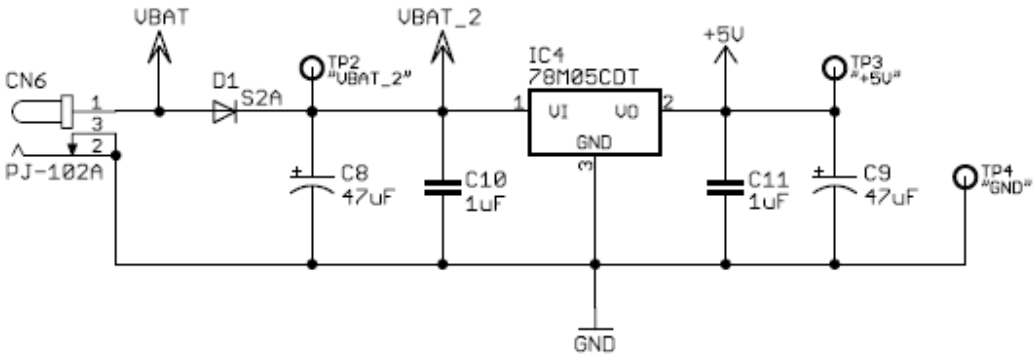
3.2.2.2 *Circuito para entrada analógica de la batería del automóvil*



Fuente: Datasheet MICRO OBD200

3.2.2.3 *Circuito correspondiente a la fuente de alimentación de 5V.* Este circuito es otorgado en la hoja de datos del componente 78M05.

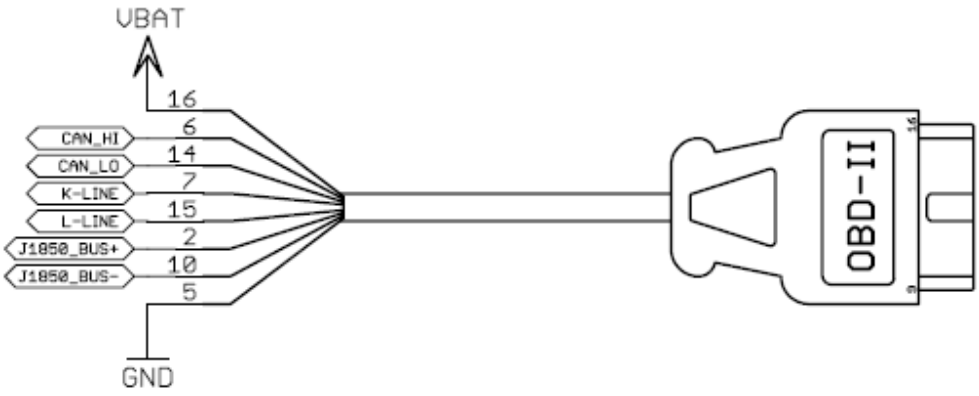
Figura38. Circuito par el regularos 78M05



Fuente: Datasheet MICRO OBD200

3.2.2.4 *Conexión de Micro OBD200 y el cable J1962 acorde al número de pin*

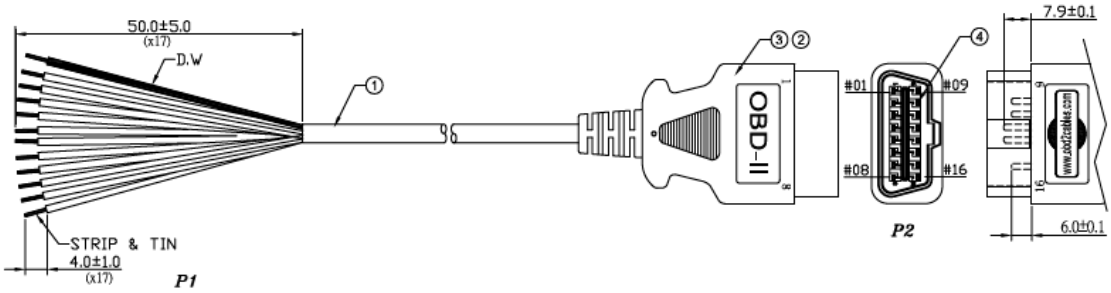
Figura39. Descripción de los pines del conector OBDII



Fuente: Datasheet MICRO OBD200

3.2.2.5 *Diagrama de conexión y especificaciones del cable OBD2 J1962*

Figura40. Conector OBDII



Fuente: DatasheetCable OBDII

Tabla 28 Descripción de los colores de cables del conector OBDII

Asignación de pines de P2	**Descripción**	**Color**
6	CAN Alto (J-2284)	Blanco/Negro
14	CAN Bajo (J-2284)	Verde/Negro
10	J1850 Bus -	Amarillo/Negro
2	J1850 Bus+	Azul/Negro
1	--	Celeste
3	--	Naranja
4	--	Negro+ D.W
5	Negativo	Marrón
7	--	Amarillo
8	--	Verde
9	--	Azul
11	--	Violeta
12	--	Gris
13	--	Blanco
15	--	Rosado
16	Positivo 12 v	Rojo

Tabla 29. Descripción general del conector OBDII

Punto	**Descripción**
1	Cable : UL2464 24 AWG (7/0.20TC)*2 Pares + 24AWG (7/0.20TC)*12C + Al/MY +D.W, OD=7.5+/-0.1mm
2	P2 con. OBD-IIA
3	Moldeado en PVC 45P
4	Tipo de Pin Delphi

Fuente: Datasheet Cable OBD2

CAPÍTULO IV

4. FUNDAMENTOS MATEMÁTICOS

4.1 Cálculo distancia recorrido

El valor correspondiente al PID 0x0D; que nos entrega el computador del vehículo, según la norma SAE J1979 está en Km/h (Véase anexo B). Dado que el vehículo se encuentra sometido al movimiento acelerado, se procede a lo siguiente.

El movimiento acelerado, cuyas relaciones dinámicas y cinemáticas, respectivamente, son:

$$a(t) = a = \frac{F}{m} = \frac{d^2x}{dt^2} = \frac{dv}{dt} = \frac{lim}{\Delta t \to 0}\frac{\Delta v}{\Delta t} \quad (1)$$

La velocidad v para un instante t dado es:

$$v(t) = \frac{dx}{dt} \quad (2)$$

Debido a que el tiempo de muestreo es muy pequeño, podemos aproximar los cálculos mediante las fórmulas de MUA. Así que: $v_f \; y \; t$ son datos que tenemos, ya que al final de cada lectura $v_f \; pasara \; a \; ser \; v_o$, el valor necesitado es el de la aceleración.

$$a = \frac{v_f - v_o}{t} \quad (3)$$

Ya que el tiempo es más fácil medir en segundos, la velocidad se trabajara en $\frac{m}{s}$ por lo que se debe dividir para la constante 3,6.

$$a = \frac{v_f - v_o}{t * 3.6} \quad (4)$$

Finalmente la posición x en función del tiempo se expresa por:

$$x_f = \frac{1}{2}at^2 + \frac{v_o t}{3.6} + x_o \quad (5)$$

Donde x_o, es la posición inicial y x_f la posición final luego de la última lectura. Al final de cada lectura, si las lecturas van a continuar, entonces $x_o = x_f$. Caso contrario x_f será la distancia total recorrida.

4.2 Cálculo consumo de combustible

Al igual que en el caso anterior el valor correspondiente al PID 0x10; que nos entrega el computador del vehículo, según la norma SAE J1979 está en $\frac{gramos_{aire}}{s}$ (Véase anexo B).

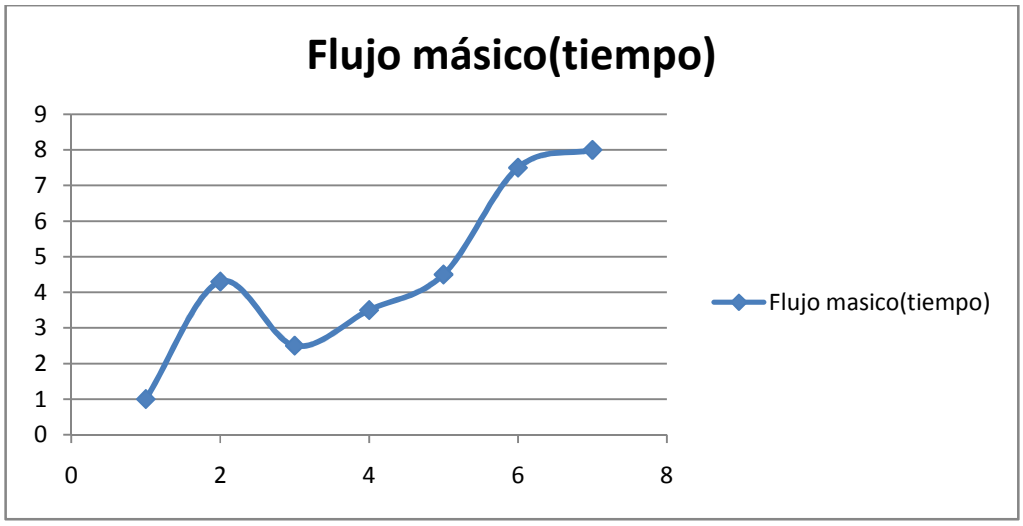
Dónde la masa de aire es transformada a masa de combustible, y posteriormente, a volumen de combustible:

1 gramo $_{aire}$	1 gramo combustible
s	14,7 gramo aire

Al realizar todos los cálculos respectivos, obtenemos una constante de transformación de 0,06803

Y al graficar estos, obtenemos una gráfica similar a la siguiente:

Figura41 Ejemplo de la gráfica de datos de flujo másico en función del tiempo



4.2.1 *Cálculo del consumo por la regla del trapecio.* En matemática la regla del trapecio es un método de integración numérica, es decir, un método para calcular aproximadamente el valor de la integral definida

$$F(x) = \int_a^b f(x)dx \quad (6)$$

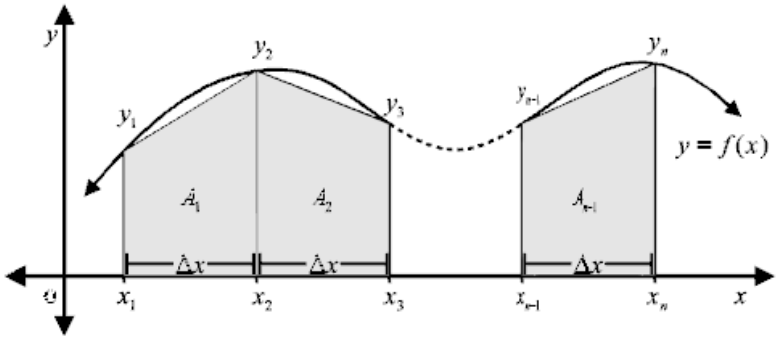
La regla se basa en aproximar el valor de la integral de f(x) por el de la función lineal que pasa a través de los puntos (a, f(a)) y (b, f (b)). La integral de ésta es igual al área del trapecio bajo la gráfica de la función lineal. Se sigue que

$$\int_a^b f(x)dx \approx \frac{(b-a)(f(a)+f(b))}{2} \quad (7)$$

Y donde el término error corresponde a:

$$-\frac{(b-a)^3}{12} f^2(\xi)^{20} \quad (8)$$

Figura42.Figura de la regla de los trapecios



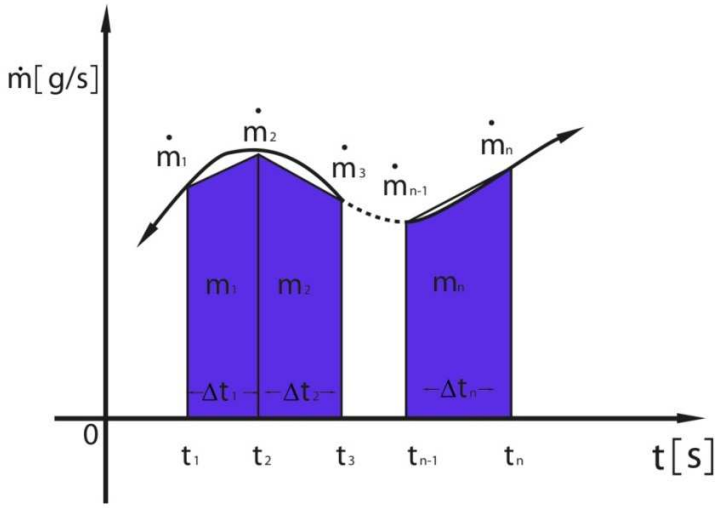
$$A_i = \frac{1}{2}(y_i + y_{i+1})\Delta x \quad (9)$$

$$A \cong \sum_{i=1}^{n-1} A_i \quad (10)$$

Para el caso de la gráfica flujo másico de combustible en función del tiempo el área bajo la curva representaría la masa de combustible consumido en un$tiempo$ (t).

[20]Siendo ξ un número perteneciente al intervalo $[a, b]$.

Figura43.Figura de la regla de los trapecios aplicados al cálculo de la masa



$$\mathrm{m} \approx \sum_{\mathrm{i}=1}^{\mathrm{n}-1} \mathrm{m_i} = \frac{1}{2}(\dot{m}_1 + \dot{\mathrm{m}}_2)\Delta t_1 + \frac{1}{2}(\dot{m}_2 + \dot{\mathrm{m}}_2)\Delta \mathrm{t}_2 + \ldots + \frac{1}{2}(\dot{m}_{\mathrm{n}-1} + \dot{m}_{\mathrm{n}})\Delta \mathrm{t}_{\mathrm{n}} \quad (11)$$

Donde el volumen seria:

$$Vol = \frac{1}{\delta} m \quad (12)$$

Siendo δ la densidad del combustible, y considerando que las variaciones de temperatura no son representativas. Se podría decir que es una constante aproximada de

$$740 \frac{g}{Litro}$$

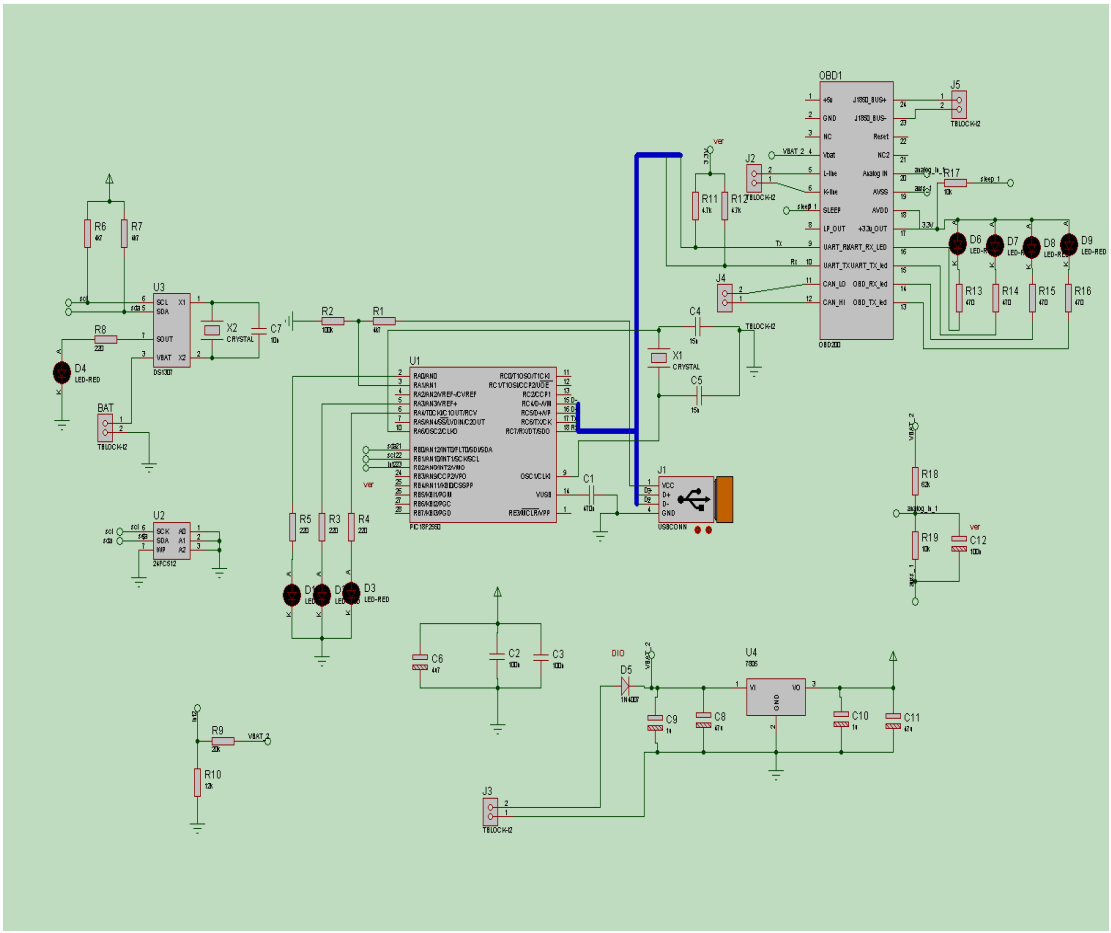
CAPÍTULO V

5. CONSTRUCCIÓN DE EQUIPOS Y SISTEMAS

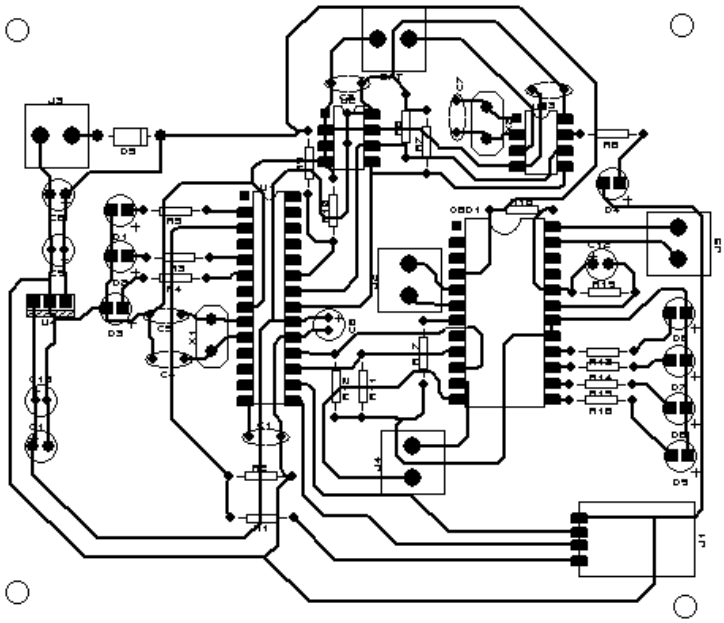
5.1 Diseño y construcción de la placa

A partir del esquema, se elaboró las pistas del circuito necesarias para la construccion de la placa.

Figura44. Diseño de pistas del circuito impreso



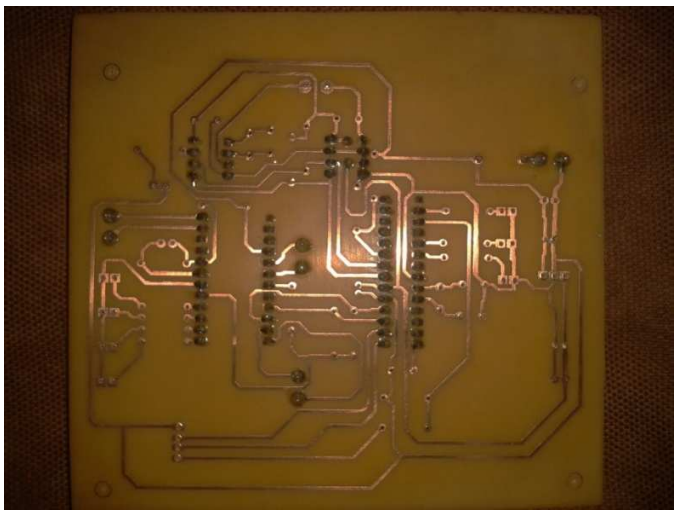
Fuente: Autores



Fuente: Autores

Una vez dibujadas las pistas se procede crear el circuito impreso sobre el cual se ensamblará los componentes del sistema.

Figura45. Construcción del circuito impreso

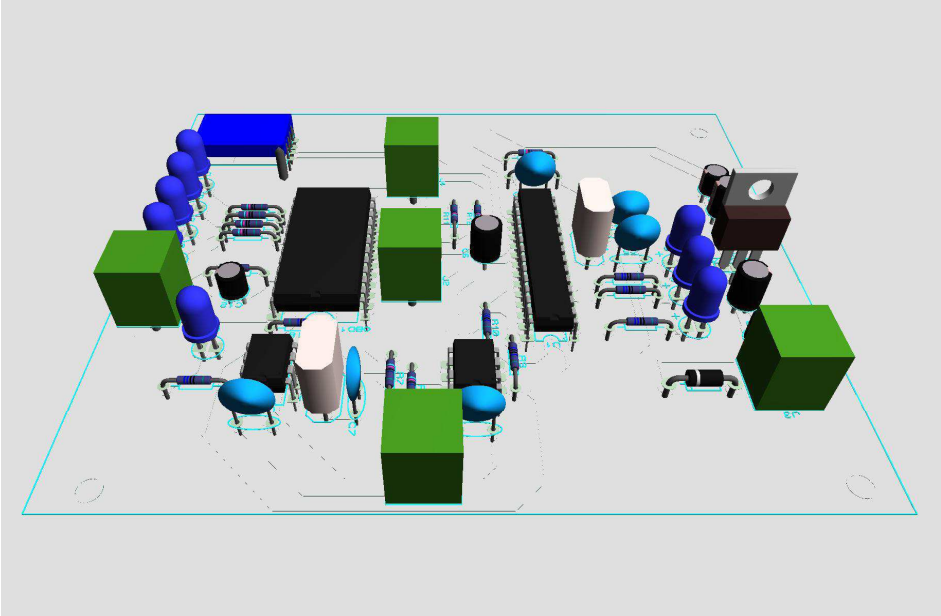


Fuente: Autores

5.2 Ensamblaje

Para determinar la ubicación de cada elemento se emplea la opción de visualización 3D que permite observar todos los componentes del circuito ubicados en su respectivo lugar.

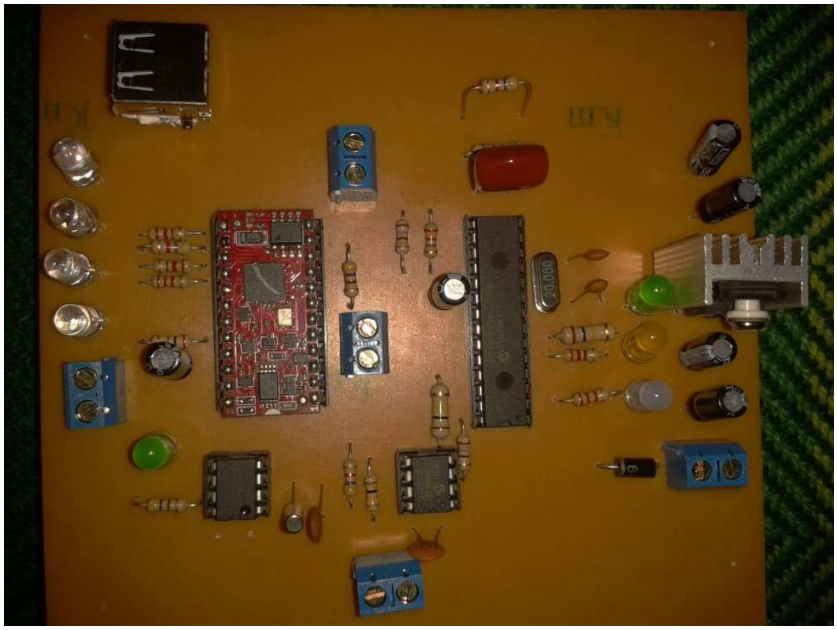
Figura46. Visualización previa del circuito



Fuente: Autores

A partir de esta imagen se facilita la ubicación de los elementos del circuito para poder soldarlos en el lugar correcto.

Figura47. Circuito terminado



Fuente: Autores

5.3 Costos

El costo total de proyecto se clasifican en.

- Costos directos
- Costos indirectos

5.3.1 *Costos directos*

Esto incluye todos los costos generados para la fabricación del hardware, los mismos que son:

- Materiales y equipos
- Costos por importación OBD 200 y cable Transporte

5.3.1.1 *Costo de materiales y equipos*

Las tablas muestran los costos totales en materiales y equipos.

Tabla 30. Costo de materiales y equipos

Cantidad	Denominación	Características	Costo unitario	Costo total (USD)
1	Micro OBD 200	Transceiver OBD a UART	69,95	69,95

1	Cable OBD2	Conexión OBD 200-Automovil	12,95	12,95
1	PIC 18F2550	Micro-controlador	12,00	12,00
1	24FC512	EEPROM I2C	3,75	3,75
1	Ds1307	Reloj tiempo real I2C	6,05	6,05
1	78M05	Regulador de voltaje 5 V	0,80	0,80
19	Resistencias ½ Watt	Diferentes valores	0,10	1,90
1	Cristal cuarzo	Oscilador 20 MHz	1,00	1,00
1	Conector USB 2.0	Hembra tipo A	1,15	1,15
1	Cable USB 2.0	Los dos extremos tipo A	3,00	3,00
1	Cristal cuarzo	Oscilador de32.768Hz	0,60	0,60
8	Diodos	LED	0,25	2,00
6	Condensador	Electrolítico	0,40	2,00
6	Condensador	Cerámico	0,15	0,90
5	Bloque terminal	2 puntos de conexión	0,25	1,25
1	Circuito impreso		35,00	35,00
1	Diodo	Rectificador 1N4007	0,20	0,20
Total				154,50

Fuente: Autores

5.3.1.2 *Costos por importación OBD 200 y cable*

Tabla 31. Costos por importación OBD 200 y cable

Denominación	**Características**	**Costo unitario**	**Costo total (USD)**
Envío	USA-ECUADOR	75,90	75,90
Impuesto 5%	Salida de capitales	16,24	16,24
Total			92,14

Fuente: Autores

5.3.1.3 *Costo directo total*

Tabla 32. Costo directo total

Descripción	**Valor (USD)**
Costo de materiales y equipos	154,50
Costos por importación OBD 200 y cable	92,14
Total	246,64

Fuente: Autores

5.3.2 *Costos indirectos*

Tabla 33. Costos indirectos

Denominación	**Cantidad**	**Costo (USD)**
Flete	10	100,00
Ingeniería	60	800,00
Imprevistos	--	95,00
Total		995,00

Fuente: Autores

5.3.3 *Costo total*

Tabla 34. Costo total

Descripción	**Valor (USD)**
Indirecto	995,00
Directo	246,64
Total	1241,64

Fuente: Autores

5.3.4 *Costo aproximado del producto.* Considerando que al producir en serie, los costos de importación de los elementos y el valor de los mismos se reducen considerablemente, hemos aproximado que se reduciría a 220,00 USD.

CAPÍTULO VI

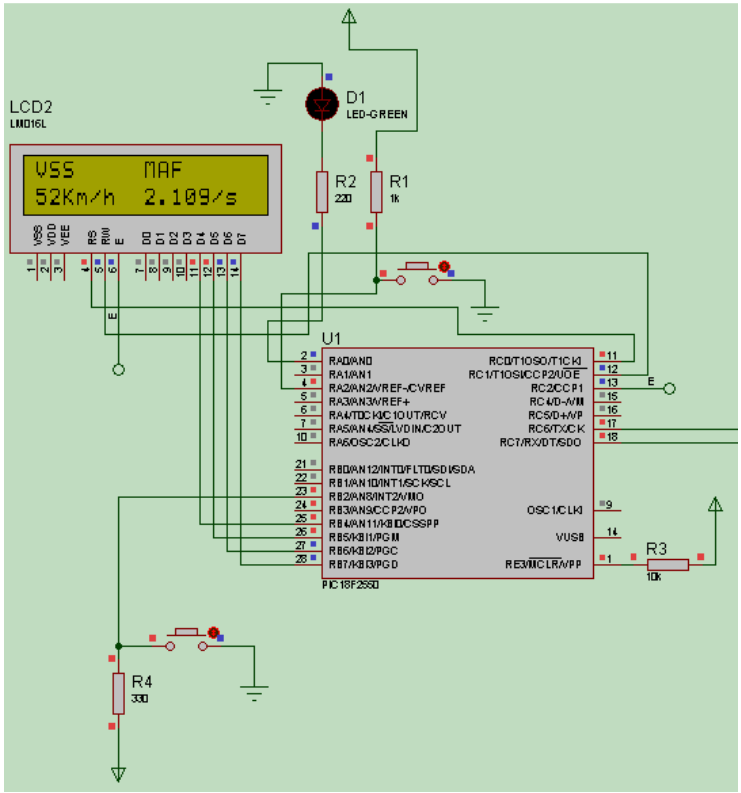
6. FASE EXPERIMENTAL

6.1 Simulación en Software

Para simular el funcionamiento del dispositivo se empleó la aplicación ISIS de Proteus con el objetivo de determinar el correcto desempeño del circuito sin tener la necesidad de ensamblarlo y de esta manera corregir posibles errores sin arriesgarnos a destruir ningún componente.

En primer lugar se determinó si el dispositivo era capaz de leer los valores provenientes de los sensores que son empleados para calcular los valores tanto de consumo de combustible como de recorrido realizado; estos sensores son: el MAF y el VSS.

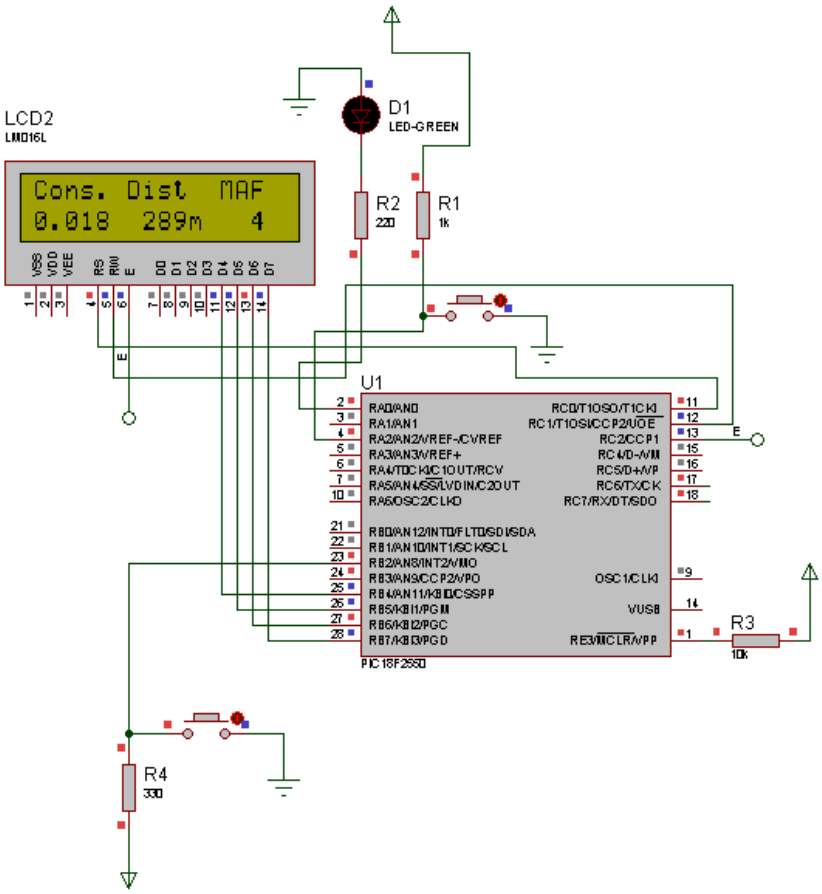
Figura48. Circuito simulado que verifica la lectura de los sensores



Fuente: Autores

Una vez comprobado que el dispositivo lee correctamente los valores entregados por los sensores iniciamos la simulación del funcionamiento final del aparato y así poder determinar si efectivamente está calculando los valores necesarios para el proyecto.

Figura49. Circuito simulado que verifica los cálculos



Fuente: Autores

6.2 Experimentación bajo parámetros reales

Una vez ensamblado el dispositivo procedemos a realizar las pruebas en una Toyota Stout IIprimero para comprobar que es capaz de leer los valores entregados por los sensores[21] obteniendo los siguientes resultados.

[21] Sensor MAF y VSS

6.2.1 *Sensor MAF*

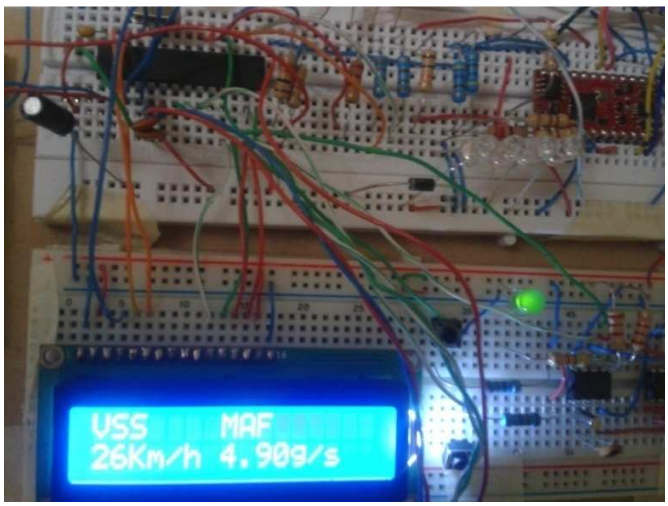
Motor en ralentí: 2 g/s de aire

Aceleración a fondo: 46 g/s de aire

Estos valores corresponden a los normales de operación para este vehículo.

6.2.2 *Sensor VSS.* Se comprobó que funciona correctamente debido a que la velocidad mostrada por el dispositivo corresponde a la velocidad que circulaba el vehículo en ese momento.

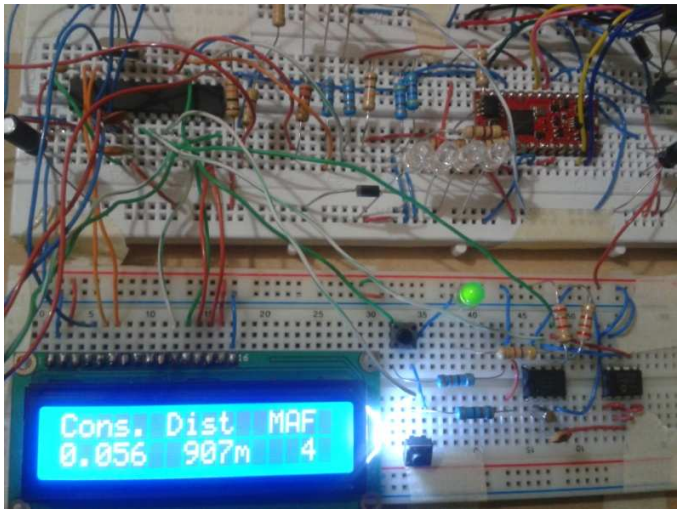
Figura50. Circuito real que verifica la lectura de los sensores



Fuente: Autores

Al verificar que se lee correctamente los valores entregados por los sensores se comprueba si el dispositivo realiza cálculos para obtener el consumo y el recorrido.

Figura51. Circuito real que verifica los cálculos



Fuente: Autores

Para poder determinar si el consumo mostrado es el real se realiza el siguiente procedimiento:

1. Llenar completamente el tanque de combustible
2. Recorrer una distancia de 35 Km
3. Leer el consumo mostrado por el dispositivo
4. Llenar nuevamente el tanque de combustible
5. Leer la cantidad de combustible que marca la bomba distribuidora de la estación de servicio
6. Comparar los valores de la bomba y el dispositivo
7. Repetir el procedimiento anterior

Valores obtenidos en el primer recorrido

Consumo mostrado por el dispositivo = 2,715 litros

Consumo mostrado por la estación de servicio = 0,673 Gal. (2,547 litros)

$$\%Error = \frac{2{,}715 - 2{,}547}{2{,}547} * 100\% = 6{,}6\%$$

Figura52.Resultados prueba 1

Fuente: Autores

Valores obtenidos en el segundo recorrido

Consumo mostrado por el dispositivo = 2,670 litros

Consumo mostrado por la estación de servicio = 0,663 Gal. (2,509 litros)

$$\%Error = \frac{2{,}670 - 2{,}509}{2{,}509} * 100\% = 6{,}4\%$$

Figura53.Resultados prueba 2

Fuente: Autores

6.2.3 *Análisis de resultados*

6.2.3.1 *Recorrido vs Consumo.*Entre todas las gráficas, esta es una de las más importantes ya que aquí se puede visualizar cualquier desorden en el consumo de combustible, debido a que en condiciones óptimas de funcionamiento esta debería tender a ser lineal.

Tabla 35. Datos obtenidos en la fase de prueba

Tiempo de func. (Minutos)	**Fecha y hora de encendido**	**Fecha y hora de apagado**	**Recorrido(Km)**	**Consumo (mL)**
38	30/08/2012 16:33	30/08/2012 17:11	30	2190
22	30/08/2012 17:35	30/08/2012 17:57	20	1460
12	31/08/2012 10:03	31/08/2012 10:15	5	365
27	31/08/2012 10:27	31/08/2012 10:54	23	1679
16	31/08/2012 14:05	31/08/2012 14:21	12	876
14	31/08/2012 16:37	31/08/2012 16:51	8	584
42	01/09/2012 11:03	01/09/2012 11:45	27	1971
19	01/09/2012 15:28	01/09/2012 15:47	18	1200
142	02/09/2012 09:43	02/09/2012 12:07	120	8195
8	02/09/2012 13:03	02/09/2012 13:11	3	219
15	03/09/2012 15:01	03/09/2012 15:16	10	800

Fuente: Autores

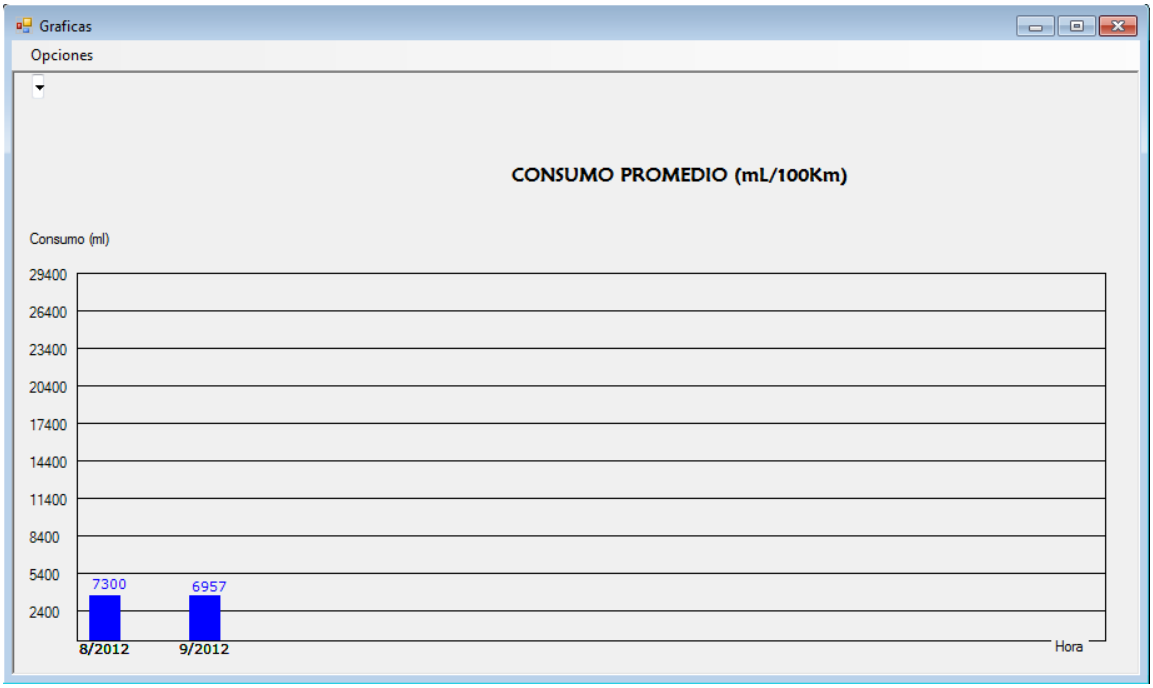
Figura54. Gráfica de consumo en función del recorrido

Fuente: Autores

Como se puede apreciar esta gráfica tiende a ser lineal lo que indica que el vehículo funciona correctamente, los puntos en los que la línea tiende a distorsionarse se deben a que en el recorrido realizado se encontró una pendiente y al momento de subirla el consumo se incrementa por el aumento de la carga del motor, en cambio al momento de bajar la misma pendiente el consumo tiende a disminuir ya que la aceleración no es necesaria.

6.2.3.2 *Consumo Mensual Promedio.* La gráfica muestra el consumo de combustible promedio mensual en relación con un recorrido de 100 km, esto nos permite identificar si existe o no diferencia entre los distintos meses.

Figura55. Gráfica de consumo promedio



Fuente: Autores

La diferencia que existe entre el consumo del mes de agosto con el de septiembre es de 343 ml, la cual está dentro de un rango moderado debido a que siempre va a existir una diferencia causada por las diferentes condiciones a las que está sujeta la precisión del dispositivo, tales como: temperatura, presión atmosférica, calidad del combustible, carga del motor, condiciones de la carretera, factores mecánicos, hábitos de manejo.

CAPÍTULO VII

7. CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

Ya que el sistema OBD2 esta normalizado en J1979 e ISO 15031-5 y en los últimos años ya se ha incorporado prácticamente todas las marcas de autos, este dispositivo se puede utilizar fácilmente en cualquier automóvil a gasolina que cumpla con las normas internacionales de control mínimo de emisiones tales como las normas EURO4 y EURO5 entre otras.

Con el uso de este dispositivo, el usuario podrá no solo adaptarse a mejores condiciones de manejo, sino como también verificar si la estación de servicio le está suministrando la cantidad indicada de combustible.

La aplicación que complementa al dispositivo está diseñada de tal manera que su utilización permita visualizar diferentes tipos de gráficas, donde el usuario puede observar consumo en función de recorrido. Ya que se podría encontrar un mismo recorrido con diferentes valores de consumo, porque este estará en función de la carga del motor, condición de carretera, factores mecánicos, hábitos de manejo, etc.

Usar los datos de los sensores del vehículo, accesibles mediante OBD2, vuelve al dispositivo portátil, fácil y rápido de conectar, por consiguiente, una reducción en los costos de fabricación.

Al ser construido con dimensiones pequeñas, el dispositivo es fácil de colocar en cualquier lugar, dando una presencia casi nula.

Con el desarrollo de este proyecto se puede observar que los valores entregados por los sensores; no solo pueden ser usados a la hora de detectar una falla en el sistema sino que también para el desarrollo de accesorios.

Tanto la relación de aire-combustible como la densidad del combustible, usada en el prototipo es la ideal, las mismas que son de 14,7-1 y $740\,\frac{gramos}{Litro}$ respectivamente. Por lo que, el error

obtenido estará en función de estos valores, los mismos que a su vez dependen de temperatura, presión atmosférica y calidad del combustible.

7.2 Recomendaciones

Verificar que los sensores MAF y VSS, usados como referencia por el dispositivo estén en perfecto estado, ya sea con la utilización de un escáner o algún otro método de diagnóstico.

Antes de empezar usar el dispositivo verificar que el vehículo pertenezca a OBD2 y que soporte los dos PIDs, pertenecientes a los sensores que el dispositivo utiliza.

El dispositivo tiene una memoria suficiente como para almacenar 2978 puestas en marcha del motor, por lo que, con un promedio de 10 encendidas diarias, se recomienda descargar los datos cada seis meses aproximadamente para evitar que se pierdan datos.

No olvide configurar la hora y fecha del dispositivo en su primera conexión. Para que se tome en cuenta estos parámetros en el control que este realiza.

Siempre que se desee desconectar el dispositivo del PC, sírvase extraer de forma segura; con el procedimiento común usado para expulsar dispositivos USB.

REFERENCIAS BIBLIOGRÁFICAS

[1] SAE, International. SurfaceVehicleStandard SAE 1979. REV APR2002. USA, 2002.Pág. 5-23.

[2]GARCIA BREIJO, Eduardo. Compilador C CCS y simulador proteus para micro controladores PIC. 1ra Ed. México, 2008. Pág. 251.

[3] AXELSON, Jan. USB complete.4ta Ed.USA:Madison, 2009. Pág. 12.

[4] AXELSON, Jan. USB complete. 4ta Ed. USA: Madison, 2009. Pág.71-75.

[5] AXELSON, Jan. USB complete. 4ta Ed. USA: Madison, 2009. Pág.147-156.

[6] AXELSON, Jan. USB complete. 4ta Ed. USA: Madison, 2009. Pág. 165.

[7] AXELSON, Jan. USB complete. 4ta Ed. USA: Madison, 2009. Pág.217-220.

[8] AXELSON, Jan. USB complete. 4ta Ed. USA: Madison, 2009. Pág.345-371.

[9] OBD Solutions. Manual reference and programming manual.2da Ed. USA, 2002. Pág. 9-14

[10] SHARP, Jhon. Visual C# paso a paso. 1ra Ed. España. Madrid, 2008. Pág. 29-52

BIBLIOGRAFÍA

AXELSON, Jan. USB complete. 4ta Ed. USA: Madison, 2009.

BEER, Ferdinand;JOHNSTON, Russell y CLAUSEN, William.Mecánica vectorial para ingenieros Dinámica. 8va edición. México: McGraw Hill, 2007.

CHAND, Mahesh. Graphics Programing with GDI+.1ra Ed. Massachusetts: Addison-Wesley

Family reference and programming manual.Phoenix Solutions.

HUERTA CEREZUELO, ANTONIO. Métodos numéricos Introducción, aplicaciones y propagación. 1ra edición. España: UPC. pp 183-187.

OBD Solutions. Manual reference and programing manual. 2da Ed. USA, 2002.

SAE, International. SurfaceVehicle Standard SAE 1979. REV APR2002. USA, 2002.

SHARP, Jhon. Visual C# paso a paso. 1ra Ed. España. Madrid, 2008.