UNIVERSIDAD NACIONAL
DE MAR DEL PLATA

Trabajo Final

Adquisidor de datos de conducción de vehículos para sistema de *Eco-driving*

Juan Manuel López

Director: Ing. Juan Carlos Bonadero
Co-director: Dr. Ing. Alejandro José Uriz

Facultad de Ingeniería
Universidad Nacional de Mar del Plata

2015

Resumen

En este informe se detalla el desarrollo de un sistema de *Eco-driving*[1] para su utilización en distintos tipos de vehículos, correspondiente al trabajo final de la carrera de Ingeniería en Electrónica. Para esto, se implementó un sistema formado por un microcontrolador, un módulo Bluetooth, GPS, acelerómetro y una interfaz OBD-2-Bluetooth. Éste le provee al conductor del vehículo una serie de indicaciones tanto en tiempo real, como para su posterior análisis, de manera que le permita modificar su manera de conducir y reducir el consumo de combustible. De esta manera, el conductor tendrá disponible la información necesaria para poner en práctica los conocimientos adquiridos en un curso de *Eco-driving*.

El microcontrolador utilizado es un Coldfire V1 MCF51JM128 de Freescale, debido a la facilidad para programar los distintos periféricos y la capacidad de procesamiento que posee al tratarse de un CPU de 32 bits.

Para realizar la comunicación entre el puerto de diagnóstico OBD-2[2] y el microcontrolador, se utilizó un módulo Bluetoothmaestro HC05 y un adaptador OBD-2-Bluetooth. De esta manera, el microcontrolador puede obtener información en tiempo real sobre el estado del auto, como puede ser la velocidad, cantidad de combustible en el tanque, y más.

El dispositivo cuenta con un display LCD que le proporciona realimentación en tiempo real al conductor, como así también la posibilidad de almacenar los distintos recorridos realizados junto con la velocidad, revoluciones por minutos (RPM) del motor y el nivel de combustible en cada instante, para su posterior análisis mediante la aplicación Google Earth.

El trabajo se organiza en tres capítulos. En el primero se describe la problemática que lleva al desarrollo del *Eco-driving*, se describe un sistema electrónico genérico para la adquisición de datos de manejo, se presentan las distintas alternativas comerciales disponibles y por último se realiza una introducción al sistema propuesto. En el capítulo 2 se analiza en detalle el sistema desarrollado, tanto el hardware como el software. Para finalizar, en el capítulo 3 se presenta el dispositivo ya implementado y las pruebas realizadas en situaciones de manejo reales.

1*Eco-driving: Es el nombre que se le da a un estilo de manejo eficiente, cuyo objetivo es el de disminuir la emisión de gases contaminantes y ahorrar combustible.*

2*OBD-2: Abreviatura de "On Board Diagnostics 2", estándar introducido a mediados de los años 90 que especifica las características del puerto de diagnóstico del vehículo, como tipo de conector, protocolos utilizados y formato del mensaje.*

Índice

1. Introducción. 1
1.1. Definición 4
1.2. Estructura de un sistema de Eco-driving 8
1.3. Soluciones disponibles 9
1.4. Solución propuesta 12
1.5. Conclusión 15
2. Descripción del dispositivo. 16
2.1. Diagrama en bloques 16
2.1.1. Adaptador OBD-2 -> Bluetooth 16
2.1.2. Módulo Bluetooth Maestro HC05 19
2.1.3. Sistema de Posicionamiento Global (GPS). 22
2.1.4. Memoria SD 25
2.1.5. Acelerómetro. 27
2.1.6. Microcontrolador 28
2.1.7. Interfaz de usuario 31
2.2. Microcontrolador 33
2.2.1. Serial Communication Interface (S08SCIV4) 34
2.2.2. Timer/PWM Module 37
2.2.3. Serial Peripheral Interface (SPI) 40
2.2.4. Multipurpose Clock Generator (S08MCGV3) 44
2.2.5. Inter-Integrated Circuit (S08IICV2) 47
2.3. Descripción del programa 51
2.3.1. Configuración 52
2.3.2. Programa principal 52
2.3.3. Rutinas de interrupción. 54
3. Diseño y pruebas. 58
3.1. Circuito esquemático 58
3.1.1. Microcontrolador 58
3.1.2. Periféricos externos 59
3.1.3. Fuente de alimentación 61
3.2. Diseño del circuito impreso 61
3.3. Dispositivo terminado 65
3.4. Desarrollo del Software 66
3.4.1. Módulo GPS 66
3.4.2. Tarjeta SD 67
3.4.3. Comunicación OBD-2-Bluetooth. 67
3.4.4. Acelerómetro 67
3.4.5. Aplicación final 68
3.5. Pruebas realizadas 68
4. Conclusiones. 72

Capítulo 1. Introducción

Actualmente existe un enfoque en disminuir la emisión de gases contaminantes como el dióxido de carbono (CO_2) y reducir el consumo de combustible. Estos esfuerzos están centrados mayoritariamente en el sector industrial. Sin embargo, estudios hechos en la Unión Europea (UE) muestran que el sector automotor de transporte es uno de los pocos que presenta una suba considerable en la emisión de gases contaminantes respecto a estudios anteriores. (un 20,5% más altas que en 1990) **[1]**. También se concluyó que el 15% de las emisiones en la UE provienen de vehículos ligeros como automóviles, y si bien los vehículos son cada vez más eficientes, los conductores deben modificar su manera de conducir para disminuir el consumo de combustible.

Son distintos factores los que influyen en mayor o en menor medida en el consumo final del automóvil. A continuación se detallan algunos:

1. **Mantenimiento del vehículo:** Circular con los neumáticos desinflados produce un aumento en la resistencia a la rodadura, debido que al tener baja presión, se deformarán generándose así una resistencia mayor y se desperdiciará mayor energía. En el caso de una presión alta, la resistencia a la rodadura disminuirá, pero el agarre también lo hará, lo que puede ser peligroso.

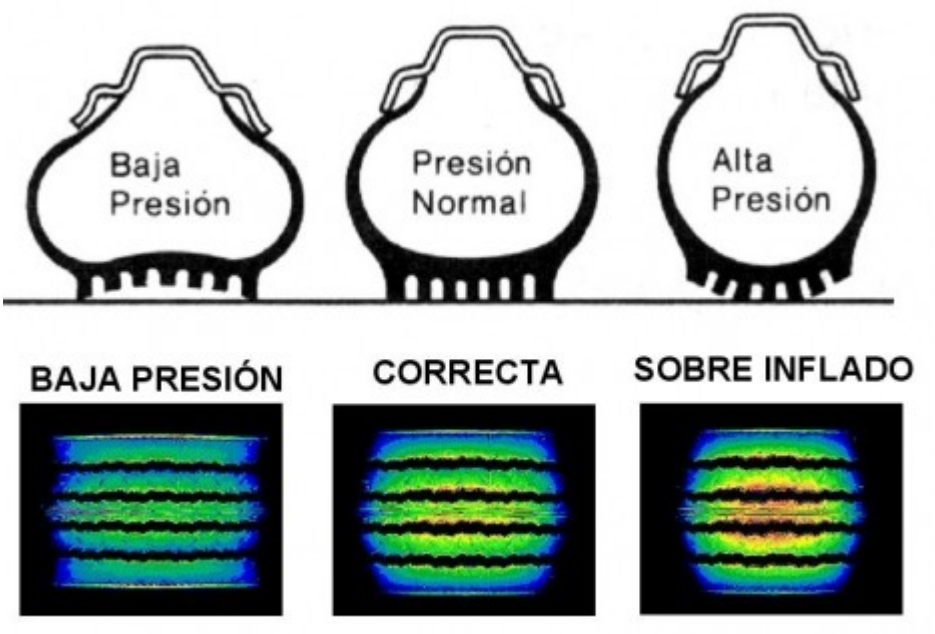


Figura 1 - Se muestra tres neumáticos con presión baja, normal y alta [2]

2. **Peso extra en el vehículo:** Cualquier peso extra sobre el vehículo conlleva a un aumento en el consumo de combustible. Un peso extra de 45 Kg produce aproximadamente un aumento del 1% en el consumo **[3]**.

3. **Selección del cambio:** La velocidad del cigüeñal de un motor se mide en revoluciones por minuto (RPM). En un vehículo con transmisión manual, el conductor puede disminuir o aumentar la velocidad del motor seleccionando distintos cambios. El cambio utilizado y las RPM influyen en el consumo de combustible, ya que si se conduce en un cambio bajo, las RPM serán altas, lo que quiere decir que el motor producirá un torque mayor y el consumo aumentará.

4. **Aceleración y frenado:**Variar la velocidad produce pérdidas de energía cuando se desacelera y luego se vuelve a acelerar. Si el conductor conduce a alta velocidad no podrá anticiparse al tráfico y a las condiciones de la carretera, de esta manera normalmente se encontrará con situaciones en las que deberá frenar de manera brusca debido al poco tiempo que tiene para reaccionar, para luego volver a acelerar. Esta energía empleada en acelerar y desacelerar continuamente, no es recuperada y se verá reflejado en un aumento del consumo de combustible.

1.1. Definición

Eco-driving es un término utilizado para describir una forma de conducir eficiente, de manera de reducir el consumo de combustible y la emisión de gases contaminantes como el CO_2.

Existen distintos cursos cuyo objetivo es el de proporcionar información al conductor y establecer las reglas básicas de conducción eficiente, de modo de que pueda modificar su manera de conducir, para luego comparar los resultados antes y después del curso. Otro método consiste en brindar información al conductor en tiempo real, o para su posterior análisis, mediante un dispositivo electrónico que obtenga datos del puerto de diagnóstico del automóvil y de módulos GPS y acelerómetro, para así de esta manera caracterizar el estilo de manejo del usuario y que esto le permita modificar su conducta de acuerdo a las indicaciones y los datos adquiridos durante el viaje.

En la actualidad, este tipo de sistemas se encuentran disponibles en vehículos de alta gama, pero que representan un porcentaje muy bajo del total de autos que hay en circulación. También existen soluciones externas para ser incorporadas en los vehículos que no poseen este tipo de prestaciones.

El *Eco-driving* establece algunas reglas sencillas para cumplir con el objetivo. Se recomienda anticiparse al flujo del tráfico, de modo de no producir cambios bruscos en la velocidad. A velocidades medias y bajas, el conductor tiene un tiempo mayor para reaccionar al tráfico, por lo que puede frenar de manera suave, evitando los cambios bruscos que producen un mayor consumo de combustible y además, en el caso de manejar en la ciudad, realizar aceleraciones bruscas con picos de alta velocidad no garantiza que se llegará más rápido a destino, ya que este tiempo depende de factores externos como la cantidad de tráfico, los semáforos, etc. Otra recomendación es evitar permanecer con el vehículo encendido pero sin movimiento. Se estima que manteniendo al automóvil en ese estado se desperdician alrededor de 1 l/h de combustible**[4]**. También se recomienda mantener una velocidad constante y pasar al siguiente cambio alrededor de las 2500 RPM en autos nafteros y las 2000 RPM en autos diesel. Como se mencionó con anterioridad, se recomienda revisar la presión de los neumáticos frecuentemente, al menos una vez al mes ya que una presión baja es un riesgo para la seguridad y desperdicia combustible. También es necesario evitar cualquier peso extra que pueda llevar el vehículo, como así también el uso de accesorios que consumen energía del automóvil como el aire acondicionado.

Un estudio llevado a cabo por la Universidad de Michigan muestra como el estilo de manejo, un correcto mantenimiento del vehículo y la selección adecuada de rutas pueden llevar a una reducción en el consumo de combustible de hasta un 45% **[5]**. En la Error: no se encontró el origen de la referencia se presentan los resultados de dicho estudio. En la misma se detalla como varía el consumo de combustible en Millas por Galón (mpg) en función de distintos factores, como un estilo de manejo agresivo, uso de aire acondicionado, entre otros.

Factor (effect on performance)	**Fuel economy (mpg)**
Nominal performance	36.0
Aggressive driving a) (25% drop)	27.0
Driving at excessively high speeds b) (6% drop)	25.4
Route selection (road type, grade, and congestion) c) (6% drop)	23.9
Out-of-tune engine d) (4% drop)	22.9
Tires with increased rolling resistance e) (4% drop)	22.0
Using air conditioner f) (4% drop)	21.1
Excessive idling g) (2% drop)	20.7
Extra weight h) (1.5% drop)	20.4
Improper oil (1.5% drop)	20.1
Under-inflated tires i) (1.5% drop)	19.8

Tabla 1 - Consumo de combustible

a) Not using cruise control included.
b) Driving at very high speeds on 20% of the total distances driven.
c) Two possible routes (with different road types, grade profiles, and/or levels of congestion) are available 20% of the total distance driven.
d) Faulty oxygen sensor (infrequent in relatively new vehicles) could result in a fuel- economy drop of 40%.
e) Replacement tires with 25% higher rolling resistance than originally equipped tires.
f) Used during 25% of the total distance driven. At very high speeds the windows are up.
g) Turning off the engine during two 1-minute idle periods per each 10 miles.
h) Extra 100 pounds of cargo.
i) Underinflation of all four tires by 5 psi.

Fuente: University of Michigan Transportation Research Institute (UMTRI).

A continuación se presentan tres gráficos con datos recolectados en distintas situaciones de manejo. El estudio fue publicado en el International Journal of Computer Applications Volume 98**[6]**.

En el Gráfico 1se muestran valores de aceleración en $\frac{m}{s^2}\frac{m}{s^2}$ para los ejes X (orientado hacia el lado derecho del vehículo), Y (orientado en la dirección de avance) y Z (orientado hacia el techo) en condiciones normales de manejo. Se puede observar como los valores en los ejes X e Y no superan los 0,8 $\frac{m}{s^2}\frac{m}{s^2}$ (0,081 g).

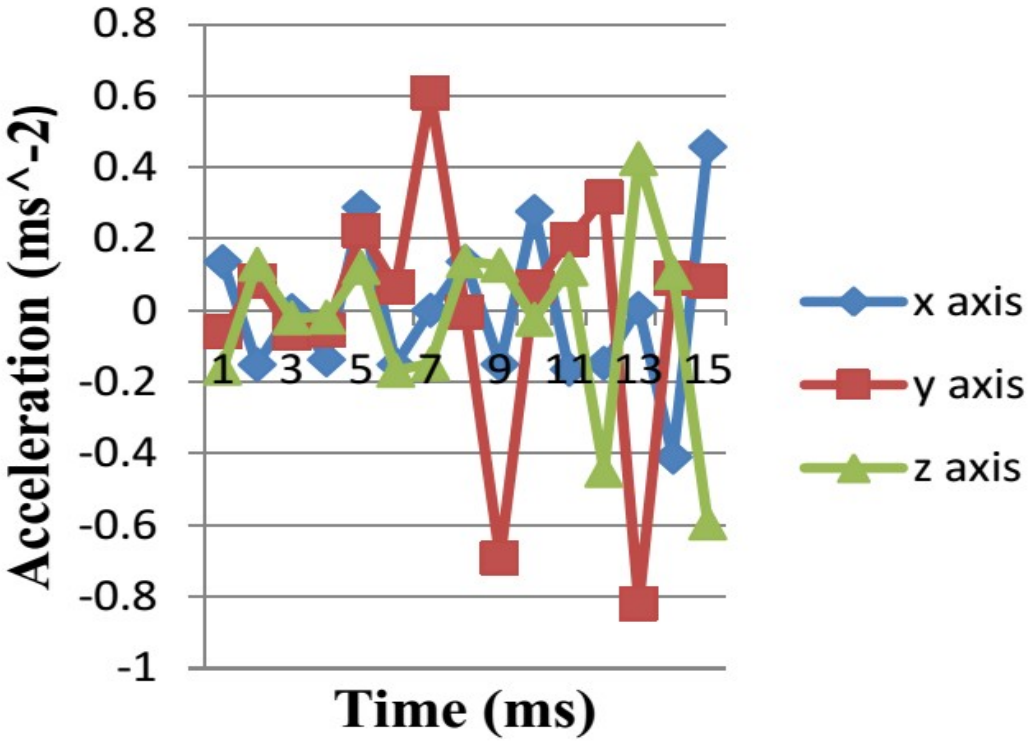


Gráfico 1 - Valores normales de aceleración [6]

se puede observar un caso en el que se aplican los frenos de forma repentina. En el eje Y se produce un pico de -3,5 $\frac{m}{s^2}\frac{m}{s^2}$ (-0,36 g).

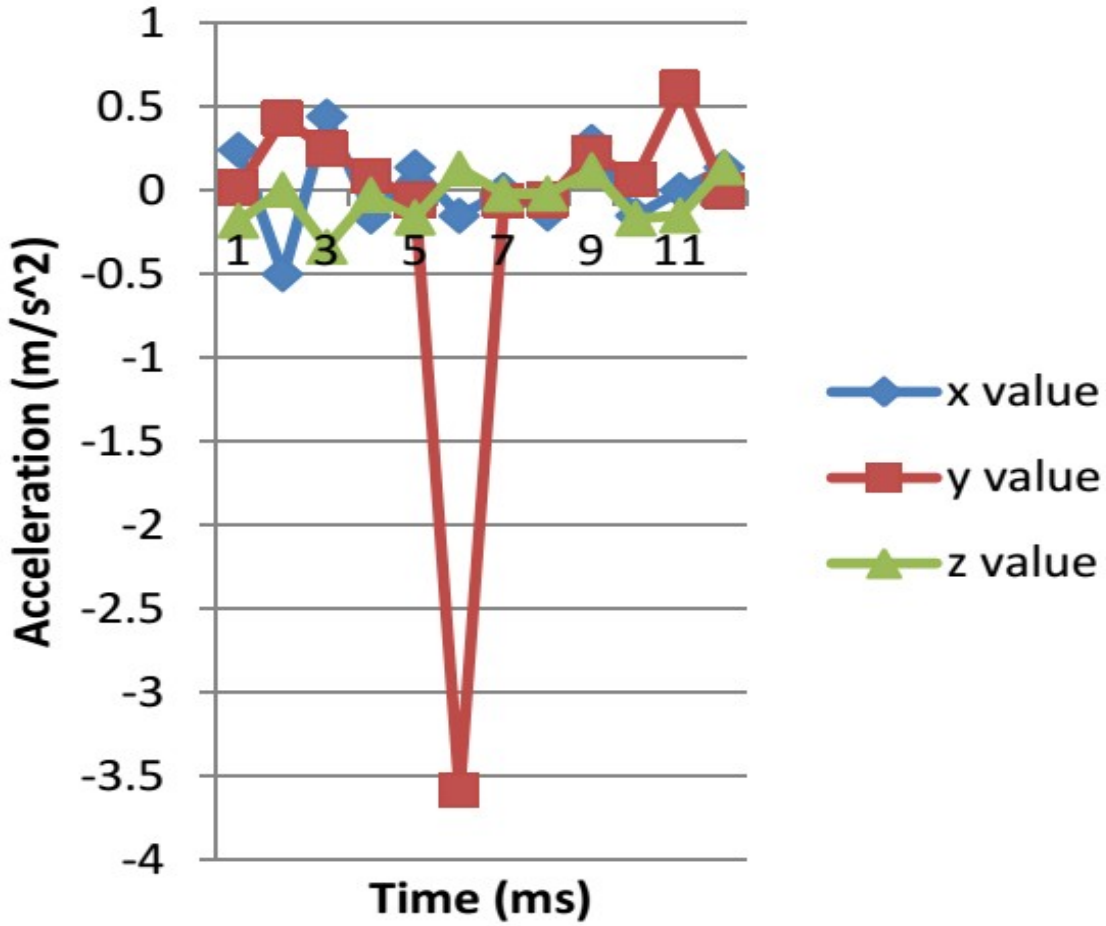


Gráfico 2–Frenado brusco [6]

ElGráfico 3muestra un caso en el que se dobla en una esquina.Se produce un pico negativo de aceleración en el eje X de -1,5$\frac{m}{s^2}\frac{m}{s^2}$ (-0,15 g).

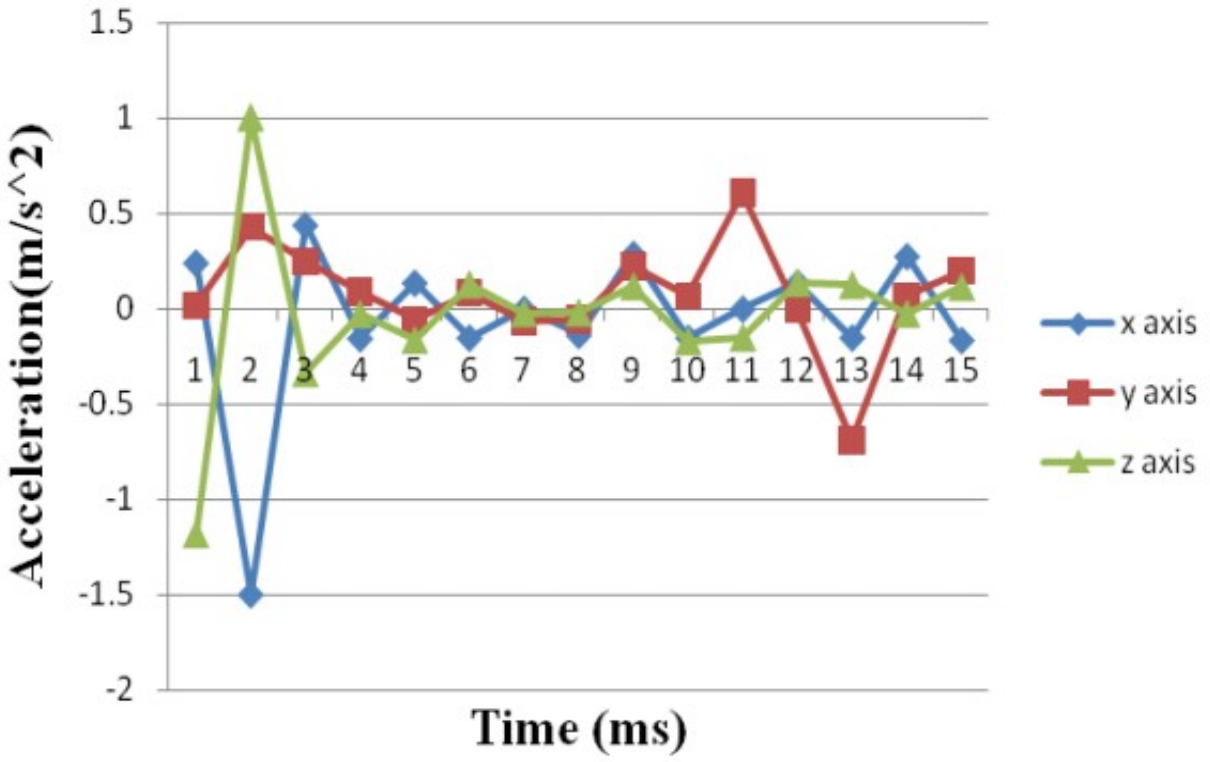


Gráfico 3 - Doblando en una esquina [6]

1.2. Estructura de un sistema de *Eco-driving*

Para monitorear el estilo de conducción de una persona es posible obtener información a través del puerto de diagnóstico OBD-2 del vehículo, como así también de un GPS y un acelerómetro.

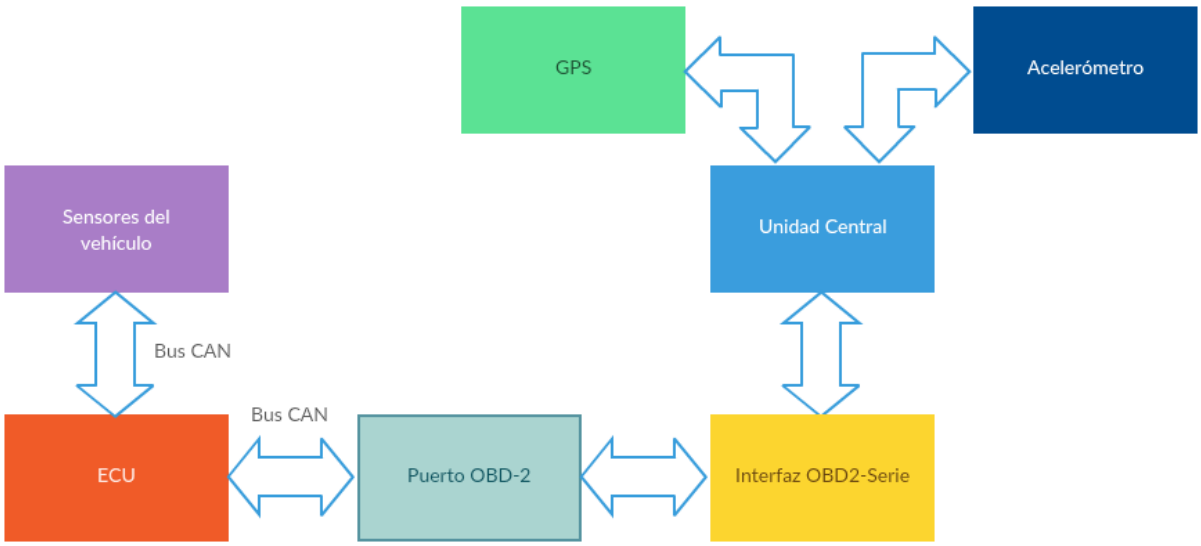


Figura 2 - Diagrama en bloques de un sistema de *Eco-driving*

Mediante la utilización de un GPS es posible obtener información detallada sobre la ubicación, velocidad y hora en la que sucedió un determinado evento. El acelerómetro permite monitorear cambios bruscos en la velocidad para determinar si el usuario posee un estilo de conducción agresivo o moderado. Con estos dos módulos es posible desarrollar un dispositivo que pueda ser instalado en cualquier automóvil, pero mediante la utilización del puerto OBD-2 (cuando esté disponible), se pueden obtener parámetros extras sobre el vehículo, como las RPM, nivel del tanque de combustible, la posición del acelerador, nivel de carga del motor, entre otros, que son muy útiles a la hora de analizar los resultados obtenidos y comprobar si se produjeron mejoras en el estilo de conducción. La interfaz entre el puerto OBD-2 y la unidad central puede realizarse de manera cableada (USB o serie) o mediante un enlace inalámbrico Bluetooth. Las interfaces más utilizadas están implementadas con en el circuito integrado ELM327 basado en un microcontrolador PIC18F2480 de Microchip, cuya función es la de proveer una capa de abstracción para los protocolos de bajo nivel utilizados en el bus CAN, y permitir un acceso rápido y sencillo a la información del bus por parte del usuario.

Toda la información es procesada por una unidad central para producir advertencias en tiempo real y almacenar los datos para su posterior análisis.

1.3. Soluciones disponibles

Existen en el mercado distintas soluciones orientadas a mejorar el estilo de manejo de las personas. Las mismas se dividen en dos grandes grupos: las que utilizan información provista por el puerto de diagnóstico del automóvil para calcular en

tiempo real el consumo de combustible, y las que utilizan sensores externos como acelerómetros y módulos GPS para detectar conductas de manejo inapropiadas.

Las primeras se basan en la información de velocidad y del sensor MAF (*Mass Air Flow)*, cuya función es medir el flujo de aire que ingresa al motor en gramos por segundo. Esta información es necesaria para que la ECU pueda entregar la cantidad correcta de combustible al motor. Utilizando estos datos es posible calcular la cantidad de combustible consumida por ahora y luego la cantidad de combustible consumida por distancia recorrida (asumiendo la condición ideal de 14,7 gramos de aire por gramo de combustible). Los pasos para calcular las millas recorridas por galón de combustible (MPG) son los siguientes:

1. Dividir el valor dado por el sensor MAF por 14,7. De esta manera obtendremos los gramos de combustible por segundo.
2. Dividir el resultado por 454 para obtener las libras de combustible por segundo.
3. Dividir el resultado por 6,701 galones de combustible por segundo.
4. Multiplicar el resultado por 3600 para obtener galones por hora (GPH).

El valor de la velocidad se obtiene en Km/h, para convertirlo a millas por hora (MPH), se lo multiplica por 0,621317. Finalmente para calcular el MPG se divide MPH por GPH.

El inconveniente principal de este método es que se basa en una condición ideal para realizar los cálculos. Además, no todos los vehículos poseen un sensor MAF accesible a través del puerto OBD-2.

Uno de los dispositivos más populares es el *Scan Gauge* desarrollado por *Linear-Logic* (Figura 3)**[7]**. El mismo utiliza la información provista por el puerto OBD-2 para calcular el consumo de combustible en MPG (Millas por galón) con el método descripto anteriormente. Tiene un sistema basado en viajes, que permite llevar cuenta del consumo de combustible y la distancia recorrida en el viaje actual y en el anterior, de manera de poder realizar comparaciones y analizar si los hábitos de manejo mejoraron. También cumple la función de escáner OBD-2, permitiendo al usuario borrar códigos de error. Este dispositivo no cuenta con módulo GPS ni acelerómetro, por lo que no es de utilidad para vehículos que no poseen puerto de diagnóstico.

Figura 3 - ScanGauge

Otro producto es el *UltraGauge*[8], que posee similares prestaciones, pero a un precio más económico.

Las aplicaciones basadas en acelerómetros utilizan la información de aceleración provista por los mismos para informar al usuario si está conduciendo de manera inapropiada. En una situación de manejo normal, los valores de aceleración no deberían sufrir cambios bruscos, ya que un estilo de conducción agresivo lleva a un mayor consumo de combustible. De esta manera, a través de alertas en tiempo real o mediante el posterior análisis de los datos almacenados, el conductor puede ir corrigiendo su estilo de manejo.

Si bien con éste método no se obtiene en tiempo real información sobre el consumo del vehículo, tiene la ventaja de que es posible instalarlo en cualquier automóvil sin puerto de diagnóstico OBD-2.

Hay una gran cantidad de aplicaciones para dispositivos Android e iOS que se basan en este método de monitoreo. La aplicación STR Eco-driving[9] registra la velocidad y los cambios de aceleración bruscos, para asignar un puntaje de manejo al finalizar el viaje.

Figura 4 - STR Eco-driving

Camélys Driving Assistance 2.0 **[10]** utiliza un sistema de puntuación basado en aceleración, frenado y velocidad. Toda la información del viaje es almacenada en su página web, desde la cual es posible exportarla en formato Excel.

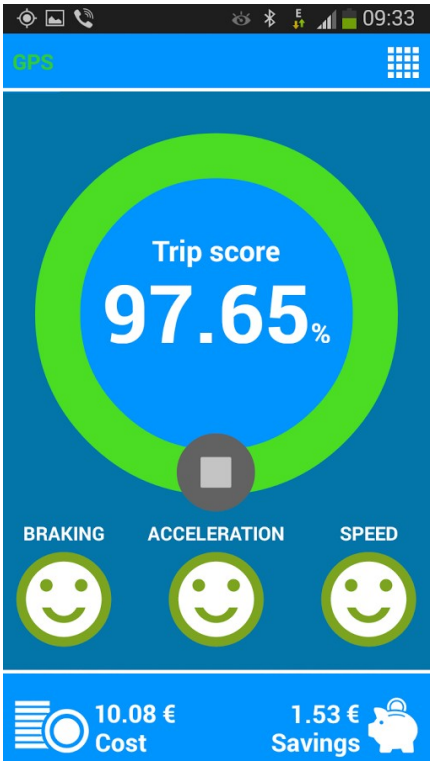


Figura 5 - Camélys Driving Assistance 2.0

1.4. Solución propuesta

Como establecen las reglas del *Eco-driving*, para conducir de manera eficiente se recomienda no producir cambios bruscos en la aceleración del vehículo, como así también mantener una velocidad constante a bajas RPM. El valor recomendado para avanzar al siguiente cambio se encuentra alrededor de las 2000 RPM dependiendo del automóvil. Tampoco se recomienda mantener el motor encendido si el auto va a estar estático por más de 1 minuto.

En la Figura 6 se muestra el sistema propuesto que será desarrollado en el Capítulo 2. El mismo consta de un microcontrolador, un acelerómetro, un módulo GPS, un módulo Bluetooth, un adaptador OBD-2 – Bluetooth y una interfaz de usuario.

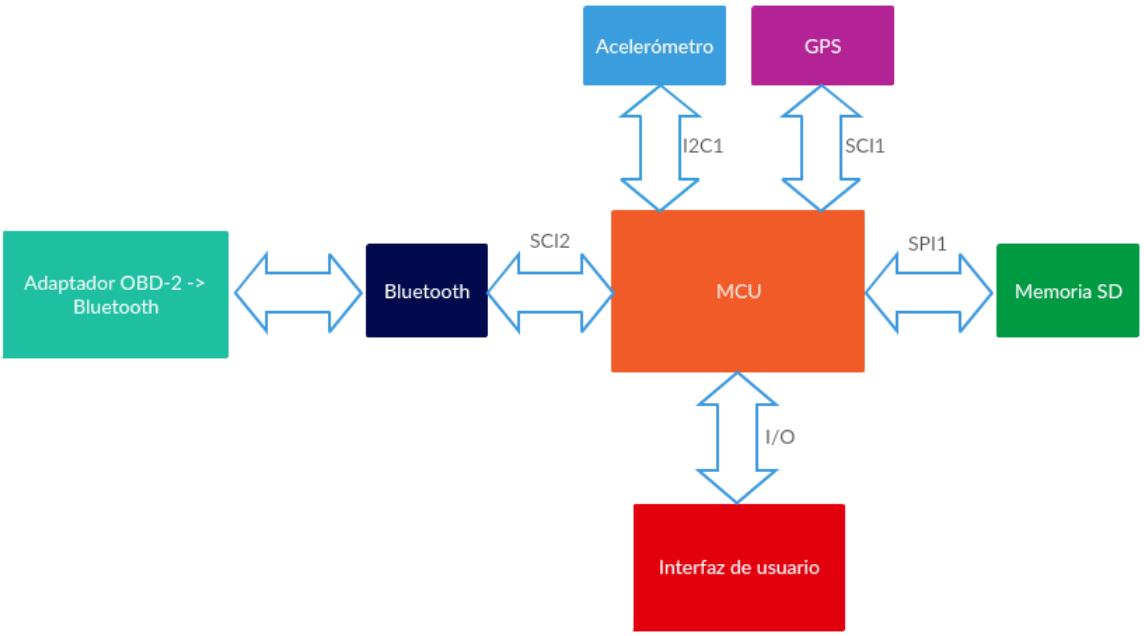


Figura 6 - Diagrama en bloques del sistema propuesto

La aplicación desarrollada utiliza la información provista por el acelerómetro para producir advertencias sonoras al conductor en caso de que se superen los umbrales predefinidos de aceleración tanto en el eje X como en el eje Y. La información del eje Z es utilizada para evitar que se disparen falsas advertencias en el caso de que el auto se encuentre en una ruta empinada, ya que el acelerómetro entregaría un valor de aceleración correspondiente con la inclinación de la ruta, que puede ser interpretado por el microcontrolador como una aceleración excesiva, cuando ese no es el caso. Así, para que una aceleración en el eje X o en el eje Y sea considerada peligrosa, la aceleración en el eje Z debe ser cercana a 1 G, valor que se corresponde con una situación de manejo sobre una ruta con poca pendiente.

Para indicar el momento adecuado para pasar al siguiente cambio, la aplicación se basa en la información provista por el puerto de diagnóstico del automóvil. El conductor recibe una indicación luminosa mediante un LED. A bajas RPM, el LED se encuentra de color verde, pero a medida que las RPM se acercan a 2000 el color cambia a naranja, para finalmente cambiar a rojo si se supera dicho valor.

Cada vez que el auto es encendido y permanece sin movimiento por más de 80 segundos, se produce una alerta sonora y a través del display LCD para recomendarle al conductor que apague el motor en caso de que vaya a permanecer en ese estado por un período de tiempo largo.

La interfaz de usuario es accesible mediante dos botones que permiten seleccionar las distintas opciones. El usuario puede elegir entre distintos modos, los cuales establecen umbrales de aceleración diferentes. También puede iniciar el almacenamiento en la tarjeta SD y detenerlo cuando sea conveniente.

Toda la información de manejo es almacenada en la tarjeta de memoria en un archivo de texto separado por comas para su posterior análisis por el usuario. Mediante la aplicación Google Earth (Figura 8) es posible visualizar el recorrido realizado, y los parámetros almacenados en cada punto (hora, latitud, longitud,

aceleración en los ejes X, Y y Z, velocidad, RPM, nivel del tanque de combustible, tiempo de viaje y tiempo de viaje con el auto en movimiento). De esta manera, el conductor puede comparar los distintos recorridos realizados y analizar si sus hábitos de conducción presentan mejoras.

Para realizar la conversión del archivo de texto al formato .kml de Google Earth**[11]**, se utiliza la web www.gpsvisualizer.com (Figura 7)**[12]**. La misma permite representar los datos almacenados en cada punto para ser visualizados por el usuario (Figura 8).

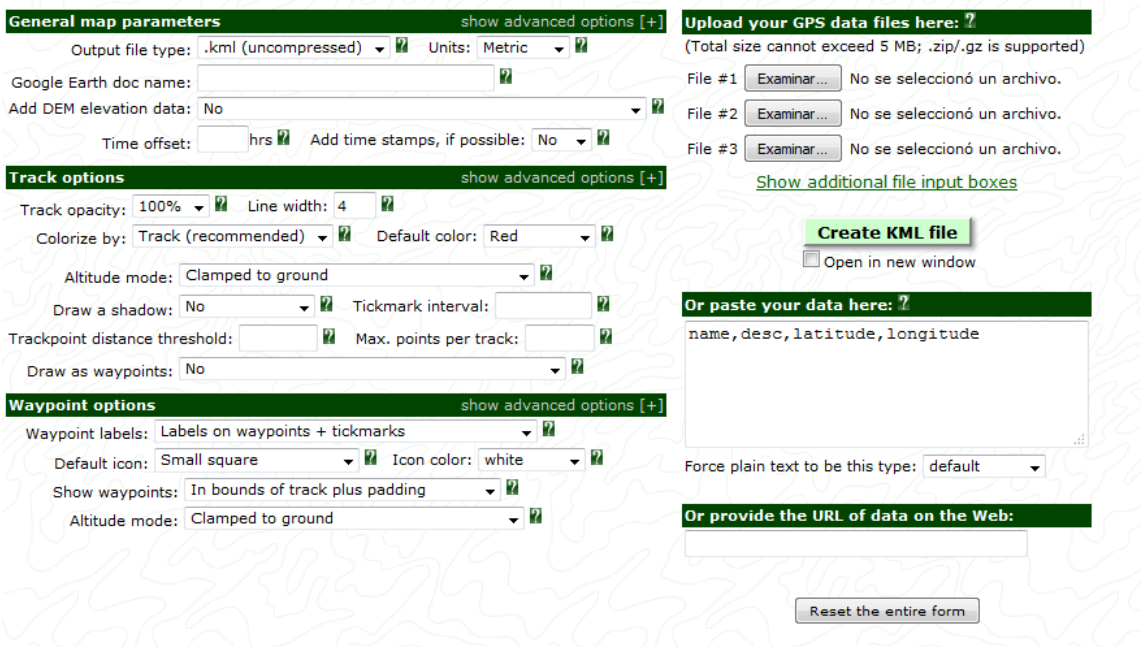


Figura 7 - www.gpsvisualizer.com

Figura 8 - Visualización del recorrido del vehículo en Google Earth junto con los datos adquiridos en cada punto

1.5. Conclusión

En este capítulo se realizó una introducción a la problemática que llevó al desarrollo del *Eco-driving*, como así también los distintos equipos existentes en el mercado que permiten corregir la manera de conducir de las personas. Por último, se hizo una descripción del sistema propuesto.

En el próximo capítulo se describirán en detalle los distintos bloques que componen el sistema desarrollado, como así también el software implementado en el microcontrolador.

Capítulo 2. Descripción del dispositivo

En este capítulo se analizarán los distintos bloques funcionales que conforman el dispositivo desarrollado, y que fueron presentados en la introducción. Por último, se describirá el programa implementado en el microcontrolador.

El capítulo se dividirá en tres secciones, la primera correspondiente a los periféricos utilizados y una breve descripción del microcontrolador, la segunda aborda con más detalle los distintos módulos internos del microcontrolador y la última se enfoca en el software desarrollado para llevar a cabo la aplicación final.

2.1. Diagrama en bloques

A continuación se describirán los distintos bloques funcionales que componen el dispositivo desarrollado (Figura 9).

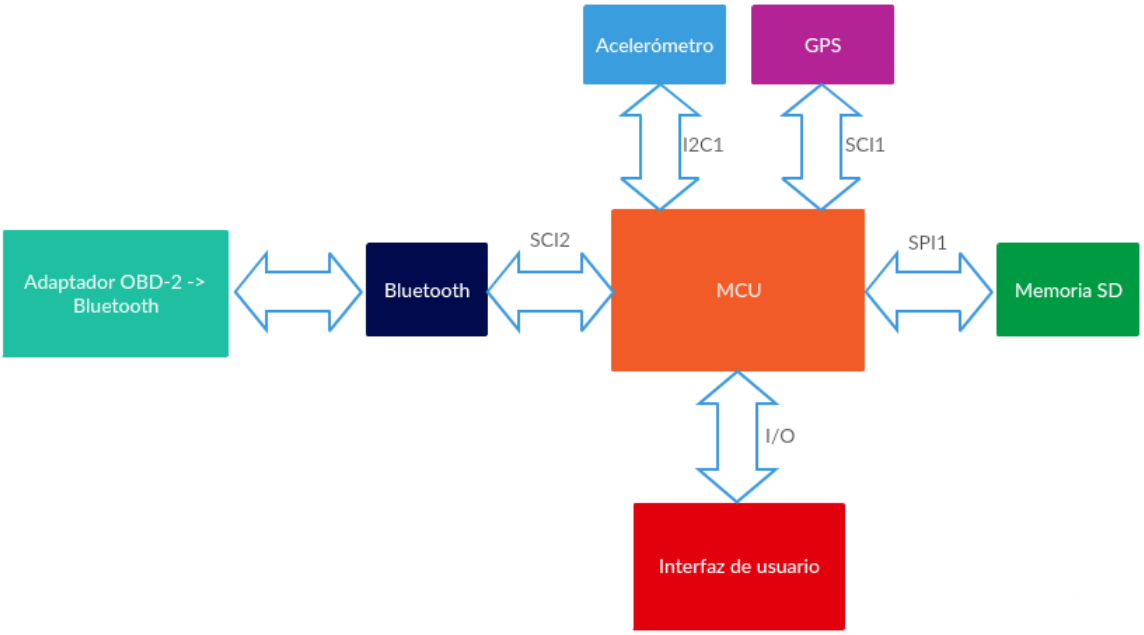


Figura 9 - Diagrama en bloques del sistema desarrollado

2.1.1. Adaptador OBD-2 -> Bluetooth

Este módulo se conecta en el puerto OBD-2 del automóvil y su función es la de transmitir los mensajes internos del vehículo a través de un enlace Bluetooth, para su posterior recepción por el microcontrolador. Está basado en el circuito integrado ELM-

327**[13]**, el cuál puede ser configurado mediante la utilización de comandos AT enviados a través de la interfaz serie (SCI). Todos los mensajes que se envíen al ELM-327 deben estar terminados con un retorno de carro (0x0D). Por ejemplo, para comprobar el estado de la conexión puede utilizarse el comando “AT Z”, y si el mensaje es recibido correctamente, el adaptador responderá con el mensaje “ELM327 v1.5”. De esta manera, se determina que el enlace se estableció correctamente.

Además de los comandos AT, el usuario puede requerir información interna del automóvil enviando comandos OBD (*On-board diagnostics*). Los protocolos especificados por el estándar OBD-2 son los siguientes:

1. SAE J1850 PWM
2. SAE J1850 VPW
3. ISO 9141 – 2
4. ISO 14230 – 4 KWP (5 BAUD INIT)
5. ISO 14230 – 4 KWP (FAST INIT)
6. ISO 15765 – 4 CAN (11 BIT ID, 500 KBAUD)
7. ISO 15765 – 4 CAN (29 BIT ID, 500 KBAUD)
8. ISO 15765 – 4 CAN (11 BIT ID, 250 KBAUD)
9. ISO 15765 – 4 CAN (29 BIT ID, 250 KBAUD)
10. SAE J1939 CAN (29 BIT ID, 250 KBAUD)

Independientemente del tipo de protocolo utilizado, en el nivel de capa de aplicación los mensajes OBD-2 poseen un formato estandarizado. En el caso de una solicitud, el formato de trama es el siguiente:

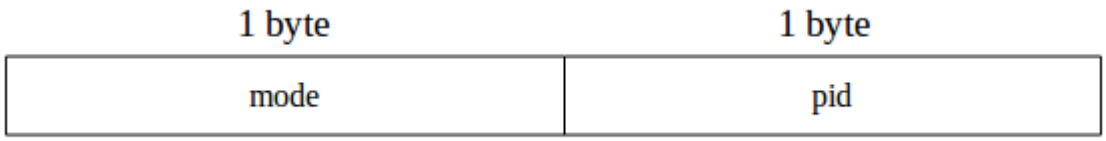


Figura 10 – Trama de solicitud OBD-2

Estos comandos son palabras en hexadecimal pero escritas como caracteres ASCII. Generalmente, están formados por uno o más pares hexadecimales. El primer par indica el modo OBD que debe utilizarse (Tabla 1), mientras que el segundo representa el Parameter ID (PID) solicitado. Algunos PID pueden observarse en la Tabla 2.

Número de modo	Descripción
1	Current Data
2	Freeze Frame Data
3	Diagnostic Trouble Codes
4	Clear Trouble Code
5	Test Results/Oxygen Sensors
6	Test Results/Non-Continuous Testing
7	Show Pending Trouble Codes
8	Special Control Mode
9	Request Vehicle Information
0A	Request Permanent Trouble Codes

Tabla 2 – No todos los PID son soportados en todos los vehículos.

PID (hex)	Descripción
0	PIDs disponibles
3	Estado del sistema de combustible
5	Temperatura del líquido refrigerante del motor
0C	RPM del motor
0D	Velocidad

Tabla 3 – Algunos PID utilizados.

Hay 10 modos OBD (Tabla 2) pero no todos son soportados por todos los vehículos. El modo 1 es el necesario para obtener información en tiempo real de los sensores.

Con respecto a la trama de respuesta, la misma posee el siguiente formato:

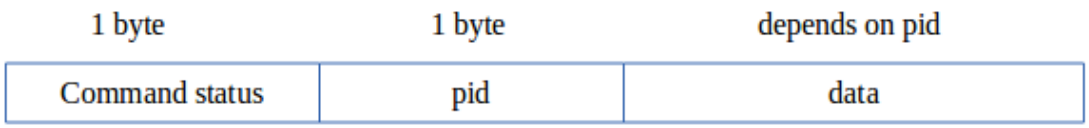


Figura 11 - Trama de respuesta OBD-2

El primer byte (Command status) indica si el comando tiene éxtio o no. Un comando exitoso resultará en un Command Status del tipo “0x40 + modo”, por lo que si se trata del modo 1, este campo tendrá el valor “0x41”. El segundo byte (PID) contiene el PID enviado en la trama de solicitud, mientras que en los bytes restantes se

encuentra la información requerida. La longitud del campo de Data dependerá del tipo de PID.

Para saber que PID son soportados por el vehículo, por ejemplo en el modo 1, se utiliza el comando “01 00”. Así, se obtiene la siguiente trama de respuesta:

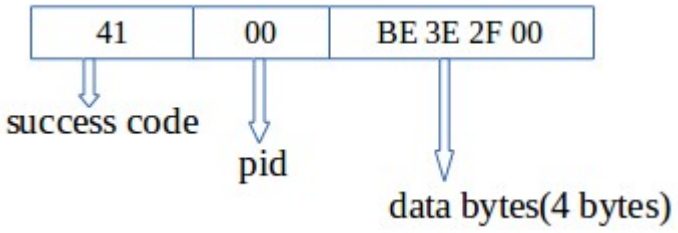


Figura 12 - Respuesta a una solicitud de los PID soportados por el vehículo

Como se mencionó anteriormente, el primer byte indica el modo solicitado y lo expresa sumándole el número de modo al 0x40. Como se trata del modo 1, se obtiene el número 0x41. El segundo byte indica el parámetro que se solicitó (0x00). Los bytes que siguen son la respuesta del mensaje, y en este caso indican los PID soportado (0xBE, 0x3E, 0x2F y 0x00).

Para obtener la velocidad del vehículo (PID 0D) deberemos enviar el comando “01 0D”, a lo que recibiremos una respuesta del tipo “41 0D 3C”. El tercer byte (0x3C) es el que nos indica la velocidad, pero para poder interpretarlo debemos convertirlo a decimal. En este caso indicaría una velocidad de 60 Km/h.

De esta manera, es posible obtener información como el nivel del tanque de combustible, la velocidad, las RPM, entre otros.

2.1.2. Módulo Bluetooth Maestro HC05

Se utilizó un módulo maestro debido a la necesidad de que el microcontrolador pueda iniciar la conexión con el adaptador OBD-2 -> Bluetooth. El módulo HC05 **[14]**está basado en el procesador BC417 de CSR **[15]** ypuede ser alimentado con tensiones entre 3,6V y 6V, pero las líneas de Tx y Rx están especificadas para un nivel lógico de 3,3V. Este dispositivo también es configurado mediante comandos AT enviados desde el microcontrolador, terminados en <CR><LF>.

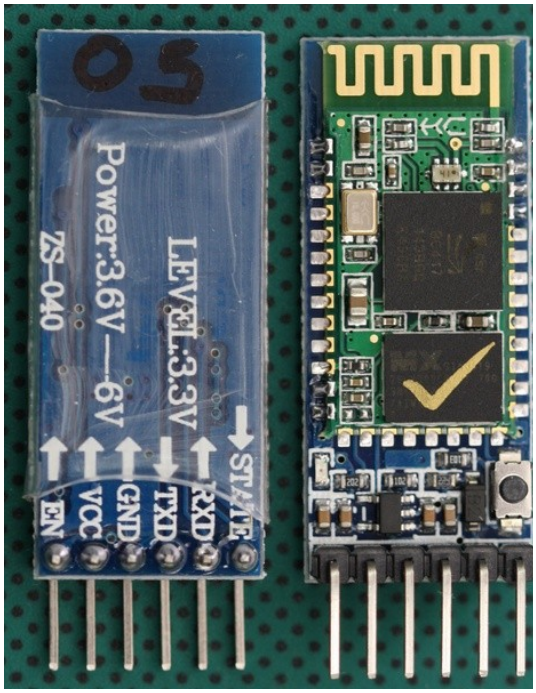


Figura 13 - Módulo Bluetooth HC05

Para ingresar en el modo de configuración AT deben seguirse los siguientes pasos:

1) Conectar los pines Tx y Rx a los pines Rx y Tx de la interfaz SCI del microcontrolador.
2) Mantener presionado el pulsador presente en el módulo.
3) Aplicar alimentación al HC05.

Si la operación tuvo éxito, el LED parpadeará con un intervalo de tiempo de 2 segundos y el HC05 está listo para ser configurado.

Algunos comandos útiles son los siguientes:

- Para obtener o cambiar el nombre del dispositivo utilizamos los siguientes comandos:

Command	Respond	Parameter
AT+NAME=<Param>	OK	Param: Bluetooth module name (Default :HC-05)
AT+NAME?	+NAME:<Param> OK (/FAIL)	

Figura 14 - Comandos para averiguar y cambiar el nombre del módulo

Ejemplo: Obtener nombre.

1. Envío el mensaje: AT+NAME?\r\n
2. Respuesta:+NAME:HC-05
   OK

- Los siguientes comandos permiten obtener o cambiar el password del módulo:

Command	Respond	Parameter
AT+PSWD=<Param>	OK	Param: PIN code (Default 1234)
AT+ PSWD?	+ PSWD : <Param> OK	

Figura 15 - Comandos para averiguar y cambiar el password del módulo

Ejemplo: Obtener password.

1. Envío el mensaje: AT+PSWD?\r\n
2. Respuesta:+PSWD:1234
   OK

- Para conocer si el dispositivo está configurado como maestro o esclavo y modificarlo según sea necesario se utilizan los siguientes comandos:

Command	Respond	Parameter
AT+ROLE=<Param>	OK	Param: 0- Slave
AT+ ROLE?	+ROLE:<Param>	
	OK	1-Master 2-Slave-Loop

Figura 16 - Comandos para configuración maestro/esclavo del módulo

Ejemplo: Configurar como maestro.

1. Envío el mensaje: AT+ROLE=1\r\n
2. Respuesta: OK

- Estos comandos permiten comprobar o modificar el baud rate del módulo:

Command	Respond	Parameter
AT+UART=<Param>,<Param2>,<Param3>	OK	Param1: Baud Param2: Stop bit Param3: Parity
AT+ UART?	+UART=<Param>,<Param2>,<Param3> OK	

Figura 17 - Comandos para averiguar y modificar el baud rate del módulo

Ejemplo: Obtener el baud rate configurado:

1. Envío el mensaje: AT+UART?\r\n
2. Respuesta:+UART:9600,0,0
   OK

2.1.3. Sistema de posicionamiento global (GPS)

Se utilizó el módulo Ublox Neo-6M**[16]**, que posee una interfaz serie para comunicarse con el microcontrolador.

Figura 18 - Módulo Ublox Neo-6M

El GPS utiliza un protocolo para transmitir información mediante mensajes NMEA que establece las siguientes reglas:

1) Todos los mensajes comienzan con el carácter $.
2) Los siguientes 5 caracteres indican el talker (dos caracteres) y el tipo de mensaje (tres caracteres). En el caso de un GPS, los primeros dos caracteres son GP, y los tres restantes dependerán del tipo de mensaje.
3) Los campos de datos siguientes se encuentran separados por comas.
4) Si algún dato no está disponible, ese campo permanece vacío.
5) Si el mensaje incluye un checksum, el último carácter después del último campo de datos, es un asterisco.
6) El asterisco es seguido por un checksum representado como un número hexadecimal de dos dígitos.
7) El mensaje termina con <CR><LF>.

Como ejemplo se muestra a continuación un mensaje GLL:

```
$GPGLL,4916.45,N,12311.12,W,225444,A
```

```
           4916.46,N    Latitude 49 deg. 16.45 min. North
12311.12,W   Longitude 123 deg. 11.12 min. West
           225444       Fix taken at 22:54:44 UTC
A            Data valid
```

Como puede observarse, los mensajes GLL contienen información de latitud, longitud y tiempo. Para interpretar los mensajes, en el caso de la latitud y la longitud, los dos dígitos a la izquierda del punto representan los minutos, mientras que a la derecha del punto son décimas de minutos.

El listado completo de mensajes NMEA es el siguiente:

Mensaje	**Descripción**
$GPAAM	Waypoint Arrival Alarm
$GPALM	GPS Almanac Data
$GPAPA	Autopilot Sentence "A"
$GPAPB	Autopilot Sentence "B"
$GPASD	Autopilot System Data
$GPBEC	Bearing & Distance to Waypoint, Dead Reckoning
$GPBOD	Bearing, Origin to Destination
$GPBWC	Bearing & Distance to Waypoint, Great Circle
$GPBWR	Bearing & Distance to Waypoint, Rhumb Line
$GPBWW	Bearing, Waypoint to Waypoint

$GPDBT	Depth Below Transducer
$GPDCN	Decca Position
$GPDPT	Depth
$GPFSI	Frequency Set Information
$GPGGA	Global Positioning System Fix Data
$GPGLC	Geographic Position, Loran-C
$GPGLL	Geographic Position, Latitude/Longitude
$GPGSA	GPS DOP and Active Satellites
$GPGSV	GPS Satellites in View
$GPGXA	TRANSIT Position
$GPHDG	Heading, Deviation & Variation
$GPHDT	Heading, True
$GPHSC	Heading Steering Command
$GPLCD	Loran-C Signal Data
$GPMTA	Air Temperature (to be phased out)
$GPMTW	Water Temperature
$GPMWD	Wind Direction
$GPMWV	Wind Speed and Angle
$GPOLN	Omega Lane Numbers
$GPOSD	Own Ship Data
$GPR00	Waypoint active route (not standard)
$GPRMA	Recommended Minimum Specific Loran-C Data
$GPRMB	Recommended Minimum Navigation Information
$GPRMC	Recommended Minimum Specific GPS/TRANSIT Data
$GPROT	Rate of Turn
$GPRPM	Revolutions
$GPRSA	Rudder Sensor Angle
$	RADAR System Data

$GPRSD	
$GPRTE	Routes
$GPSFI	Scanning Frequency Information
$GPSTN	Multiple Data ID
$GPTRF	Transit Fix Data
$GPTTM	Tracked Target Message
$GPVBW	Dual Ground/Water Speed
$GPVDR	Set and Drift
$GPVHW	Water Speed and Heading
$GPVLW	Distance Traveled through the Water
$GPVPW	Speed, Measured Parallel to Wind
$GPVTG	Track Made Good and Ground Speed
$GPWCV	Waypoint Closure Velocity
$GPWNC	Distance, Waypoint to Waypoint
$GPWPL	Waypoint Location
$GPXDR	Transducer Measurements
$GPXTE	Cross-Track Error, Measured
$GPXTR	Cross-Track Error, Dead Reckoning
$GPZDA	Time & Date
$GPZFO	UTC & Time from Origin Waypoint
$GPZTG	UTC & Time +A1:B59to Destination Waypoint

Tabla 4 - Listado de mensajes NMEA

Este módulo en particular envía por defecto mensajes de manera continua cada 1 segundo, a una velocidad de 9600 bps, por lo tanto es necesario implementar en el microntrolador un algoritmo que sea capaz de diferenciar los mensajes recibidos y extraer la información necesaria para finalmente representarla en un formato que pueda interpretar el usuario.

También es posible configurarlo para que solo envíe los mensajes que necesitamos. Si se requiere información de latitud, longitud, tiempo y velocidad, con

los mensajes GGA y VTG será suficiente. Por ejemplo, si queremos deshabilitar los mensajes GSV, debemos enviar la siguiente cadena de caracteres:

"$PUBX,40,GSA,0,0,0,0*4E\r\n"

Mientras que para habilitarlos:

"$PUBX,40,GSA,1,1,1,0*4F\r\n"

2.1.4. Memoria SD

Para implementar el sistema de archivos FAT32 y el control de la tarjeta SD, se utilizó la siguiente librería desarrollada por Suan-Aik Yeo**[17]**. La misma fue desarrollada para ser utilizada en el microcontrolador de 16 bits 9S12C32 de Freescale**[18]**, pero debido a la similitud entre los periféricos de las familias Coldfire V1 y 9s12, se pudo portar el código de manera sencilla. Los únicos archivos que dependen del tipo de hardware utilizados son uart.h y SPI.h.

En uart.h deben modificarse los siguientes defines:

```
/*
 * 9s12c32 microcontroller dependent locations that you might have to
change!
 */
#define SCI_TX_ENABLE SCICR2_TE /* SCI Transmitter enable bit */
#define SCI_RX_ENABLE SCICR2_RE /* SCI Receiver enable bit */
#define SCI_BAUD_LOW SCIBDL /* SCI Baud Rate Register (Low Byte) */
#define SCI_TDRE SCISR1_TDRE /* SCI Transmit Data Register Empty flag
*/
#define SCI_DATA_LOW SCIDRL  /* SCI Data Register (Low Byte) */
#define SCI_RDRF SCISR1_RDRF /* SCI Receive Data Register Full flag */
```

Por los siguientes:

```
#define SCI_TX_ENABLE SCI2C2_TE /* SCI Transmitter enable bit */
#define SCI_RX_ENABLE SCI2C2_RE /* SCI Receiver enable bit */
#define SCI_BAUD_LOW SCI2BDL /* SCI Baud Rate Register (Low Byte) */
#define SCI_TDRE SCI2S1_TDRE /* SCI Transmit Data Register Empty flag
*/
#define SCI_DATA_LOW SCI2D  /* SCI Data Register (Low Byte) */
#define SCI_RDRF SCI2S1_RDRF /* SCI Receive Data Register Full flag */
```

Mientras que en SPI.h deben modificarse:

```
/*
 * 9s12c32 microcontroller dependent locations that you might have to
change!
 */
#define SPI_ENABLE SPICR1_SPE /* SPI System Enable Bit */
#define SPI_MASTER_SEL SPICR1_MSTR  /* SPI Master/Slave Mode Select
Bit */
```

```
#define SPI_CLK_PHASE SPICR1_CPHA /* SPI Clock Phase Bit */
#define SPI_BAUD_RATE SPIBR /* SPI Baud Rate Register */
#define SPI_SS_DIR DDRM_DDRM3 /* SPI Slave Select Pin Data Direction
Register */
#define SPI_SS PTM_PTM3 /* SPI Slave Select Pin */
#define SPI_TX_EMPTY SPISR_SPTEF /* SPI Transmit Empty Interrupt Flag
*/
#define SPI_INTERRUPT SPISR_SPIF /* SPI Interrupt Flag */
#define SPI_DATA SPIDR  /* SPI Data Register */
```

Por:

```
#define SPI_ENABLE SPI1C1_SPE /* SPI System Enable Bit */
#define SPI_MASTER_SEL SPI1C1_MSTR  /* SPI Master/Slave Mode Select
Bit */
#define SPI_CLK_PHASE SPI1C1_CPHA /* SPI Clock Phase Bit */
#define SPI_BAUD_RATE SPI1BR /* SPI Baud Rate Register */
#define SPI_SS_DIR PTFDD_PTFDD5 /* SPI Slave Select Pin Data Direction
Register */
#define SPI_SS PTFD_PTFD5 /* SPI Slave Select Pin */
#define SPI_TX_EMPTY SPI1S_SPTEF /* SPI Transmit Empty Interrupt Flag
*/
#define SPI_INTERRUPT SPI1S_SPRF /* SPI Interrupt Flag */
#define SPI_DATA SPI1D  /* SPI Data Register */
```

Por el lado del hardware, en la Figura 19 se muestra el módulo utilizado.

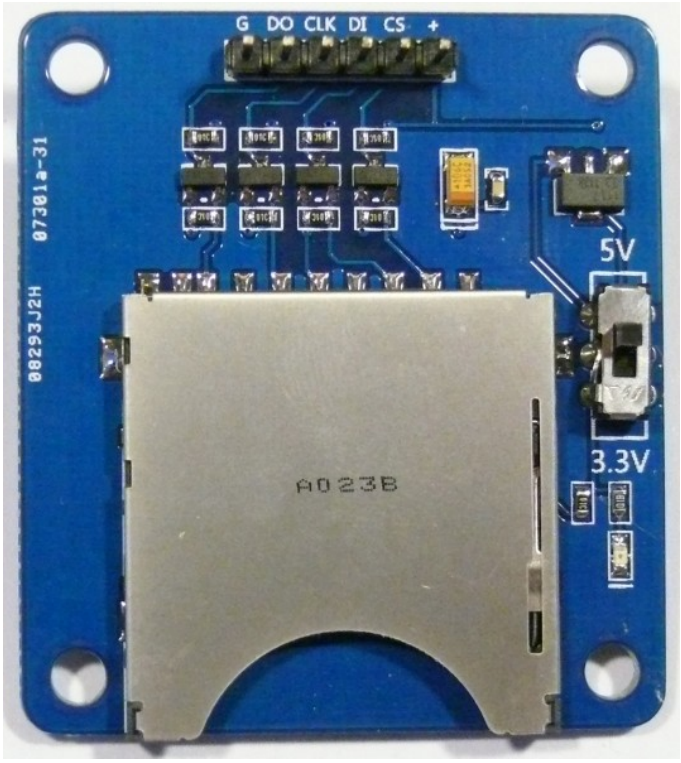


Figura 19 - Módulo de tarjeta SD

El mismo se comunica mediante una interfaz SPI y cuenta con los conversores de nivel necesarios para poder adaptar el nivel lógico de la tarjeta SD (3.3V) al nivel utilizado por el microcontrolador (5V).

2.1.5. Acelerómetro

Se utilizó el acelerómetro de 3 ejes MMA8452 **[19]** de Freescale.

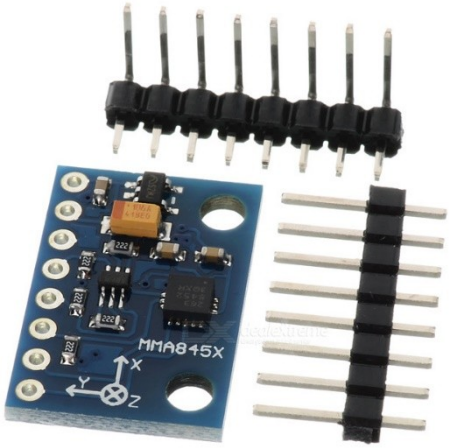


Figura 20 - Módulo con acelerómetro MMA8452

Características principales:

- Comunicación I2C.
- Resolución de 12 bits.
- 2 pines de interrupción.
- Escalas seleccionables +/-2g, +/-4g, +/-8g.
- Filtro pasa altos.

Si bien el acelerómetro trabaja con señales lógicas de 3,3V, este módulo en particular viene preparado para ser utilizado con señales de 5V.

2.1.6. Microcontrolador

El microcontrolador utilizado es el MCF51JM128 **[20]** de la familia Coldfire V1 de Freescale. El mismo puede funcionar a una frecuencia de hasta 50,33MHz y posee un encapsulado QFP de 64 pines, por lo que dispone de una gran cantidad de puertos para comunicarse con los periféricos externos requeridos. También permite la utilización de cristales externos de hasta 16MHz y posee un PLL interno para incrementar esa frecuencia. Sus características principales son las siguientes:

- Frecuencia de clock de hasta 50,33MHz (los periféricos internos funcionan a la mitad de esa frecuencia, con excepción del RGPIO).
- 128KBytes de memoria Flash.
- 16KBytes de memoria RAM.
- Módulo BDM para programación *in-circuit y debugging*.
- Comparador analógico (ACMP).
- ADC de 12 bits con 12 canales seleccionables.
- Controlador CAN.
- 2 módulos I2C.
- 8 líneas de interrupción por teclado (KBI).
- Detector de bajo voltaje (LVD).
- Módulo MCG (*Multipurpose Clock Generator*).
- 2 interfaces serie (SCI1, SCI2).
- 2 interfaces SPI (SPI1, SPI2).
- RTC de 8 bits (*Real Time Counter*).
- TPM1 (6 canales) y TPM2 (2 canales) de 16 bits (*Timer/PWM*).
- Controlador USBOTG (*USB On-The-Go*).
- RGPIO (*Rapid GPIO*).

En las Figura 21 y Figura 22 se muestra un diagrama en bloques interno y la distribución de pines del MCF51JM128.

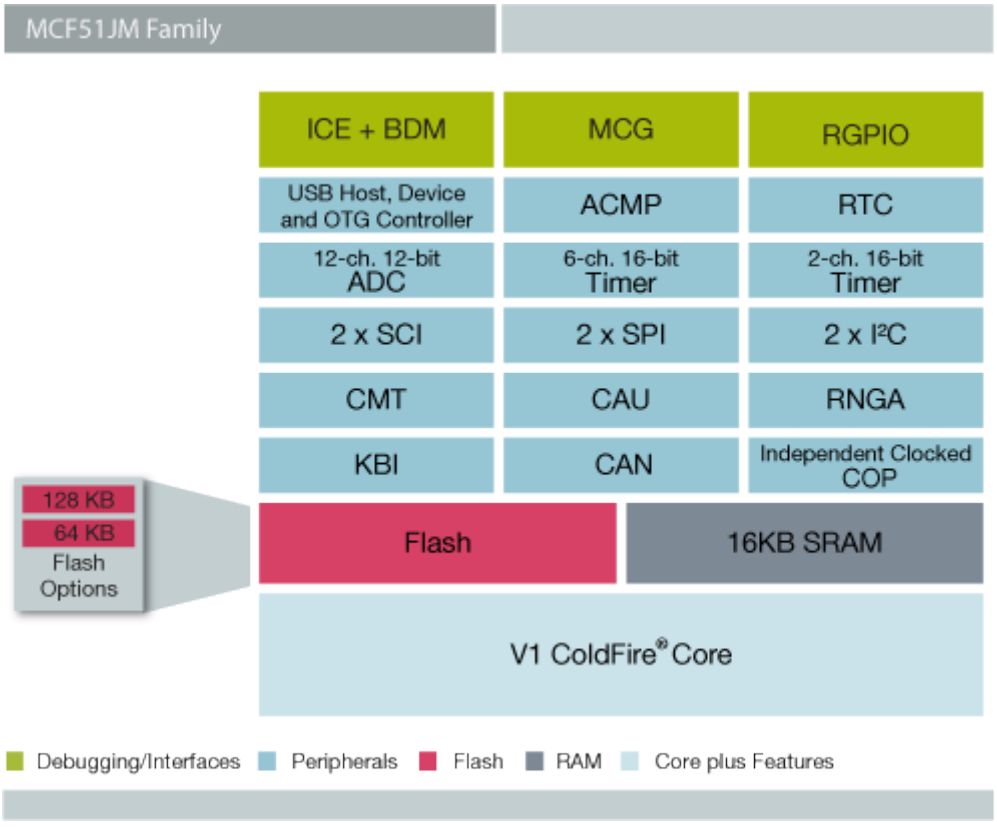


Figura 21 - Estructura interna del microcontrolador

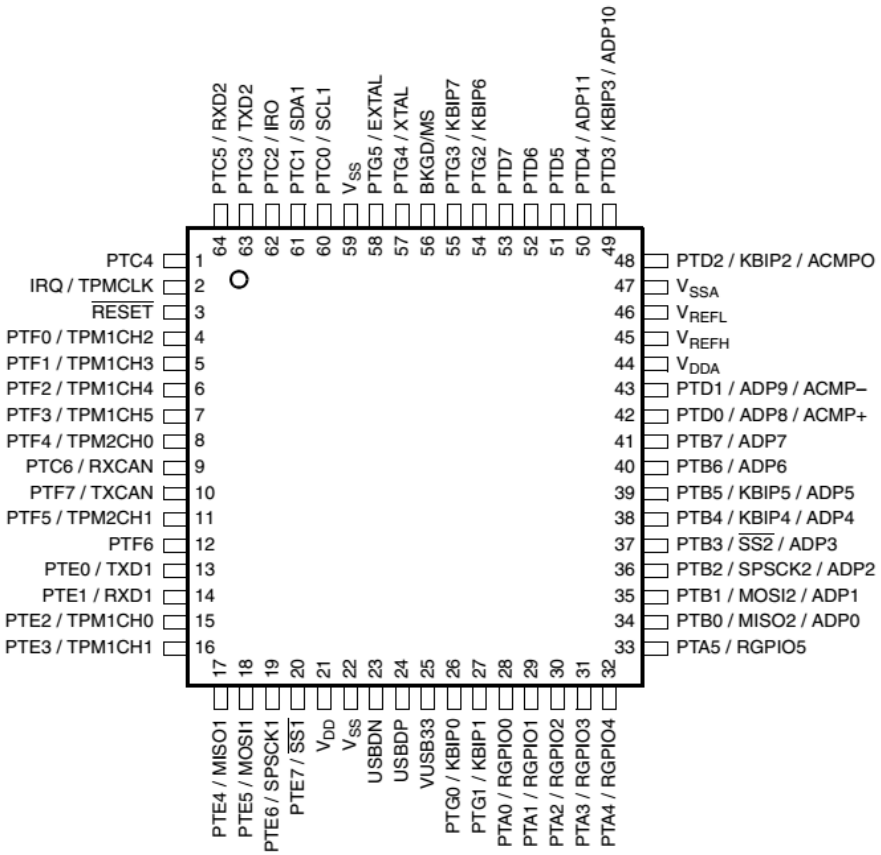


Figura 22 - Distribución de pines del microcontrolador

2.1.7. Interfaz de usuario

La interfaz de usuario cuenta con un display LCD de 20 caracteres por 4 líneas basado en el controlador HD44780**[21]** (Figura 23), dos pulsadores y un LED bicolor.

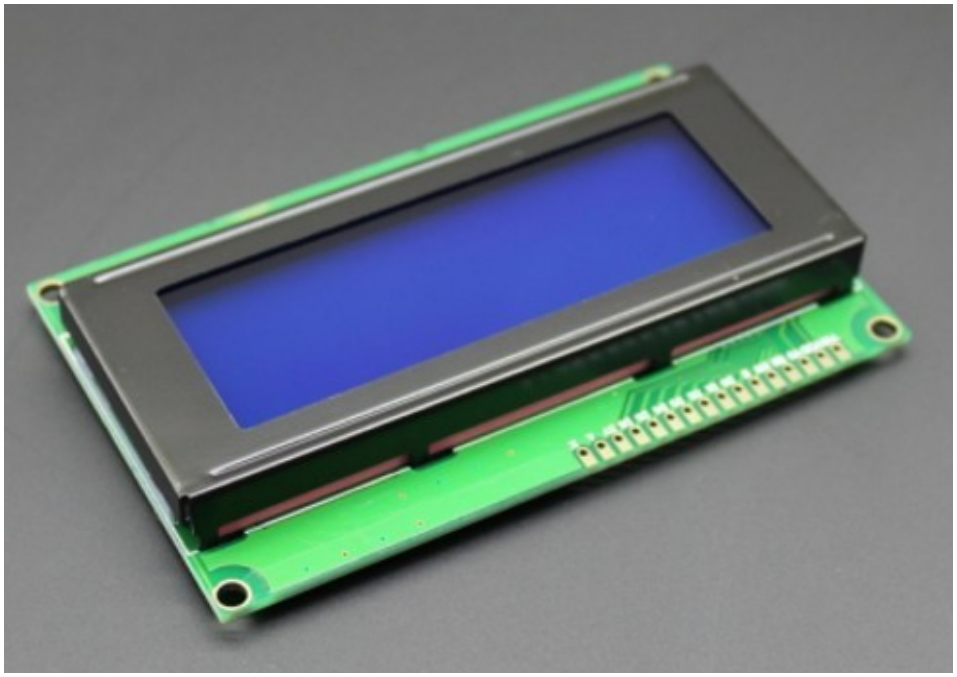
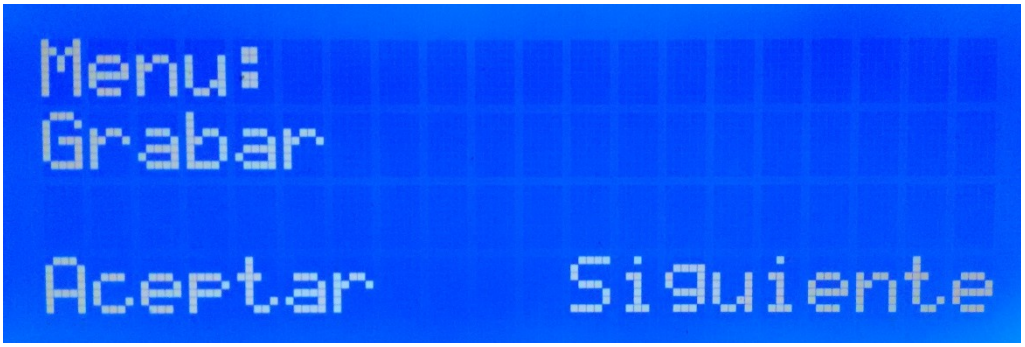


Figura 23 - Display LCD de 20 caracteres por 4 lineas

Se implementó un menú de opciones mediante el cual el usuario, con la utilización de los dos pulsadores presentes, puede iniciar la grabación en la tarjeta SD o configurar los distintos modos de operación, mediante los cuales se establecen los umbrales de aceleración a partir de los que se emitirá una alerta.

En la Figura 24 se presentan las opciones de grabación y de selección de modo como se ven en el display LCD. En el primer caso, presionando el pulsador 1 se inicia el proceso de grabación, mientras que con el pulsador 2, se pasa al segundo caso, donde se ingresa al menú de selección de modo con el pulsador 1, y con el pulsador 2 se sale del menú.



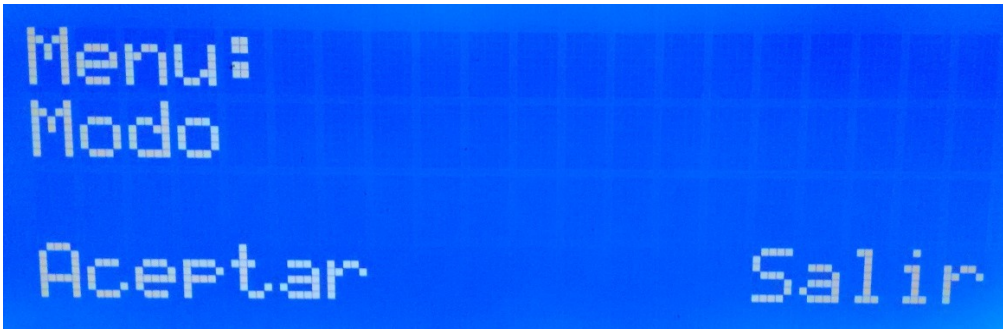


Figura 24 - Menú principal

La Figura 25 y la Figura 26 muestran el menú de selección de modo. La forma de navegar por este menú es similar a la descripta anteriormente, con el pulsador 1 se selecciona el modo y con el pulsador 2 se avanza a la siguiente opción.

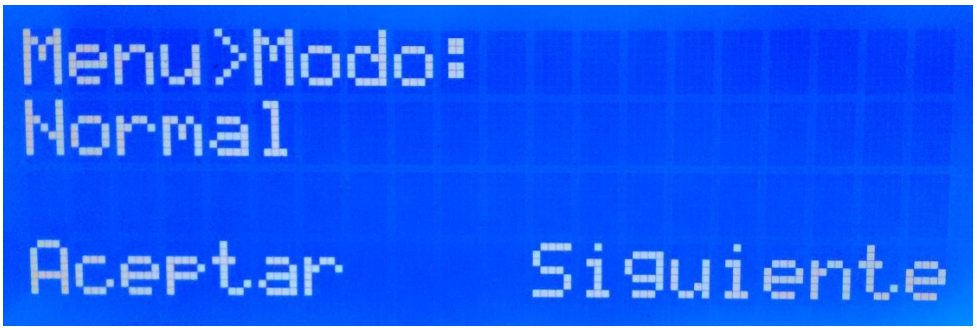


Figura 25 - Selección de modo (Normal)

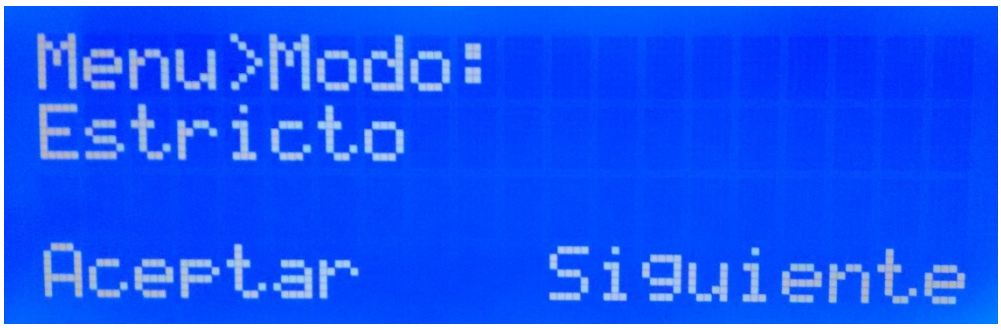


Figura 26 - Selección de modo (Estricto)

En el LCD también se presentan datos en tiempo real, estos son la velocidad, el porcentaje del tanque de combustible, las RPM, el estado de conexión del GPS, aceleración en los ejes X e Y en g, el tiempo transcurrido desde que se inició el viaje y el tiempo que el auto estuvo en movimiento (Figura 27, Figura 28 y Figura 29). Con el pulsador 1 se seleccionan las distintas pantallas.

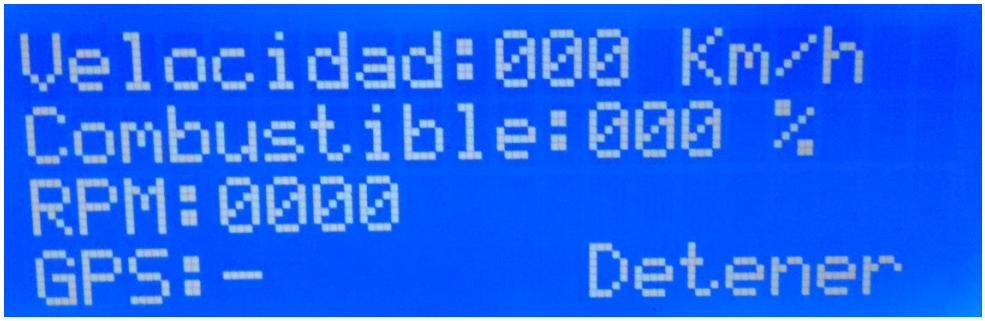


Figura 27 - Datos de velocidad, combustible, RPM y estado del GPS

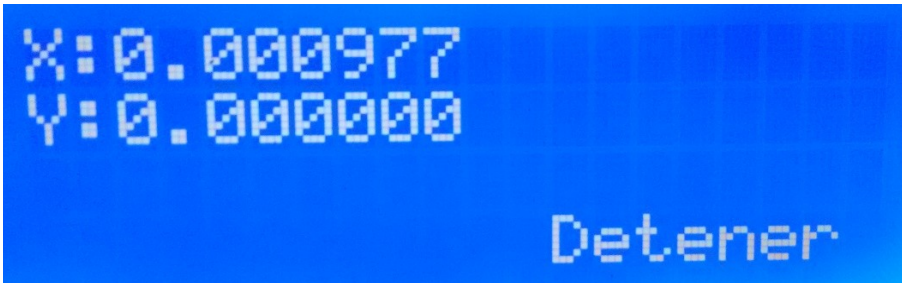


Figura 28 - Aceleración de los ejes x e y en G

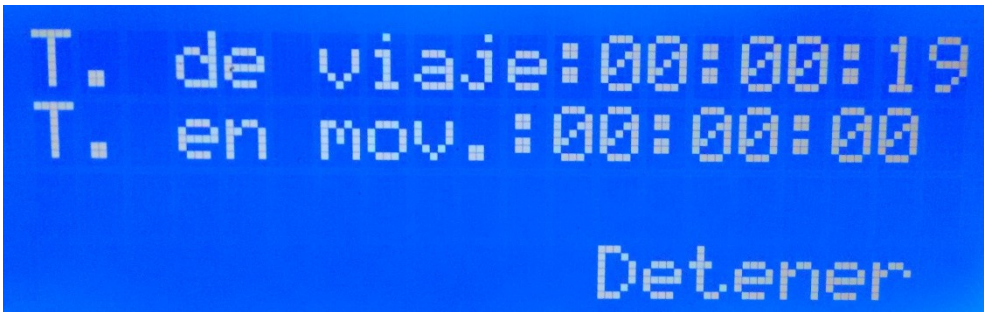


Figura 29 - Tiempo desde que se inició el viaje y tiempo que el auto estuvo en movimiento

Para finalizar la grabación de datos, se utiliza el pulsador 2. Una vez hecho esto, se mostrará en pantalla la duración del viaje (Figura 30) y luego se regresará al menú principal.

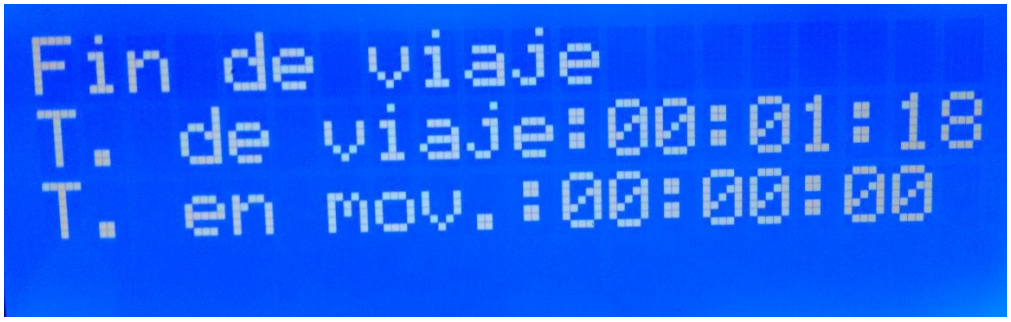


Figura 30 - Cuando finaliza el viaje se muestra en pantalla la duración del mismo

2.2. Microcontrolador

Se utilizó el microcontrolador MCF51JM128 debido a que posee un *core* de 32 bits, y sus periféricos son compatibles con los microcontroladores de la familia S08 de 8 bits, por lo que su programación es muy sencilla y existe abundante información disponible sobre la utilización de los mismos. Además, a diferencia de microcontroladores de costo similar, cuenta con dos interfaces serie que fueron indispensables para el desarrollo de este proyecto.

A continuación se describe en más detalle los periféricos internos del microcontrolador**[22]**, junto con los registros utilizados en este proyecto.

2.2.1. Serial Communication Interface (S08SCIV4)

El MCF1JM128 cuanta con dos interfaces serie independientes, SCI1 y SCI2, también conocidas comúnmente como *Universal Asynchronous Receivers/Transmitters* (UARTs).

Para poder establecer una comunicación a través de un enlace serie es necesario configurar correctamente la velocidad de transmisión dada en bits por segundo (bps). Para esto, la interfaz SCI cuenta con un generador de *Baud Rate* de 13 bits, lo que permite configurar distintas velocidades de acuerdo a lo requerido. Este generador es independiente para cada una de las interfaces disponibles en el microcontrolador.

A continuación se listan las características principales de este periférico:

- *Full-duplex*, Formato estándar *non-return-to-zero* (NRZ)
- Transmisor y receptor con doble buffer y *enable* independientes
- *Baud Rate* programable (Divisor de módulo de 13 bits)
- Funcionamiento mediante interrupciones o encuesta:
    - — Registro de transmisión de datos vacíp y transmisión completa
    - — Registro de recepción de datos lleno
    - — *Overrun* de recepción, error de paridad, error de *framing*, error por ruido
    - — Detección de receptor inactivo
    - — Flanco activo en el pin de recepción
- Hardware de generación y comprobación de paridad
- Longitud de caracteres de 8 ó 9 bits
- Polaridad de salida del transmisor seleccionable.

Cuenta con 8 registros que permiten configurar una gran cantidad de parámetros, como trabajar en modo *loopback* (para comprobar el correcto funcionamiento sin necesidad de utilizar otro microcontrolador como receptor del mensaje), longitud de caracteres de 8 o 9 bits, paridad, entre otros. Los utilizados en este proyecto son:

- **SCI *Baud Rate Registers* (SCIxBDH, SCIxBDL):** En estos dos registros se configura el valor del divisor de 13 bits para generar el *baud rate*. En SCIxBDH se encuentra la parte alta del divisor (SBR[12:8]), mientras que en SCIxBDL la parte baja (SBR[7:0]).

Figura 31 - Registro SCIxBDH

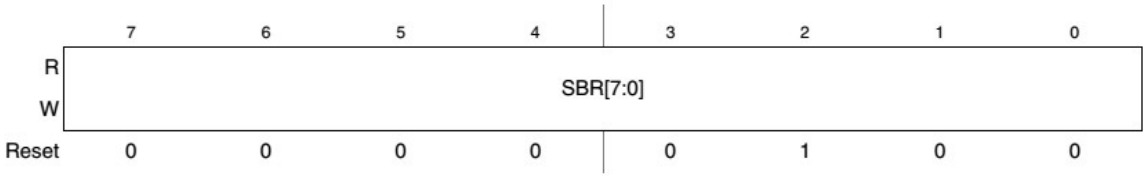


Figura 32 - Registro SCIxBDL

La ecuación 1 es la utilizada para calcular el baud rate.

Ec. 1 $$Baud\ Rate = \frac{BUS_{CLK}}{16 \times BR} Baud\ Rate = \frac{BUS_{CLK}}{16 \times BR}$$

Donde BUS_CLK es la frecuencia *clock* del sistema dividido 2, y BR el divisor de 13 bits.

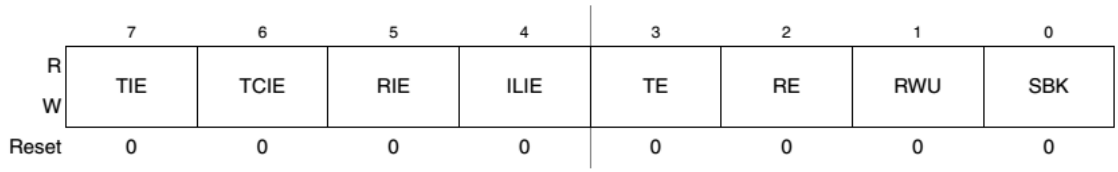
Considerando una *clock* del bus de datos de 24MHz, si se quiere obtener el valor del divisor para un *baud rate* de 38400 bps, solo debemos despejarlo de la ecuación 2:

Ec. 2 $$BR = \frac{24MHz}{16 \times 38400} = 39\ BR = \frac{24MHz}{16 \times 38400} = 39$$

De esta manera, debemos agregar la siguiente línea de código para que la configuración sea efectiva:

```
SCI2BD = 39;     //38400 baudios a 24 MHz
```

- **SCI *Control Register 2* (SCIxC2):** En este registro se configuran las interrupciones por transmisión y recepción, entre otras opciones.
-



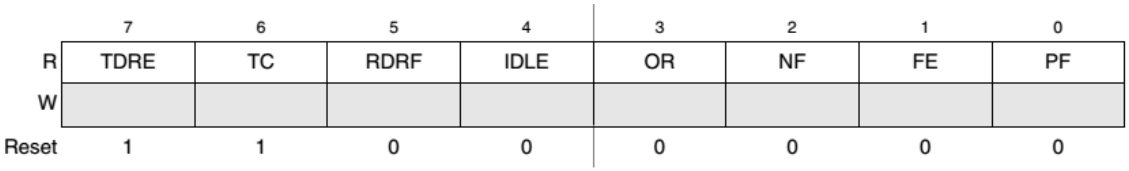
	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Figura 33 - Registro SCIxC2

El estado inicial del registro luego de un *reset* es con todos los bits en “0”. Para habilitar interrupciones por recepción y habilitar los módulos transmisor y receptor, debemos escribir un “1” en los bits RIE, TE y RE respectivamente.

```
SCI2C2 = 0b00101100;          //interrupcionesporRx, Tx y Rxactivados
```

- **SCI *Status Register 1* (SCIxS1):** Este registro contiene 8 flags de estado de solo lectura. TDRE toma el valor “0” cuando el buffer de transmisión se encuentra lleno, y el “1” cuando está vacío. TC vale “0” cuando el transmisor se encuentra activo, y “1” cuando está sin actividad. RDRF toma el valor “0” cuando el registro de recepción de datos está vacío, y “1” cuando está lleno.



	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

Figura 34 - Registro SCIxS1

- **SCI *Data Register* (SCIxD):** Está formado por dos registros, uno de escritura y otro de lectura. Las lecturas devuelven el valor almacenado en el buffer de recepción de datos, mientras que las escrituras escriben en el buffer de transmisión de datos.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figura 35 - Registro SCIxD

Si queremos transmitir un dato, debemos hacer:

```
while (!SCI2S1_TDRE);  // Espera a que se vacíe el buffer de Tx
    SCI2D = dato1;      // Escribe el dato a enviar en el buffer de Tx
```

2.2.2.*Timer/PWM Module*

El TPM es un *timer* que posee 8 canales y soporta las funciones de input capture, output compare y Modulación por Ancho de Pulsos (PWM). Es un contador de 16 bits y es posible ajustar la frecuencia mediante un *prescaler* y el módulo del contador. Características principales:

- 8 canales:

  — Cada canal puede ser configurado como *input capture*, *output compare*, o *edge-aligned PWM*

  — Disparo de *input capture* por flanco ascendente, flanco descendente o cualquier flanco

  — Funciones de output compare: Setear, borrar, o conmutar

  — Polaridad seleccionable en las salidas PWM
- Distintas fuentes de reloj disponibles:

  — *Prescaler* de 1, 2, 4, 8, 16, 32, 64, o 128

  — *Clock* del bus de datos

  — Pin de *clock* externo
- Contador de 16 bit con modos de cuenta libre o por módulo, con operación up/down
- Una interrupción por cada canal más interrupción por cuenta

Los registros utilizados fueron los siguientes:

- **TPM *Status and Control* Register (TPMxSC):** Este registro contiene el bit de indicación de *overflow* (TOF), bit para configurar las interrupciones por *overflow* (TOIE), configuración de PWM (CPWMS), fuentes de *clock* (CLKS), y el *prescaler* (PS).

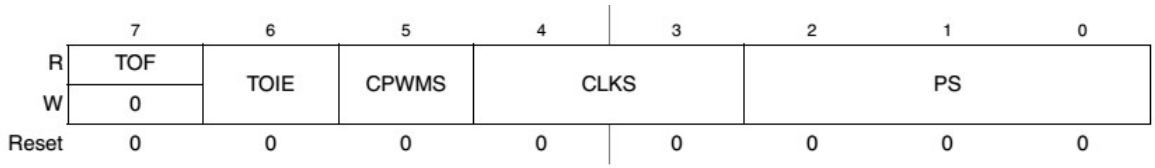


Figura 36 – Registro TPMxSC

Cada vez que el *timer* llega al valor de módulo configurado, o si se produce un *overflow*, el bit TOF tomará el valor “1”, caso contrario, permanecerá en “0”.

Para activar las interrupciones por *overflow* debemos escribir un “1” en el bit TOIE.

Mediante la configuración de los bits 4 y 3 (CLKSB y CLKSA) es posible elegir la fuente de reloj del *timer* (Tabla 5).

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Tabla 5 - Selección de la fuente de reloj

Los bits 2 al 0 permiten configurar un *prescaler* para así obtener distintas configuraciones de tiempos (Tabla 6).

PS	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Tabla 6 - Selección de *prescaler*

- **TPM *Counter Modulo Registers* (TPMxMODH, TPMxMODL):** Este registro de 16 bits contiene el valor del módulo del contador. Una vez llega hasta el valor de modulo establecido, la cuenta continua desde 0x0000 y el bit TOF del registro TPMxSC toma el valor “1”.

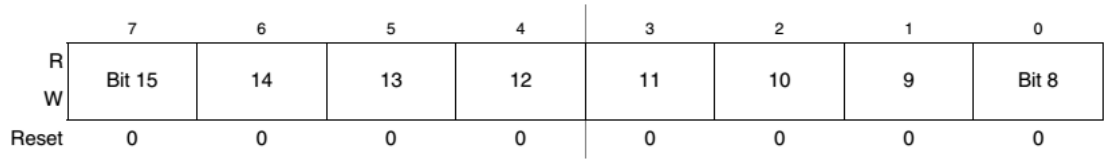


Figura 37 - Parte alta del registro de modulo

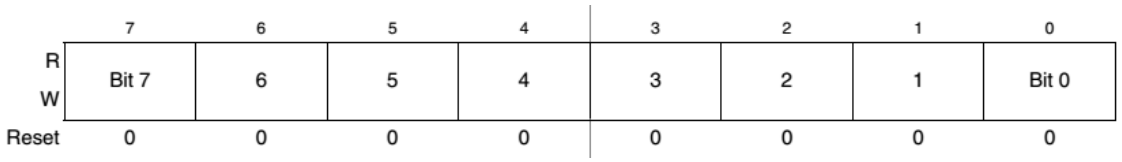


Figura 38 - Parte baja del registro de modulo

Para configurar el módulo TPM para producir interrupciones por *overflow* cada 250ms, con un reloj del bus datos de 24MHz y un *prescaler* de 128, se debe escribir un “1” en los bits TOIE, CLKSA y en los tres bits correspondientes al *prescaler* (PS0, PS1, PS2), del registro TPMxSC. Mientras que con la ecuación 3 Podemos calcular el valor del módulo a cargar en el registro TPMxMOD.

Ec. 3 $$Modulo = \frac{Tiempo\ de\ cuenta}{Prescaler} \times Frecuencia\ TPM = \frac{250ms}{128} \times 24MHz = 46875$$

$$Modulo = \frac{Tiempo\ de\ cuenta}{Prescaler} \times Frecuencia\ TPM = \frac{250ms}{128} \times 24MHz = 46875$$

```
/*
/ Configuración de módulo TPM2
*/
TPM2MOD = 46875;  // Interrupcióncada 250 ms

TPM2SC = 0b01001111     //Interrupcionespor OVF, BUS CLK, DIV 128
```

2.2.3.*Serial Peripheral Interface* (SPI)

Es un protocolo de comunicación desarrollado por Motorola que permite el intercambio de datos de manera sencilla entre distintos procesadores y periféricos externos, como pueden ser memorias, Conversores Analógico Digital (ADC), Conversores Digital Analógico (DAC), entre otros.

Se transmite sobre 4 líneas de manera sincrónica y *full-duplex*, con una topología maestro-esclavo. Posee una línea de reloj (SPSCK), una salida de datos serie (MOSI: *Master Output-Slave Input*), una entrada de datos serie (MISO: *Master Input-Slave Output*) y una de selección (SS: *Slave Select*).

El maestro inicia la comunicación seleccionando al esclavo mediante la línea SS, y por cada flanco de reloj, envía un bit de manera serie a través de la línea MOSI. En el caso del MCF51JM128, el módulo SPI puede configurarse para enviar datos de 8 o 16 bits. En modo maestro el *baud rate* máximo corresponde a la frecuencia del bus de datos dividida 2, mientras que en modo esclavos corresponde a dicha frecuencia dividida 4. Puede ser utilizado tanto en modo encuesta como por interrupciones.

Características principales:

- Operación en modo maestro o esclavo
- Modo *Full-Duplex* o *Single-Wire* bidireccional
- Velocidad de transmisión programable
- Opciones de fase y polaridad de la señal de *clock* serie
- Salida de selección de esclavo
- Modo de transmisión *MSB-first* o *LSB-first*
- Longitud de datos de 8 o 16 bits
- FIFO de 64bits (SPI2)

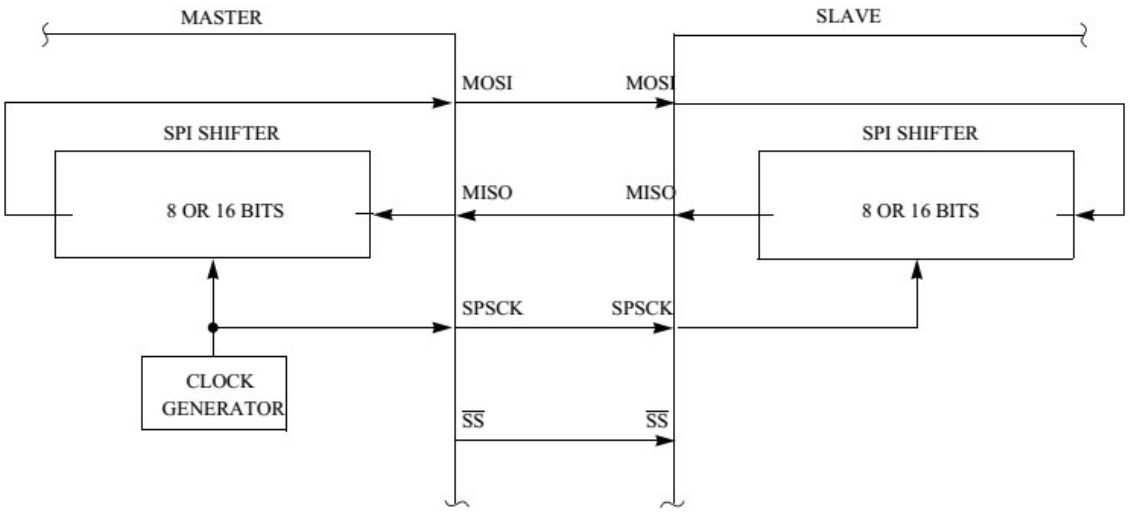


Figura 39 - Conexionado de un sistema SPI

Se utilizaron los siguientes registros:

- **SPI *Control Register 1* (SPIxC1):** Desde este registro se puede habilitar el módulo SPI, activar interrupciones y distintas opciones de configuración que dependerán del dispositivo con el cual queramos comunicarnos.

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

Figura 40 - Registro SPIxC1

Escribiendo un “1” en el bit SPIE se habilitan las interrupciones por recepción de datos.

Para habilitar el módulo SPI debemos escribir un “1” en el bit SPI. En caso de dejarlo desactivado, los pines MISO, MOSI, SPCK y SS pasan a funcionar como pines de I/O.

SPTIE controla las interrupciones por transmisión. Si escribimos un “1” en dicho bit, se producirá una interrupción cuando el buffer de transmisión de datos se vacíe.

Con MSTR se configura al microcontrolador como maestro (“1”) o como esclavo (“0”).

La polaridad de la señal de reloj se controla con el bit CPOL. Un valor “0” produce un valor bajo cuando el reloj está inactivo, mientras que un “1” produce un valor alto.

CPHA establece la fase del reloj. Con un “0”, el primer flanco del reloj ocurre a la mitad del primer ciclo de una transmisión de datos. Con un “1”, ocurre al comienzo del primer ciclo de una transmisión de datos.

El bit SSOE se utiliza en combinación del bit MODFEN del registro SPIxC2, y su función es la de configurar el comportamiento del pin SS.

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

Tabla 7 - Configuración del pin SS

En el caso que se necesite controlar manualmente la selección del esclavo, MODFEN deberá tomar el valor “0” y SSOE podrá valer tanto “0” como “1”.

Con LSBFE se configura si se transmite primero el bit menos significativo (“1”) o si s e transmite el más significativo (“0”).

- **SPI *Baud Rate Register* (SPIxBR):** Cuando el módulo SPI se encuentra configurado en modo maestro, con este registro podremos configurar el *prescaler* y el divisor de *bit rate* para obtener la velocidad de transmisión deseada.

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figura 41 - Registro SPIxBR

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

Tabla 8 - *Prescaler*

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

Tabla 9 - Divisor

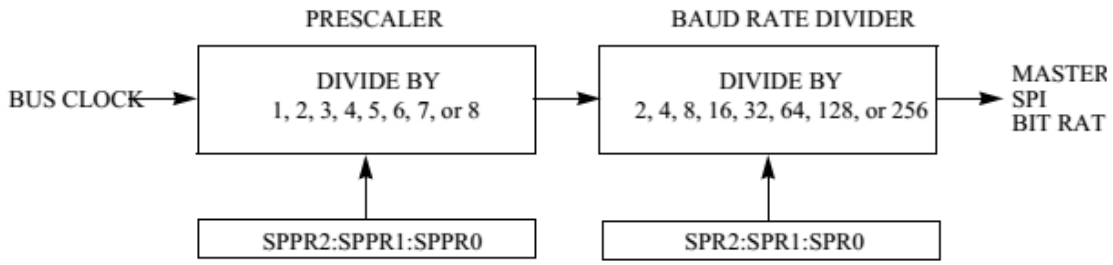


Figura 42 - Generación de reloj

La ecuación para calcular el *baud rate* es la siguiente:

Ec. 4

$$Baud\ Rate = \frac{Bus\ Clock}{(SPPR[2:0]+1) \times 2^{(SPR[2:0]+1)}}$$

$$Baud\ Rate = \frac{Bus\ Clock}{(SPPR[2:0]+1) \times 2^{(SPR[2:0]+1)}}$$

Para el caso de un baud rate de 375 KHz, considerando una frecuencia del bus de datos de 24 MHz, configuramos los bits SPRR[2:0] en “0:0:0” y los bits SPR2[2:0] en “1:0:0”. Con la siguiente línea de código realizamos la configuración:

```
SPI1BD = 0x04;       // BR = 375KHz
```

Para transmitir un dato por SPI sólo debemos comprobar que no haya una transmisión en curso y escribir el dato en el registro SPI1D:

```
while(!SPI1S_SPTEF);  // Espero a que finalice la transmisión
SPI1D = dato;         // Escribo el dato en el buffer de Tx
```

2.2.4. *Multipurpose Clock Generator* (S08MCGV3)

El sistema de generación de reloj del MCF51JM128 contiene un lazo de enganche de frecuencia (FLL) y un lazo de enganche de fase (PLL), los cuales pueden ser utilizados tanto por el reloj de referencia interno como por el externo.

Luego de un *reset*, el microcontrolador funciona en modo FLL con referencia interna (FEI). Se utiliza dicha referencia de 32,768KHz para producir una frecuencia de bus de datos es de 4.194304MHz.

Para este proyecto se utilizó un cristal de 12MHz como referencia externa y el modo PLL *Engaged External* (PEE). El PLL produce una frecuencia de referencia de 48MHz, que luego es dividida para producir la referencia del bus de datos de 24MHz.

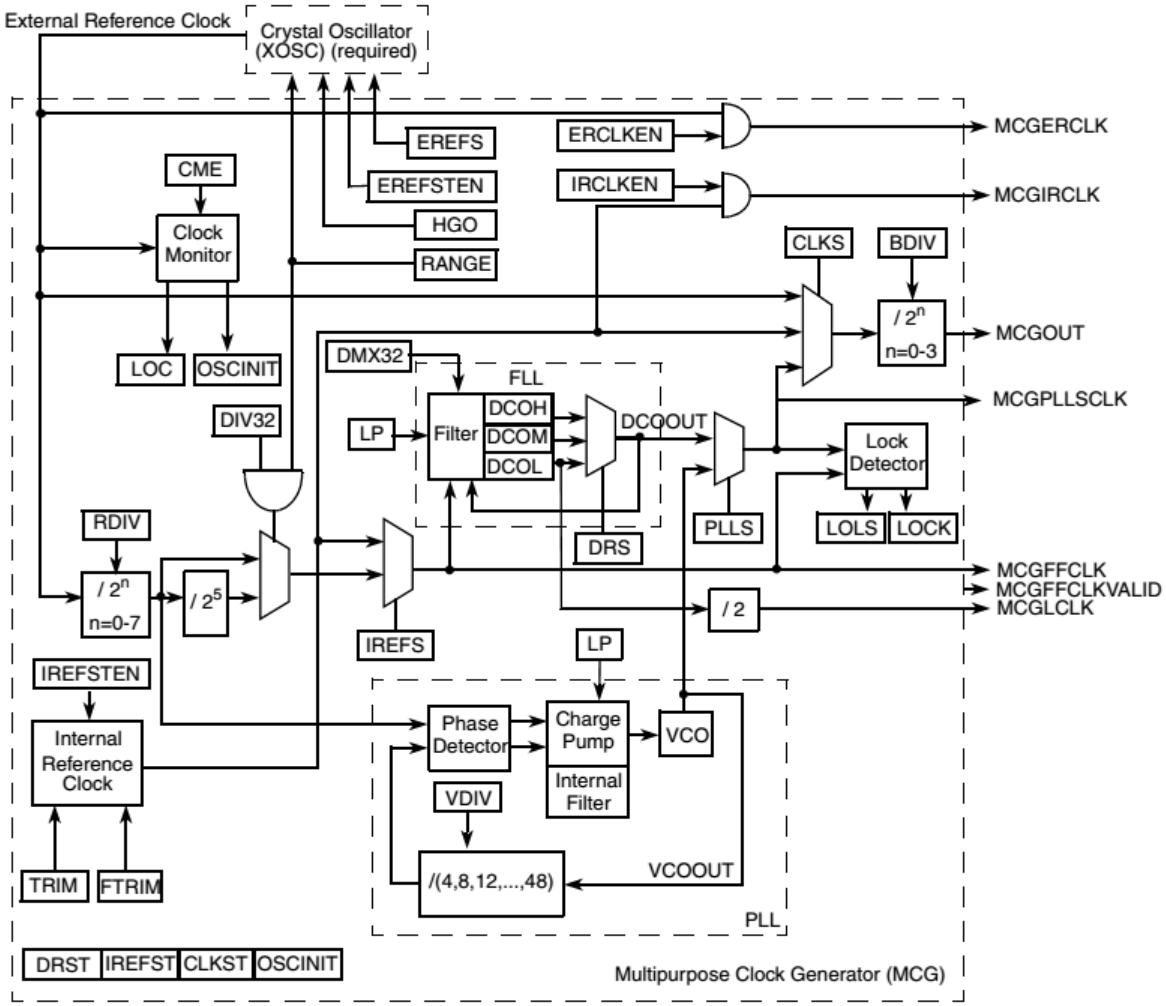


Figura 43 - Diagrama en bloques del módulo MCG

Debido a la complejidad del módulo se optó por utilizar Processor Expert para generar el código necesario para inicializarlo. Processor Expert es una herramienta de configuración de periféricos provista por Freescale en el software Codewarrior**[23]**.

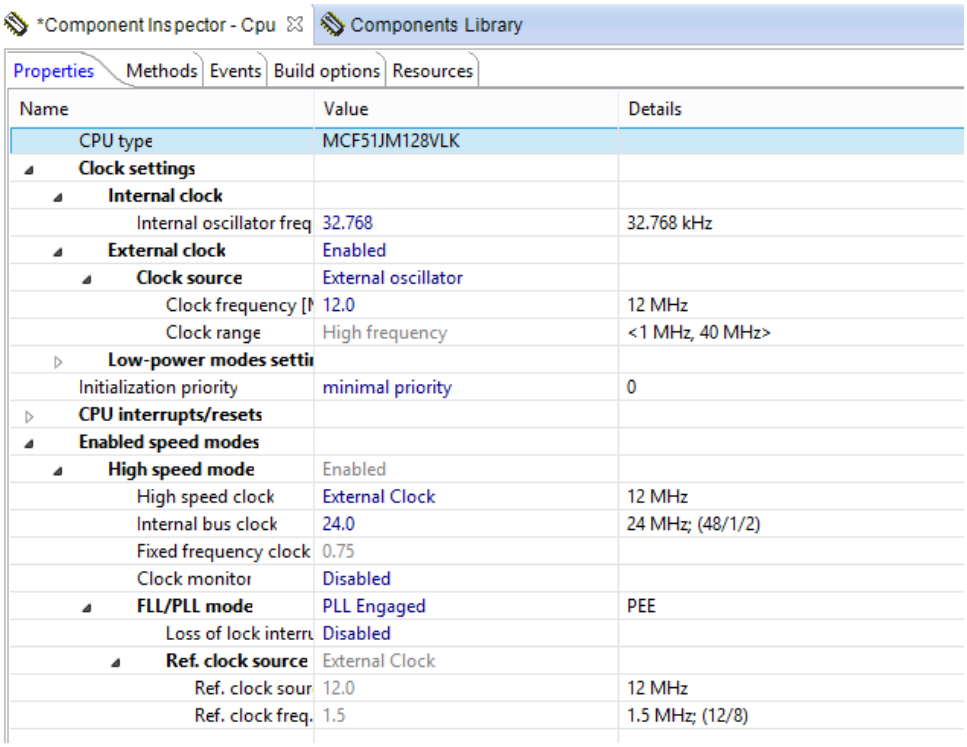


Figura 44 - Ventana de configuración de Processor Expert

A continuación se muestra el código generado a partir de la configuración de la Figura 44.

```
/*  System clock initialization */
if (*(unsignedchar*)0x03FFU != 0xFFU) { /* Test if the device trim
value is stored on the specified address */
MCGTRM = *(unsignedchar*)0x03FFU; /* Initialize MCGTRM register from a
non volatile memory */
MCGSC = *(unsignedchar*)0x03FEU;  /* Initialize MCGSC register from a
non volatile memory */
              }

/* MCGC2: BDIV=0,RANGE=1,HGO=1,LP=0,EREFS=1,ERCLKEN=1,EREFSTEN=0 */
setReg8(MCGC2, 0x36U);                /* Set MCGC2 register */
while(MCGSC_OSCINIT == 0U) {          /* Wait until external reference
is stable */
              }
/* MCGC3: DIV32=1 */
setReg8Bits(MCGC3, 0x10U);
/* MCGC1: CLKS=2,RDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
setReg8(MCGC1, 0x9AU);                /* Set MCGC1 register */
while(MCGSC_IREFST != 0U) {           /* Wait until external reference
is selected */
            }
/* MCGC3: LOLIE=0,PLLS=0,CME=0,DIV32=1,VDIV=3 */
setReg8(MCGC3, 0x13U);                /* Set MCGC3 register */
/* MCGC4: ??=0,??=0,DMX32=0,??=0,??=0,??=0,DRST_DRS=0 */
setReg8(MCGC4, 0x00U);                /* Set MCGC4 register */
```

```
while((MCGSC & 0x0CU) != 0x08U) {    /* Wait until external clock is
selected as a bus clock reference */
          }
/* MCGC2: BDIV=0,RANGE=1,HGO=1,LP=1,EREFS=1,ERCLKEN=1,EREFSTEN=0 */
setReg8(MCGC2, 0x3EU);               /* Set MCGC2 register */
/* MCGC1: CLKS=2,RDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
setReg8(MCGC1, 0x9AU);               /* Set MCGC1 register */
/* MCGC3: LOLIE=0,PLLS=1,CME=0,DIV32=0,VDIV=3 */
setReg8(MCGC3, 0x43U);               /* Set MCGC3 register */
while(MCGSC_PLLST == 0U) {           /* Wait until PLL is selected */
            }
/* MCGC2: LP=0 */
clrReg8Bits(MCGC2, 0x08U);
while(MCGSC_LOCK == 0U) {            /* Wait until PLL is locked */
            }
/* MCGC1: CLKS=0,RDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
setReg8(MCGC1, 0x1AU);               /* Set MCGC1 register */
while((MCGSC & 0x0CU) != 0x0CU) {    /* Wait until PLL clock is
selected as a bus clock reference */
               }
```

2.2.5.*Inter-Integrated Circuit* (S08IICV2)

Es un protocolo de comunicación desarrollado por Phillips en los 80s, con el objetivo de simplificar el conexionado de sus chips, pero en la actualidad todavía se le realizan actualizaciones al estándar.

Posee una topología maestro-esclavo basada en una interface de dos cables bidireccionales, SDA (*Serial Data*) y SCL (*Serial Clock*), las cuales cuentan con una configuración *open drain*, por lo que es necesario utilizar resistencias de *pull-up* externas.

Al igual que SPI, es utilizado para comunicarse con una gran cantidad de dispositivos, pero a diferencia de éste, las velocidades alcanzadas son menores ($\frac{Bus\ Clock}{20}$ $\frac{Bus\ Clock}{20}$ en el caso del MCF51JM128, dependiendo de la carga del bus), pero permite que hasta 127 dispositivos compartan el mismo bus de datos. Este protocolo hace que agregar dispositivos al bus resulte sencillo, ya que cada uno tendrá su propia dirección, lo que permitirá seleccionará sólo el dispositivo con el que se quiere iniciar una comunicación.

Características principales:

- Compatible con el estándar IIC
- Permite más de un maestro en el bus
- Transferencia de datos por interrupción byte a byte
- Generación y detección de señales de start stop
- Generación y detección de bit de ACK
- Detección de bus ocupado

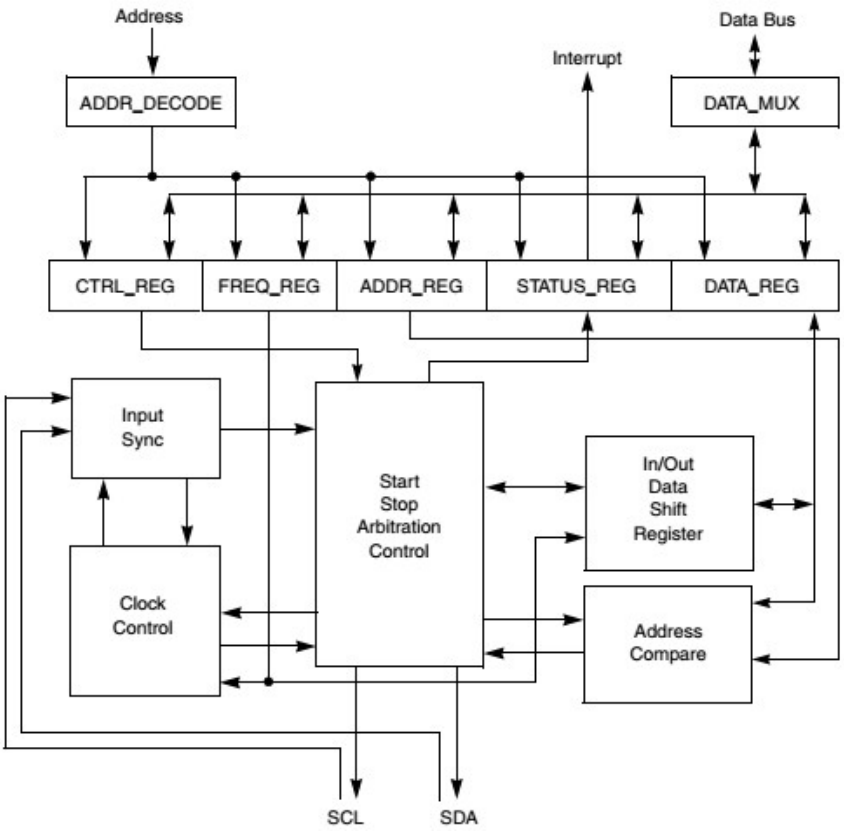


Figura 45 - Diagrama en bloques del módulo IIC

Los registros utilizados fueron los siguientes:

- **IIC *Frequency Divider Register* (IICxF):** Este registro permite configurar el *baud rate* del módulo, el tiempo de espera de la línea SDA y el tiempo de espera de comienzo de la línea SCL.

Figura 46 - Registro IICxF

Los bits del campo MULT multiplican al divisor SCL para obtener el *baud rate*. Mientras que los bits del campo ICR se utilizan para dividir el reloj del bus de datos.

A continuación se presenta la ecuación para calcular el *baud rate*.

Ec. 5

$$Baud\ Rate = \frac{bus\ clock\ [Hz]}{MULT \times SCL\ divider}$$

$$Baud\ Rate = \frac{bus\ clock\ [Hz]}{MULT \times SCL\ divider}$$

La Tabla 10 muestra valores posibles de tiempo de espera para obtener un *baud rate* de 100 Kbps si la velocidad del bus de datos es de 8 MHz.

MULT	ICR	Hold Times (µs)		
		SDA	SCL Start	SCL Stop
0x2	0x00	3.500	4.750	5.125
0x1	0x07	2.500	4.250	5.125
0x1	0x0B	2.250	4.000	5.250
0x0	0x14	2.125	4.000	5.250
0x0	0x18	1.125	3.000	5.500

Tabla 10 - Valores de tiempo de espera para distintos valores de MULT e ICR para un clock del bus de datos de 8 MHz

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SDA Hold (Stop) Value
00	20	7	6	11
01	22	7	7	12
02	24	8	8	13
03	26	8	9	14
04	28	9	10	15
05	30	9	11	16
06	34	10	13	18
07	40	10	16	21
08	28	7	10	15
09	32	7	12	17
0A	36	9	14	19
0B	40	9	16	21
0C	44	11	18	23
0D	48	11	20	25
0E	56	13	24	29
0F	68	13	30	35
10	48	9	18	25
11	56	9	22	29
12	64	13	26	33
13	72	13	30	37
14	80	17	34	41
15	88	17	38	45
16	104	21	46	53
17	128	21	58	65
18	80	9	38	41
19	96	9	46	49
1A	112	17	54	57
1B	128	17	62	65
1C	144	25	70	73
1D	160	25	78	81
1E	192	33	94	97
1F	240	33	118	121

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
20	160	17	78	81
21	192	17	94	97
22	224	33	110	113
23	256	33	126	129
24	288	49	142	145
25	320	49	158	161
26	384	65	190	193
27	480	65	238	241
28	320	33	158	161
29	384	33	190	193
2A	448	65	222	225
2B	512	65	254	257
2C	576	97	286	289
2D	640	97	318	321
2E	768	129	382	385
2F	960	129	478	481
30	640	65	318	321
31	768	65	382	385
32	896	129	446	449
33	1024	129	510	513
34	1152	193	574	577
35	1280	193	638	641
36	1536	257	766	769
37	1920	257	958	961
38	1280	129	638	641
39	1536	129	766	769
3A	1792	257	894	897
3B	2048	257	1022	1025
3C	2304	385	1150	1153
3D	2560	385	1278	1281
3E	3072	513	1534	1537
3F	3840	513	1918	1921

Tabla 11 - Divisor SCL y valores de espera

- **IIC *Control Register* (IICxC1):** Este registro permite activar el módulo IIC, activar las interrupciones, seleccionar el modo de operación (maestro o esclavo), seleccionar la dirección de la transferencia de datos, y setear o borrar el bit de ACK.

	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figura 47 - Registro IICxC1

2.3. Descripción del programa

El software fue desarrollado con el IDE Codewarrior 10.6, basado en Eclipse**[24]**, que en su versión gratuita posee un límite de 64KB de código en el caso del microcontrolador utilizado.

El programa se basa en el uso de interrupciones para adquirir los distintos datos requeridos por la aplicación, mientras que en el programa principal se analizan y procesan dichos datos.

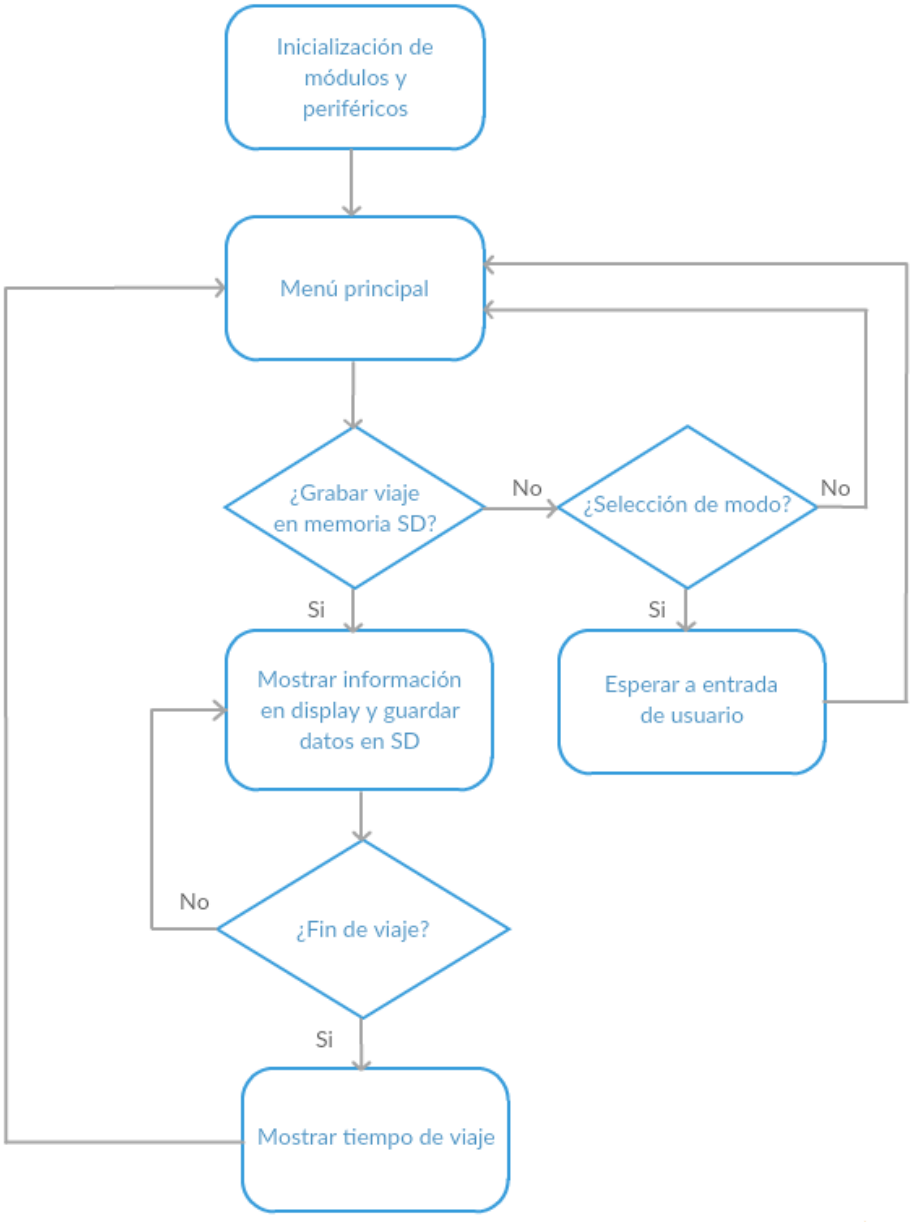


Figura 48 - Diagrama de flujo del programa principal

2.3.1. Configuración

En esta sección se inicializan los periféricos utilizados. La función MCU_Init() configura la frecuencia del bus de datos del microcontrolador en 24MHz, en base a un cristal de 12MHz (modo *PLL Engaged External*). También se inicializa el módulo I2C para realizar la comunicación con el acelerómetro. Se configuran las dos interfaces serie, SCI1 y SCI2, la primera es la correspondiente al módulo GPS por lo que se establece un *baud rate* de 9600 bps, mientras que la segunda es utilizada con el módulo Bluetooth HC05, que es la interface con el adaptador OBD2-Bluetooth, y la velocidad de transmisión se configura en 38400 bps. En ambas se habilitan las interrupciones por recepción y los módulos transmisor y receptor.

El *timer* 1 es configurado con interrupciones por *overflow* y se utiliza el reloj del bus de datos con un *prescaler* de 64 como fuente de reloj para el módulo. El módulo del contador se establece en 426, lo que produce que se ingrese a la rutina de interrupción cada 1,1ms. Este *timer* es usado para producir los tonos de advertencia mediante el zumbador externo.

El *timer* 2 se configura de la misma manera, pero el *prescaler* usado es de 128 y el módulo del contador es de 46875, lo que produce que se ingrese a la rutina de interrupción cada 250ms. La función de este *timer* es la de generar una base de tiempo para ejecutar diferentes tareas en el programa principal.

Por último se configuran los pines de entrada/salida correspondientes al LED bicolor, zumbador, los dos pulsadores y el pin de interrupción utilizado por el acelerómetro.

Luego de esto, se ejecutan las funciones SPI_init(), SD_init(), init_FAT_fs(), necesarias para la comunicación con la tarjeta SD. SPI_init() inicializa el módulo SPI a una frecuencia de 400 KHz. Las funciones SD_init() e init_FAT_fs() inicializan la tarjeta SD y el sistema de archivos FAT respectivamente.

Finalmente, se identifica el acelerómetro y se ejecutan las funciones Accelerometer_Init() y Calibrate(), la primera configura el acelerómetro para trabajar en el rango de +/- 2G y la segunda realiza la rutina de calibración requerida por el acelerómetro.

2.3.2. Programa principal

El programa principal consta de distintas tareas, las cuales se ejecutan en intervalos de tiempo definidos por los *timers* 1 y 2, o dependiendo de la recepción de datos por los módulos SCI1 y SCI2.

La recepción de datos está basada en un buffer circular mediante el uso de interrupciones. A medida que se reciben datos, si lo mismos corresponden a un

mensaje válido, se almacenan en el arreglo rxbuffer[], hasta que se detecta que se recibió el último caracter, momento en el que se habilita una bandera que indica que hay un dato completo en el buffer para ser analizado. De esta manera, en el programa principal se recorre el buffer de recepción para extraer la información necesaria. En el caso del GPS, que envía mensajes cada 1 segundo, primero se analiza si el mensaje corresponde a una sentencia GPGGA, si esto es así, se cuentan comas para extraer la información correspondiente a la hora, latitud y longitud.
Estructura de mensaje GPGGA:

```
$GPGGA,HHMMSS.SS,DDMM.MMMMM,K,DDDMM.MMMMM,L,N,QQ,PP.P,AAAA.AA,M,
±XX.XX,M,SSS,RRRR*CC<CR><LF>
```

Por ejemplo, contando una coma, y guardando los próximos 9 caracteres, obtendremos la información correspondiente a la hora.

Para los mensajes OBD-2, se envía una solicitud cada 250ms. Los mensajes solicitados son los correspondientes a la velocidad ("010D1\r"), RPM ("010C1\r") y el nivel del tanque de combustible ("012F1\r"). Luego, cuando se habilita la bandera de mensaje recibido en la interrupción de recepción, en el programa principal se lee el buffer de recepción para extraer la información. Como se mencionó anteriormente, un mensaje válido OBD-2 comienza con los caracteres 0x40 + número de modo, en este caso, como se trata de mensajes de modo 1, comenzarán con la cadena de caracteres 0x41, y a continuación sigue el PID solicitado (0x0D en el caso de la velocidad). De esta manera, el método para diferenciar las distintas respuestas consiste en comparar estos caracteres de respuesta correcta con los almacenados en el buffer de recepción. Si coinciden, sabremos de que tipo de mensaje se trata y podremos extraer el dato requerido.
Ejemplo de obtención del nivel del tanque de combustible:

```
if (strncmp(rxbuffer_OBD2,"41 2F",5) == 0)      /*Recibí un mensaje de
                                                 de combustible? */
      {
            fuel_raw_str[0] = rxbuffer_OBD2[6]; /*Guardo los datos*/
            fuel_raw_str[1] = rxbuffer_OBD2[7];
            fuel_uchar = (hex2uint8(fuel_raw_str)*100/255); /*Convierto
      de ASCII a binario*/
            fuel_ok = 1;      /*Indico que el mensaje se recibió
                              correctamente*/
      }
```

En el ejemplo se comparan los 5 primeros caracteres del buffer con "41 2F", que corresponde a una respuesta válida para una solicitud del nivel de combustible. Si la comparación es correcta, los próximos dos caracteres contendrán el dato requerido, por lo que se almacenan en el arreglo fuel_raw_str[2] para luego ser convertidos a formato *unsigned char* mediante la función hex2uint8() (convierte una cadena de caracteres en formato hexadecimal a un byte). También se lo multiplica por un factor para escalar el dato y convertirlo en un porcentaje del tanque de combustible. El mismo procedimiento se realiza para los mensajes restantes.

Cada vez que el acelerómetro envía la información de aceleración de los ejes X, Y y Z, se produce una interrupción en la que se habilita una bandera que indica que se recibió un dato. En el programa principal se almacena la información de aceleración en los 3 ejes, para ser convertida a fuerza G. En el caso que los valores de aceleración superen el umbral establecido, con el *timer* 1 se generará una señal de 440Hz que será reproducida por el zumbador y la misma servirá de alerta para el conductor.

Cuando el almacenamiento de datos se encuentra habilitado, los mismos se guardarán en la tarjeta de memoria en intervalos de tiempo de 1 segundo, establecidos por el *timer* 2. Los datos se escriben en el archivo generado en la rutina de inicialización y se encuentran separados por comas (formato .csv), lo que permite que sean convertidos a formato .kml de Google Earth para su posterior análisis. La información almacenada corresponde a la hora, latitud, longitud, aceleración en los 3 ejes, velocidad, RPM, nivel del tanque de combustible, tiempo de viaje, y tiempo de viaje con el auto en movimiento.

Los datos son representados en el display cada 50ms.

2.3.3. Rutinas de interrupción

El programa posee 5 rutinas de interrupción:

1. **Interrupción SCI1:** Se ejecuta con la recepción de cada mensaje del GPS. La rutina analiza si el carácter recibido se corresponde con el símbolo '$', de ser así, se almacenan los caracteres siguientes en el arreglo rxbuffer[]. Cuando se recibe el carácter '\n', se habilita la bandera rx_flag_GPS, para indicar que se recibió un mensaje completo y así realizar el análisis del buffer en el programa principal. Además se comprueba si el puntero de escritura del buffer circular superó el tamaño del buffer de recepción, y de ser así, se lo reinicia. Debido a que el módulo GPS envía mensajes cada 1 segundo, se utilizó esta característica para generar el reloj que lleva la cuenta del tiempo de viaje.

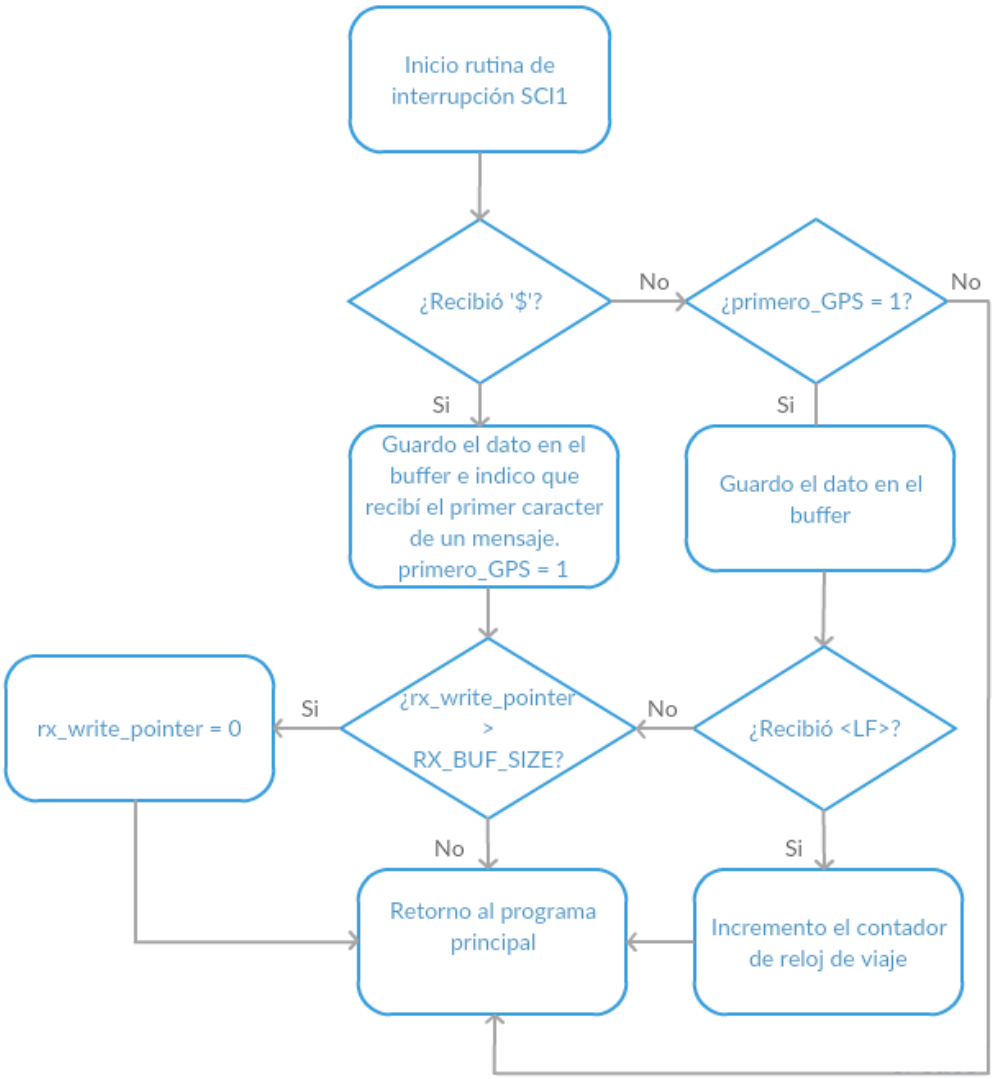


Figura 49 - Diagrama de flujo de la interrupción SCI1

2. **Interrupción SCI2:** Se ejecuta cuando se recibe un dato OBD-2. La estructura es similar a la interrupción del módulo SCI1, pero en este caso se analiza si el primer caracter es ‘4’.

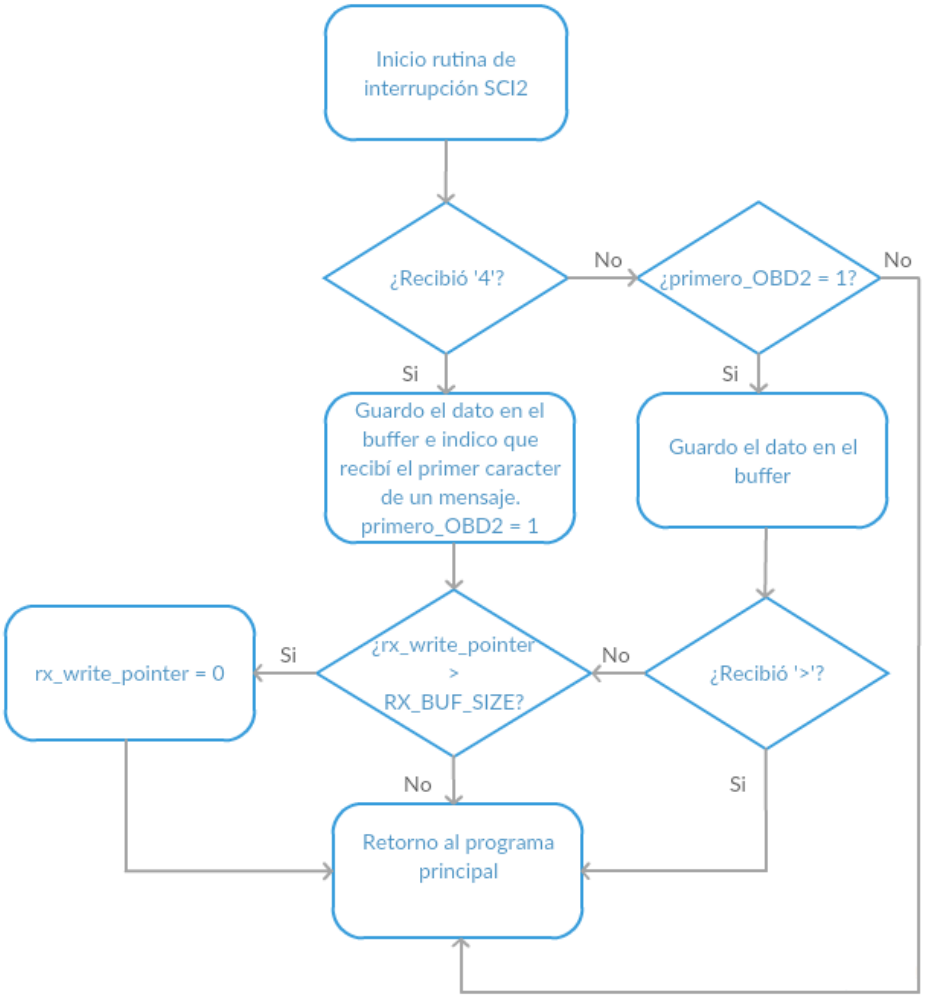


Figura 50 - Diagrama de flujo de la interrupción SCI2

3. **Interrupción TPM1:** Se genera cada 1,1ms. Su función es la de cambiar el estado del pin correspondiente al zumbador, para generar los tonos de advertencia. También genera la base de tiempo de 50ms para actualizar los datos en el display LCD.

4. **Interrupción TPM2:** Se produce cada 250ms. Produce 3 bases de tiempo, la primera de 250ms para enviar una solicitud al módulo OBD-2, la segunda de 1 segundo para escribir en la tarjeta de memoria en caso de que dicha funcionalidad se encuentre activada, y la última de 80 segundos, que se ejecuta en caso de que el auto se encuentre encendido, pero no se haya movido en un período de 80 segundos, lo cual generará un mensaje de advertencia al conductor.

5. **Interrupción PTG3:** Corresponde al pin 3 del puerto G y es utilizada por el acelerómetro para informar que posee un dato listo para ser leído por el microcontrolador.

Capítulo 3. Diseño y pruebas

El objetivo de este capítulo es presentar el desarrollo ya integrado. Para ello, se describirán las funcionalidades de cada uno de los bloques que lo componen. También se presentará el circuito esquemático y el diseño del circuito impreso del dispositivo. Luego se describirá el proceso de desarrollo del software y finalmente se mostrarán las pruebas realizadas sobre el dispositivo.

Se encuentra dividido en 5 secciones. La primera dedicada al circuito esquemático, la segunda al diseño del circuito impreso, en la tercera se mostrará el dispositivo terminado, en la cuarta se describirá el proceso de desarrollo del software y por último en la quinta se presentarán las pruebas realizadas.

3.1. Circuito esquemático

A continuación se presentará el circuito esquemático, dividido en dos secciones. La primera dedicada al microcontrolador y la circuitería externa mínima para el funcionamiento del mismo, y la segunda a los periféricos externos necesarios para la aplicación final.

3.1.1. Microcontrolador

En la Figura 51 se muestra el circuito esquemático mínimo del microcontrolador MCF51JM128.

En el mismo se encuentran los condensadores de desacople, que deben ir montados lo más cerca posible a los pines de alimentación del microcontrolador.

Se utilizó un pulsador normal abierto para el pin de *reset*, junto con un resistor de *pull-up* para mantener el pin en estado alto y evitar que quede flotando, y un condensador electrolítico que cumple la funcionalidad de anti-rebotes.

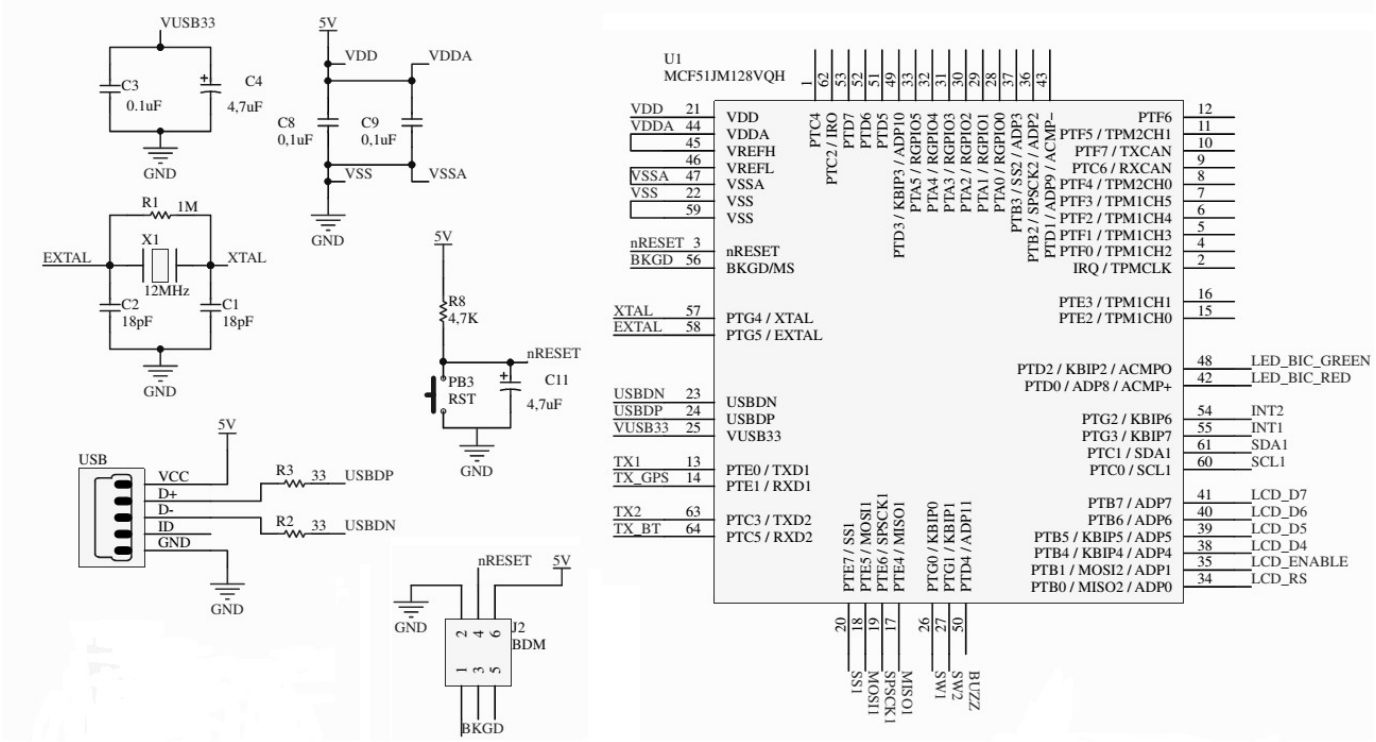


Figura 51 - Esquema del microcontrolador

Como se explicó en el capítulo 2, después de un *reset*, el microcontrolador utiliza un reloj interno generado por el módulo MCG, para luego utilizar el cristal externo en caso de lo que hubiere y se lo configure adecuadamente. En este caso se utilizó un cristal de 12 MHz con dos condensadores de 18pF y un resistor de 1MΩ. La función de dicho resistor es la de mantener al cristal dentro de su rango lineal durante la inicialización. Si bien el valor no es crítico, valores muy altos son sensibles a la humedad y valores muy bajos reducen la ganancia y pueden evitar que el cristal oscile, por lo que en la hoja de datos del microcontrolador se recomiendan valores entre 1MΩ y 10MΩ.

Para *debug* se utiliza la interfaz BDM similar a la disponible en los microcontroladores S08 de 8 bits. La misma utiliza una sola línea para comunicarse con el *debugger*. El conector estándar es de 6 pines distribuidos en dos filas de 3 pines cada una con una separación de 2,54mm, de los cuales se utilizan 4 pines, BKGD (transmisión y recepción de datos), RESET, 5V y GND.

Para la conexión USB se utilizó un conector mini-USB.

3.1.2. Periféricos externos

En la Figura 52 se muestran los periféricos externos. El módulo GPS y el módulo Bluetooth se encuentran conectados a las interfaces serie SCI1 y SCI2 respectivamente. Ambos poseen un divisor resistivo en la línea de recepción ya que el microcontrolador

está funcionando con una tensión de 5V, por lo que el nivel lógico alto de la línea de transmisión del mismo corresponde a 5V, mientras que las líneas de datos de los módulos antes mencionados funcionan con niveles lógicos de 3,3V. Este divisor no es necesario en las líneas de transmisión de los módulos, ya que el microcontrolador reconoce 3,3V como un nivel lógico alto.

La tarjeta SD se conecta a SPI1. En este caso el módulo ya cuenta con los conversores de nivel necesarios para adaptar los 5V del microcontrolador a los 3,3V con los que trabajan las tarjetas SD.

Los pulsadores SW1 y SW2 son los utilizados en la interfaz de usuario y están conectados a los pines PTG0 y PTG1 del microcontrolador.

El LED bicolor es el indicador visual de RPM y el ánodo rojo se encuentra conectado a PTD0 y el ánodo verde a PTD2, mientras que el cátodo está a masa. De esta manera es posible obtener tres colores, ya que si se escribe un “1” en los dos ánodos al mismo tiempo, el LED tendrá un color anaranjado.

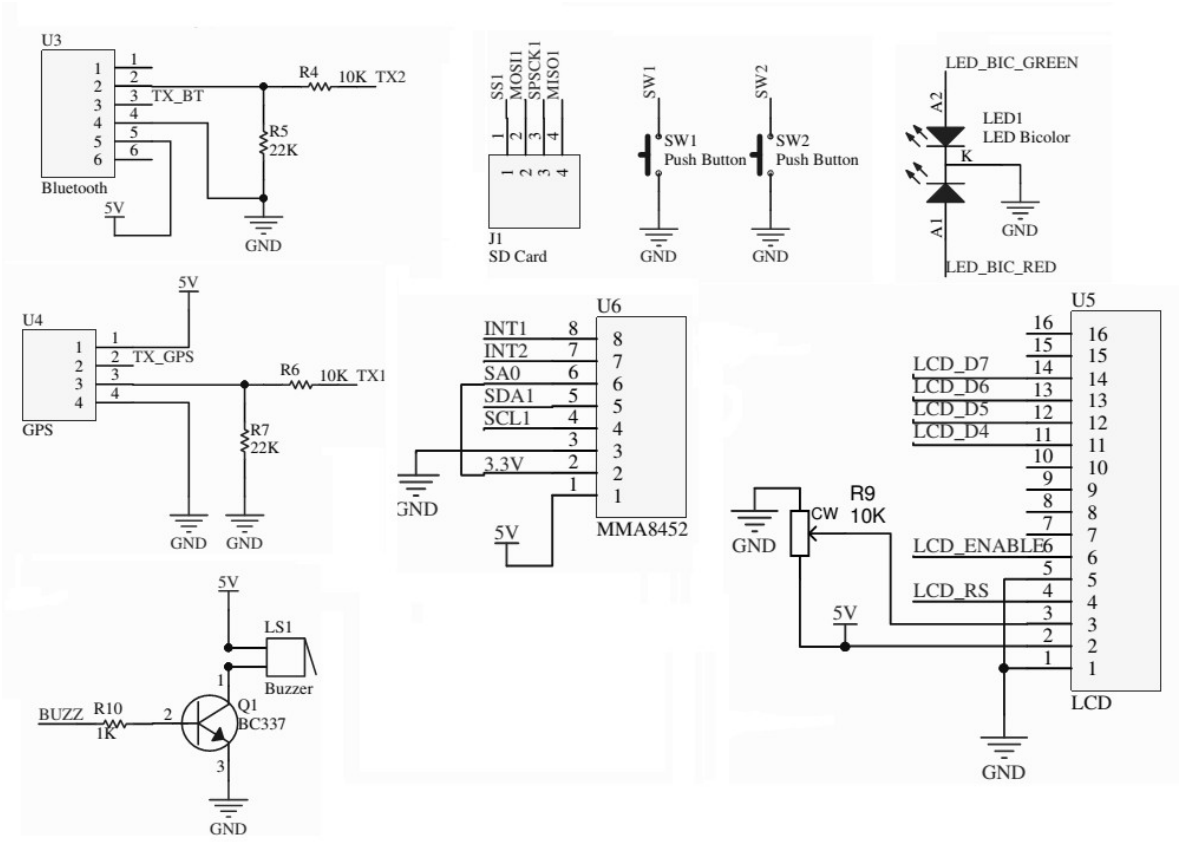


Figura 52 - Esquemático de los periféricos externos

El acelerómetro utiliza la interfaz IIC1 para los pines SDA y SCL. El pin de interrupción INT1 está conectado a PTG3 y genera una interrupción cada vez que los datos están disponibles para ser leídos por el acelerómetro. El pin de dirección SA0 se conecta a 3,3V, lo que define una dirección del dispositivo IIC de 0x1D.

El display fue utilizado en modo de 4 bits, por lo que sólo se necesitan las líneas de datos D4 a D7, conectadas a los pines PTB4 a PTB7. Se utiliza un preset para ajustar el contraste del display.

El pin PTD4 está conectado a un transistor NPN BC337 para controlar el zumbador de 5V cuya función es la de emitir alertas sonoras al conductor.

3.1.3. Fuente de alimentación

Para la fuente de alimentación (Figura 53), se utilizó un regulador lineal LM7805 **[25]** que entrega 5V de tensión continua a partir de los 12V disponibles en el automóvil.

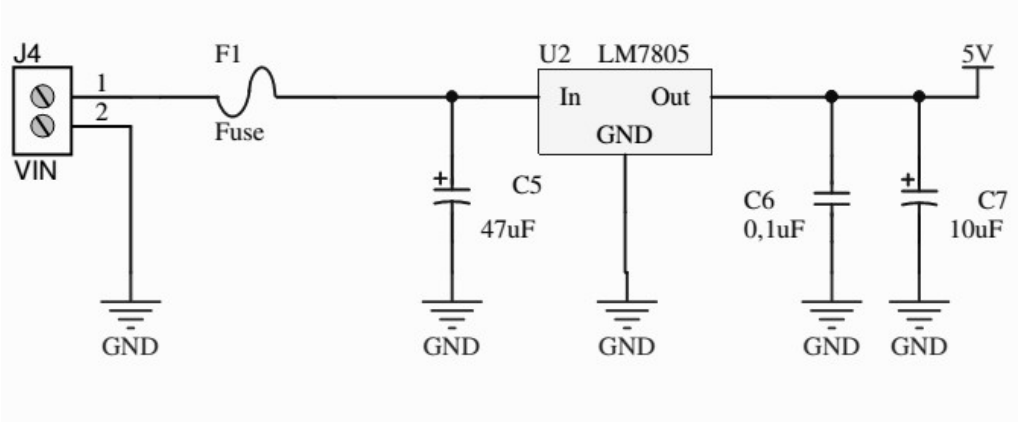


Figura 53 - Esquema de la fuente de alimentación

3.2. Diseño del circuito impreso

En la Figura 54se muestra el diseño del circuito impreso. En el mismo se encuentra el microcontrolador y posee los conectores para agregar los módulos GPS, acelerómetro, SD y LCD. Se utilizó un diseño modular, de modo que resultara sencillo realizar modificaciones y reparaciones durante la etapa de desarrollo.

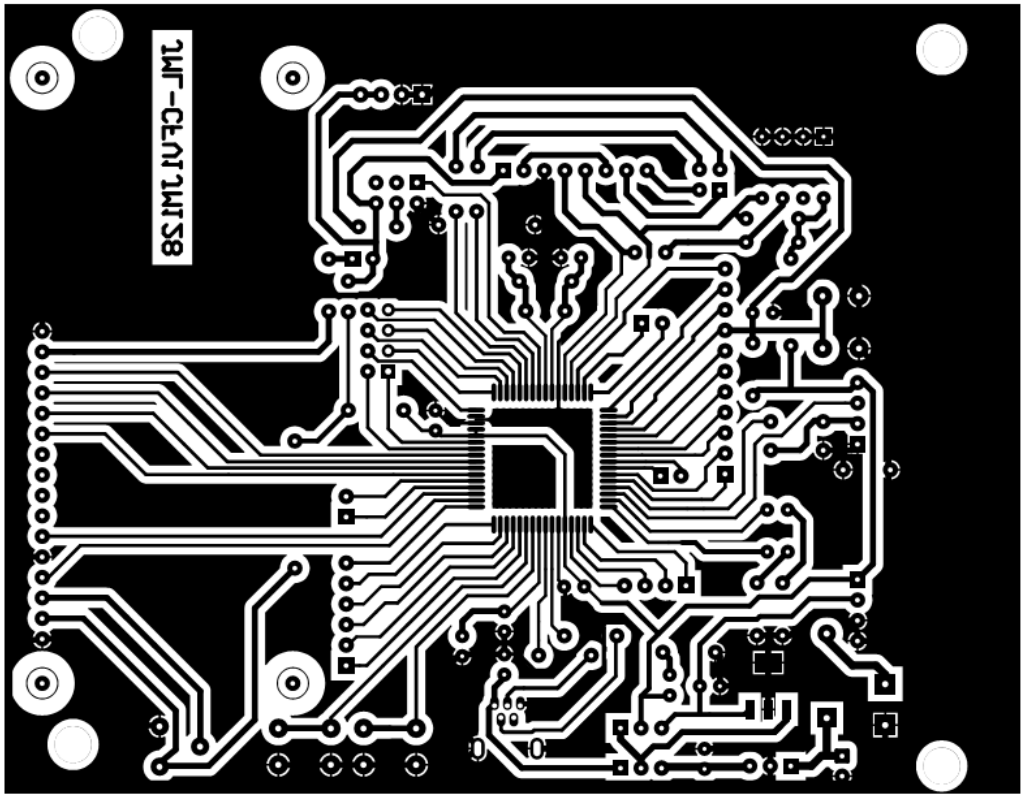


Figura 54 - Diseño de circuito impreso

El módulo Bluetooth y el zumbador fueron montados en una plaqueta de prueba (Figura 55), conectada mediante un cable plano a la placa principal.

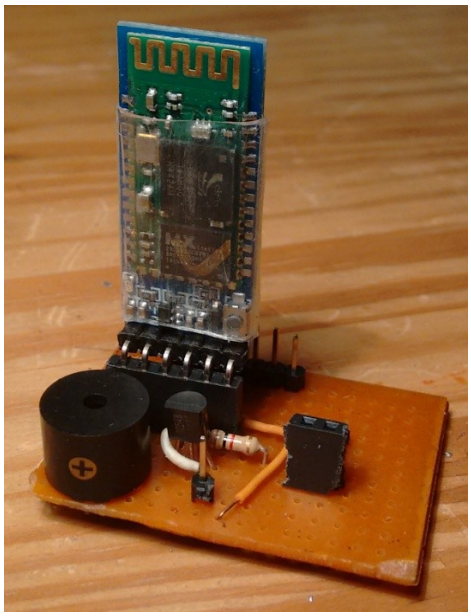


Figura 55 – Esta plaqueta contiene el módulo Bluetooth y el zumbador.

La Figura 56 y la Figura 57 muestran la plaqueta terminada. En la primera se ve un primer plano del microcontrolador MCF51JM128, mientras que en la segunda se muestra la plaqueta junto con un display de 16x2 (luego cambiado por uno de 20x4), el módulo GPS y el acelerómetro.

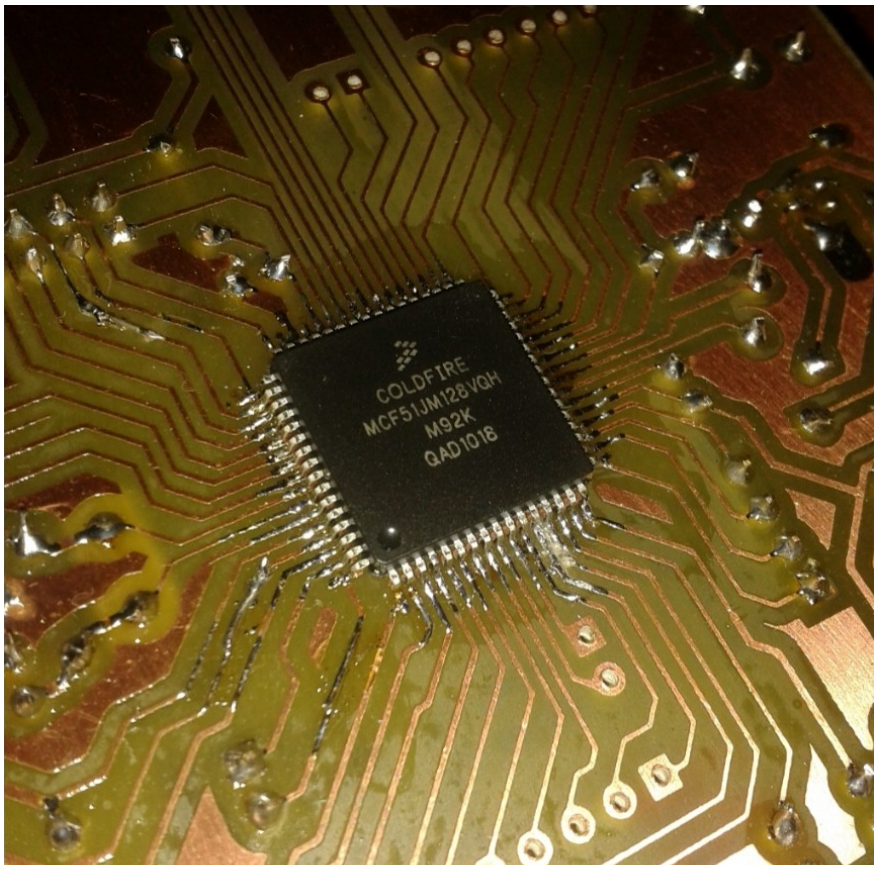


Figura 56 - Microcontrolador MCF51JM128 soldado en la plaqueta

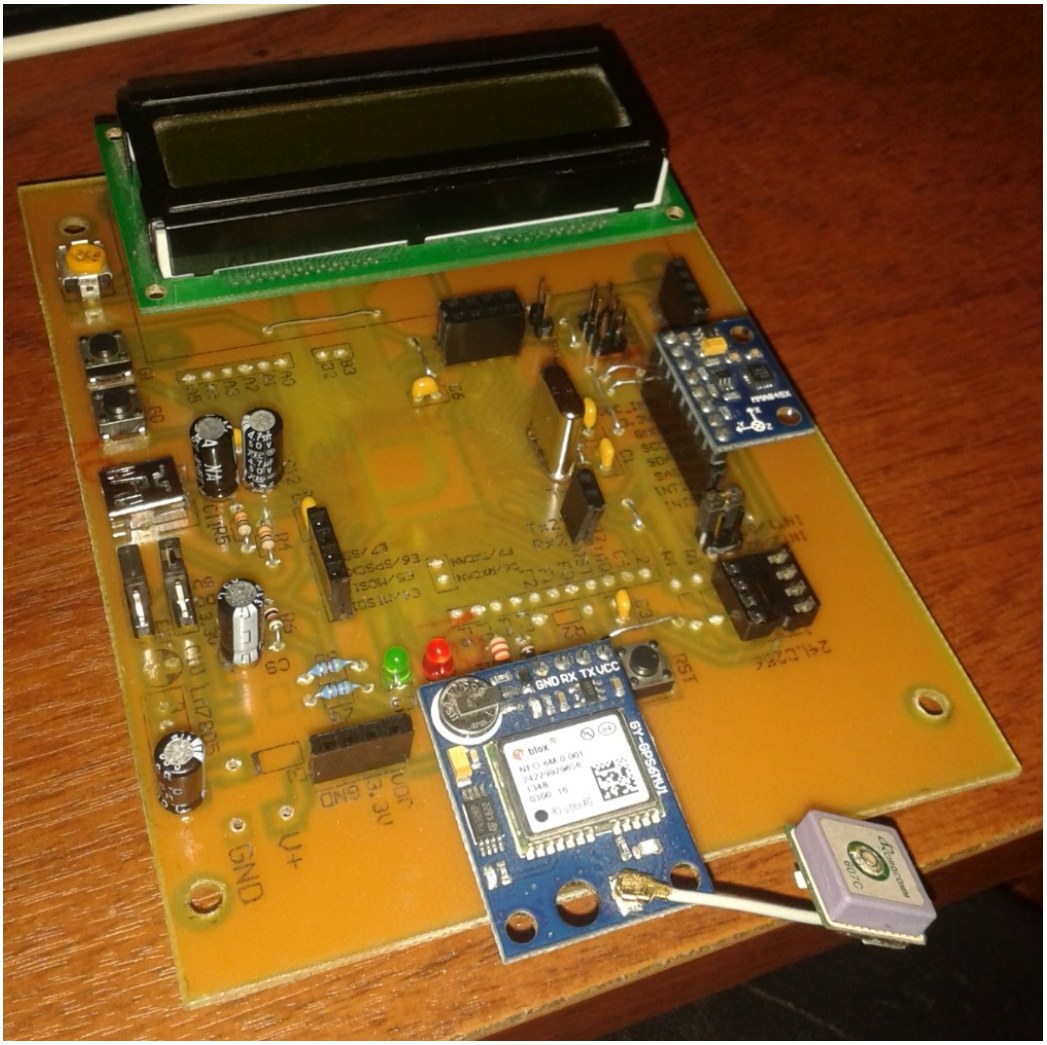


Figura 57 - Plaqueta con un display de 16x2, módulo GPS y acelerómetro

3.3. Dispositivo terminado

En esta sección se presenta el dispositivo terminado dentro de su gabinete.

Figura 58 - Frente del dispositivo con el display de 20x4, los dos pulsadores y el LED bicolor

Figura 59 - Jack de alimentación

Figura 60 - Ranura para la tarjeta SD, porta-fusible y puerto mini-USB

3.4. Desarrollo del software

A continuación se describirá el proceso de desarrollo del software del dispositivo, comenzando por la implementación de los distintos módulos, hasta la aplicación final.

Para simplificar esta tarea, se comenzó implementando programas sencillos para cada módulo de manera individual y una vez que se comprobó el correcto funcionamiento de los mismos, se combinaron en un solo programa.

3.4.1. Módulo GPS

En primera instancia se buscó implementar el código necesario para manejar el módulo GPS. Inicialmente, el programa funcionaba recibiendo todos los mensajes que enviaba el GPS y se los identificaba dentro de la rutina de interrupción de la interfaz serie. Esto traía como inconveniente que el microcontrolador se encontraba constantemente recibiendo datos que no necesitaba y se perdía mucho tiempo dentro de la rutina de interrupción. Sin embargo, esta primera iteración del programa fue útil para comprobar el correcto funcionamiento del módulo GPS, comparando los datos obtenidos con los proporcionados por el GPS de un teléfono celular.

Para evitar los inconvenientes mencionados se implementó un nuevo programa, esta vez configurando al GPS para que envíe solamente los mensajes NMEA necesarios y se utilizó un *buffer* circular para almacenar los datos recibidos en la rutina de interrupción. De esta manera, la búsqueda de un mensaje específico y el posterior procesamiento del dato se realiza en el programa principal, manteniendo la rutina de interrupción lo más chica posible.

3.4.2. Tarjeta SD

El siguiente paso fue adaptar la librería de la tarjeta SD para poder ser utilizada con el MCF51JM128, como se describió en el Capítulo 2. Una vez hecho esto, se unieron los dos programas (GPS y tarjeta SD) y se implementó un GPS *logger*. El mismo guarda los datos de hora, latitud y longitud cada 1 segundo en la tarjeta de memoria en un archivo de texto separado con comas, que luego es convertido a formato .kml para su visualización en Google Earth.

3.4.3. Comunicación OBD-2-Bluetooth

El próximo paso consistió en implementar el algoritmo para comunicarse con el puerto OBD-2 del vehículo. En un principio se utilizó un teléfono celular para enviar comandos OBD al adaptador OBD-2-Bluetooth, de manera de comprobar el buen funcionamiento del enlace y corroborar las respuestas recibidas con la información provista por la hoja de datos del ELM327. Así, se pudo conocer los PID soportados por el vehículo y que fueron utilizados para realizar la aplicación final.

Una vez hecho esto, se implementó un programa que envía solicitudes de datos OBD cada 250ms en el programa principal, mientras que las respuestas se guardan en un *buffer* en la rutina de interrupción de la interfaz serie, para luego ser procesadas en el programa principal.

3.4.4. Acelerómetro

Se implementó un programa para leer la información de aceleración de los tres ejes. El mismo se basa en la función del MMA8452 de generar una interrupción cada vez que los datos están disponibles para ser leídos.

3.4.5. Aplicación final

Con todos los módulos funcionando, se procedió a desarrollar la aplicación final. Primero se combinaron los módulos en un único programa y se comprobó el correcto funcionamiento de los mismos. Para esto se realizaron distintas pruebas de manejo, inicialmente adquiriendo solo un tipo de datos por la interfaz OBD-2, para que sea sencillo encontrar errores en el programa en caso de que algo no funcione correctamente. Una vez corroborado el buen funcionamiento del sistema, se agregaron los mensajes necesarios para la aplicación final y se realizaron nuevas pruebas para determinar los umbrales de aceleración peligrosos de modo de emitir las alertas sonoras. Finalmente, se implementó una interfaz de usuario sencilla mediante un display LCD y dos pulsadores, para facilitar la configuración del dispositivo por parte del conductor.

3.5. Pruebas realizadas

A continuación se presentan tres gráficos con datos recolectados en distintas situaciones de manejo. Los mismos se realizaron con el software Excel, importando el archivo generado por el microcontrolador como un archivo de texto separado por comas. De esta manera, Excel **[26]** interpreta cada separación como columnas distintas, cada una correspondiente a un tipo de datos. El eje X se encuentra se encuentra orientado en la dirección de avance, el eje Y hacia la izquierda y el eje Z hacia el techo del vehículo.

En el Gráfico 4se muestran valores de aceleración en g (1 G = 9,80665 $\frac{m}{s^2}\frac{m}{s^2}$) para los ejes X, Y y Z. Se puede observar como los valores en los ejes X e Y no superan los 0,1 G.

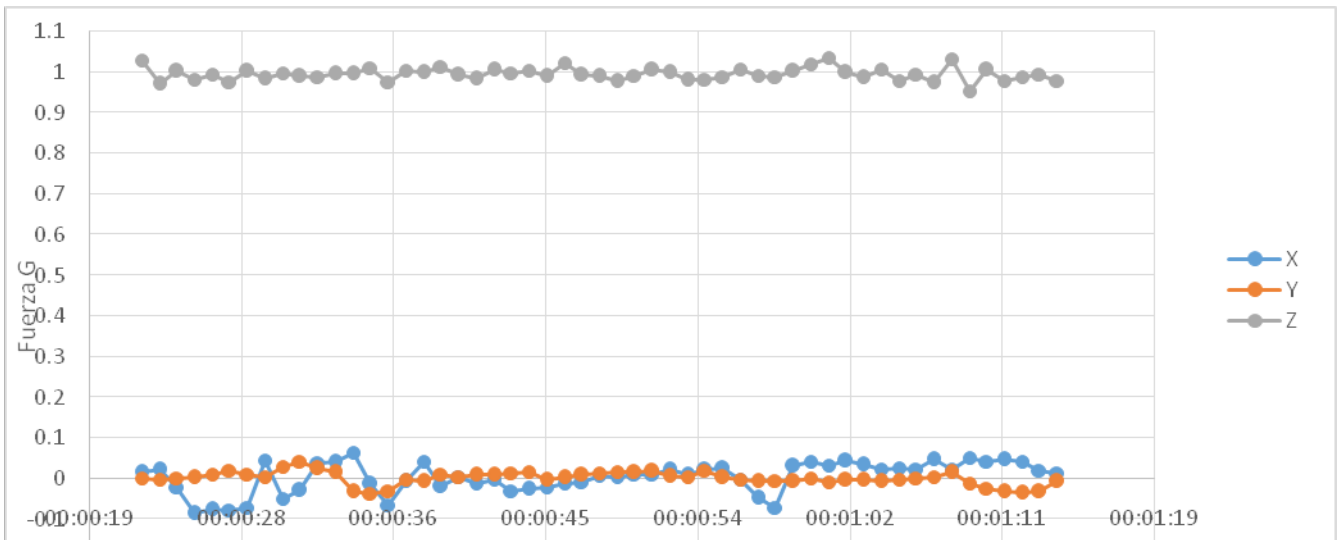


Gráfico 4 - Valores normales de aceleración

En el Gráfico 5se pueden observar distintos casos. Entre los minutos 7:36 y 7:40 se frena de manera normal y el valor de aceleración no supera los -0,15 G. A continuación entre 7:45 y 7:49 minutos se produce una aceleración excesiva llegando a un valor de 0,27 G. Finalmente, entre 7:49 y 7:53 se frena de forma brusca, llegando a una aceleración de -0,31 G.

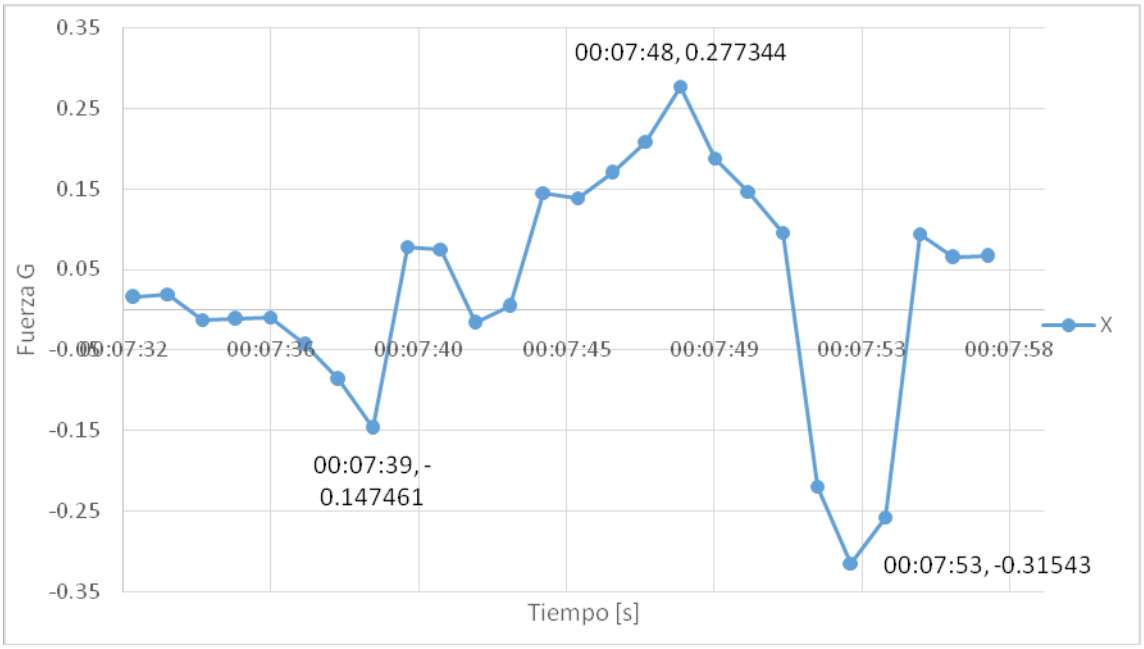


Gráfico 5 - Cambios bruscos en la aceleración

El Gráfico 6muestra un caso en el que se dobla en una esquina de manera agresiva. En 6:33 se produce un pico negativo de aceleración en el eje Y de -0,33 G.

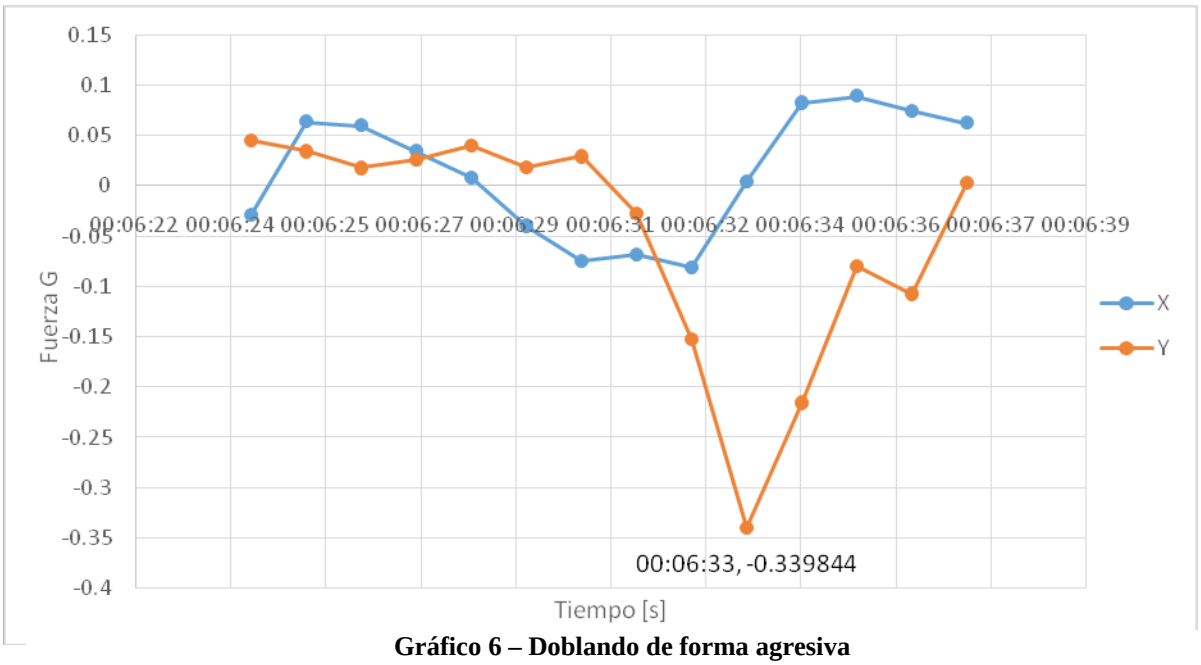


Gráfico 6 – Doblando de forma agresiva

Teniendo en cuenta los valores de aceleración en condiciones de manejo normal, y los valores correspondientes a los casos de manejo agresivo, fue posible establecer los umbrales de aceleración que producirán las alertas sonoras.

Ahora se muestran dos recorridos (Figura 61 y Figura 62), en los cuales se pueden observar los datos adquiridos que son guardados en la tarjeta SD cada 1 segundo.

Figura 61– Recorrido

Figura 62 – Recorrido

En cada punto se guarda hora, latitud, longitud, aceleración, velocidad, RPM, nivel del tanque de combustible, tiempo de viaje y tiempo de viaje con el auto en movimiento. Estos datos fueron corroborados con un GPS y la información provista por el tablero del auto.

En base a las pruebas realizadas puede decirse que el equipo cumplió con las expectativas, ya que las funciones implementadas respondieron de manera adecuada. En general los datos de ubicación recolectados con el GPS fueron precisos, al igual que los datos adquiridos por el puerto de diagnóstico y en el caso de los archivos de texto almacenados en la tarjeta de memoria, en ningún momento se produjo un archivo corrupto que no pueda ser leído por la PC. Las alertas sonoras también se produjeron de manera correcta de acuerdo a los umbrales de aceleración establecidos.

Capítulo 4. Conclusiones

En este proyecto se realizó un sistema de *Eco-driving* para ser instalado en distintos tipos de vehículos. El principal objetivo fue el de brindar asistencia técnica al conductor a que optimice sus hábitos de manejo, de modo de conducir más eficientemente y por consecuencia, que el automóvil emita menor cantidad de gases contaminantes. El trabajo fue dividido en cuatro capítulos.

En el Capítulo 1 se realizó una introducción estableciendo la problemática que da origen al *Eco-driving*. También se presentaron las características que debe tener un equipo electrónico para adquirir datos de manejo en tiempo real, como puede ser contar con GPS, acelerómetro, interfaz OBD-2, entre otros. Luego se describieron algunas alternativas disponibles comercialmente, disponibles en forma de aplicaciones para teléfonos celulares, y otras implementadas en hardware dedicado para tal fin. Para finalizar, se presentó el sistema propuesto que fue implementado en este trabajo, junto con las distintas partes que lo componen.

En el Capítulo 2 se describieron en detalle todos los módulos que componen el equipo, con sus características técnicas y comandos necesarios para que cumplan la función requerida. Estos módulos son el GPS, acelerómetro, tarjeta SD, adaptador OBD-2-Bluetooth, Bluetooth HC05 y display de 20 caracteres por 4 líneas. También se dedicó una sección al microcontrolador utilizado, el MCF51JM128 de Freescale, explicando los periféricos internos utilizados, juntos a sus registros de configuración. Éste fue seleccionado debido a la facilidad para escribir código para el mismo y al poder de procesamiento, ya que se trata de un microcontrolador de 32 bits y además permite frecuencias de reloj de hasta 50,33 MHz. También cuenta con los periféricos necesarios para comunicarse con los módulos externos, como dos puertos series, dos interfaces SPI, una interfaz IIC y una gran cantidad de puertos de entrada/salida. Finalmente se desarrolló el código implementado en el microcontrolador, explicando las funciones creadas para llevar a cabo la aplicación requerida.

En el Capítulo 3 se presentó el sistema completo. Esto incluye el esquema circuital del equipo, como así también el diseñó del circuito impreso. Luego se describió el proceso de desarrollo del software del microcontrolador, empezando por tareas sencillas como leer un dato del módulo GPS, hasta la implementación de la aplicación final. Por último, se mostraron las pruebas realizadas para determinar los umbrales de aceleración peligrosa, el correcto funcionamiento del GPS y corroborar la validez de los datos adquiridos mediante el puerto de diagnóstico.

El equipo desarrollado cumplió con las expectativas, ya que permite obtener datos confiables de manejo en tiempo real y gracias al almacenamiento de los mismos en la tarjeta de memoria, es posible analizar el desempeño del conductor, comprobando si realizó maniobras peligrosas o ineficientes. También la interfaz de usuario es simple e intuitiva, de manera que el conductor puede configurar rápidamente el equipo antes de iniciar el viaje, sin necesidad de tener que manipularlo

hasta finalizar el mismo. En el futuro se planea implementar un algoritmo para calcular la distancia recorrida en base a los datos de longitud y latitud provistos por el GPS, debido a que no es posible solicitar la información del odómetro mediante un PID estándar a través del puerto de diagnóstico, siendo este un dato que solo puede ser adquirido mediante un PID propietario, en el caso que el fabricante del vehículo haya decidido implementarlo. También se añadirá una memoria EEPROM para almacenar los parámetros configurados por el usuario y un bootloader USB, para facilitar las actualizaciones al software del microcontrolador. También se diseñará un nuevo circuito impreso, con el objetivo de disminuir las dimensiones del equipo.

Bibliografía

[1]European Commission website.
ec.europa.eu/clima/policies/transport/vehicles/index_en.htm

[2] Diesel o Gasolina website.
http://www.dieselogasolina.com/Noticias/View/neumaticos-como-afectan-al-consumo-de-combustible.html

[3] Impact of Vehicle Weight Reduction on Fuel Economy for Various Vehicle Architectures. Research Report. Ricardo Inc., 2008.
http://www.drivealuminum.org/research-resources/PDF/Research/2008/2008-Ricardo-Study.pdf

[4] Evaluating Eco-driving Advice using GPS and CANBus data. Karsten Jakobsen y Sabrine C. H. Mouritsen. Aalborg University.
http://projekter.aau.dk/projekter/files/77310278/EvaluatingEcodrivingAdviceUsingGPSAndCANBusData.pdf

[5] Eco-driving: Strategic, tactical, and operational decisions of the driver that improve vehicle fuel economy. Michael Sivak y Brandon Schoettle, University of Michigan Transportation Research Institute, 2011.
http://deepblue.lib.umich.edu/bitstream/handle/2027.42/86074/102758.pdf;jsessionid=710E207010350665F479EE525B08FFF4?sequence=1

[6] Analyzing Driving and Road Events via Smartphone. Nidhi Kalra, Gunjan Chugh y Divya Bansal. International Journal of Computer Applications, Volume 98, 2014.

[7] ScanGauge website. http://www.scangauge.com

[8] UltraGauge website. http://www.ultra-gauge.com/ultragauge/

[9] STR Eco-driving website. https://play.google.com/store/apps/details?id=com.cgi.str.ecodrivinglight&hl=es_419

[10] Camélys - Driving assistance website. https://play.google.com/store/apps/details?id=com.apikod.camelys.full&hl=es_419

[11] Google Earth website. https://www.google.com/earth/

[12] GPS Visualizer website. http://www.gpsvisualizer.com

[13] ELM-327 Data Sheet, ELM, 2014.
http://elmelectronics.com/DSheets/ELM327DS.pdf

[14] Módulo Bluetooth HC05 datasheet. http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf

[15] CSR website. http://www.csr.com

[16] NEO-6 Data Sheet, U-blox, Rev. E,2011.
https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf?utm_source=en%2Fimages%2Fdownloads%2FProduct_Docs%2FNEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf

[17] Librería tarjeta SD. http://github.com/suan/Freescale-SD-FAT-Library

[18] Freescale Semiconductor website. http://www.freescale.com

[19] MMA8452 Data Sheet, Freescale Semiconductor, Rev. 9.2,
2015.http://www.freescale.com/files/sensors/doc/data_sheet/MMA8452Q.pdf

[20] MCF51JM128 Freescale website.
http://www.freescale.com/products/more-processors/32-bit-mcu-and-mpu/coldfire-plus-coldfire-mcus-mpus/coldfire-mcus/v1-mcu/flexis-32-bit-v1-coldfire-usb-microcontroller:MCF51JM

[21] HD44780 Data Sheet, Hitachi, Rev. 0.0, 1999.
https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[22] MCF51JM128 Reference Manual, Freescale Semiconductor, Rev. 2, 2009.
http://www.freescale.com/files/32bit/doc/ref_manual/MCF51JM128RM.pdf

[23] Codewarrior IDE website.
http://www.freescale.com/tools/software-and-tools/software-development-tools/codewarrior-development-tools:CW_HOME?tid=vanCODEWARRIOR

[24] Eclipse website. https://eclipse.org

[25] LM7805 Data Sheet, Texas Instruments, Rev.2013.
http://www.ti.com/lit/ds/symlink/lm7815c.pdf

[26] Microsoft Excel website.https://products.office.com/es/excel