UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA MECÁNICA

"DISEÑO DE UNA CAJA NEGRA CON SUS SISTEMAS ELECTRÓNICOS, PARA UN VEHICULO VOLKSWAGEN POLO"

TRABAJO DE GRADUACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECÁNICO AUTOMOTRIZ

AUTORES:

SERGIO ALBERTO AGUILAR AGUILAR

PEDRO EDUARDO ASTUDILLO OCHOA

DIRECTOR:

ING. HUGO TORRES SALAMEA

CUENCA, ECUADOR

2010

DEDICATORIA

Primeramente a Dios que con su grandeza supo iluminarnos y darnos la fuerza para sobrellevar todos los obstáculos que se fueron presentando a lo largo de la carrera y que ahora vemos reflejado en nuestros logros. Yo Sergio Aguilar dedico este trabajo a mis Padres, Sergio y Amanda quienes con su ejemplo y cariño siempre estuvieron apoyándome a la distancia durante toda la etapa universitaria, a mi hermana Mayra que con sus buenos consejos y razonamientos ayudaron para finalizar con éxito mi carrera. Yo Pedro Astudillo dedico este trabajo a mi padre Pedro que con esfuerzo me ha brindado todo lo necesario para seguir adelante. A mi Madre Anita que con sus consejos me ha llevado a ser una persona mejor. A mis hermanos Patricia y Pablo, mis grandes amigos, siempre a mi lado, comprendiendo y apoyándome a pesar de todo en familia. Y por último a mis compañeros y amigos de carrera, con quienes hemos compartido tantas penas y alegrías.

AGRADECIMIENTO

Agradecemos a aquellas personas que nos dieron su ayuda no solo en el transcurso del trabajo sino en todo el proceso de aprendizaje en la universidad, al Ing. Paúl Tenesaca que fue un pilar importante en el desarrollo de la tesis, al Ing. José Cordero, que nos instruyo en cuestiones de programación, y al Ing. Hugo Torres por su ayuda y comprensión en todo el proceso de organización.

RESUMEN

El presente trabajo de grado se basa en el Diseño y Simulación de una caja negra para un vehículo Volkswagen Polo. El proceso incluyó un esquema de los principales circuitos electrónicos que recopilan señales básicas para el posterior análisis, guardando datos sobre la: aceleración, impacto, velocidad, cinturón de seguridad, pedal del freno, y posición del volante de dirección.

La tarjeta de almacenamiento de datos está constituida entre otros elementos por un microcontrolador previamente programado a conveniencia, para receptar las señales de dichos sensores. Cada uno de estos diseños vienen detallados en su ubicación y mecanismo electrónico dentro del vehículo.

ABSTRACT

In the present work, the design and simulation of a Black box for a Volkswagen Polo car was developed. The process included a scheme of the main electronic circuits, which stores basic signals to further analysis. Data from acceleration, impact, speed, safety belt, brake pedal and wheel drive position were stored in this devise.

Data collecting card is build up from a microcontroller previously programmed, to receive the signals from the sensors. The design process is developed regarding the position and electronic mechanism at the vehicle.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

Programación en MPLAB

Programación en MATLAB

Aguilar Aguilar Sergio Alberto

Astudillo Ochoa Pedro Eduardo

Trabajo de Graduación

Ing. Hugo Torres Salamea

Noviembre del 2010

DISEÑO DE UNA CAJA NEGRA CON SUS SISTEMAS ELECTRÓNICOS, PARA UN VEHÍCULO VOLKSWAGEN POLO

INTRODUCCION

La electrónica en la actualidad está siendo implementada en los vehículos modernos, para deducir el comportamiento del automotor a partir de varios sensores, que como resultado nos proporciona una mayor velocidad, potencia, seguridad, y se aprovecha de mejor forma la capacidad de los combustibles, mejorando las condiciones de confort, rendimiento y beneficiando de esta manera el transporte muy necesario en la actualidad, sin embargo, aun existen problemas con la impericia de algunos conductores que no respetan ciertas leyes establecidas en la sociedad, dando un mal uso al vehículo y siendo autores de accidentes con resultados en ocasiones fatales, muchas de las veces por descuido y en ciertas ocasiones por fallas mecánicas, dejando grandes dudas en el momento de señalar culpables, es por ese motivo que hemos visto la necesidad de diseñar un elemento que recepte las señales digitales de elementos clave al momento de una colisión, esto permitirá reconstruir las circunstancias de un accidente.

La utilización de un dispositivo de estas características ayudará a explicar que posibles problemas fueron causantes del suceso. La justificación de este trabajo de investigación es que al contar con un mecanismo que recepte información de ciertos sensores, se beneficiará a la ciudadanía, puesto que mediante este podremos determinar en qué condiciones se produjo determinado hecho.

Implementado un control adecuado de las diferentes señales que generan estos sensores determinaremos si el conductor del automotor respeta los límites de velocidad permitidos, así como las diferentes normas que se debe tener en cuenta al momento de conducir.

Esta tecnología permitiría conocer si el usuario del automotor incurrió en una conducta inapropiada al mando del mismo. Los propietarios de seguro estarían de acuerdo en implementar en su vehículo un dispositivo que pueda testificar sobre las circunstancias del accidente. Puesto que en ocasiones la falla tiene que ver con condiciones netamente mecánicas. Es lógico suponer que las compañías de transporte y aseguradoras aceptarían muy bien la intención de verificar el desempeño de sus empleados y el buen funcionamiento de la maquinaria.

Para nosotros uno de los objetivos es estudiar la forma de comportarse de los sensores que a nuestra consideración deben ser básicos en el sistema de una caja negra, para así poder realizar circuitos electrónicos que nos permitan receptar sus señales e interpretarlas. Los mecanismos que hemos seleccionado son: el sensor de aceleración, sensor de velocidad, sensor del pedal del freno, pulsante del cinturón de seguridad, sensor de impacto, sensor de posición angular del volante de dirección. Los parámetros que tendremos en cuenta son los alcances que tiene el modelo Volkswagen Polo con algunas variaciones que por defecto de diseño no cuenta este vehículo, dichas prestaciones y cambios iremos señalando en el transcurso del tema. Además del diseño de los circuitos electrónicos nos ayudaremos con un software que desarrollado a conveniencia nos permita captar, leer, receptar, convertir y visualizar gráficas donde se muestren los rangos de funcionamiento de los sensores analizados, para que sean traducidas por el investigador.

Intentando de esta manera brindar un proyecto que ya desarrollado ayude a esclarecer circunstancias muchas de las veces incomprensibles de los accidentes de tránsito que en nuestro medio son del diario vivir, pretendemos dar una herramienta novedosa adicional para contribuir al desarrollo de la tecnología en nuestra sociedad.

CAPITULO I

DESCRIPCIÓN DE LOS ELEMENTOS A UTILIZAR EN EL DISEÑO DE LA CAJA NEGRA

1.1 Generalidades

1.1.1 Introducción a los microcontroladores

"El PIC (*Programable Integrated Circuit*) microcontrolador es un circuito integrado programable, siendo su fabricante Microchip. Programable quiere decir que se puede planificar la manera de cómo va a funcionar, para poder adaptar a nuestras necesidades. En otras palabras el integrado es capaz de modificar su comportamiento en función de una serie de instrucciones que es posible comunicarle.

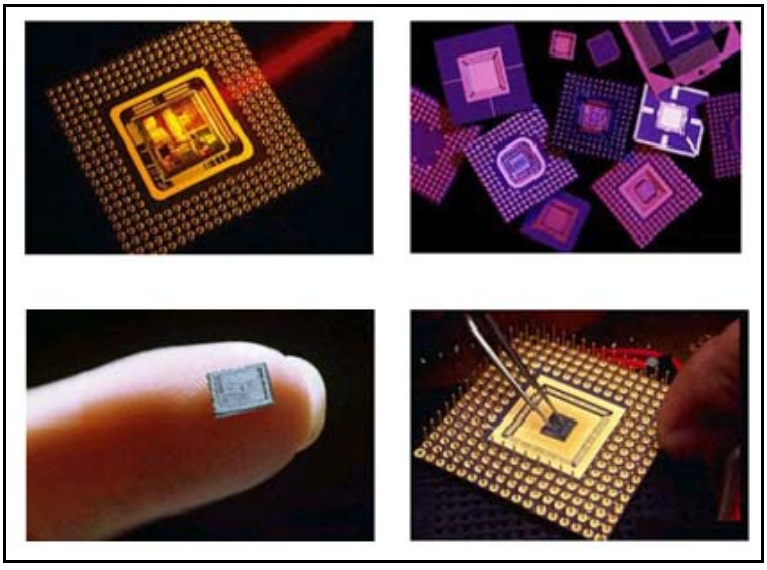


Figura 1: Microcontroladores

Fuente: RICHARD, James. 2008. Microcontroladores [en línea]. España. <http://www. corbisimages.com/Search#q=microcontroladores>. [consulta 17 de abril 2009]

PIC y PICMICRO para todos los fines prácticos describen el mismo microcontrolador ya que en 1997 Microchip registró el nombre PIC Micro para su línea de microcontroladores.

Los microcontroladores PIC de Microchip *Technology Inc*. combinan una alta calidad, bajo coste y excelente rendimiento. Un gran número de estos microcontroladores son usados en muchas aplicaciones tan comunes como: sistemas de control industrial, bioelectrónica, robótica, autotrónica, y por último en seguridad y aplicaciones dentro del sector de las telecomunicaciones."[1]

1.1.1.1 Aplicaciones de los microcontroladores:

- "En sistemas de comunicación: centrales telefónicas, transmisores, receptores, teléfonos fijos, celulares, fax, etc.
- En electrodomésticos: lavarropas, hornos de microondas, heladeras, lavavajillas, televisores, reproductores de dvd, minicomponentes, controles remotos, etc.
- Industria informática: Se encuentran en casi todos los periféricos tanto de entrada como de salida; ratones, teclados, impresoras, escáner, etc.
- Domótica: sistemas de alarma y seguridad, control de procesos hogareños a distancia, etc.
- Automoción: climatización, seguridad activa, seguridad pasiva, gestión con el motor, etc.
- Industria: Autómatas, control de procesos, etc.
- Otros: Instrumentación, electromedicina, ascensores, calefacción, aire acondicionado, sistemas de navegación, etc."[2]

---

[1] ANGULO, José María. 2003. Microcontroladores PIC: diseño práctico de aplicaciones. Tercera Edición. McGraw Hill. España. pp. 1,3

[2] Edibosco. 2002. Curso de Electrónica II: Componentes y circuitos básicos de la micro electrónica. Editorial Edibosco. Ecuador. Pp. 352, 353

1.1.1.2 Principales características del PIC18F4550

- "Procesador de arquitectura RISC avanzada: 16bit con 8 bit de datos
- Set de 77 instrucciones.
- Hasta 64Kbytes para la Memoria de Programa, tipo FLASH (hasta 2 Mbytes en ROMless).
- Multiplicador Hardware 8x8.
- Frecuencia máxima de reloj 48Mhz.
- Hasta 2048 Bytes de memoria de datos RAM.
- Hasta 256 Bytes de memoria de datos EEPROM.
- Hasta 20 fuentes de interrupción internas y externas.
- Periféricos de comunicación avanzados (CAN y USB).
- Modos de direccionamiento directo e indirecto.
- Temporizador *Power-on* (POP) y Oscilador Temporizador *Start-Up* (OST).
- Perro Guardián (WDT).
- Código de protección programable.
- Modo *Sleep* de bajo consumo. Solo necesita 5V para programarlo en este modo.
- Arquitectura tipo Harvard"[3] (*figura: 2*).

La arquitectura Harvard dispone de dos memorias independientes, una para datos y otra para instrucciones, permitiendo accesos simultáneos.

[3] MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›. [consulta 29 de mayo 2009]. Pp. 5

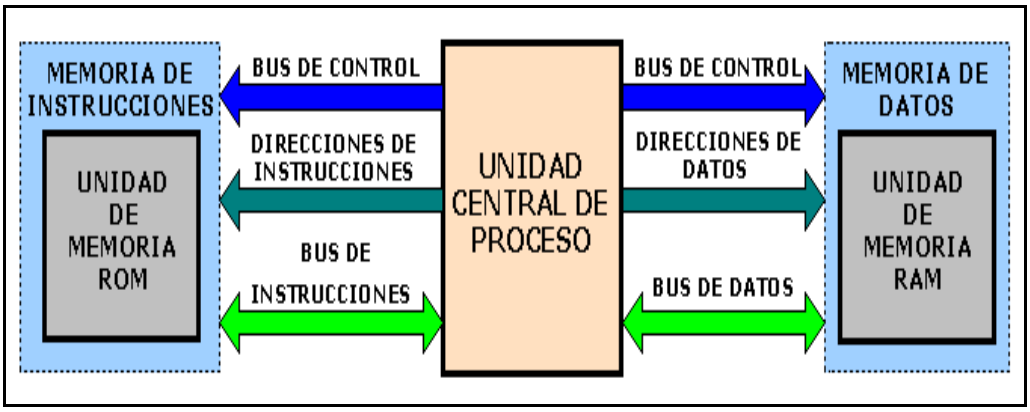


Figura 2: Arquitectura Tipo Harvard,

Fuente: ANGULO, José María. 2003. Microcontroladores PIC: diseño práctico de aplicaciones. Tercera Edición. McGraw Hill. España.

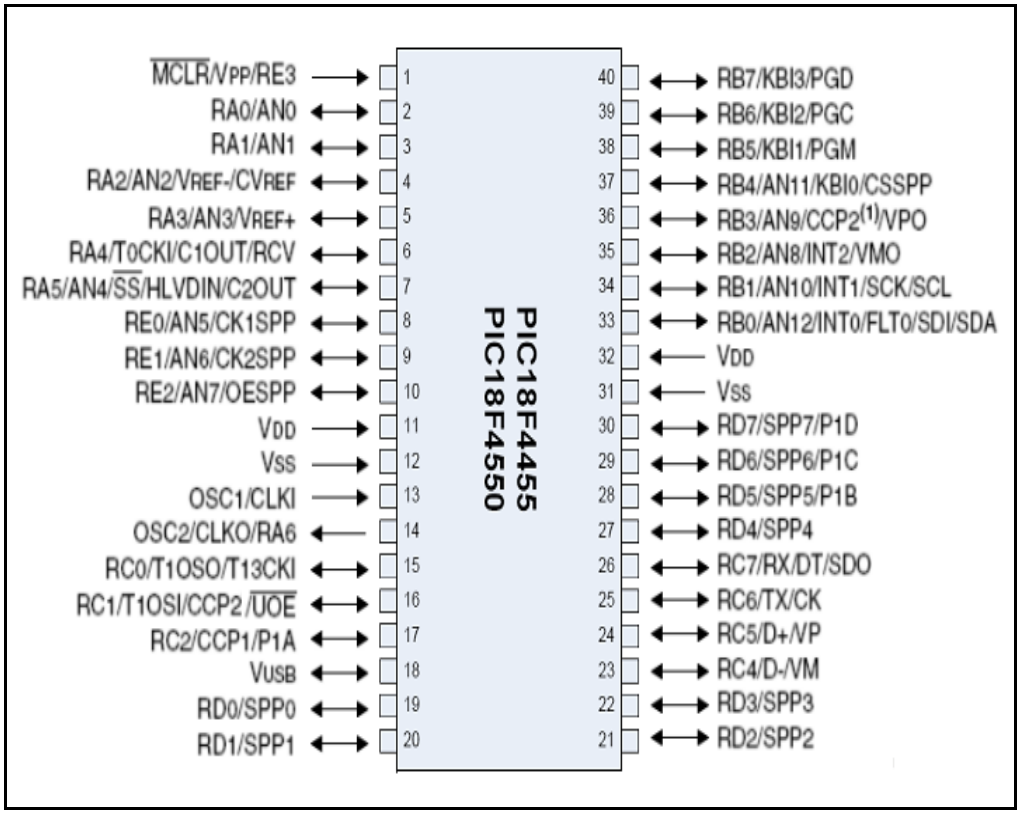


Figura 3: Diagrama de pines del PIC18F4550

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›. [consulta 29 de mayo 2009]. pp. 2

1.1.2 Recursos especiales del procesador:

1.1.2.1 Puertos de entrada y salida E/S

"Las entradas y salidas permiten que el microcontrolador se comunique con los dispositivos externos al Microcontrolador.

Entradas: Un dispositivo externo otorga al microcontrolador una señal en estado alto o bajo. El nivel lógico es leído por el microcontrolador como un bit sencillo de información de entrada.

Salidas: El microcontrolador forza uno de sus pines a un estado alto o bajo. El voltaje de salida en el pin corresponde a un bit sencillo de información.

Puertos: Un puerto es un grupo de pines utilizado para enviar o recibir información. Un puerto puede tener únicamente salidas, entradas o incluso una combinación de pines de entradas y salidas. Actualmente la mayoría de los puertos son bi-direccionales, es decir pueden ser configurados como pines de entrada o salida dependiendo de los requerimientos del usuario."[4]

Todas estas son líneas que manejan altas corrientes hasta 25 mA, se comporta como drenador o surtidor.

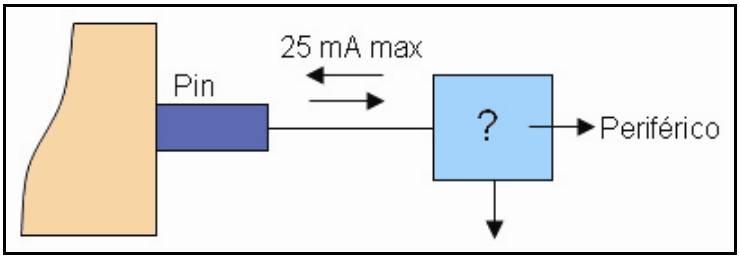


Figura 4: Líneas de entrada y salida

Las líneas de puerto incluyen internamente un b*uffer*.

[4] MANDADO, Enrique. 2006. Autómatas programables: entorno y aplicaciones. Primera edición. Editores Paraninfo. España. Pp. 338,345.

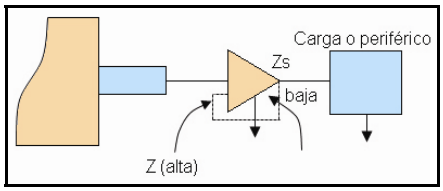


Figura 5: Líneas de puerto

El *buffer* es un circuito aislador de impedancia z; posee una impedancia de entrada alta y una impedancia de salida baja.

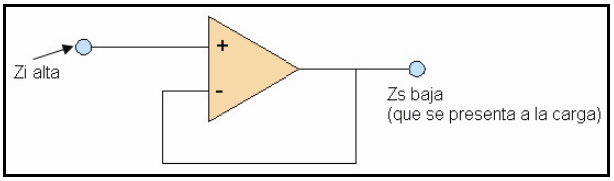


Figura 6: Buffer

Una de las características de los puertos es que pueden conectarse leds (diodo emisor de luz) sin necesidad de b*uffers*. Las líneas de entrada y salida se pueden programar como entradas o salidas pin a pin; modificando un bit de un registro denominado TRIS. Ejemplo 1:

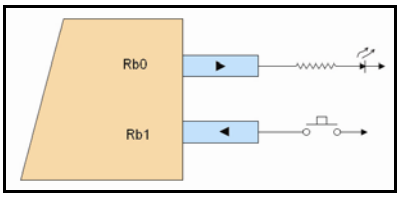


Figura 7: Periféricos de entrada y salida

Rb0: Periférico programado como salida.

Rb1: Periférico programado como entrada, son líneas de entrada y salida adyacentes.

Conclusión: Cada puerto tiene asociado un registro TRIS; por ejemplo un puerto A tiene un TRIS A. Por lo tanto si en un bit del registro TRIS correspondiente se escribe un 0, esto significa que el pin asociado a ese bit será una salida. Consecuentemente si se escribe un 1, significa que el pin en cuestión será una entrada. La función de la mayoría de las líneas de entrada y salida pueden multiplexarse con varias funciones.

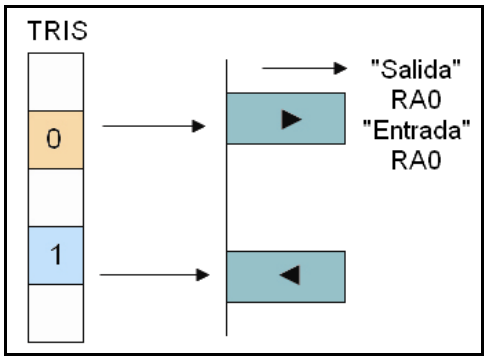


Figura 8: Registro tris

1.1.2.1.1 Características de las líneas E/S:

- "Unidireccionales o bidireccionales.
- Se configuran a través de un registro específico.
- Debe estar mapeados.
- Se accesan por medio de una dirección.
- Pueden ser TTL, CMOS, según sea el dispositivo con el que se comunican."[5]

Para el desarrollo de esta aplicación en concreto se utilizará el PIC MICRO 18F4550, que pertenece a la familia alta de MICROCHIP. "El PIC18F4550 dispone 5 puertos de E/S que incluyen un total de 35 líneas digitales de E/S:

PUERTO	LINEAS DE ENTRADA/SALIDA
PUERTO A	7 Líneas De Entrada/Salida
PUERTO B	8 Líneas De Entrada/Salida
PUERTO C	6 Líneas De Entrada/Salida + 2 Líneas De Entrada
PUERTO D	8 Líneas De Entrada/Salida
PUERTO E	3 Líneas De Entrada/Salida + 1 Líneas De Entrada

Tabla 1: Puertos del Microcontrolador 18F4550

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›. [consulta 29 de mayo 2009]. Pp.69

[5] LEHMANN, Stefan. 2008. Microcontroladores PIC: prácticas de programación. Primera edición. Ediciones Marcombo. España. Pp. 237, 239

Todas las líneas digitales de E/S disponen de al menos una función alternativa asociada a alguna circuitería especifica del microcontrolador. Cuando una línea trabaja en el modo alternativo no puede ser utilizada como línea digital de E/S estándar."[6]

1.1.2.2 Módulo convertidor analógico/digital (A/D)

El convertidor Analógico-digital (A/D) tiene 13 entradas para los dispositivos de 40/44 pines. Este módulo permite conversión de un signo de la entrada analógico a un 10-bit correspondiente el número digital.

A través de la entrada analógica se aplica la señal analógica a un condensador de captura y retención que después se introduce en el convertidor. El convertidor A/D es de aproximaciones sucesivas da como resultado una palabra de 10 bits. El módulo del convertidor A/D puede seleccionar como tensión de referencia la interna, es decir entre VDD y masa o bien una externa que se introduzca entre AN3 y AN2.

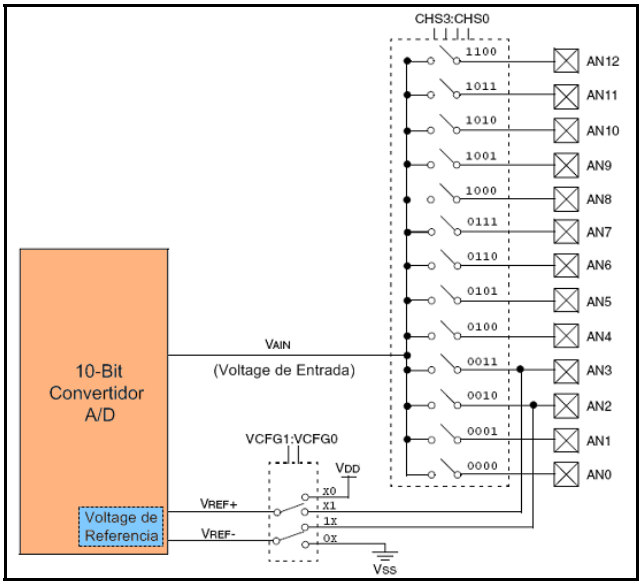


Figura 9: Diagrama del Convertidor Analógico/Digital (A/D)

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf› [consulta 29 de mayo 2009]. Pp.262

[6] MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/ en/DeviceDoc/80278a.pdf›. [consulta 31 de mayo 2009]. Pp.69

1.1.2.2.1 Características:

- “10 bits de resolución
- 13 canales multiplexados
- Señal de reloj de conversión configurable
- Tiempo de adquisición programable (0 a 20TAD)
- Posibilidad de establecer el rango de tensiones de conversión mediante tensiones de referencia externas.
- Permite que el sistema microcontrolador pueda procesar una variable analógica.
- Valor mínimo y máximo ajustable.
- Resolución: indica la precisión de la conversión realizada.
- Entre más cantidad de bits, más es la resolución del convertidor.
- Requieren configuración a través de registros especiales del microcontrolador.”[7]

El convertidor A/D tiene como característica especial de ser capaz de seguir trabajando mientras el dispositivo esté en el modo *Sleep*. Para ello el oscilador interno RC debe conectarse al conversor A/D. El módulo conversor A/D tiene asociados cinco registros de control.

Registros de control:

- ADCON 0
- ADCON 1
- ADCON 2

Registro de resultado:

- ADRES H (B – 0) Parte alta del resultado de la conversión

[7] MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/down loads/ en/DeviceDoc/80278a.pdf›. [consulta 3 de junio 2009]. Pp.107

- ADRES L (B – 1) Parte baja del resultado de la conversión

1.1.2.2.1.1 Registro ADCON 0

	-0	-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0
ADCON0	-	-	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON

Tabla 2: Registro ADCON 0

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf. [consulta 4 de junio 2009]. Pp. 108

- CHS3 a CHS0: Bits selección del canal de conversión A/D (13 canales)
- GO/DONE: Bit de inicio y de monitorización del estado de la conversión A/D:
  - GO/DONE= 0 : Proceso de conversión parado
  - GO/DONE= 1 : Proceso de conversión en marcha
- ADON: Bit de habilitación del convertidor A/D
  - ADON= 0 : Convertidor A/D desactivado
  - ADON= 1 : Convertidor A/D

1.1.2.2.1.2 Registro ADCON 1

	-0	-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0	L/E-0
ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0

Tabla 3: Registro ADCON 1

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf. [consulta 4 de junio 2009]. Pp. 109

- VCFG1: Bit de configuración de la tensión de referencia VREF-:

- VCFG1= 0 : VREF- se conecta a VSS
- VCFG1= 1 : VREF- se conecta a la línea física RA2

- VCFG0: Bit de configuración de la tensión de referencia VREF+:
  - VCFG1= 0 : VREF+ se conecta a VDD
  - VCFG1= 1 : VREF+ se conecta a la línea física RA3
- PCFG3 a PCFG0: Bits configuración de los puertos de conversión A/D. Mediante estos bits se establecen las líneas físicas (RA5 a RA0, RB4 a RB0, RE1 y RE0) que van a trabajar como entradas del convertidor A/D

1.1.2.2.1.3 Registro ADCON 2

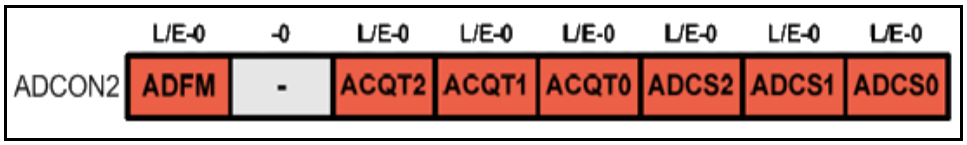


Tabla 4: Registro ADCON 2

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. http://www. microchip.com/downloads/en/DeviceDoc/80278a.pdf. [consulta 3 de junio 2009]. Pp. 110

- ADFM: Bit de configuración del tipo de almacenamiento del resultado de la conversión en los registros ADRESH y ADRESL:
  - ADFM= 0 : El resultado de la conversión se almacena con justificación a izquierdas
  - ADFM= 1 : El resultado de la conversión se almacena con justificación a derechas
- ACQT2 a ACQT0 : Bits de configuración del tiempo de adquisición
- ADCS2 a ADCS0: Bits selección de la señal de reloj del convertidor A/D.

1.1.3 Periféricos integrados a los microcontroladores

- **Periférico:** Dispositivo externo que intercambia datos con el procesador.
- **Temporizador:** Cuenta una base de tiempo fija que incrementa con una base de tiempo fija.
- **Contador:** Cuenta cuantos datos externos hay, muchas veces asincrónicos o asíncronos; no depende de un reloj, ejemplo: cantidad de pulsos.

1.1.3.1 *Timer* Integrado

1.1.3.1.1 Módulo *Timer* 0

"En la *Figura 10* se muestra un diagrama de bloques del t*imer* 0 y el *preescaler* que comparte con el WDT. El modo temporizador se selecciona poniendo a cero el bit T0CS (registro OPTION_REG <5>). En el modo temporizador, el módulo t*imer* 0 se incrementa en cada ciclo de instrucción (sin el *preescaler*). Si el registro TMR0 se escribe, el incremento se inhibe durante los siguientes dos ciclos de instrucción. El usuario puede trabajar teniendo en cuenta esto y ajustando el valor a cargar en el TMR0.

El modo contador se selecciona poniendo a uno el bit T0CS (registro OPTION_REG <5>). El modo contador, t*imer* 0 se incremento en cada flanco de subida o de bajada de la señal que le llega por RA4/TOCK1. El flanco de incremento se determina por el bit T0SE (registro OPTION_REG <4>). Poniéndose a cero T0SE se selecciona el flanco ascendente. El preescaler se comparte exclusivamente entre el *timer* 0 y el WDT. Y además no es de lectura/escritura."[8]

[8] ANGULO, José María. 2003. Microcontroladores PIC: diseño práctico de aplicaciones. Tercera Edición. McGraw Hill. España. Pp. 111

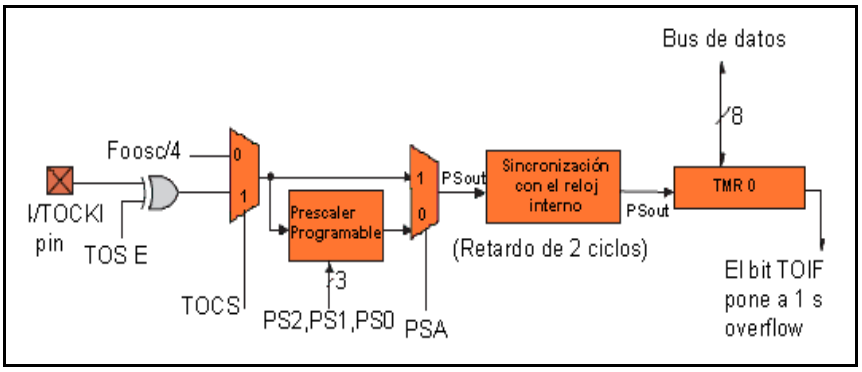


Figura 10: Diagrama de bloques del timer 0

Fuente: ANGULO, José María. 2003. Microcontroladores PIC: diseño práctico de aplicaciones. Tercera Edición. McGraw Hill. España. Pp. 98

El módulo *timer* 0 (*figura: 10*), es un temporizador/contador con las siguientes características:

- "El temporizador / contador dispone de 8 bits.
- Puede escribirse y leerse.
- Preescaler programable por software de 8 bits.
- Puede trabajar con el reloj interno o con una señal de reloj externa.
- Dispone de una interrupción por desbordamiento al pasar de FFh a 00h.
- Selección de flanco ascendente o descendente para el flanco del reloj externo."[9]

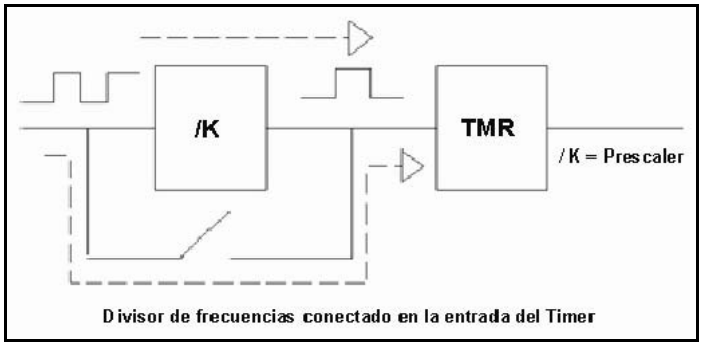


Figura 11: Módulo timer 0

Fuente: FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Cuenca, Ecuador. *Pp. 25*

[9] FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Cuenca, Ecuador. Pp. 52

1.1.3.2 Módulo CCP (modulación de anchura de pulsos)

"Los dispositivos de PIC18F4550 tienen dos CCP (*Capture*/ *Compare*/ PWM) (*figura: 12*). Cada módulo contiene un registro de 16-bit que puede operar como un registro de Captura de 16-bit. En los dispositivos de 40/44 pines, CCP1 se lleva a cabo como un módulo de CCP reforzado, con la captura normal compara y refuerza los modos de PWM."[10]

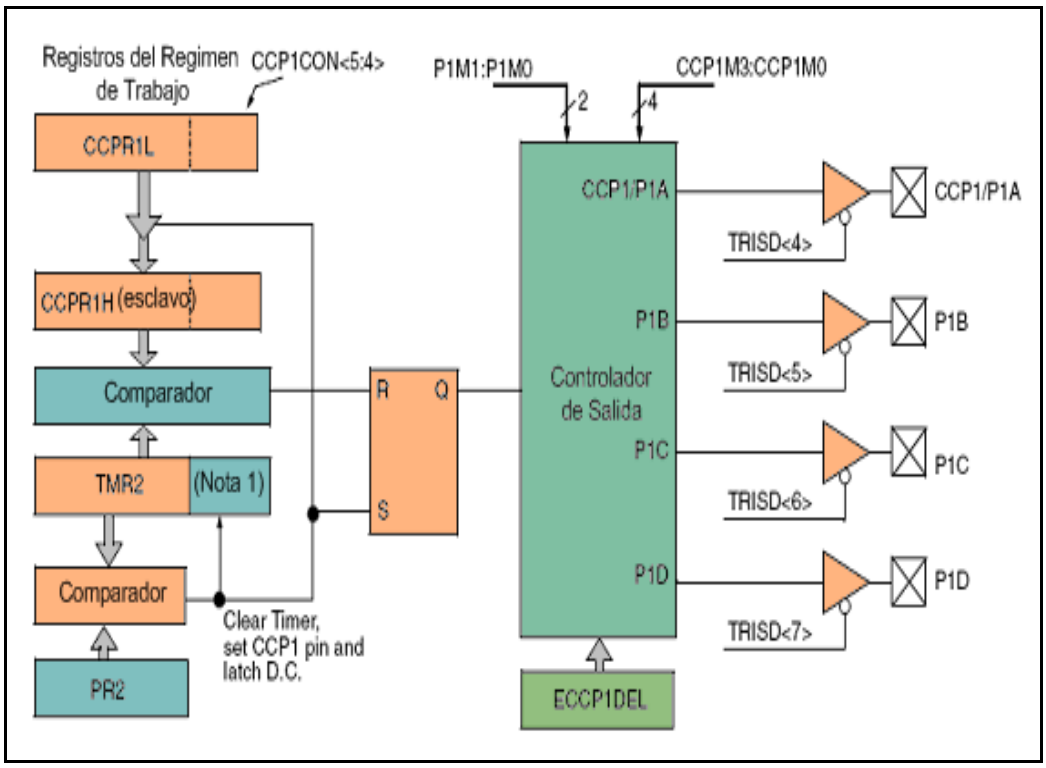


Figura 12: Diagrama de Bloques del módulo CCP en modo PWM

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf. [consulta 12 de junio 2009]. Pp. 151

Con el modo de modulación de anchura de pulsos se pueden conseguir impulsos a nivel alto de anchura variable.

[10] MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›. [consulta 13 de junio 2009]. Pp. 149

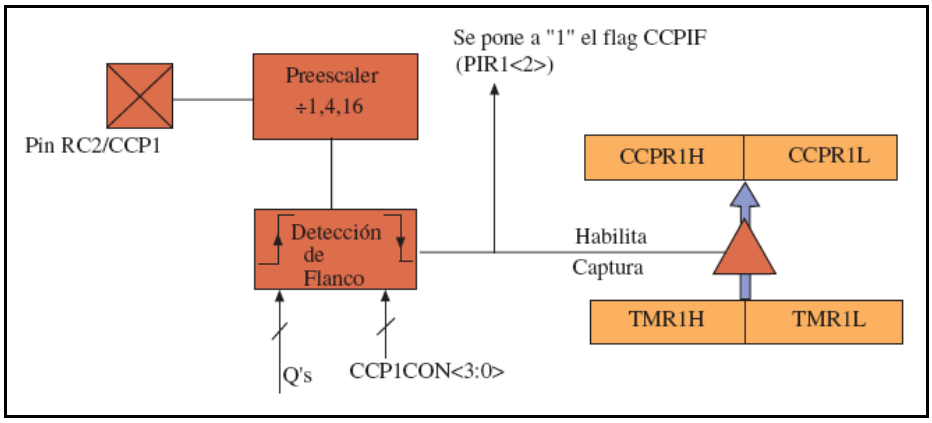


Figura 13: Diagrama de bloques del módulo de captura

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf. [consulta 15 de junio 2009]. Pp. 58

Es una técnica de modulación que codifica información en la variación del ciclo útil (*dutty cicle*) de un tren de pulsos, manteniendo siempre constante el período T, (*figura: 14*).

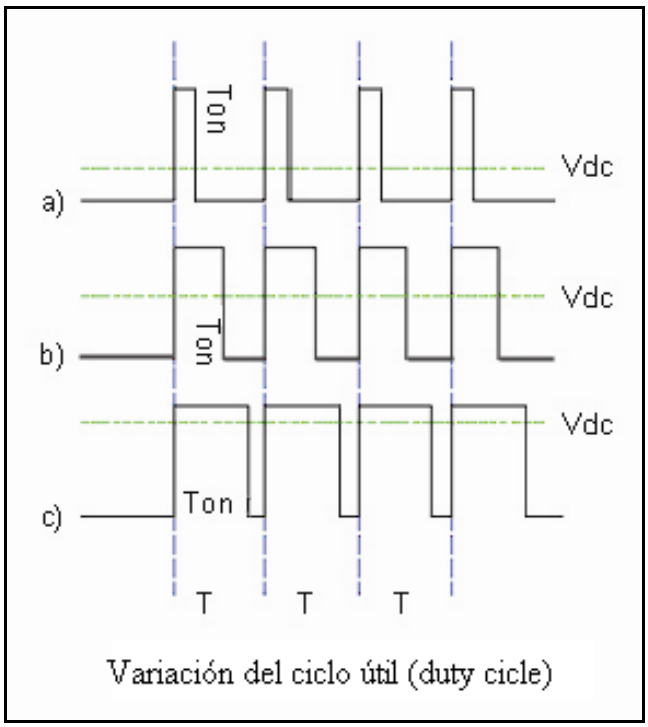


Figura 14: Modulación por ancho de pulsos

Fuente: FLORES, Marcelo. Curso de microcontroladores CEE. Ecuador. 2009. Pp. 46

Es un dispositivo que recibe un código digital de n bits, y de acuerdo con el valor, genera una señal cuadrada con un pulso alto de duración proporcional al valor recibido.

Duty Cicle:

C Util = Tiempo encendido/T = Ton/T

C Util % = Ton/T x 100

1.1.3.2.1 Aplicaciones:

- **Modulación:** Esta se la puede realizar de la siguiente manera:
  - Generación de señales BF, (*figura: 15*).
  - Control de reguladores de potencia como SCR y TRIAC.
  - Con el uso de ambos dispositivos (TRIAC y PWM) se regula la potencia que se aplica a una carga de corriente alterna, por ejemplo motores, bombillas, resistencias de hornos, etc. Regular la potencia de un motor de corriente alterna significa regular la velocidad de giro del mismo.

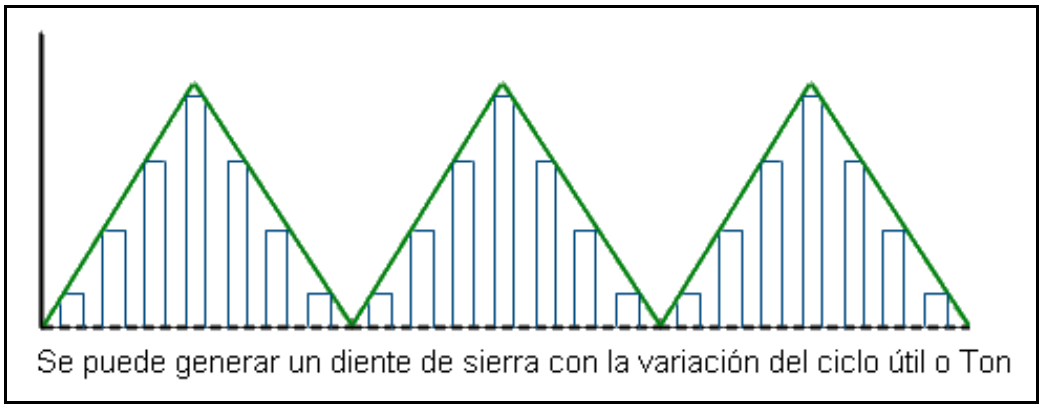


Figura 15: Generación de señales BF

Fuente: FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Cuenca, Ecuador. Pp. 52

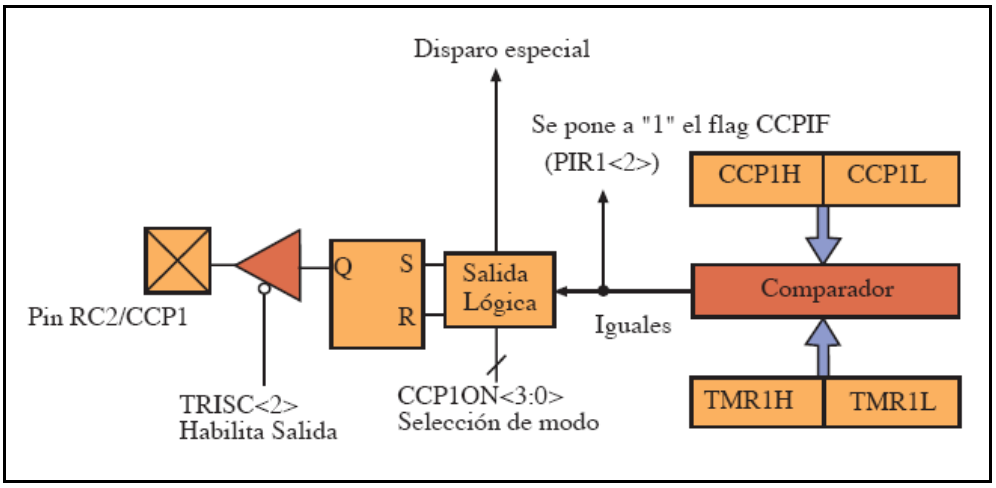


Figura 16: Diagrama de bloques de la forma de operar en el modo comparación

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›. [consulta 28 de junio 2009]. Pp. 58

1.1.4 La unidad central de proceso o CPU

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. "La función del CPU es realizar las tareas de:

- ***Fetch*:** La dirección de memoria de programa que se encuentra almacenada en el registro apuntador (PC) es utilizada para capturar la instrucción localizada en ésta dirección. La instrucción es copiada al registro de instrucciones (IR). El registro PC es incrementado para apuntar a la siguiente instrucción disponible.
- ***Decode*:** La instrucción localizada en el registro IR es decodificada. Es decir la representación en bits en el registro IR especifican determinada acción y es entonces cuando se generan señales de control y ajuste para preparar la ejecución de la instrucción.
- ***Execute*:** Las señales de control se distribuyen por todo el Microcontrolador, causando que la acción deseada se realice"[11].

[11] FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Cuenca, Ecuador. Pp. 12

Existen tres tipos de CPU en cuanto a la forma de procesar las instrucciones:

- **CISC:** Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones Complejo). Disponen de más de 80 instrucciones de máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución. Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros.
- **RISC:** Tanto la industria de los computadores comerciales como la de los microcontroladores, están decantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones de máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.
- **SISC:** En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es específico; o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores de Juego de Instrucciones Específico)."[12]

1.1.5 Memorias

"Una memoria es un dispositivo capaz de guardar el estado de un bit durante cierto tiempo, posee casillas o localidades cada una con la capacidad de almacenar un dato generalmente de tamaño byte (8 bits). Tiene un bus de direcciones para identificar cada una de las localidades y un bus de datos por donde entran y salen datos a cada una de las casillas o localidades de la memoria. Existen distintos tipos de memorias en un microcontrolador:

[12] MONOGRAFIAS.COM. 2009. Tipos de memorias [en línea]. México. ‹http://www.monografias.com/trabajos12/microco/microco.shtml#MERCA›. [consulta 10 de julio 2009].

- **RAM** (*Random Access Memory*)**:** Almacenamiento temporal de datos, pierde la información capturada cuando se le desconecta alimentación.
- **ROM** (*Read Only Memory*)**:** Es una memoria no volátil de sólo lectura, cuyo contenido se graba durante la fabricación del chip, conserva el contenido aun cuando se desconecta.
- **OTP** (*One Time Programmable*)**:** Es una memoria no volátil de sólo lectura programable una sola vez por el usuario. Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde una PC. La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas.
- **EPROM** (*Erasable Programable Read Only Memory*)**:** Funciona con el principio de fusibles, puede borrarse mediante luz ultravioleta, Se reprograma eléctricamente.
- **EEPROM** (*Electrical Erasable Programmable Read OnIy Memory*): son memorias de sólo lectura, programables y borrables eléctricamente EEPROM a través de la aplicación de una tensión de predisposición Vpp. Tanto la programación como el borrado se realizan eléctricamente desde el propio grabador y bajo el control programado de una PC. No disponen de ventana de cristal en la superficie. Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito.
- **FLASH**: Funciona igual que la EEPROM pero a una velocidad de operación y programación mayor".[13]

1.1.5.1 Características de las memorias

Algunas de las características fundamentales de las memorias de cualquier tipo son las que a continuación se detalla:

[13] ANGULO, José María. 2003. Microcontroladores PIC: diseño práctico de aplicaciones. Tercera Edición. McGraw Hill. España. Pp. 6, 7

- **"Volatilidad:** Se dice que la información almacenada en una memoria es volátil siempre y cuando corra el riesgo de verse alterada en caso de que se produzca algún fallo de suministro de energía eléctrica (memorias biestables). Por esta simple razón específica, las memorias RAM y ROM son volátiles.
- **Tiempo de Acceso:** Es el tiempo que transcurre desde el instante en que se lanza la operación de lectura en la memoria y el instante en que se dispone de la primera información buscada. Es simplemente eso, el tiempo que se solicita a la memoria para poder ejecutar cualquier operación específica.
- **Capacidad:** A mayor capacidad, mayor velocidad. A la hora de escoger una memoria, se debe elegir un valor que sea óptimo (sea de 512 megabytes, 1 gigabyte) para tener mejor rendimiento en la computadora."[14]

1.1.6 Ventajas y desventajas de los PIC18F4550

1.1.6.1 Ventajas:

- "Reducción de la cantidad de espacio en la implementación de un diseño dado.
- Reduce el costo de implementación.
- Desarrollo de aplicaciones específicas de manera más rápida y eficiente.
- Los fabricantes dan mucho soporte sobre las aplicaciones más comunes.
- Sencillez de manejo.
- Compatibilidad del software en todos los modelos dentro de una misma familia
- fiabilidad: disminuye el riesgo de averías, se precisan menos ajustes.
- Mayor flexibilidad: las características de control están programadas por lo que su modificación sólo necesita cambios en el programa de instrucciones.
- Como el microcontrolador sólo se destina a una tarea en la memoria ROM, sólo hay que almacenar un único programa de trabajo.

[14] GOMEZ, Eddy. 2008. Características de las memorias [en línea]. México. ‹http://www.Mi tecnologico.com/Main/CaracteristicasMemorias›. [consulta 22 de julio 2009].

- Algunas de las familias de microcontroladores actuales se desarrollaron pensando en el sector automotriz (18% de la producción mundial) que es uno de los mercados más exigentes en el que los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. y seguir siendo fiables.

1.1.6.2 Desventajas:

- Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada.
- No existen sistemas de almacenamiento masivo como disco duro o *disquetes*"[15].

1.2 Diagramas de bloque

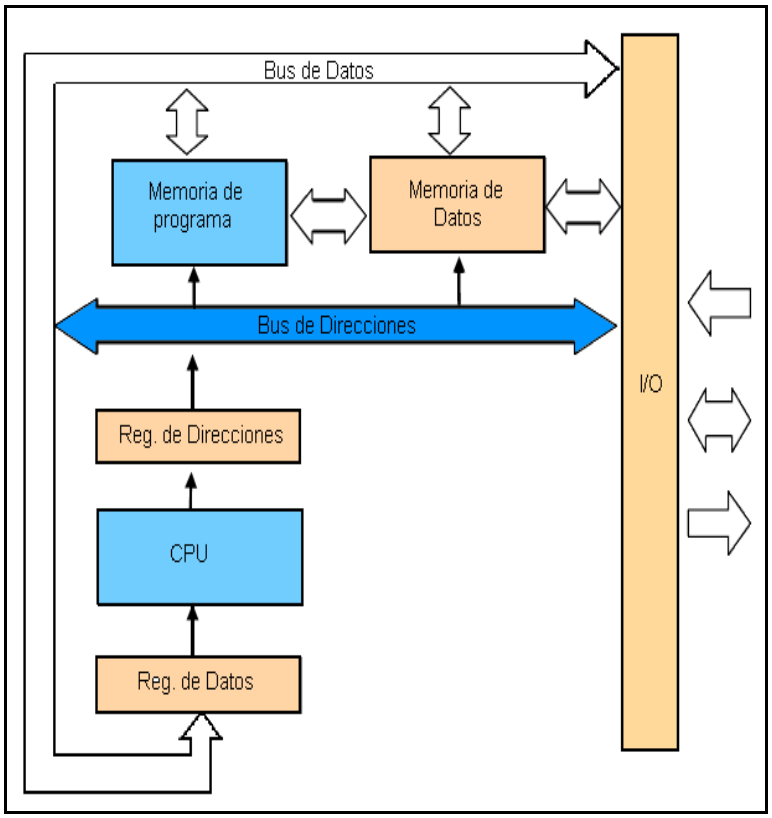


Figura 17: Diagrama de bloque del sistema de la caja negra

[15] FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Cuenca, Ecuador.

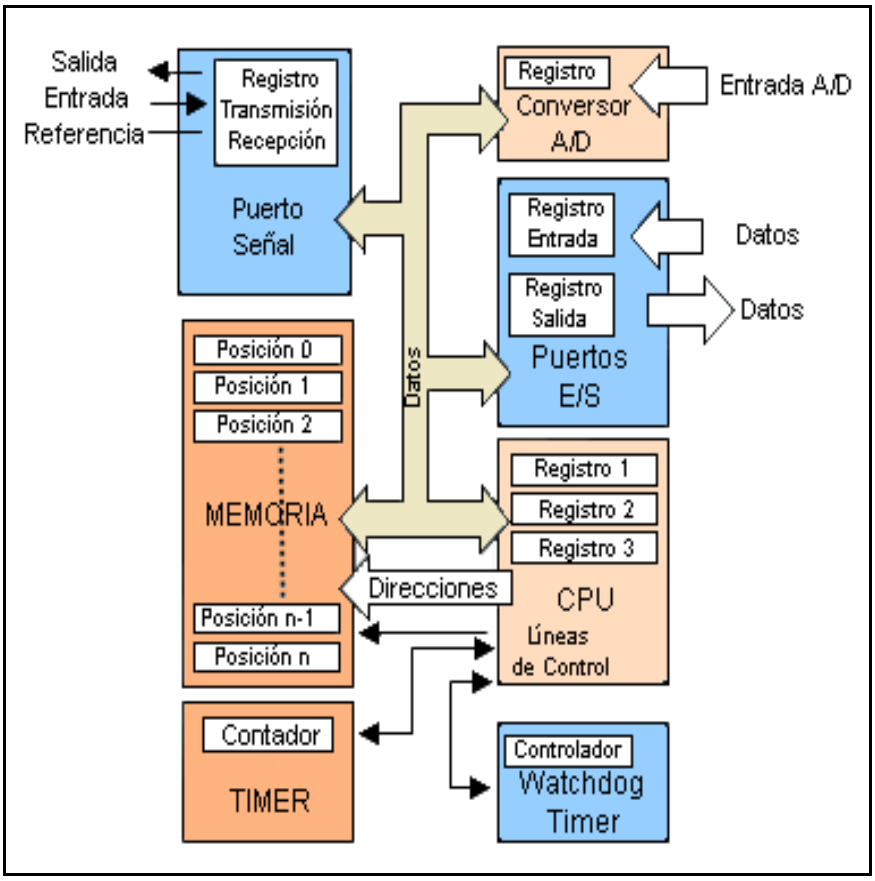


Figura 18: Estructura interna del PIC18F4550

Fuente: MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›. [consulta 5 de agosto 2009].

1.3 Sensores

"Un sensor es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas (*figura: 19*). Las variables de instrumentación dependen del tipo de sensor y pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, etc. Una magnitud eléctrica obtenida puede ser una resistencia eléctrica, una capacidad eléctrica (como en un sensor de humedad), una tensión eléctrica (como en un termopar)."[16]

[16] PALLAS, Ramón. 2007. Sensores y acondicionadores de señal. 4ta Edición. Editores Marcombo. España.

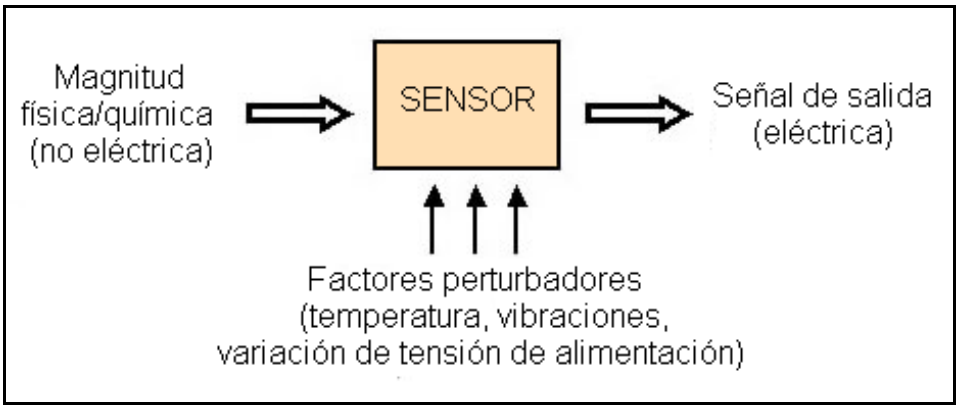


Figura 19: Transformación de magnitudes de un sensor

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www. mecanicavirtual.org/sensores3.htm›. [consulta 15 de agosto 2009].

El objetivo de un sensor es transmitir información respecto al funcionamiento del motor. Es un dispositivo que aprovecha una de sus propiedades con el fin de adaptar la señal que mide para que la pueda interpretar otro dispositivo. Áreas de aplicación de los sensores: Industria automotriz, Industria aeroespacial, Medicina, Industria de manufactura, Robótica, etc.

1.3.1 Características de un sensor

- "Rango de medida: dominio en la magnitud medida en el que puede aplicarse el sensor.
- Precisión: Desviación de la lectura de un sensor con respecto a un patrón de entrada conocido. Se expresa en % del rango.

  Ej.: Sensor que mide 0º - 100ºC (Exactitud 5 %).
- Offset o desviación de cero: valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, habitualmente se establece otro punto de referencia para definir el offset.
- Linealidad o correlación lineal: cuan lineal es la curva de calibración, un sistema mientras más lineal sea más exacto es.

- Sensibilidad de un sensor: relación entre la variación de la magnitud de salida y la variación de la magnitud de entrada.
- Resolución: mínima variación de la magnitud de entrada que puede apreciarse a la salida.
- Rapidez de respuesta: puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.
- Repetitividad: error esperado al repetir varias veces la misma medida."[17]

1.3.2 Sensor de Aceleración

"La aceleración es una magnitud que está relacionada directamente con las variaciones de la velocidad en el momento de un impacto, un vehículo percibe aceleraciones negativas o desaceleración inmediatamente, es por esta razón que al igual que en la velocidad el o los dispositivos a emplearse deberán funcionar bajo esta condición convirtiendo la desaceleración mecánica en señal eléctrica."[18]

1.3.2.1 Acelerador electrónico

El objetivo del acelerador electrónico es eliminar la conexión mecánica entre el pedal del acelerador y la unidad de mando de la válvula de aceleración, permitiendo así mediante una señal que proporciona un potenciómetro manejar en forma fiable la válvula de aceleración.

---

[17] VOLKSWAGEN. 2009. Acelerador electrónico Volkswagen [en línea]. Ecuador. <http://www.volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0 /sicherheitspedale.index.html>. [consulta 18 de agosto 2009].

[18] CARDENAS, Javier. Técnico en mecánica y electrónica automotriz, Tomo 2. Codesis Editorial. Colombia. Pp. 337

1.3.2.1.1 Ventajas:

- Permite un mejor control sobre las emisiones contaminantes.
- Permite variar la relación entre la posición del acelerador y la apertura de la válvula de aceleración con mayor precisión.
- Corrige errores de accionamiento del acelerador por parte del conductor.

Consta de las siguientes partes: (*figura: 20*)

- "El módulo del pedal acelerador con los transmisores de posición del acelerador.
- La unidad de control del motor.
- La unidad de mando de la válvula de aceleración.
- El testigo de avería para el acelerador electrónico.

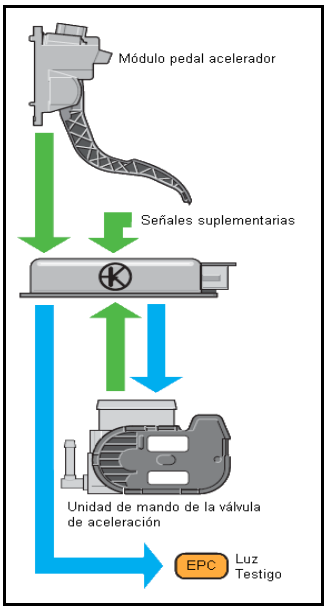


Figura 20: Acelerador electrónico

Fuente: VOLKSWAGEN. 2009. Acelerador electrónico Volkswagen [en línea]. Ecuador. <http://www.volkswagen. com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0/sicherheitspedale.index.html>. [consulta 18 de agosto 2009].

- **El módulo pedal acelerador:** Detecta la posición momentánea del acelerador a través de sus transmisores y emite una señal correspondiente a la unidad de control del motor.

- **La unidad de control del motor:** Analiza esta señal y calcula con ella los deseos expresados por el conductor a través del acelerador, para transformarlos en un par específico. A esos efectos, excita el mando de la válvula de aceleración, con objeto de abrir o cerrar un poco más la válvula.
- **La unidad de mando de la válvula de aceleración:** Se encarga de establecer el paso de la masa de aire necesaria. Mediante transmisores de ángulo para la posición de la válvula de aceleración se detecta la posición de la válvula y se realimenta en forma de las señales correspondientes a la unidad de control del motor.
- **Testigo de avería del acelerador electrónico:** Indica al conductor, que existe una avería en el sistema del acelerador electrónico."[19]

1.3.2.2 Sensor del pedal de acelerador

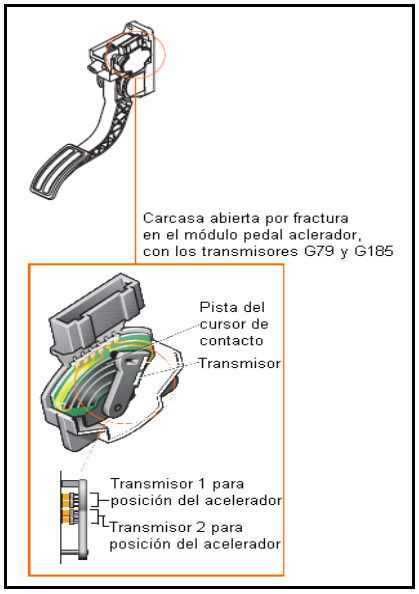


Figura 21: Sensor del pedal de acelerador

Fuente: VOLKSWAGEN. 2009. Acelerador electrónico Volkswagen [en línea]. Ecuador. <http://www.volkswagen. com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/ innovation/0/sicherheitspedale.index.html>. [consulta 18 de agosto 2009].

[19] VOLKSWAGEN. 2009. Acelerador electrónico Volkswagen [en línea]. Ecuador. <http://www.Volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0/sicherheitspedale. index.html>. [consulta 18 de agosto 2009].

Se emplean dos transmisores, para contar con los máximos niveles de fiabilidad posibles. El módulo pedal acelerador consta de las siguientes partes: (*figura: 21*).

- Pedal del acelerador.
- Transmisor 1 para posición del acelerador G79.
- Transmisor 2 para posición del acelerador G185.

1.3.2.2.1 Aplicaciones de la señal

A través de las señales procedentes de ambos transmisores de posición del acelerador, la unidad de control del motor detecta la posición momentánea del pedal acelerador.

Ambos transmisores son potenciómetros variables, que van fijados en un eje compartido. Con cada modificación que experimenta la posición del acelerador, varían las resistencias de los potenciómetros de cursor variable y las tensiones que transmiten a la unidad de control del motor."[20]

1.3.3 Sensor de velocidad

Figura 22: Sensor de velocidad

Fuente: VOLKSWAGEN. 2009. Sensor de velocidad [en línea]. Ecuador. <http://www. volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation /0/raddrehz ahlsensor.index.html>. [consulta 18 de septiembre 2009].

[20] VOLKSWAGEN. 2009. Sensor de velocidad [en línea]. Ecuador. <http://www. Volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0/rad drehzahlsensor. index.html>.

"La función del sensor de velocidad del vehículo VSS (*Vehicle Speed Sensor*) es proporcionar la velocidad del automóvil al módulo de control del motor (ECM). El ECM inspecciona la válvula de control de la velocidad en vacío, tiempo de encendido y cantidad de combustible inyectado con el propósito de mejorar la manejabilidad y reducción del gas de salida dependiendo de la velocidad del vehículo. El sensor genera 4 pulsos con una rotación (*figura: 23*), el ECM calcula la velocidad del vehículo contando el número de pulsos por segundo.

Figura 23: Señal del Sensor de Velocidad

Fuente: VOLKSWAGEN. 2009. Sensor de velocidad [en línea]. Ecuador. <http://www.volkswagen. com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw /innovation/0/raddrehz ahlsensor.index.html>. [consulta 18 de septiembre 2009].

El sensor de velocidad del Volkswagen Polo es del tipo hall, se encuentra ubicado en la cubierta del engranaje diferencial. Monitorea la velocidad de salida del cambio. El sensor VSS puede ser comprobado mediante el código de fallo y los datos de la corriente. Para comprobarlo más precisamente, la forma de onda puede medirse. El conector consiste en un sensor que funciona con un rango de 0 – 12v y posee tres pines para alimentación, señal y tierra, la frecuencia varía con la velocidad del vehículo (*figura: 24*)."[21]

[21] VOLKSWAGEN. 2009. Sensor de velocidad [en línea]. Ecuador. <http://www. volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0/raddrehz ahlsensor.index.html>.

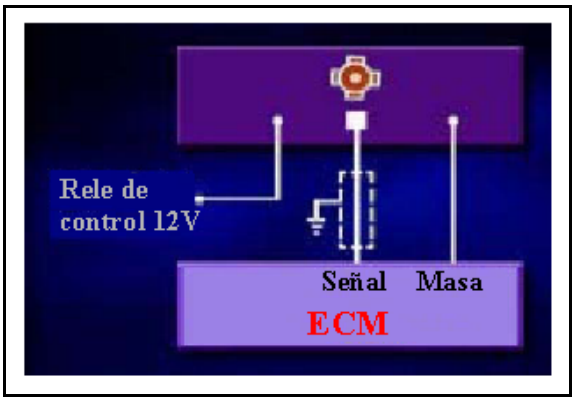


Figura 24: Conexión del sensor de velocidad

Fuente: VOLKSWAGEN. 2009. Sensor de velocidad [en línea]. Ecuador. <http://www.volkswagen. com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw /innovation/0/raddrehz ahlsensor.index.html>. [consulta 18 de septiembre 2009].

1.3.3.1 Sensor de efecto hall

Si fluye corriente por un sensor hall y se aproxima a un campo magnético que fluye en dirección vertical al sensor, entonces el sensor crea un voltaje saliente proporcional al producto de la fuerza del campo magnético y de la corriente. Si se conoce el valor de la corriente, entonces se puede calcular la fuerza del campo magnético; si se crea el campo magnético por medio de corriente que circula por una bobina o un conductor, entonces se puede medir el valor de la corriente en el conductor o bobina.

"Principio de funcionamiento del sensor de efecto hall (*figura: 25*).

- Una pastilla de semiconductor es sometida a un campo magnético externo. La pastilla genera una señal que polariza la base de un transistor. La señal recogida por el voltímetro es de máxima en este caso.
- En esta situación el transistor se hace conductor por lo que circula corriente y pone el colector a masa.
- La señal recogida en este momento por el voltímetro es mínima"[22].

[22] PESANTEZ, Henry. 2008. Inyección de gasolina. Cuenca, Ecuador. Pp. 7

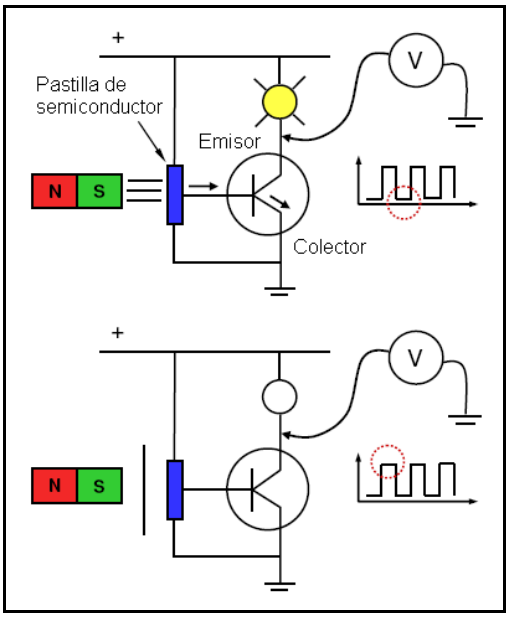


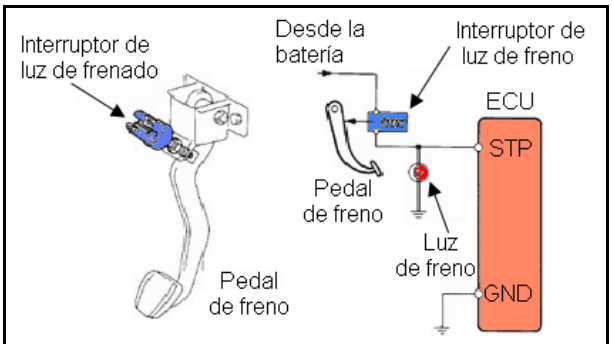
Figura 25: Principio de funcionamiento en el interior de un sensor hall

Fuente: PESANTEZ, Henry. 2008. Inyección de gasolina. Cuenca, Ecuador. Pp. 7

1.3.4 Sensor del pedal del freno

Para comprobar si el conductor frenó o no en la colisión, tomaremos de referencia el interruptor del pedal, el mismo que varía su intensidad de 0 a 5 voltios, dependiendo de su posición, este pulsante posee únicamente dos pines.

En el Volkswagen Polo las señales de frenada son provenientes de la unidad electrónica de control (*figura: 27*), de donde es posible obtener la señal hacia la caja negra.



Pedal del Freno	STP Terminal de Voltaje
Pisado	5 V
Suelto	0 V

Figura 26: Señal del pedal de freno

Fuente: CAMPOS, Guillermo. 2008. Sistemas de diagnóstico del ABS. Cuenca, Ecuador. Pp.88

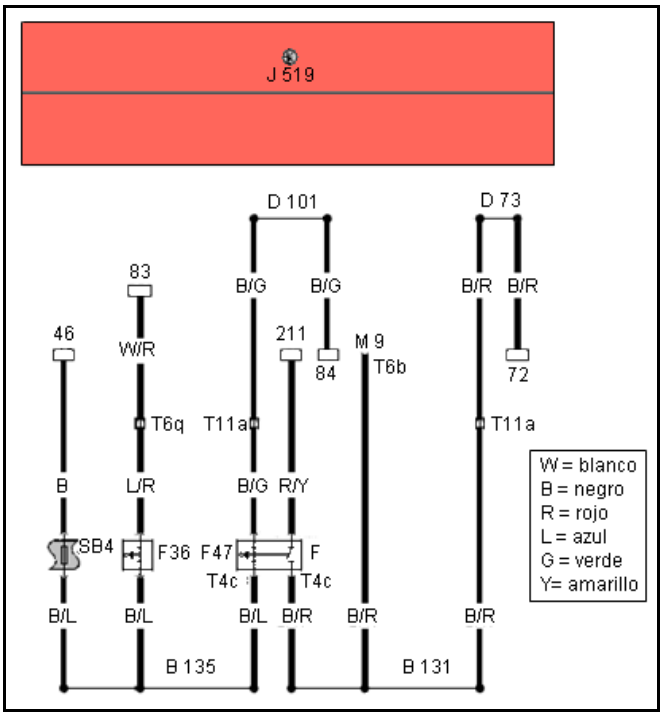


Figura 27: Esquema circuito de corriente freno

Fuente: VOLKSWAGEN. 2009. Esquema del circuito de corriente del freno [en línea]. Ecuador. <http://www.volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0/elektronische_bremskraftverteilung.index.html>

F.- Interruptor de la luz de freno.

F36.- Interruptor de pedal de embrague.

F47.- Interruptor de pedal de freno.

J519.- Comando p/red eléctrica de bordo.

M9.- Lámpara de luz de freno.

SB4.- Fusible (4) del porta-fusibles.

T4c.- Conector de encaje, cuádruplo, en el interruptor de la luz de freno.

T4p.- Conector de encaje, cuádruplo, en el interruptor del pedal de embrague.

T6b.- Conector de encaje, séxtuplo, en la lámpara de la luz de freno.

T6q.- Conector de encaje, séxtuplo, en el componente del motor.

T11a.- Conector de encaje, 11 polos, en la esquina del componente del motor.

B131.- Conexión (54) en el juego de cables del interior.

B135.- Conexión positiva -1- (15a) en el juego de cables del interior.

D73.- Conexión positiva /54) en el juego de cables del componente del motor.

D101.- Conexión -1- del juego de cables del componente del motor.

1.3.5 Pulsante del cinturón de seguridad

"Un cinturón de seguridad es un arnés diseñado para sujetar a un ocupante de un vehículo si ocurre una colisión y mantenerlo en su asiento, los cinturones son dispositivos de seguridad pasiva más importantes que posee el vehículo. El objetivo de los cinturones de seguridad es minimizar las heridas en una colisión, impidiendo que el pasajero se golpee con los elementos duros del interior o contra las personas en la fila de asientos anterior, y que sea arrojado fuera del vehículo. Actualmente los cinturones de seguridad poseen tensores que aseguran el cuerpo en el momento del impacto mediante un resorte o un disparo (tensor pirotécnico)."[23]

[23] JAMAICA, Arturo. 2009. Pulsante del cinturón de seguridad [en línea]. Argentina. <http://walhez.Com/2008/04/ campana-usatu-cinturon-de-seguridad>. [consulta 8 de octubre 2009].

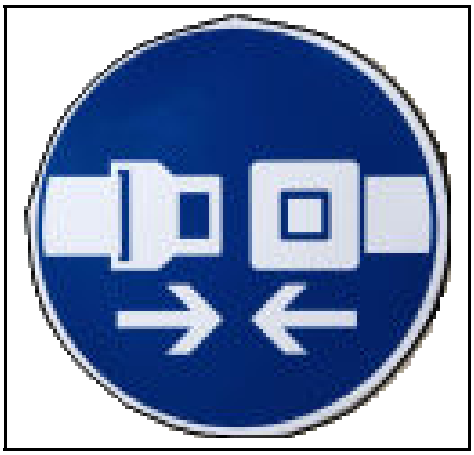


Figura 28: Pulsante del cinturón de seguridad

Fuente: JAMAICA, Arturo. 2009. Pulsante del cinturón de seguridad [en línea]. Argentina. <http://walhez.com/2008/04/campana-usatu-cinturon-de-seguridad>. [consulta 8 de octubre 2009]. Pp.1

1.3.5.1 Sensores de contacto

Los sensores de contacto son utilizados para determinar si el conductor utilizó el cinturón de seguridad, en este proyecto se utilizarán sensores de tipo pulsante. Existen distintos interruptores que detectan el contacto de una superficie sobre otra y los mismos pueden ser de funcionamiento mecánico (interruptores y pulsadores), magnéticos, eléctricos, etc.

1.3.5.1.1 Características:

- Es un interruptor mecánico tipo pulsante.
- Se acciona por el contacto físico con algún elemento.
- Posee contactos normalmente abiertos.
- Es resistente a los golpes.
- Está diseñado para funcionar con cualquier corriente por tratarse de un interruptor.
- Es de tamaño adecuado para ser colocado en el vehículo.

- Su instalación es sencilla.
- No consume energía por tratarse de un interruptor que solo permite el paso de corriente.
- Son de bajo costo.

1.3.6 Sensor de impacto

1.3.6.1 Magnitudes de medición

"Los sensores de aceleración y de vibraciones son apropiados para la regulación contra la detonación (picado) en motores de combustión interna, en este proyecto nos sirven para activar sistemas de protección de los pasajeros (*airbag*, tensores de cinturón, arco contra el vuelco) y para detectar aceleraciones en las curvas y variaciones de velocidad en vehículos de tracción integral equipados con el sistema antibloqueo ABS, o con un sistema de regulación del tren de rodaje. La magnitud de medición es la aceleración a, que con frecuencia se indica como múltiple de la aceleración de la gravedad g (1g = 9,81 m/s2) para valores típicos de los automóviles."[24]

Aplicación	Campo de medición
Regulación contra la detonación	1……10g
Protección de los pasajeros:	
Airbag tensor de cinturón	50 a 100g
Arco contra el vuelco	4 a 6g
Bloqueador de cinturón	0, 4 a 1g
ABS, ESP	0,8g……1,2g
Regulación del tren de rodaje:	
Carrocería	1g
Eje	10g

Tabla 5: Sensores de aceleración y vibraciones

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanica virtual.org/sensores3.htm›. [consulta 26 de octubre 2009]. Pp.1

[24] MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanicavirtual.org/ sensores3.htm›. [consulta 20 de octubre 2009]. Pp.3

1.3.6.2 Principios de medición

"Todos los sensores de aceleración miden en principio, con arreglo a la ley fundamental de la mecánica, las fuerzas F, ejercidas por la aceleración a sobre las masas (inertes) m, sea ya de modo únicamente dinámico (sensores de vibraciones) o también estático:

$$F = m \bullet a$$

Como en el caso de medición de una fuerza, existen sensores que miden un desplazamiento y otros que miden esfuerzos mecánicos. El encapsulamiento en estos sensores tiene una importancia decisiva para la calidad de la detección. En su función de sensores de inercia detectan la magnitud de medición sin la menor comunicación con el exterior; puede encapsularse pues fácilmente de modo hermético. Han de disponer, sin embargo, de medios apropiados para un acoplamiento mecánico lo más rígido posible al cuerpo a medir, pues elementos intermedios adicionales elásticos o sueltos alterarían considerablemente la medición. Este acoplamiento rígido y fijo no debe dar lugar, sin embargo, a que las posibles dilataciones térmicas del cuerpo a medir se transmitan por ejemplo al sensor, lo que podría influir en el valor medido.

Ejemplo de sensores de aceleración:

- Sensores de aceleración de efecto hall.
- Sensores de aceleración piezoeléctricos.
- Sensores de aceleración micro mecánicos."[25]

[25] MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanicavirtual.org/ sensores3.htm›. [consulta 3 de noviembre 2009]. Pp.3

1.3.6.3 Sensores de aceleración

Los sensores de aceleración realizados por micro-mecánica de superficie y destinados a los sistemas de retención de pasajeros detectan los valores de aceleración de un choque frontal o lateral y provocan la activación de los tensores de cinturón, el disparo de los airbag, la actuación del arco antivuelco y en nuestro caso el final de la grabación en la caja negra (*figura: 29*).

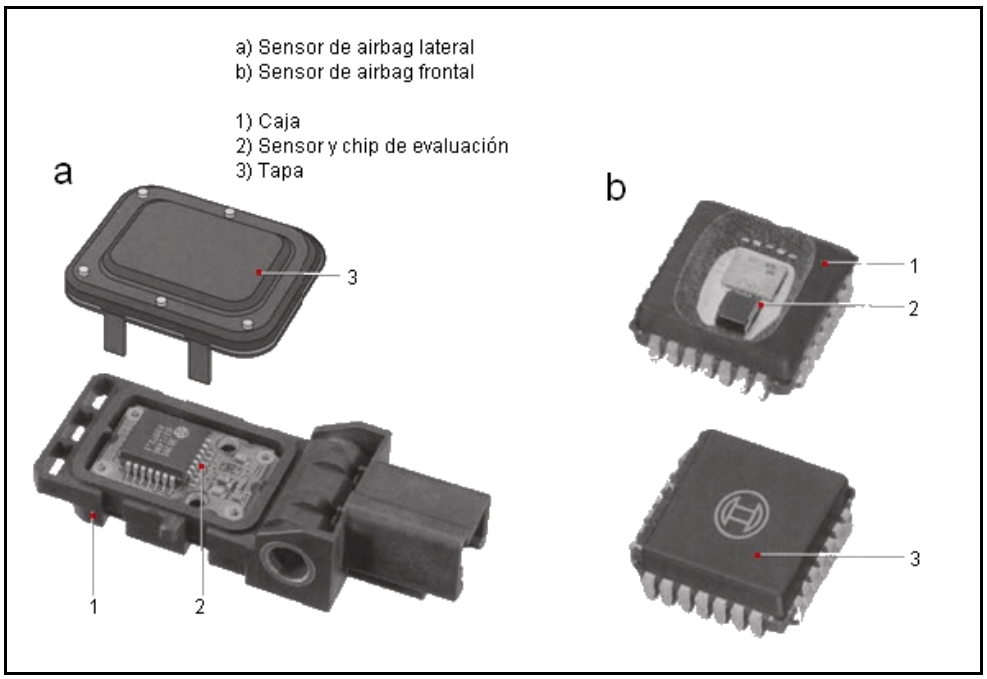


Figura 29: Sensores de aceleración utilizados para el disparo del Airbag

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanica virtual.org/sensores3.htm›. [consulta 3 de noviembre 2009]. Pp.3.

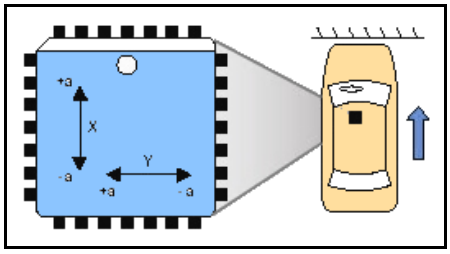


Figura 30: Ubicación del sensor de aceleración en el vehículo

Fuente: FONT, José. 2001. Tratado sobre automóviles. Alfa-omega grupo editor. España. Pp.14.73, 14.75

1.3.6.3.1 Estructura y funcionamiento

"Estos sensores, utilizados primero para detectar altas aceleraciones (50 a 100 g) en sistemas de protección de pasajeros, son también apropiados para medir aceleraciones de reducida intensidad. Comparados con los sensores de silicio realizados por micro mecánica de volumen, son mucho más compactos y están alojados junto con la electrónica de evaluación (ASIC) en una caja estanca al agua. Su sistema de masa-resorte está montado sobre la superficie de la plaquita de silicio por un procedimiento aditivo.

La masa sísmica, cuyos electrodos tienen la forma de un peine, está suspendida elásticamente dentro de la célula de medición (*figura: 31*). A ambos lados de esos electrodos móviles hay colocados sobre el chip electrodos fijos, asimismo en forma de peine (6). Esta disposición de electrodos fijos y móviles corresponde a una conexión en serie de dos condensadores diferenciales C1 y C2 (capacidad de la estructura de peine, aprox. 1 pF). A los bornes de estos condensadores se aplican tensiones alternas de fases opuestas, cuya superposición es detectada en el punto CM (capacidad de medición) entre los condensadores, o sea, en la masa sísmica. Como la masa sísmica está suspendida de resortes (2), una aceleración lineal a en el sentido de detección ocasiona una variación de la distancia entre los electrodos fijos y móviles y, por consiguiente, una variación de capacidad en los condensadores C1 y C2.

De ello resulta una variación de la señal eléctrica que en la electrónica de evaluación (ASIC) es amplificada, filtrada y digitalizada para su transmisión a la unidad de control de los airbag. Por razón de la reducida capacidad de aprox. 1 pF, la electrónica de evaluación está integrada directamente en el sensor sobre el mismo chip o estrechamente unida al sensor. Es posible la realización de sistemas reguladores de posición con vuelta electrostática al estado inicial."[26]

[26] MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanicavirtual.org/indice_cursos_electr.html# sensor›. [consulta 11 de noviembre 2009]. Pp. 3

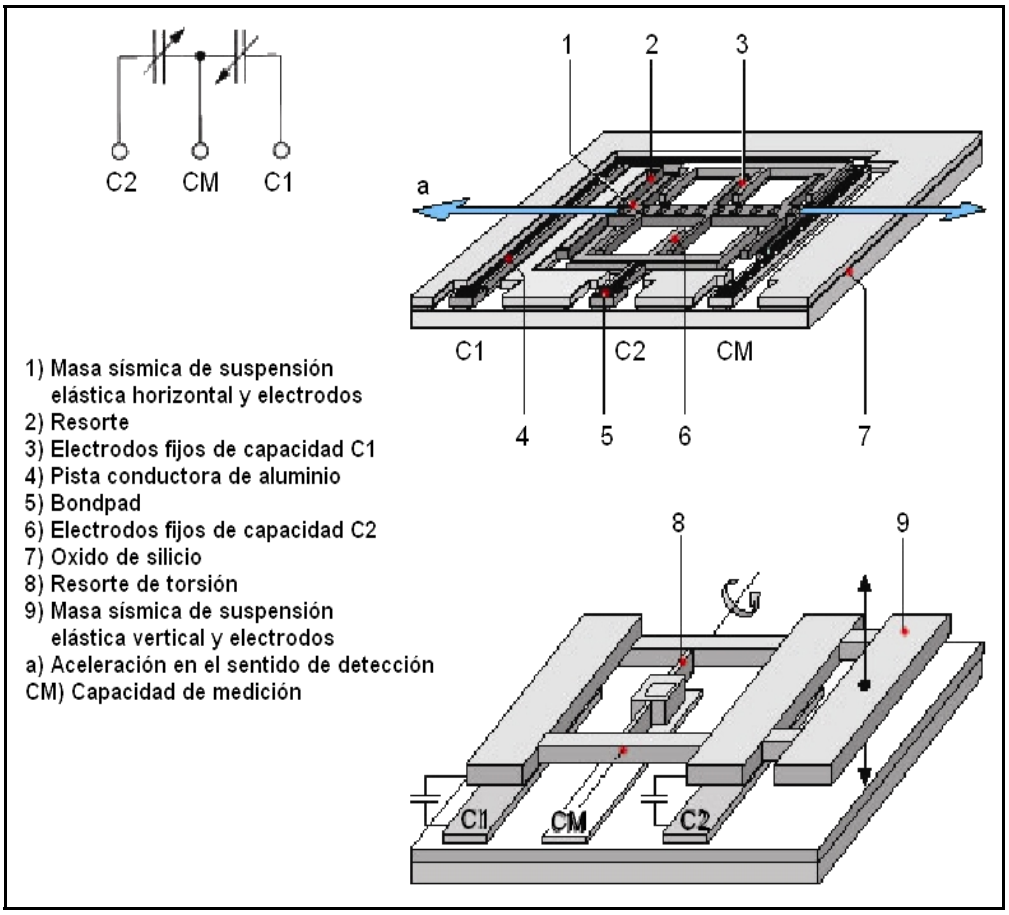


Figura 31: Esquema del sensor de aceleración realizado por micro mecánica de superficie, de detección capacitiva.

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanica virtual.org/sensores3.htm›. [consulta 10 de noviembre 2009]. Pp. 4

El circuito de evaluación dispone también de una compensación de desviaciones del sensor y de una auto-diagnosis durante la fase de puesta en funcionamiento. Para la auto-diagnosis, unas fuerzas electrostáticas provocan el desplazamiento de la estructura en forma de peine y simulan así el proceso que tiene lugar durante la aceleración en el vehículo.

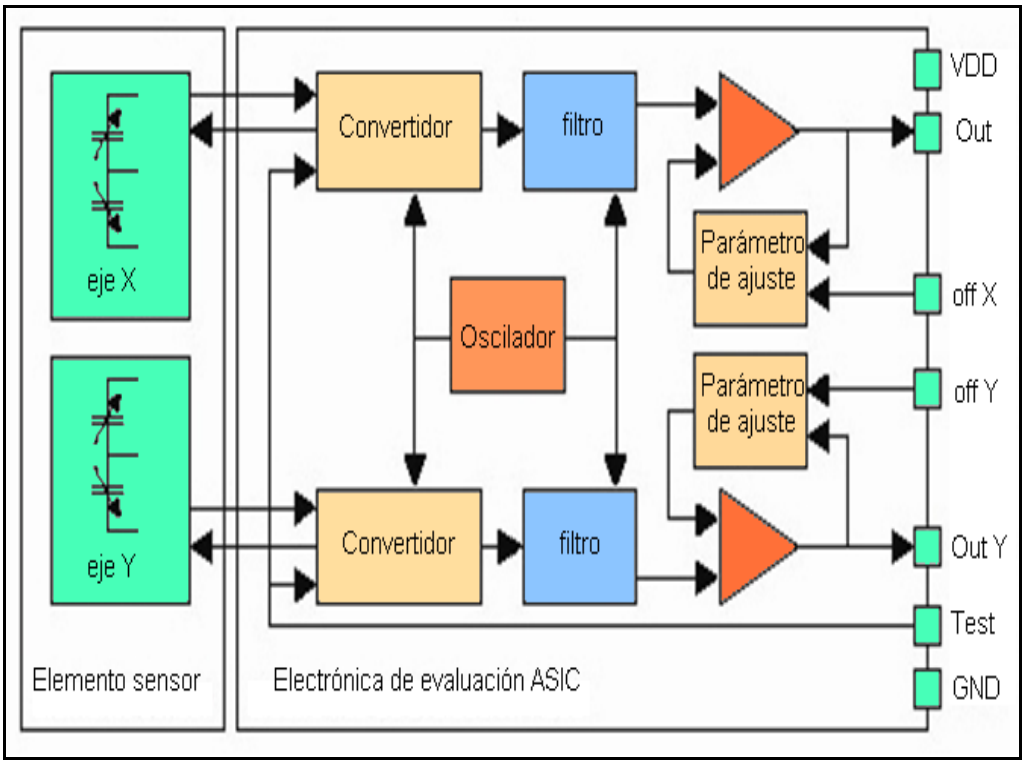


Figura 32: Diagrama de bloques del sensor y la electrónica de evaluación

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanica virtual.org/sensores3.htm›. [consulta 10 de noviembre 2009]. Pp. 4

1.3.6.3.2 Ubicación en el vehículo

Puede estar ubicado debajo del capot, detrás de la plancha de a bordo, debajo de los asientos o incorporado al módulo de control (*figura: 33*). Es un elemento sensible a una combinación de fuerza de aceleración y duración de la misma, lo cual contribuye a evitar un disparo accidental del sistema.

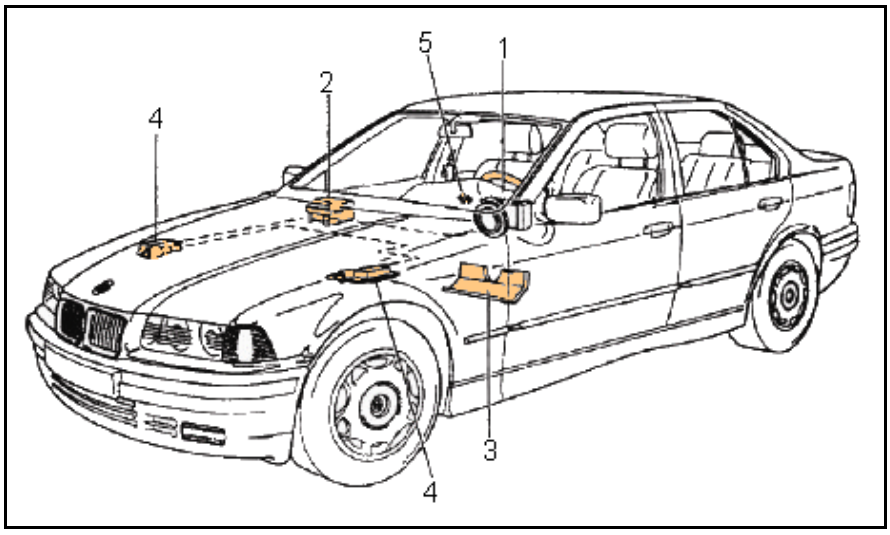


Figura 33: Ubicación del sensor de aceleración en el vehículo

Fuente: CAMPOS, Guillermo. 2008. Sistemas de diagnóstico del ABS. Curso de Graduación. Cuenca, Ecuador. Pp. 25

1) Volante con *airbag* (con cojín hinchable y unidad de control).
2) Unidad de diagnosis.
3) Protección para las rodillas.
4) Sensores de impacto.
5) Testigo de control *airbag*.

"El impacto de un vehículo provoca de manera instantánea severos cambios en la velocidad y aceleración, reduciendo los valores de estas magnitudes en el caso de un impacto frontal e incrementándolos en el caso de un impacto lateral. Los sensores de impacto frontales serán colocados en el travesaño frontal del chasis por debajo del radiador y la distancia entre ellos será distribuida de manera uniforme cubriendo así toda el área frontal del vehículo. Los sensores de impacto laterales serán ubicados en la parte central del chasis, y enviarán la señal de activación al cerrarse sus contactos por efecto del movimiento o arrastre violento del vehículo en sentido transversal."[27]

[27] MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanicavirtual.org/direccion.htm›. [consulta 28 noviembre 2009]. Pp. 5

1.3.7 Sensor de posición del volante de dirección

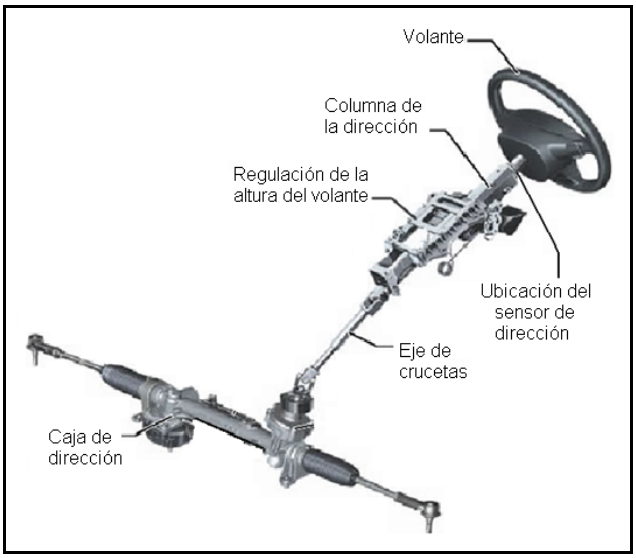


Figura 34: Componentes de la dirección

Fuente: MARQUEZ, Danny. 2009. Dirección [en línea]. España. ‹http://www.mecanica virtual.Org/dirección.htm›. [consulta 28 de noviembre 2009]. Pp. 5

La ubicación del sensor de dirección es entre el mando combinado y el volante (*figura: 34*); su misión es la de suministrar la señal para la determinación del ángulo de dirección, destinándola a la unidad electrónica de control y de esta manera expresar la trayectoria deseada por el conductor.

1.3.7.1 Optoacopladores

Los optoacopladores pueden servir para determinar la posición del volante de dirección, la señal de entrada es aplicada al fotoemisor y la salida es tomada del fotoreceptor. Los optoacopladores son capaces de convertir una señal eléctrica en una señal luminosa modulada y volver a convertirla en una señal eléctrica. La gran ventaja de un optoacoplador reside en el aislamiento eléctrico que puede establecerse entre los circuitos de entrada y salida (*figura: 35*).

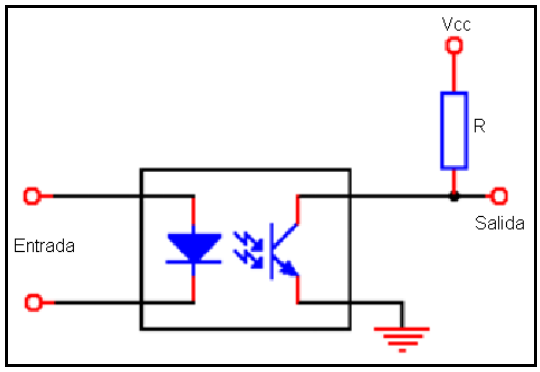


Figura 35: Circuito del optoacoplador

Fuente: BLACK, Jason. 2009. Optoacopladores [en línea]. USA. <http://www.scribd.com/doc/5516426/Optoacopladores>. [consulta 7 de diciembre]. Pp. 1

1.3.7.1.1 Características:

- **Aislamiento de alto voltaje:** "El aislamiento de alto voltaje entre entradas y salidas son obtenidos por el separador físico entre el emisor y el sensor. Estos dispositivos pueden resistir grandes diferencias de potencial, dependiendo del tipo de alcance medio.
- **Aislamiento de ruido:** El ruido eléctrico en señales digitales recibidas en la entrada del optoacoplador es aislado desde la salida por el acople medio, desde el diodo de entrada el ruido de modo común es rechazado.
- **Ganancia de corriente:** La ganancia de corriente de un optoacoplador es en gran medida determinada por la eficiencia de los sensores npn y por el tipo de transmisión usado.
- **Tamaño:** Las dimensiones de estos dispositivos permiten ser usados en tarjetas impresas estándares."[28]

[28] TELLEZ, Carlos. 2008. Optoelectrónica [en línea]. Argentina. <http://www.webelectronica.com.Ar/news10/nota014/optoelectronica2.htm> [consulta 8 de diciembre 2009]. Pp. 9

1.3.7.1.2 Tipos de Optoacopladores:

- **Fototransistor: "**Transforma una variación de corriente de entrada en una diferenciación de tensión de salida. Se utiliza en acoplamientos de líneas telefónicas, periféricos, audio, además es un transistor bipolar sensible a la luz, en la *figura 36*, tenemos el símbolo de un fototransistor.

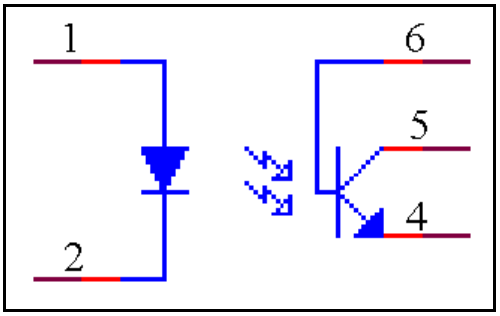


Figura 36: Símbolo del Fototransistor

Fuente: BLACK, Jason. 2009. Optoacopladores [en línea]. USA. <http://www.scribd.com/ doc/5516426/Optoacopladores>. [consulta 7 de diciembre]. Pp.2

La radiación luminosa se hace incidir sobre la unión colector base, en esta unión se generan los pares electrón – hueco, que provocan la corriente eléctrica.

- **Fototiristor:** Diseñado para aplicaciones donde sea preciso un aislamiento entre señal lógica y la red.

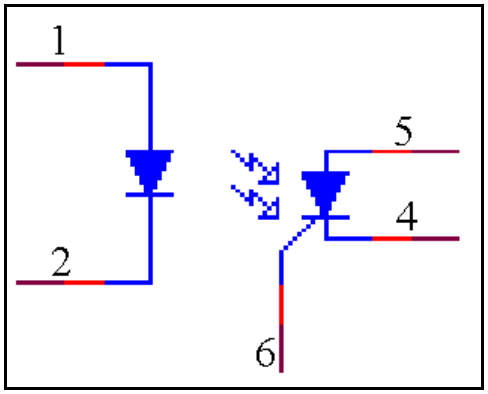


Figura 37: Símbolo del Fototiristor

Fuente: BLACK, Jason. 2009. Optoacopladores [en línea]. USA. <http://www.scribd.com/ doc/5516426/Optoacopladores>. [consulta 8 de diciembre 2009]. Pp.3

- **Fototriac:** Se compone de un optoacoplador con una etapa de salida formada por un triac."[29]

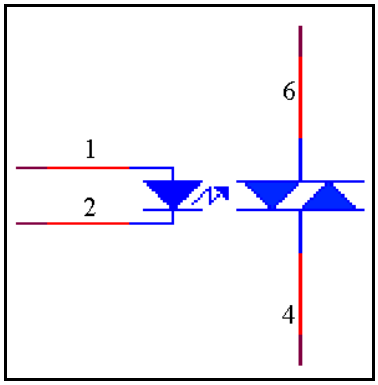


Figura 38: Símbolo del fototriac

Fuente: BLACK, Jason. 2009. Optoacopladores [en línea]. USA. <http://www.scribd.com/doc/5516426/Optoacopladores>. [consulta 9 de diciembre 2009]. Pp.3

1.3.7.2 Sensor del ángulo de dirección

"El sensor de ángulo de dirección puede detectar 1044º de ángulo (casi 3 vueltas de volante). Se dedica a sumar los grados angulares. De esa forma, al sobrepasar la marca de los 360º reconoce que se ha ejecutado una vuelta completa del volante. La configuración de la caja de la dirección permite dar 2,76 vueltas al volante de la dirección.

El anillo de incrementos esta dividido en 5 segmentos de 72º cada uno (*figura: 39*) y es explorado por una pareja de barreras luminosas. De ahí resulta la codificación de los segmentos. El anillo de absolutos viene a determinar el ángulo. El mismo que es explorado por 6 parejas de barreras luminosas.

[29] GIL, Hermógenes. 2002. La electrónica en el automóvil. Grupo editorial CEAC. España. Pp. 50, 55

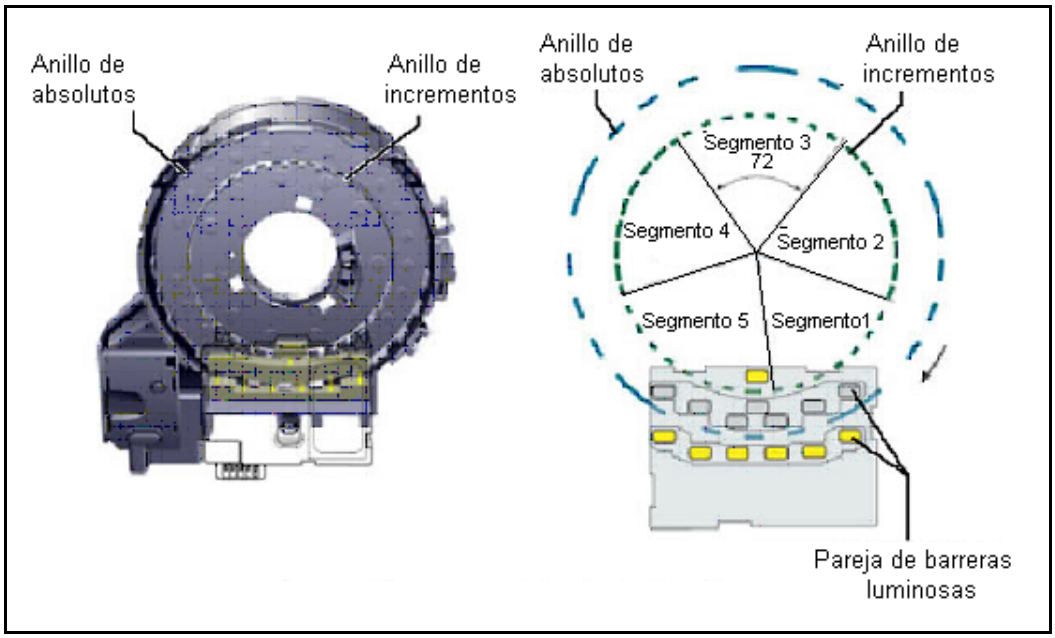


Figura 39: Esquema de un sensor de dirección

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹ http://www.mecanica virtual.org/direccionasistida-electr.htm›. [consulta 17 de diciembre 2009]. Pp. 5

Para la detección del ángulo del volante son apropiados en principio todos los tipos de sensores angulares. No obstante, con objeto de garantizar la seguridad se requieren versiones cuya aplicación se pueda comprobar fácilmente o que, mejor aún, posean una función de autocontrol. En la mayoría de sensores utilizados es necesario sin embargo registrar y memorizar constantemente la posición actual del volante, ya que los sensores angulares usuales pueden medir como máximo 360°.

La medición del ángulo se realiza según el principio de la barrera luminosa. Cuando la luz incide en el sensor al pasar por una almena del anillo se engendra una señal de tensión. Al cubrirse la fuente luminosa se vuelve a interrumpir la tensión de la señal. Al mover ahora el anillo de incrementos se produce una secuencia de señales de tensión.

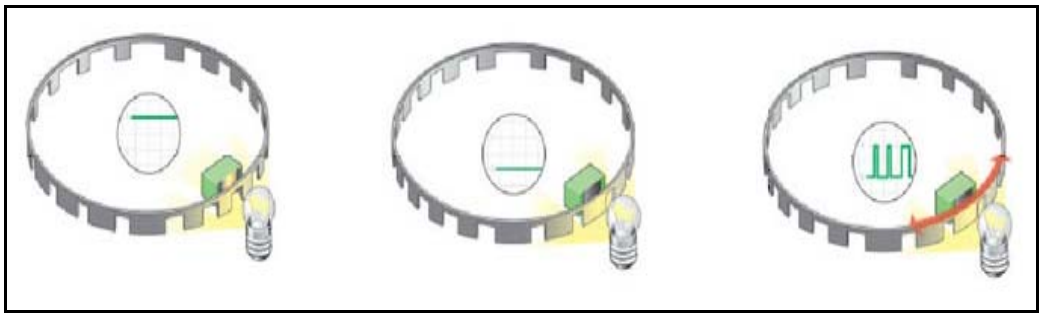


Figura 40: Principio de accionamiento del sensor de dirección

Fuente: MARQUEZ, Danny. 2008. Dirección asistida eléctricamente [en línea]. España. ‹http://www.mecanica virtual.org/direccionasistida-electr.htm›. [consulta 21 de diciembre 2009]. Pp.8

De esa misma forma se genera una secuencia de señales de tensión en cada pareja de barreras luminosas aplicadas al anillo de valores absolutos. Previa comparación de las señales, el sistema puede calcular a qué grados han sido movidos los anillos. Durante esa operación determina también el punto de inicio del movimiento en el anillo de valores absolutos."[30]

[30] MARQUEZ, Danny. 2008. Dirección asistida eléctrica [en línea]. España. <http://www.mecanica virtual.org/ direccion-asistida-electr.htm›. [consulta 21 de diciembre 2009].

1.4 Conclusiones

- Al principio del proyecto seleccionamos un microcontrolador de familia media (16F871) pero con el transcurso del mismo nos vimos en la necesidad de cambiar a uno de familia alta como es el 18F4550, siendo este de similar número de pines, pero con mayores prestaciones.
- El 18F4550 puede incluir hasta 13 señales diferentes de conversión A/D, lo que nos permite un mayor control en caso de necesitar ampliar la cobertura.
- La Arquitectura tipo Harvard permite accesos simultáneos tanto a instrucciones como datos, lo que implica mayor eficacia.
- Los PIC´s son procesadores tipo RISC, trabajan con pocas instrucciones, dando mayor rapidez de funcionamiento.
- El set de instrucciones es ortogonal (se envía en bloque).
- Con una sola instrucción se pueden realizar accesos a entidades diferentes.
- Cada puerto tiene asociado un registro TRIS; por ejemplo un puerto A tiene un TRIS A. Por lo tanto si en un bit del registro TRIS correspondiente se escribe un 0, esto significa que el pin asociado a ese bit será una salida. Consecuentemente si se escribe un 1, significa que el pin en cuestión será una entrada.
- Es preciso recordar que el microcontrolador solo registra señales digitales, por lo que es necesario una previa conversión analógica/ digital.
- Se seleccionó una memoria del tipo EEPROM porque no es volátil, y en caso de faltar la alimentación no se pierden los datos registrados en ella.
- Además la micro SD se puede reescribir por vía eléctrica.
- Nuestro proyecto puede ser modificado sin hacer cambios en el hardware ni en el software para que utilice memorias SD y micro SD, facilitando su lectura en caso de querer ampliar la capacidad de memoria.
- El sensor micromecánico nos da un mayor rango de medición de fuerzas g logrando un control en caso de un accidente.
- Los optoacopladores cubren 1044° de movimiento del volante de dirección, haciendo una referencia bastante exacta de la posición del volante de dirección.

CAPITULO II

DISEÑO DEL HARDWARE

Una vez dadas las explicaciones pertinentes de las herramientas a utilizar, ya sean estas elementos externos con sus tipos de señales, herramientas internas del microcontrolador y conexiones en general, procederemos en este capítulo con la etapa del diseño de todos los circuitos que serán necesarios en la adquisición de datos, para darnos una idea clara de la ubicación de los componentes y diseño de las tarjetas se detallará cada circuito por separado en su respectivo sensor. Estos son colocados con las conexiones de todos sus pines y además las uniones de herramientas adicionales para el correcto funcionamiento del circuito.

Tomando en cuenta que la tarjeta madre es un dispositivo con capacidad para trabajar con 6 diferentes señales al mismo tiempo, por poseer un controlador de la familia alta de microchip, la variación se dará simplemente en el sensor descrito, puesto que la tarjeta central es la misma con excepción del circuito extra para la memoria de datos, colocada así para tener la facilidad de variar su tamaño desde 64Kb en adelante de acuerdo a las necesidades.

2.1 Diseño de la caja negra

2.1.1 Sensores en el automóvil

El procesamiento de estas mediciones permitirá finalmente evaluar con rapidez los parámetros mencionados en los apartados anteriores, preparándolos para las funciones previstas en nuestro diseño.

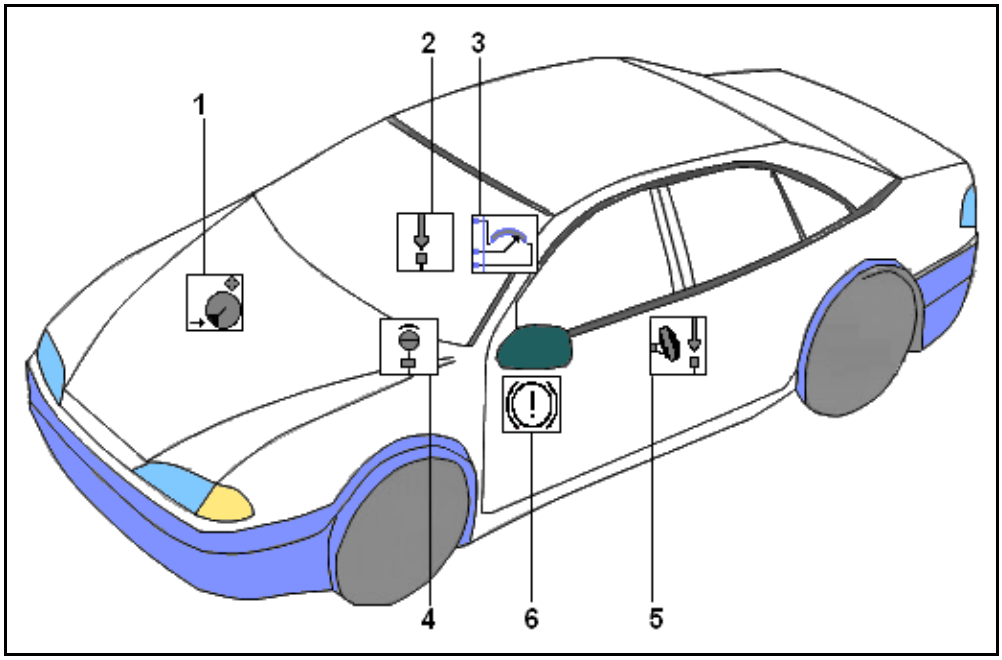


Figura 41: Ubicación de los sensores

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www. mecanicavirtual.org/sensores3.htm›. [consulta 10 de noviembre 2009].

1) Sensor de velocidad.
2) Pulsante del cinturón de seguridad.
3) Sensor de posición del pedal del acelerador.
4) Sensor de ángulo del volante de dirección.
5) Sensor de impacto (airbag).
6) Sensor del pedal de freno.

2.1.2 Características de la caja negra

- La velocidad del vehículo (dos segundos antes del impacto).
- La posición del pedal del freno (dos segundos antes del impacto).
- La posición del pedal del acelerador (dos segundos antes del impacto).
- El estado del interruptor de cinturón de asiento del conductor (On/Off).
- Señal del sensor de impacto (bolsa de aire de pasajero On/Off).
- Posición del volante de la dirección (dos segundos antes del impacto).

2.1.2.1 Protección externa de la caja negra

Si bien no está en nuestro proyecto el diseño de la parte externa de la caja negra, hemos visto la necesidad de requerir de ciertos materiales y ciertas especificaciones para que el dispositivo que estamos desarrollando pueda resistir posibles condiciones como un calor extremo, golpes violentos y la presión máxima que puede darse en un accidente de tránsito. También se muestra la ubicación recomendada para la tarjeta de almacenamiento de datos en el vehículo, a fin de minimizar las fuerzas a las que estará sometido y el dar fácil acceso a su mantenimiento. Cabe señalar que las recomendaciones que se hacen están basadas en datos teóricos, mas no en datos obtenidos por cuenta propia de los autores.

Para la parte protección es necesario tres capas de material, el circuito electrónico, va metido dentro de una sólida caja negra que aísla y protege la pila de memoria que almacena la información digitalizada. Detallaremos sobre la memoria y la parte electrónica más adelante. Los materiales que proveen de una barrera, empezando en la parte más interna y hacia el exterior, son los siguientes componentes:

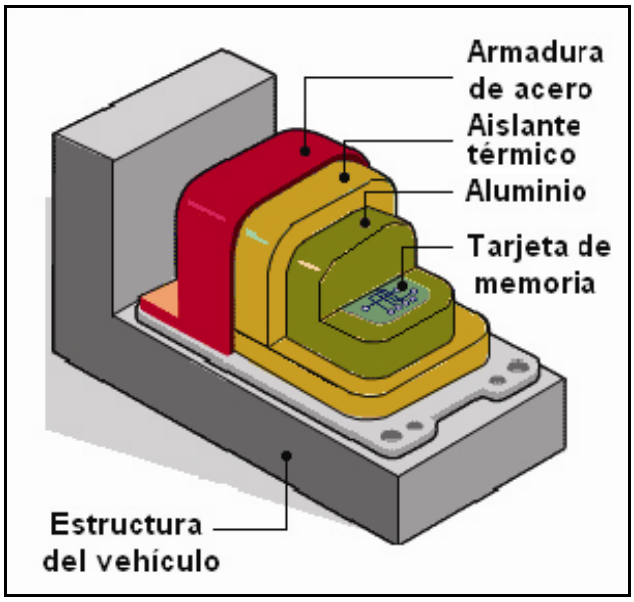


Figura 42: Protección externa de la caja negra.

Fuente: MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www. mecanicavirtual.org/caja-negra›. [consulta 7 de enero 2010].

- **"Aluminio:** Hay una fina capa de aluminio alrededor de las tarjetas de memoria de 0,2 mm de espesor.
- **Aislante para altas temperaturas:** Una capa de aislante especial de 1,5 centímetros, provee de una gran resistencia a las altas temperaturas. Esto es lo que mantiene las memorias a salvo durante un accidente donde el fuego sea intenso. El material recomendado es una aleación de silicio.
- **Una capa de acero inoxidable:** El material aislante contra las temperaturas altas es contenida dentro de una carcasa de acero inoxidable de 1 mm de espesor. También se puede utilizar titanio para fortalecer esta importante capa de protección."[31]

[31] ELECTRÓNICA-BÁSICA.COM. 2008. Caja negra después de un accidente [en línea]. México. <http://www.electronica-basica.com/caja-negra-de-avion.html>. [consulta 15 de enero 2010]

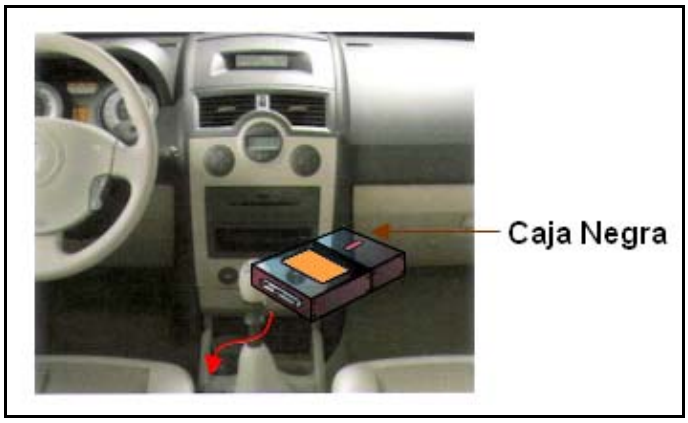


Figura 43: Ubicación de la caja negra en el vehículo.

Fuente: RUEDA, Jesús. 2006. Manual técnico de fuel inyección. Tercera Edición. Diseli Editores. Ecuador.

La ubicación recomendada de nuestra caja negra en el vehículo es entre el asiento del conductor y el asiento delantero del pasajero (*figura: 43*), debido a las siguientes ventajas:

- "Ubicación cercana al centro de gravedad del vehículo.
- Fácil acceso para mantenimiento.
- Estructura fácil de proteger contra oxidaciones y corrosiones.
- Disminución de vibraciones y ruidos, alargando la vida útil de la caja negra.
- Resistente a impactos frontales, laterales y mayor protección en caso de vuelco.
- Parte del bastidor que está dotado de la rigidez más conveniente, tanto a flexión como torsión."[32]

[32] CASCAJOSA, Manuel. 2005. Ingeniería de Vehículos. Sistemas y Cálculos. Segunda Edición. Alfaomega Grupo Editor. México.

Figura 44: Caja negra.

Fuente: CAMPOS, Jorge. 2008. Caja negra [en línea]. México. ‹http://www.autohoy.net /acce sorios-para-autos/para-que-sirve-la-caja-negra-en-los-autos.html›. [consulta 22 de enero 2010]. Pp. 1

2.1.2.2 Tipos de ruidos

El ruido es la señal acústica, eléctrica o electrónica formada por una mezcla aleatoria de longitudes de onda. El término ruido designa una señal que no contiene información. En nuestro proyecto, utilizaremos mecanismos de aislamiento para evitar la interferencia del ruido que influye directamente sobre un sistema de medición.

2.1.2.2.1 Ruido conductivo:

"Asociado a los cables de conexión de retornos, principalmente de sensores, fuentes y elementos de potencia. Se debe a la impedancia de los cables de conexión; la misma que debe tomarse en cuenta al diseñar el esquema de alumbrado para el sistema de medición, jamás se debe unir los cables de retorno. La solución es aislar los retornos, sensor y carga, deben tener conexión a tierra diferente. El ruido conductivo puede minimizarse eliminando los lazos de tierra en las conexiones entre la fuente de señal y el sistema de medición, separando los retornos de las señales de baja potencia y las de alta potencia en el diseño.

2.1.2.2.2 Ruido capacitivo:

Se produce cuando se establece un campo eléctrico no deseado entre el circuito de medición y alguno próximo. La solución para la disminución del ruido capacitivo es emplear una guarda o pantalla capacitiva. La guarda o pantalla debe ser de aluminio adaptable, no cobre debido a que se parte. Una guarda capacitiva consiste de una cubierta metálica que envuelve a los cables de señal y facilita un camino para la corriente de ruido inducida, de modo que esta no llegue a circular por los cables. En la *figura 45* se observa la conexión adecuada de la guarda, la misma que se ha aterrado solo un extremo.

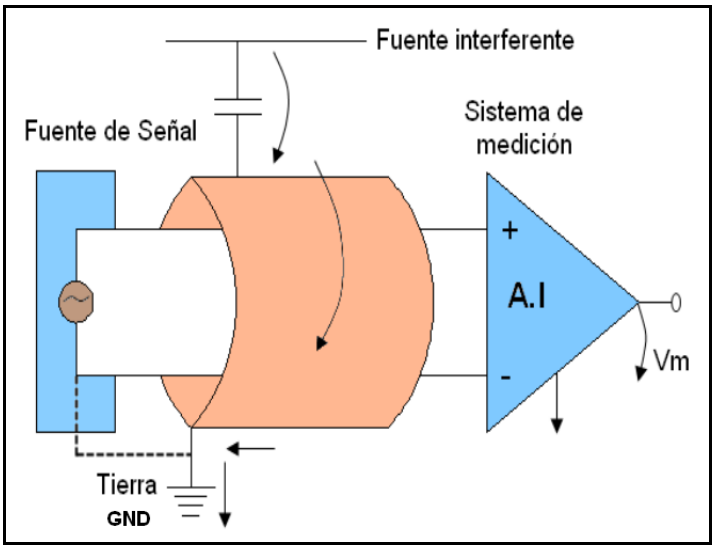


Figura 45: Colocación de la guarda para evitar el ruido capacitivo.

Fuente: HERRADON, Rafael. 2006. Curso de introducción a los sistemas de radio comunicaciones móviles [en línea]. España. ‹http://www.Ocw.upm.es/teoria-de-la-senal-y-comuni caciones-1/radiocomunicación/contenidos/presentaciones/radiocomunicación-07.pdf. [consulta 2 de febrero 2010].

La guarda debe ser colocada entre los conductores (2 cables de señal). Es importante saber que únicamente uno de los extremos de la guarda debe ser conectado a tierra, porque al ser conectados ambos extremos, es posible que circulen por la guarda corrientes significativas que generarán una diferencia de potencial entre ambos extremos de la misma.

2.1.2.2.3 Ruido inductivo:

Es el campo magnético no deseado (interferente). El ruido inductivo en los sistemas de medición es ocasionado por campos magnéticos variables. Los campos magnéticos pudieran ser generados por la circulación de corrientes en circuitos ruidosos cercanos. La solución es reducir el área encerrada en el circuito de señal, trenzando los cables que conectan la fuente de señal con el sistema de medición (*figura: 46*).”[33]

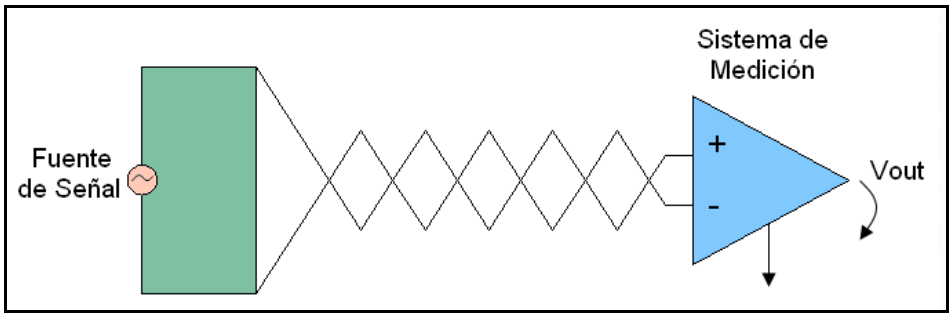


Figura 46: Esquema de trenzado para minimizar el campo próximo a un circuito.
Fuente: HERRADON, Rafael. 2006. Curso de introducción a los sistemas de radio comunicaciones móviles [en línea]. España. ‹http://www.Ocw.upm.es/teoria-de-la-senal-y-comunica ciones-1/radiocomunicación/contenidos/presentaciones/radiocomunicación-07.pdf›. [consulta 2 de febrero 2010].

2.1.3 Características generales de las tarjetas *secure digital* (SD)

“Algunas versiones poseen una marca en su lado izquierdo que permite habilitar o no la escritura en la tarjeta SD, generalmente utilizan un sistema de archivos pre formateado a FAT32, la tarjeta puede ser reformateada a cualquier sistema soportado por el sistema operativo. No es necesaria la desfragmentación de la memoria, no se acelera la lectura-escritura de la tarjeta, de hecho reduce su tiempo de vida útil.

[33] HERRADON, Rafael. 2006. Curso de Introducción a los Sistemas de Radiocomunicaciones Móviles [en línea]. España. <http://www.ocw.upm.es/teoria-de-la-senal-y-comunicaciones-1/radio comunicación/contenidos/presentaciones/radiocomunicación-07.pdf> [consulta 3 de febrero 2010].

2.1.3.1 Dimensiones de la tarjeta SD

Figura 47: Tarjeta SD.

Fuente: HERNANDEZ, Leonardo. 2009. Tarjeta micro secure digital (SDHC) HP [en línea]. Venezuela. <https://h10025.www1.hp.com/ewfrf/wc/document?docname=c01664451 &tmp_task=prodinfoCategory&lc=es&dlc=es&cc=bo&product=3934937&lang=es> [consulta 12 febrero 2010]. Pp. 1

Micro SD es un formato para tarjetas de memoria flash derivado del *TransFlash* de SanDisk. Es especialmente usado en teléfonos móviles, dispositivos GPS portátiles, reproductores de MP3, consolas de videojuegos y unidades de memoria USB. Tiene un tamaño de 15mm × 11mm × 0,7mm, lo que es un cuarto del tamaño de una tarjeta de memoria SD. Opera siempre con un voltaje de 3.3 V y posee 9 pines. Existen adaptadores que permiten usar tarjetas micro SD en dispositivos compatibles con tarjetas SD, mini SD, *Memory Stick*, etc., de todas maneras no son compatibles universalmente. Existen diferentes velocidades disponibles para las tarjetas SD. La velocidad se mide en múltiplos de 150 KB/segundo. Velocidad de 1x equivale a 150 KB/s. Las tarjetas SD básicas tienen una velocidad de 6x (0,9 MB/seg). Las tarjetas SD de alta velocidad alcanza los 66x (10 MB/seg) y las tarjetas de más alta velocidad alcanzan los 150x o superior.

2.1.3.1.1 Compatibilidades

Pueden insertarse en los *slots* de las tarjetas SD comunes utilizando un simple adaptador pasivo. En tanto, con un adaptador activo, las tarjetas SD pueden ser usadas en slots de tarjetas *Compact Flash* y PC *Card*. El formato SD es lo suficientemente abierto como para que otras empresas puedan acceder al contenido

de las tarjetas SD. Este formato es menos abierto que los *Compact Flash* o las unidades de memoria flash USB. Pero la tarjeta SD es más abierta que la *Memory Stick* de Sony, del cual no existe documentación disponible"[34].

NUMERO DE PIN	NOMBRE	DESCRIPCION
1	DAT3/CS	Línea de Dato
2	CMD/DI	Línea Cmd
3	VSS 1	Masa
4	Vdd	Voltaje (3,3v)
5	Clock	Reloj
6	Vss 2	Masa
7	DAT0/D0	Línea de Dato 0
8	DAT1/IRQ	Línea de Dato 1
9	DAT2/NC	Línea de Dato 2

Tabla 6: Descripción de los pines de la tarjeta SD.

Fuente: ALEGSA. 2009. Definición de security digital [en línea]. Argentina. ‹http://www. alegsa.com.ar/Dic/secure%20digital.php›. [consulta 15 de febrero 2010].

Para explicar las conexiones que necesita la memoria micro SD y poder funcionar hemos diseñado un esquema electrónico de fácil comprensión y lectura, donde se observa todos los integrados y herramientas con sus valores propios, además de los valores indicamos voltajes y resistencias indispensables, (*figura 48*).

[34] ALEGSA. 2009. Definición de security digital [en línea]. Argentina. http://www.alegsa.com.ar/ Dic/secure% 20digital.php. [consulta 15 de febrero 2010]. Pp. 1,2

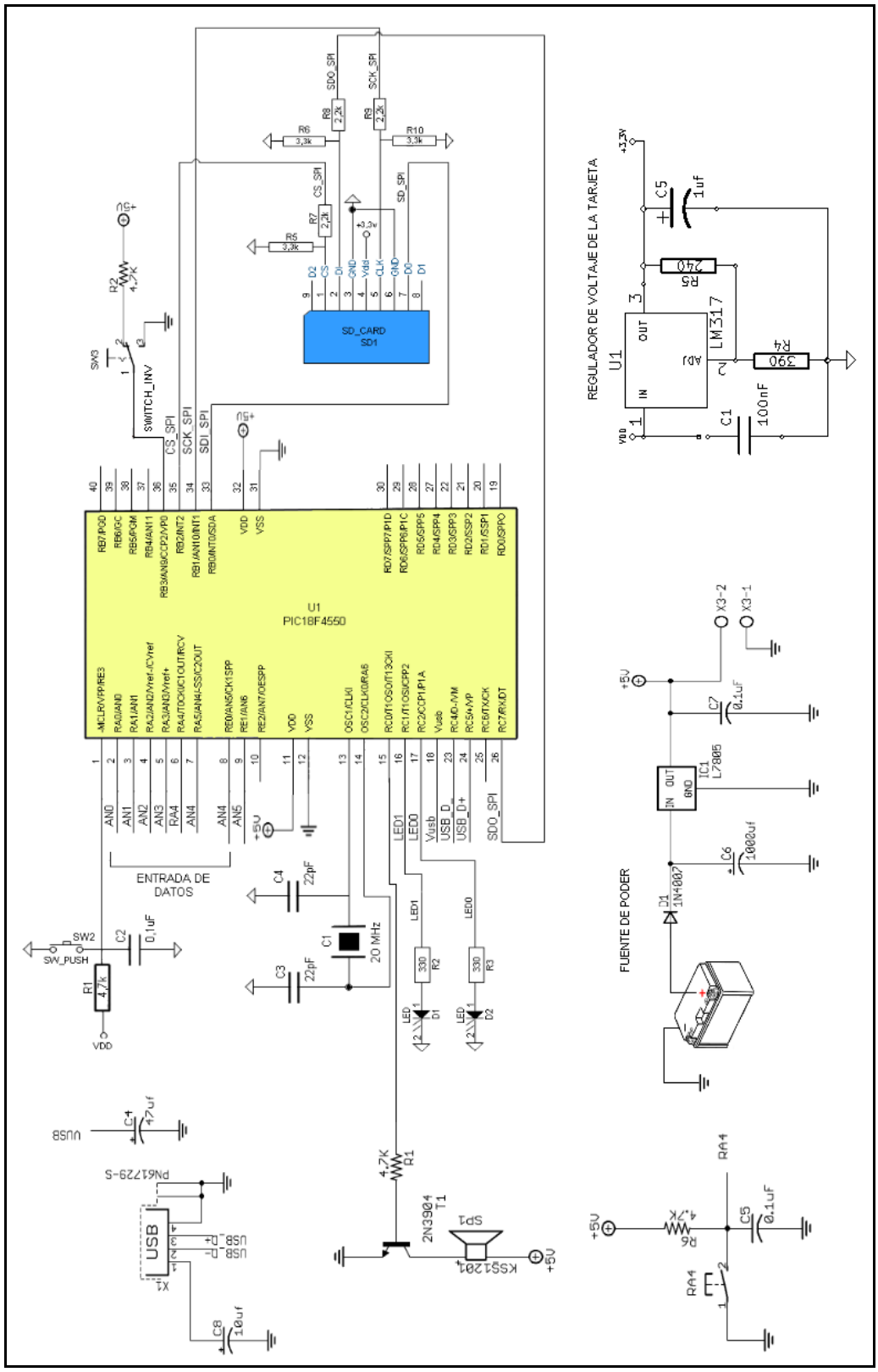


Figura 48: Circuito de la tarjeta SD

2.2 Análisis de las señales de los sensores del vehículo

En el transcurso del trabajo se viene detallando los diversos sensores en función de sus características y parámetros que los definen. En esta parte del desarrollo de nuestro proyecto se planifica mostrar las señales que cada uno de estos sensores entregan, así como también su forma de onda siendo esta analógica o digital, dichos datos almacenaremos en nuestra caja negra, buscando con esto dejar una idea de lo que será la información recopilada.

2.2.1 Sensor de aceleración

Para la adquisición del dato de la aceleración, nuestro objetivo no es enfocarnos en el pedal del acelerador, sino más bien en el sensor de posición de la válvula de la aceleración que de ahora en adelante denominaremos como TPS, donde su funcionamiento varia proporcionalmente a la velocidad y torque en que se encuentre el vehículo, desde totalmente cerrada (ralentí) hasta totalmente abierta (plena carga). La posición de la válvula del sensor del acelerador electrónico define la señal que viene dada de la siguiente manera:

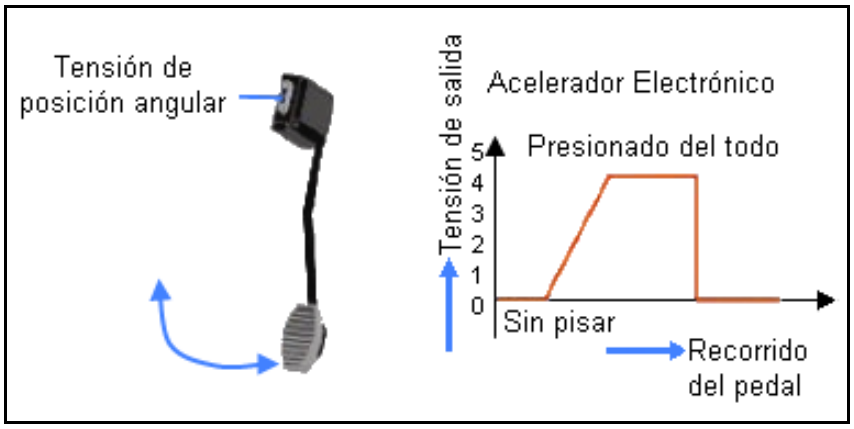


Figura 49: Tensión de salida del sensor de aceleración.

Fuente: TENESACA, Paúl. 2007. Diseño e implementación de un sistema de diagnóstico para ESP con microcontrolador. Cuenca, Ecuador. Pp. 38

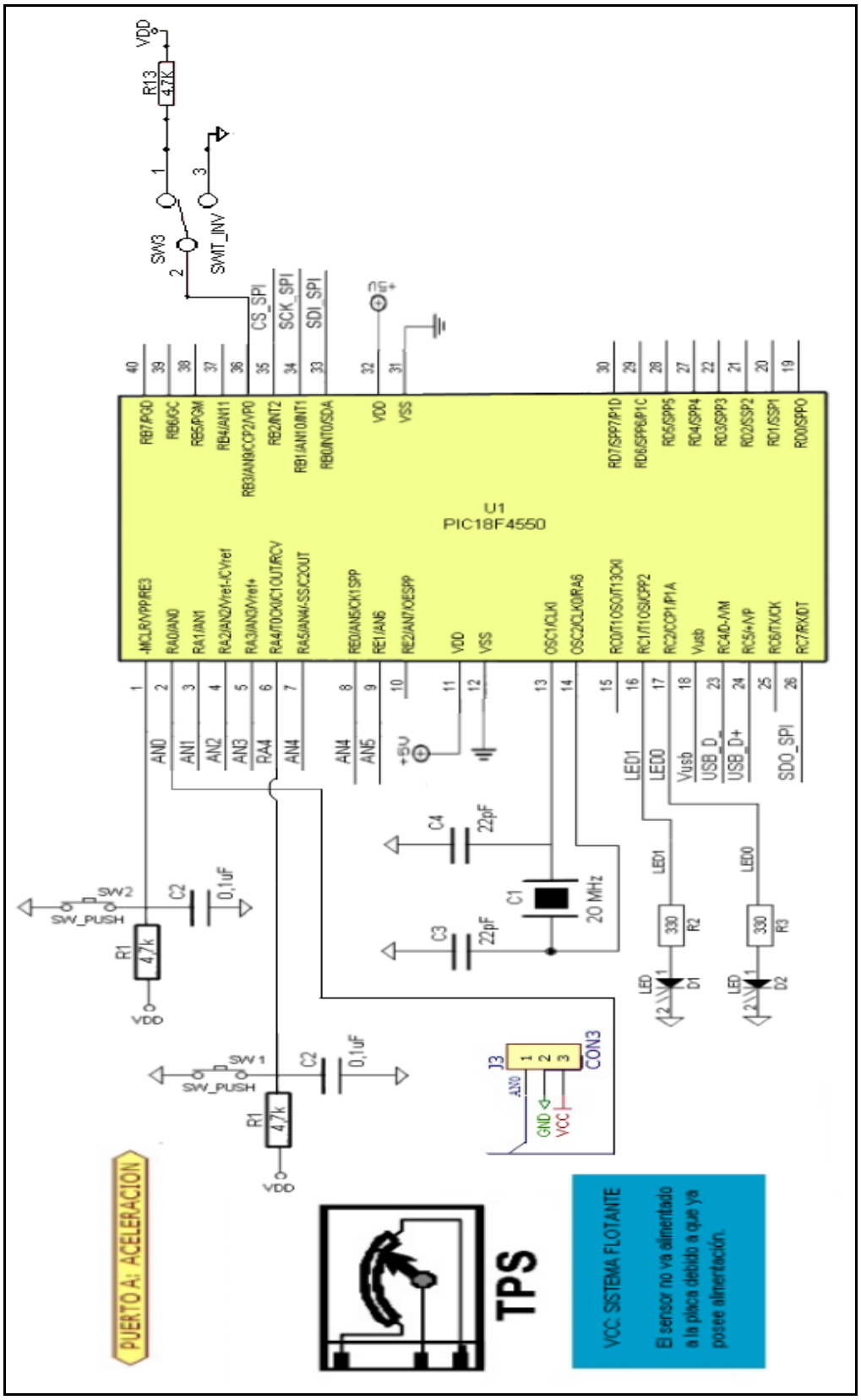


Figura 50: Circuito del sensor de aceleración

El potenciómetro consiste en una resistencia y un cursor que se desplaza sobre ella. La resistencia se alimenta con un voltaje regulado a 5v y del resistor fijo obtenemos un voltaje que va ser proporcional al desplazamiento producido. Podemos encontrarlos de diferentes formas: lineales, circulares, logarítmicos, etc. Para nuestro proyecto utilizaremos del tipo lineal.

"En algunos casos la electrónica viene incluida en el propio sensor, entregando a la salida un valor digital de fácil manejo para la unidad de control, según sea su grado de integración, o también, esta señal puede ser analizada por la propia ECU."[35]

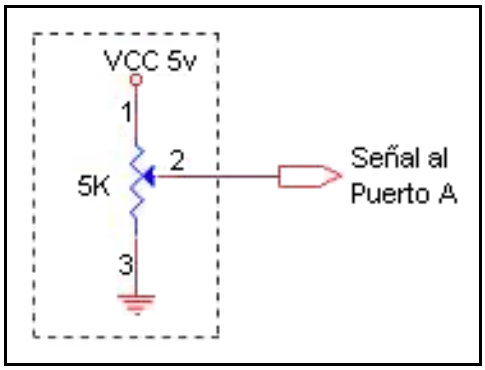


Figura 51: Potenciómetro Lineal.

Fuente: GONZÁLEZ, Guillermo. 2003. Proyecto Instrumento de Medida, parte 2 [en línea]. España. <http://www.arantxa.ii.uam.es/~gdrivera/labetcii/curso0203/pract2.htm>. Pp.2

[consulta 2 de marzo 2010]. Pp.4

2.2.2 Sensor del pedal de freno

Para obtener una señal analógica precisa sobre el pisado del freno durante un accidente, utilizaremos un potenciómetro con su respectiva explicación de los niveles que asumiremos a fin de tener una idea muy clara del rango de frenado utilizado, este circuito también será acompañado con un convertidor A/D para poder guardar los datos en la caja negra.

[35] TENESACA, Paúl. 2007. Diseño e implementación de un sistema de diagnóstico para ESP con microcontrolador. Cuenca, Ecuador. Pp. 39

"Para la simulación de la señal del pedal del freno utilizaremos un potenciómetro lineal, el cual nos dará a conocer la posición del pedal del freno. El potenciómetro es un dispositivo electromecánico, este consta de una resistencia de valor fijo sobre la cual se desplaza un contacto deslizante (cursor) que la divide eléctricamente."[36]

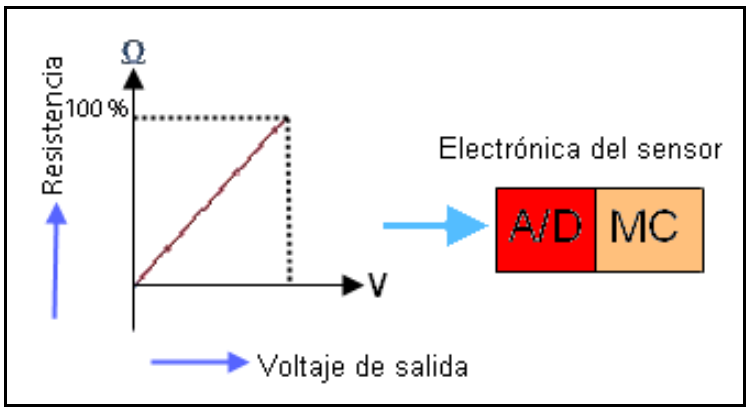


Figura 52: Tensión de salida del potenciómetro.

Fuente: TENESACA, Paúl. 2007. Diseño e implementación de un sistema de diagnóstico para ESP con microcontrolador. Cuenca, Ecuador. Pp. 47

El potenciómetro a utilizar será de tipo lineal de 5 KΩ ya que este nos brinda una salida de voltaje lineal casi similar a la utilizada en el pedal del acelerador anteriormente descrito.

[36] TENESACA, Paúl. 2007. Diseño e implementación de un sistema de diagnóstico para ESP con microcontrolador. Cuenca, Ecuador. Pp. 47

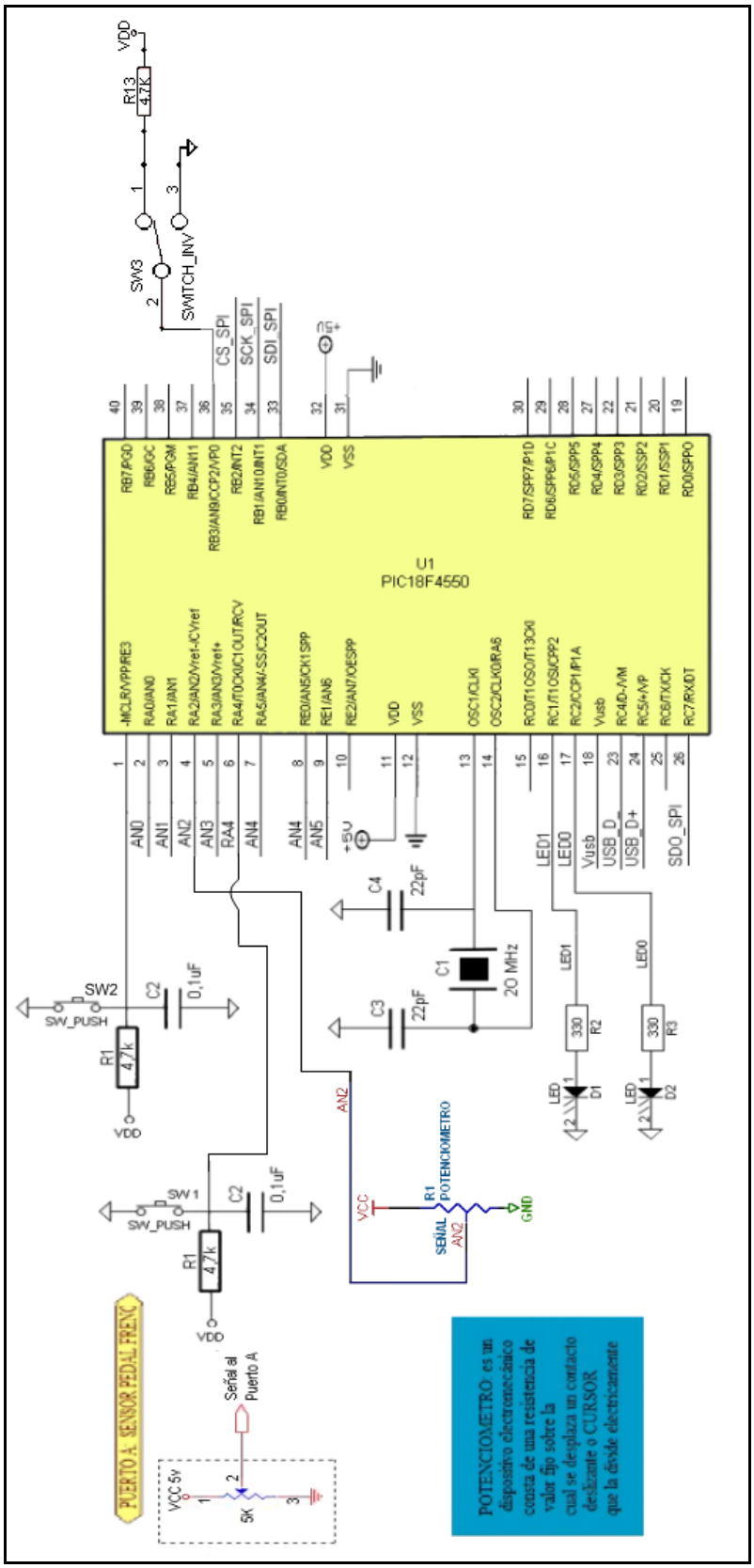


Figura 53: Circuito del sensor de pedal del freno

2.2.3 Sensor de velocidad

La señal que proviene del sensor de velocidad ubicado en la caja de cambios será aprovechada para saber a qué velocidad se produjo el accidente, el sensor utilizado es de tipo hall por lo que no necesita ser transformada su señal para ser almacenada, al ser esta ya de tipo digital, este dato será registrado en la caja negra mediante una conexión mecánica directa hacia el pin generador de la onda, a continuación detallaremos el funcionamiento de dicho mecanismo.

2.2.3.1 Sensor de giro de tipo hall

"El sensor de giro de señal pulsante tiene características completamente diferentes a la alterna, en el caso del sensor de efecto hall, la señal cambia a dos posiciones fijas que son 5 voltios o 0 voltios"[37].

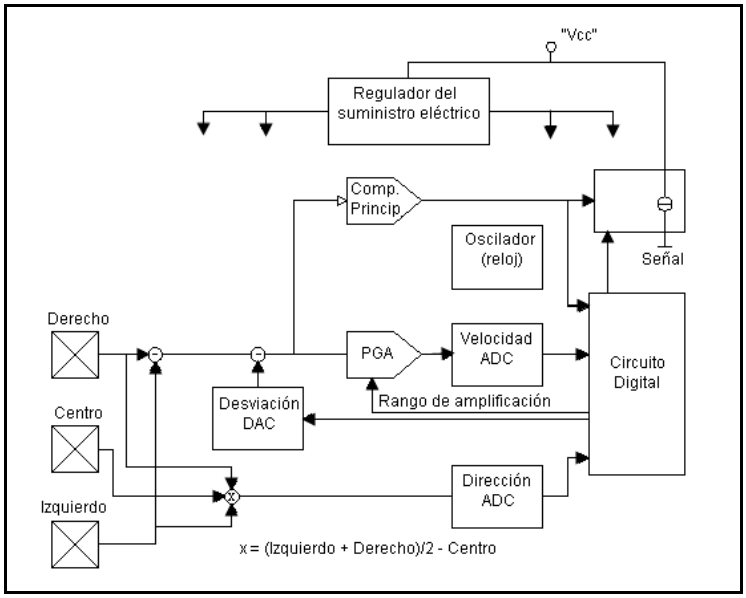


Figura 54: Esquema de bloque del IC hall.

Fuente: JURGEN, Heinz. 2005. Sistemas para la estabilidad del vehículo. Primera Edición. Alemania.

[37] JURGEN, Heinz. 2005. Sistemas para la estabilidad del vehículo. 1ra Edición. Alemania. Pp. 14

Una vez que el mecanismo del sensor efecto hall recibe un cambio de posición del eje, se genera dentro del sensor un cierre de esta señal de referencia a masa, lo cual hace que en el osciloscopio la línea del voltaje baje a 0. En la *figura 55*, se puede apreciar que el sensor tuvo un cambio interno, el cual actúa como un interruptor que envía la señal de referencia a masa, este cambio lo observamos en el osciloscopio como una señal cuadrada, que depende de las veces que el sensor coloca a masa la señal de referencia.

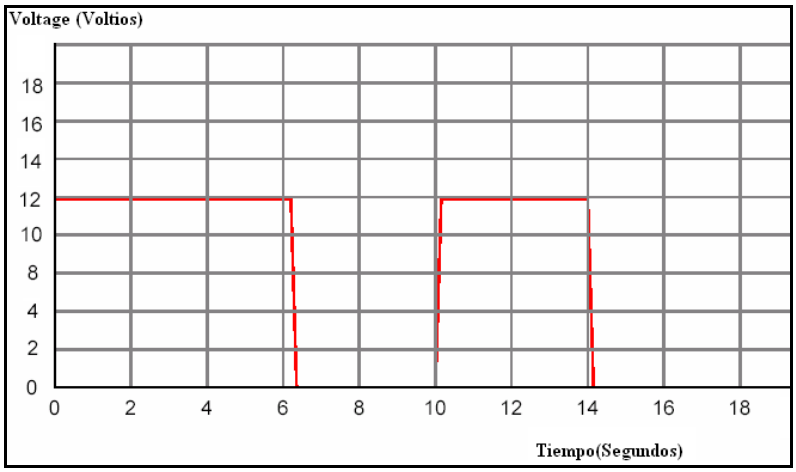


Figura 55: Señal variable de 12 a 0v en el osciloscopio.

Fuente: JURGEN, Heinz. 2005. Sistemas para la estabilidad del vehículo. Primera Edición. Alemania. Pp.16

En la *figura 56*, se muestra el mecanismo por medio del cual funciona el sensor.

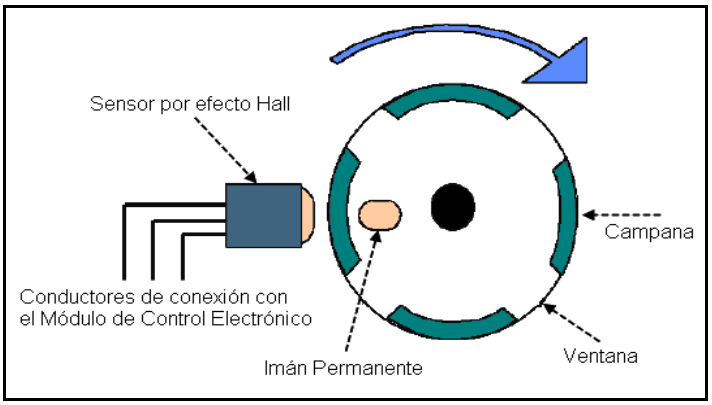


Figura 56: Funcionamiento del sensor hall.

Fuente: JURGEN, Heinz. 2005. Sistemas para la estabilidad del vehículo. Primera Edición. Alemania. Pp.16

"Cada vez que el sensor enfrenta el imán permanente, dentro del sensor se genera un mecanismo que aterriza la señal a masa, una vez que es separado nuevamente el sensor del imán vuelve a subir al voltaje de referencia. De acuerdo a la velocidad de giro del motor se tendrán números de señales cuadradas por unidad de tiempo."[38]

La *figura 57* muestra el esquema electrónico que se debe seguir para la toma de datos de este sensor, junto con un pequeño circuito anexo que nos servirá para simular la onda cuadrada que se generaría en condiciones reales, pero bien se podría prescindir de este anexo y tomar la onda del propio vehículo sin realizar ninguna modificación en el esquema global.

[38] JURGEN, Heinz. 2005. Sistemas para la estabilidad del vehículo. Primera Edición. Alemania. Pp.16

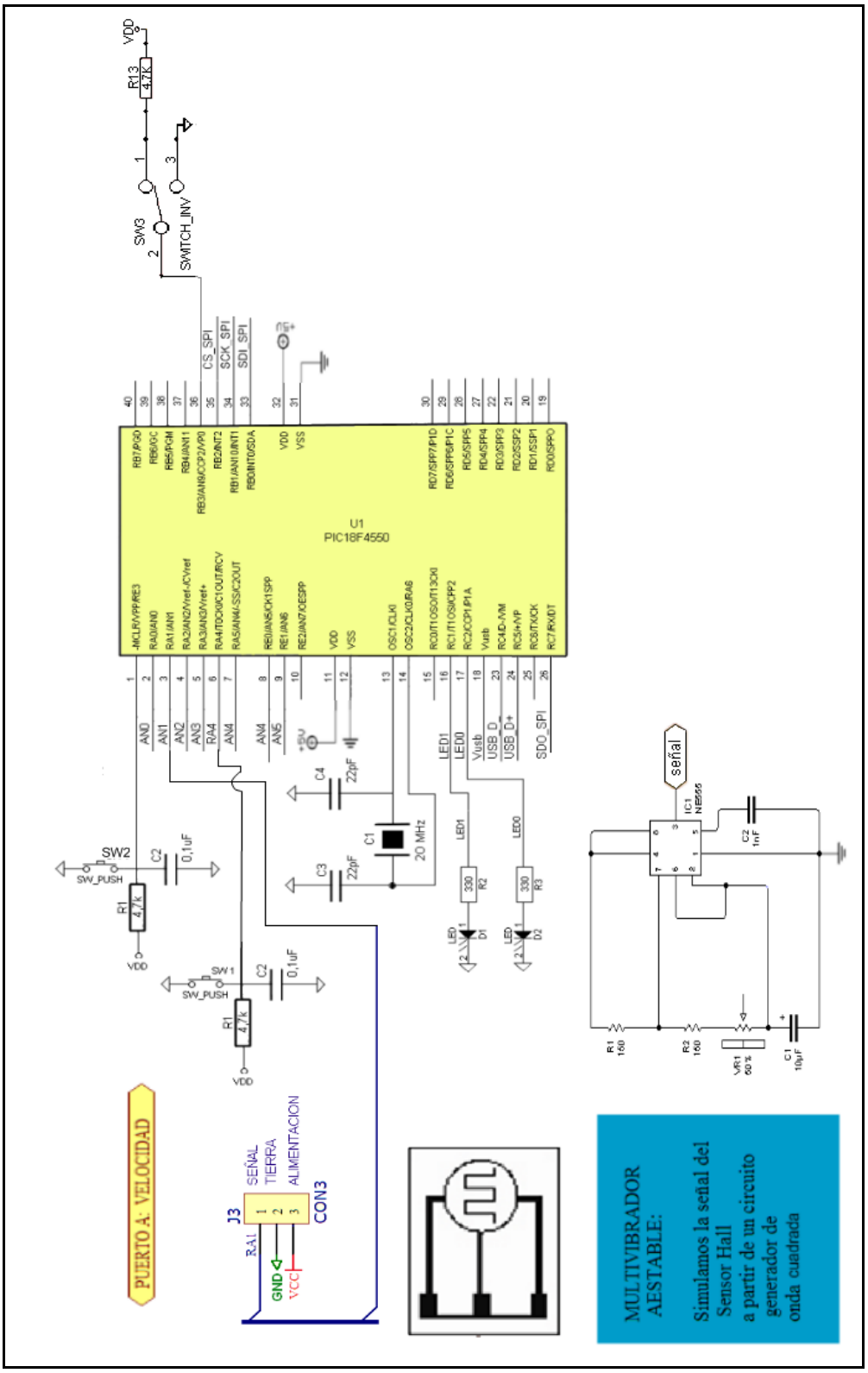


Figura 57: Circuito del sensor de velocidad

2.2.4 Pulsante del cinturón de seguridad

Algunos vehículos vienen dotados de un pulsante que emite una señal visible en el tablero de instrumentos que indica si el conductor e incluso el acompañante no está utilizando el cinturón de seguridad, dicha señal será aprovechada para tener una idea más clara de los sucesos ocurridos en el accidente, este dato no es más que un pulso que se mantendrá continuo hasta que el usuario decida quitarse o en defecto colocarse el cinturón de seguridad. En nuestro proyecto utilizaremos un interruptor que será suficiente para cumplir la función de sensor debido a que nos indicará dos posiciones definidas.

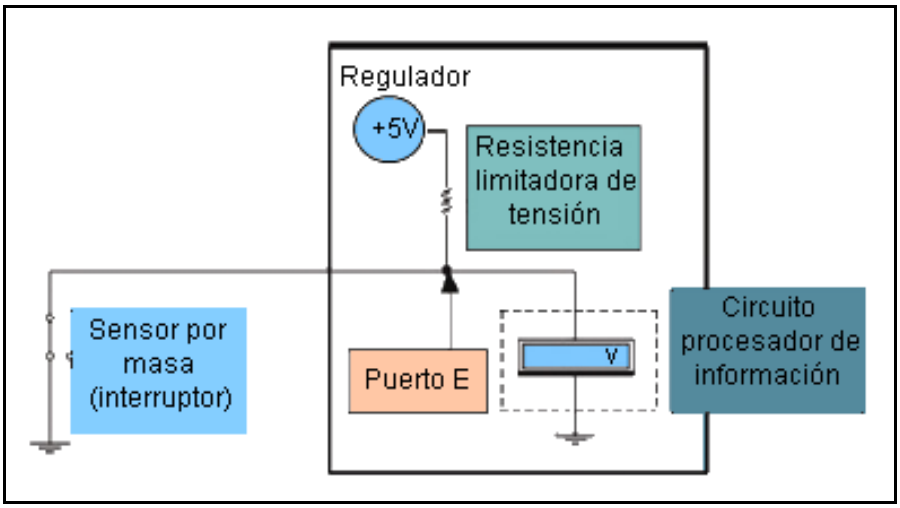


Figura 58: Módulo de control electrónico.

Fuente: CISE ELECTRÓNICA. 2009. Electrónica automotriz parte 3 [en línea]. USA. <http://www.cise.com/ Download/files/ELECTRONICA AUTOMOTRIZ parte 3.pdf.> [consulta 24 de abril 2010]. Pp.1

En los circuitos que utilizan un interruptor como sensor de posición, el interruptor puede estar referido a masa (negativo) o referido a la tensión de referencia (positivo). En nuestro caso particular utilizamos el pulsante referido a masa y su explicación es la siguiente:

"Durante la operación normal del circuito (*figura: 59*), cuando el interruptor se encuentra abierto, el circuito se completa desde el regulador de tensión (+ 5 Voltios), la resistencia limitadora de corriente (2,2 KW), cerrándose a masa a través del circuito procesador de información. El valor de la resistencia de entrada del circuito procesador de información debe ser por lo menos 10 veces mayor que el valor de la resistencia limitadora de corriente, para que el nivel de la tensión de información en el puerto E esté prácticamente en 5 Volt".[39]

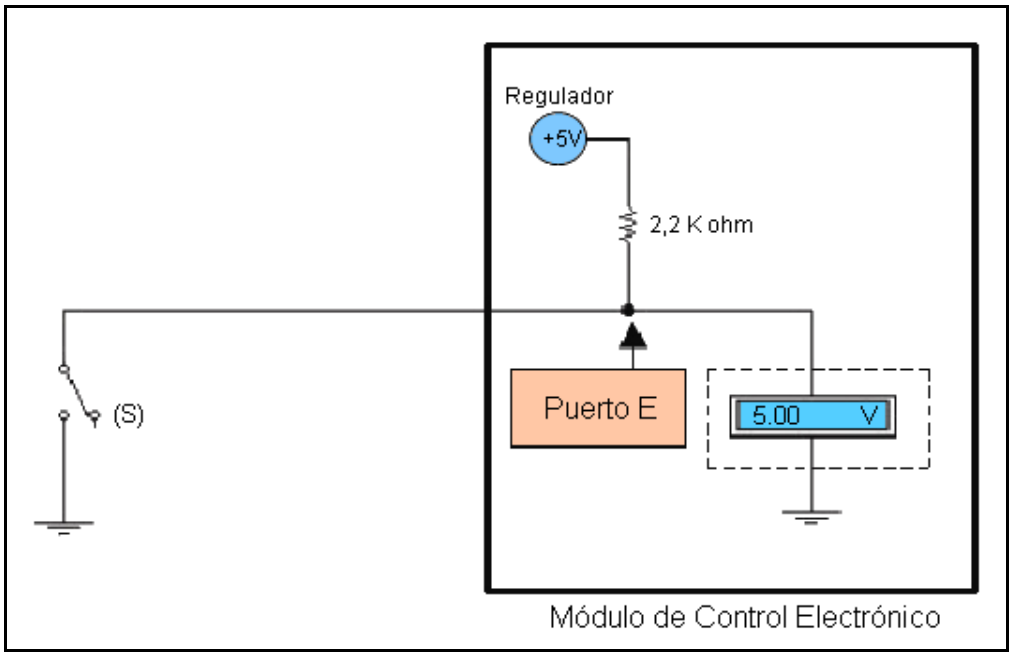


Figura 59: Sensor de posición con el interruptor abierto.

Fuente: CISE ELECTRÓNICA. 2009. Electrónica automotriz parte 3 [en línea]. USA. <http://www. cise.com/Download/files/ELECTRONICAAUTOMOTRIZ parte3.pdf>. [consulta 26 de abril 2010]. Pp.2

2.2.4.1 Pull ups

Ciertos puertos poseen "*Pull up*" internos (En nuestro caso en el puerto A) que pueden habilitarse por software, un *pull up* es una resistencia que se conecta desde un pin de entrada a la fuente. Su función es para que el pin que no se usa quede a un nivel estable.

[39] CISE ELECTRÓNICA. 2009. Electrónica automotriz parte 3 [en línea]. USA. <http://www.cise. com/Download/files/ELECTRONICAAUTOMOTRIZ parte 3.pdf>. [consulta 5 de mayo 2010]. Pp. 2

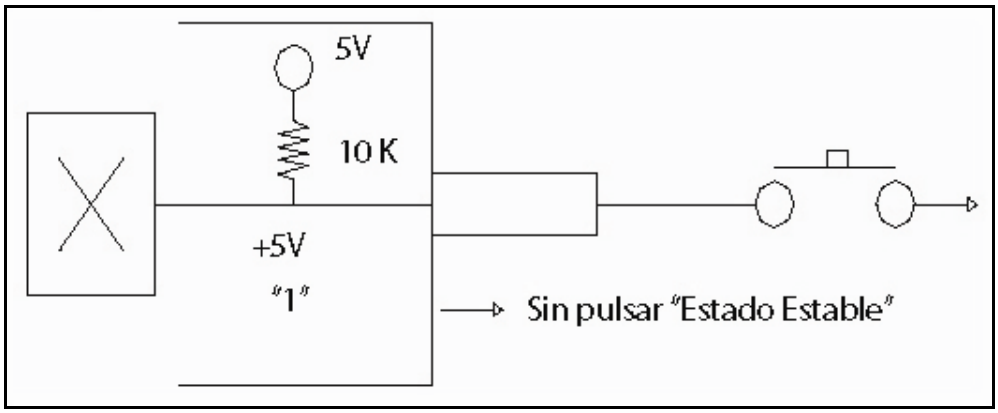


Figura 60: Puertos pull ups.

Fuente: CISE ELECTRÓNICA. 2009. Electrónica automotriz parte 3 [en línea]. USA. <http://www. cise.com/Download/files/ELECTRONICAAUTOMOTRIZ parte3.pdf>. [consulta 7 de mayo 2010]. Pp. 25

El circuito electrónico que se muestra en la *figura 61* no es el que se encuentra en el vehículo es una adaptación que se hizo a una placa para la explicación, el real tendrá variaciones en cuanto a resistencias y voltajes pero su esquema será muy similar al graficado a continuación:

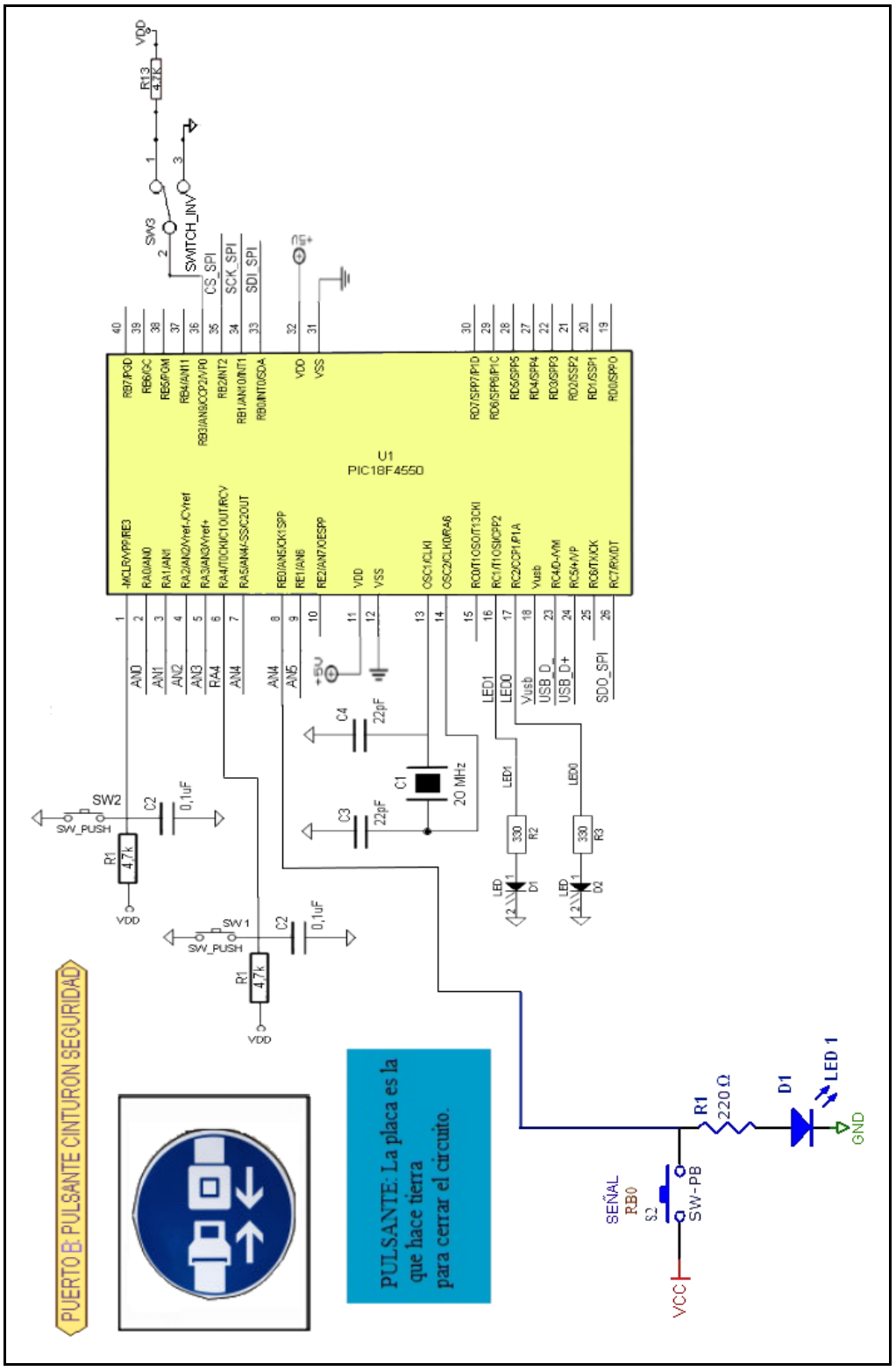


Figura 61: Circuito del pulsante del cinturón de seguridad

2.2.5 Sensor de impacto

Este tipo de sensores son utilizados en los sistemas a*irbags* como un aviso de que la detonación de la bolsa se debe activar, son muy precisos y para nuestro proyecto funcionan haciendo que la recopilación de datos se detenga como un método de seguridad para no perder lo guardado anteriormente, al ser de tipo micro mecánicos como ya se explicó anteriormente permiten variar la intensidad del impacto con el que se activan hasta 100g de fuerza, la señal enviada sería de tipo digital por lo que su conexión se simplifica. Para nuestro proyecto empleamos el diagrama del circuito de a*irbag*, el cual se ha modificado hasta obtener un esquema que nos proporcione una señal del sensor de impacto que sea clara y comprensible para nuestro microcontrolador.

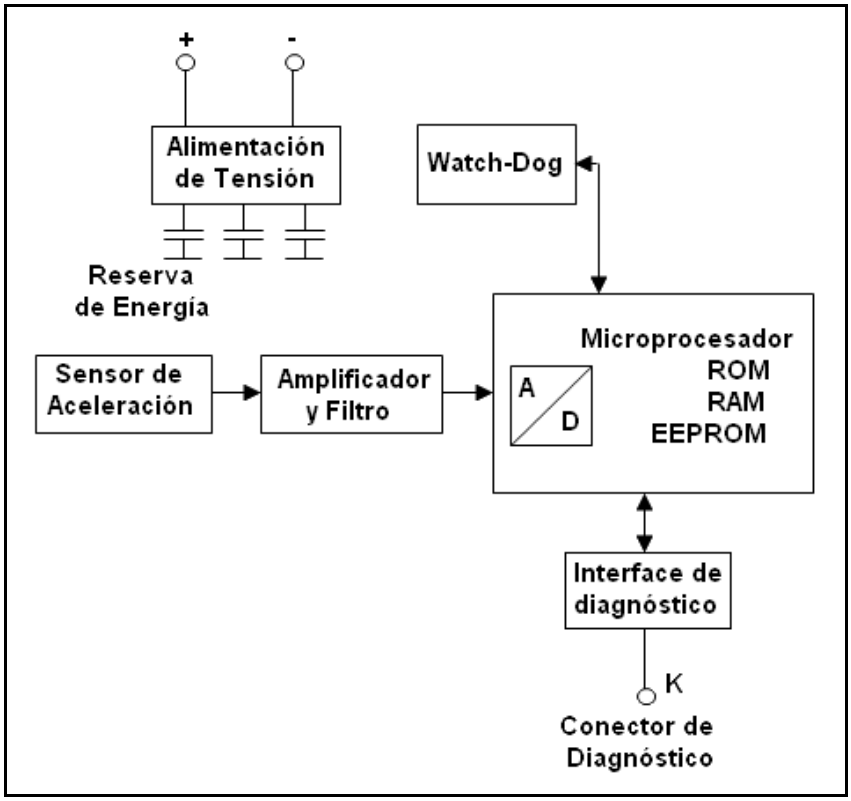


Figura 62: Diagrama del sensor de impacto

En este circuito se digitaliza la señal, luego se amplifica y filtra la misma, posteriormente se adquiere el dato, para finalmente convertirla a forma digital.

2.2.5.1 Sensor piezoeléctrico

"El tratamiento de la señal se logra mediante un comparador de tensión, que tendrá una salida TTL, la cual ingresa al microcontrolador PIC18F4550. De esta forma se abre una ventana de tiempo de un segundo y se mide la cantidad de pulsos que son generados en dicho intervalo.

Los sensores piezoeléctricos pre-amplificados producen un valor de tensión proporcional a la excitación aplicada en la salida del amplificador y su comportamiento resulta independiente del conexionado exterior, puesto que carga y resistencia de entrada del amplificador se mantienen constantes siempre. Este tipo de sensores precisa alimentación.

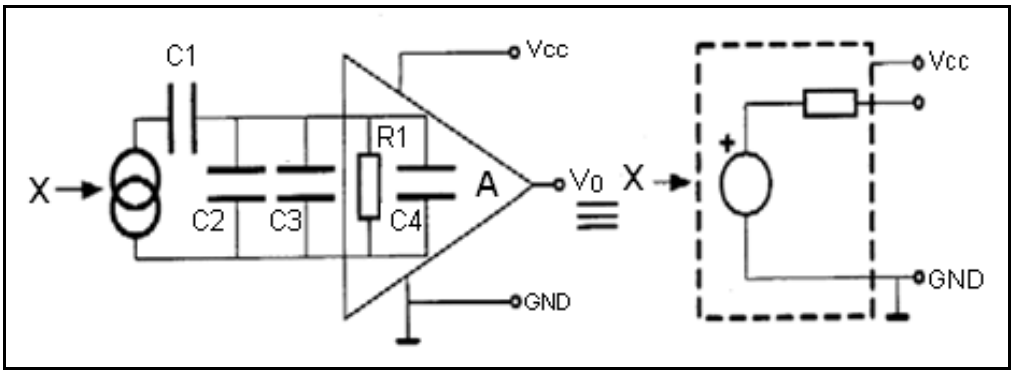


Figura 63: Circuito simplificado del sensor de vibración

La salida de este sensor, provee una tensión proporcional a la deformación del material, luego es ingresada a la entrada no inversora de un amplificador operacional para obtener una salida digital TTL compatible con el microcontrolador utilizado."[40]

[40] MILLAN, A. 2004. Determinación de los Parámetros Mecánicos de una Máquina de Inducción Mediante Mediciones en una Máquina de Corriente Continua Acoplada a su Eje. Venezuela. Pp. 418, 421

2.2.5.2 *Watchdog*

El *watchdog* digital consiste en un dispositivo formado por un contador descendente que puede ser utilizado para recuperar el control del microcontrolador cuando se produce una perturbación en el software. Aunque la traducción sea perro guardián, es un concepto de protección usado para reiniciar el programa cuando éste se pierde o realiza una acción no prevista. Los *watchdog* existen físicos o por programa, funcionan contando cada ciertos pulsos de reloj en un determinado tiempo esperando algún evento generado por el programa, si no le llega tal el *watchdog* se activa y hace que todo empiece de nuevo y si le llega el evento, entonces todo está bien y no hace nada.

La *figura 64* muestra el circuito de conexión del sensor de impacto, junto con los anexos necesarios para su correcto acoplamiento a nuestro dispositivo, por ser este muy delicado se pretende minimizar los efectos que podrían desencadenar en una explosión imprevista e innecesaria del dispositivo *airbag*.

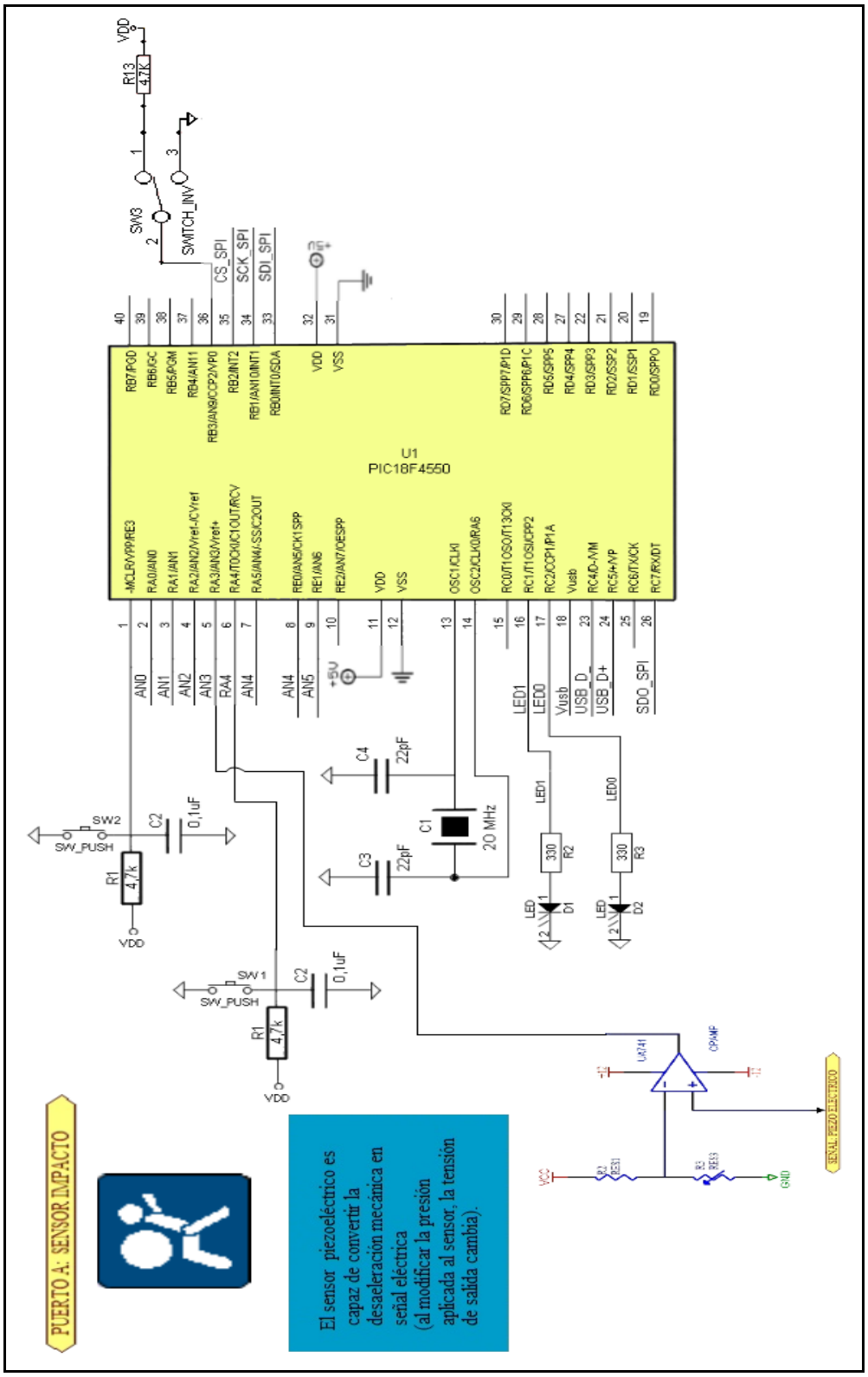


Figura 64: Circuito del sensor de impacto

2.2.6 Sensor de posición del volante de dirección

Cuando se produce el accidente es necesario en ciertas ocasiones saber cuál era la dirección del vehículo, para esto ocuparemos una señal analógica bien definida por un optoacoplador que nos dará la posición exacta del volante a cada momento, esta señal será convertida para poder ser registrada y almacenada en la caja negra.

2.2.6.1 Optoacoplador

Consiste en un diodo como etapa de entrada y un fototransistor npn como etapa de salida (*figura: 65*). El medio de acople entre el diodo y el sensor es un transmisor infrarrojo (IR) de cristal. Los fotones emitidos desde el diodo (emisor) tienen ciertas longitudes de onda establecidas.

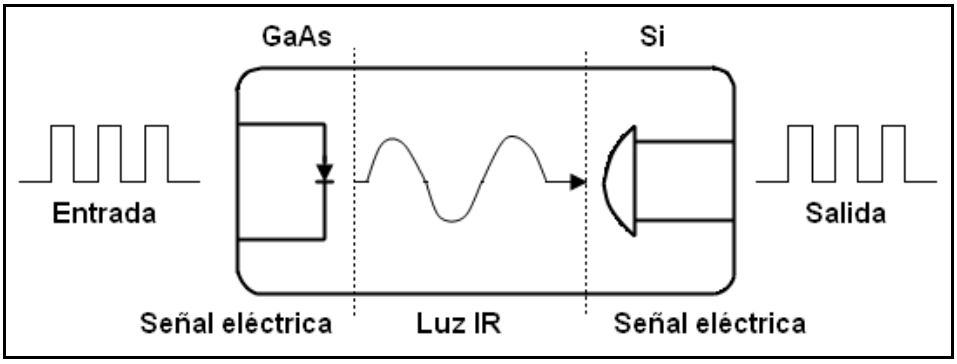


Figura 65: Descripción de un optoacoplador.

Fuente: GOMEZ, José. 2006. Fotónica aplicada a la informática [en línea]. USA. ‹http:// personales.upv.es/jogomez/fai/tema08.html›. [consulta 30 de mayo 2010]. Pp. 3

Si la tensión de entrada varía, la cantidad de luz también lo hará, lo que significa que la tensión de salida cambia de acuerdo con la tensión de entrada. De este modo el dispositivo puede acoplar una señal de entrada con el circuito de salida. La figura 66 muestra un pequeño circuito que digitaliza la señal del optoacoplador, permitiendo su desarrollo y conexión a nuestro dispositivo

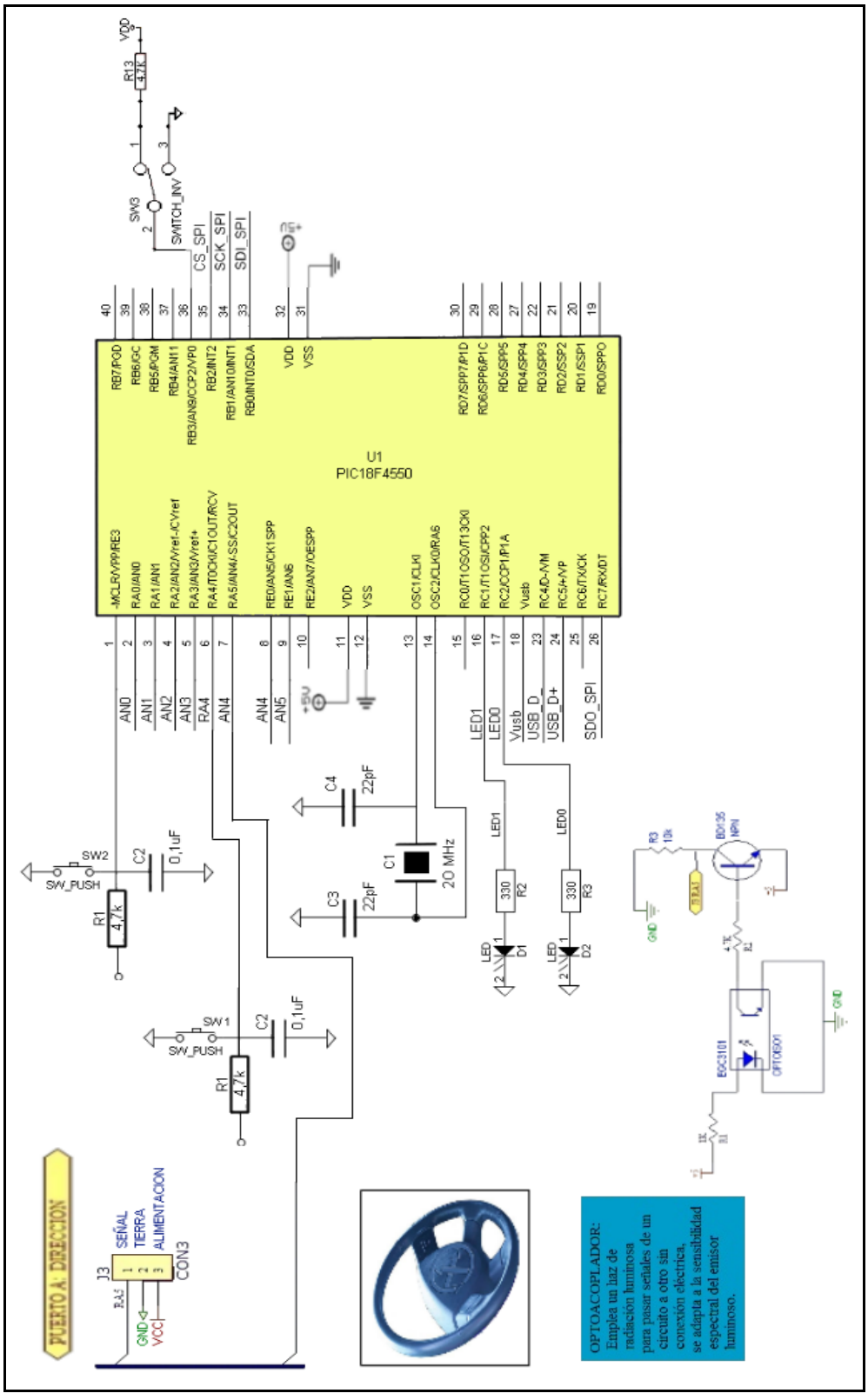


Figura 66: Circuito del sensor de dirección

2.3 Conclusiones

- Se debe tomar en cuenta las especificaciones en el recubrimiento y ubicación de la caja negra, al ser este decisivo en la conservación intacta de los datos guardados en ella.
- Las memorias micro SD permiten una mayor compatibilidad de lectura con respecto a otras memorias de distintos formatos.
- El potenciómetro del TPS es de tipo angular debido a la sencillez y costo de implementación, sin embargo existen otros como el láser o el ultrasónico que no son muy utilizados.
- La conexión de nuestro sensor de velocidad presenta las siguientes especificaciones:
  - Cable de alimentación del circuito del sensor puede ser de 5 o 12 voltios.
  - Un cable de masa que utilizaremos en el sensor para aterrizar la señal, donde encontramos un voltaje de máximo 30mv.
  - La señal cambiará de acuerdo a la velocidad de giro del eje en el cuál esté montado el sensor de onda cuadrada, lo cual representa una señal pulsante.
- Para el pedal de freno seleccionamos el potenciómetro de tipo lineal debido a las siguientes características:
  - Bajo costo.
  - Fácil de manejar.
  - Poseen las características suficientes para generar nuestra aplicación debido a su comportamiento ideal.
- El puerto A, en nuestra placa está orientado al manejo de señales analógicas y pulsantes, el puerto E está dedicado a atender el pulsante del cinturón de seguridad que es digital.
- Al abrir el circuito entre el módulo electrónico de control y el interruptor (pulsante del cinturón de seguridad), dará como resultado un nivel de tensión constante de 5 Voltios en el puerto E.

- Si el circuito entre el módulo electrónico de control y el interruptor (pulsante del cinturón de seguridad) se corto circuita a masa, dará como resultado un nivel de tensión constante de 0 Voltios en el puerto E.
- Es evidente que cualquiera de estas dos situaciones dará una falsa información al microcontrolador.

CAPITULO III

DISEÑO DEL SOFTWARE

Como complemento del trabajo de tesis hemos considerado necesario para la realización de un software que este encargado de encaminar los datos provenientes de los diversos sensores y almacenarlos dentro de nuestro proyecto, dicho programa es realizado en MPLAB, el cual es una herramienta muy utilizada en la programación de microcontroladores como por ejemplo nuestro PIC18F4550, no podemos colocar el programa realizado por lo extenso de este pero si explicaremos de forma detallada las funciones utilizadas en y lo anexaremos en la tesis para mayor facilidad de manipulación, también hablaremos del detalle de cada pin ocupado en el microcontrolador y las funciones que cumplen cada uno de estos.

La tarjeta de almacenamiento de datos es capaz de comunicarse con un computador por medio de un puerto USB (Bus Serial Universal), lo que facilita la visualización de los datos en forma real, para esto se puede variar el comportamiento del dispositivo a conveniencia, en modo ACQ para adquisición datos y en modo DEVICE para hacerlo esclavo del computador y ayudarnos con la visualización de los datos que el microcontrolador ha ido grabando previamente en nuestra memoria micro SD, con esto se desea dejar especificado todos los aspectos de diseño de la tarjeta, así como su programación.

3.1 Diseño de la etapa de los microcontroladores

Para poder realizar una programación primero debemos conocer la parte tangible, por lo que en esta etapa hablaremos sobre la parte denominada hardware, este viene dado de acuerdo a las especificaciones dadas en el capítulo II y solo lo adaptamos un poco a conveniencia de exposición, al ser un prototipo, carece de alimentación propia, por lo que simplemente variamos esta al incluir un adaptador pero su funcionamiento es totalmente independiente de los circuitos del vehículo y del computador. Este se encuentra constituido por: circuitos, tarjetas, fuente de alimentación, etc. Nuestro proyecto constará de una tarjeta madre (*figura: 67*), provista de un controlador central, una fuente externa, conexión USB un circuito anexo de memoria, etc, todo esto para visualizar los datos y varios componentes anexos que iremos detallando a lo largo del tema.

En cuanto al software en cada sistema deberá cumplir ciertas condiciones para su operación como una tensión de funcionamiento de 5 voltios, con una intensidad de corriente mínima requerida de 500mA. Para la medición de las respuestas de cada sistema utilizaremos el PIC programado mediante el lenguaje ensamblador MPLAB.

3.1.1 Características generales de la placa

A continuación se describen los puertos que utilizamos en nuestra placa madre (*figura: 67*), tales como: puertos (A, B, C, E), MCLR (*reset*), cristal, microcontrolador 18F4550, fuente de alimentación, bus serial universal (USB). Esto es necesario para que al conocer bien qué función cumple cada elemento de nuestra tarjeta, sea más fácil la comprensión y por ende su correcta ejecución que depende tanto del hardware como del software.

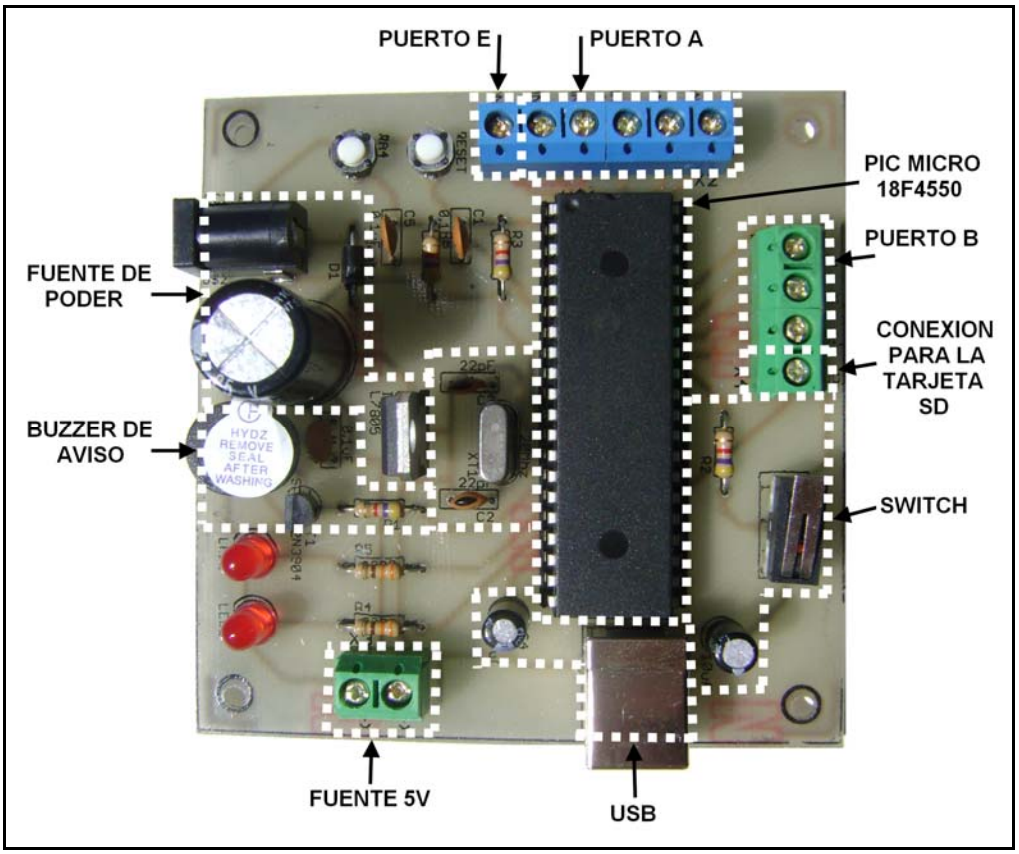


Figura 67: Placa madre (Foto autores)

3.1.1.1 Distribución de recursos en la placa madre

Para un mejor entendimiento de la utilización de los diversos componentes en nuestra placa madre detallamos los recursos utilizados junto con las funciones de los pines que ocupamos del microcontrolador (*tabla 7*), al ser el órgano principal en nuestra caja negra nos facilita la explicación el mostrar sus conexiones y su unión con la tarjeta:

Puerto	Denominación del Pin	Función	Conector Asociado	Pin #
A	RA0	Entrada Analógica o E/S Digital Externa	J1(1)	2
	RA1	Entrada Analógica o E/S Digital Externa	J1(2)	3
	RA2	Entrada Analógica o E/S Digital Externa	J1(3)	4
	RA3	Entrada Analógica o E/S Digital Externa	J1(4)	5
	RA4	Entrada digital externa para *reset*	SW1	6
	RA5	Entrada Analógica o E/S Digital Externa	J1(6)	7
	RA6	Puerto para conexión de Cristal oscilador	Osc2	14
B	RB0	Puerto para conexión de Tarjeta SD, SDI_SPI	J2 (1)	33
	RB1	Puerto para conexión de Tarjeta SD, SCK_SPI	J2 (2)	34
	RB2	Puerto para conexión de Tarjeta SD, CS_SPI	J2 (3)	35
	RB3	Puerto para conexión de cambio de modo de ACQ a DEVICE	SW3	36
C	RC0	Pin de aviso	Buzzer	15
	RC1	Pin de aviso de funcionamiento	LED1	16
	RC2	Pin de aviso de funcionamiento	LED0	17
	RC3	Filtro rectificador para USB	Vusb	18
	RC4	Control de USB.	USB_D-	23
	RC5	Control de USB	USB_D+	24
	RC7	Puerto para conexión de Tarjeta SD, SDO_SPI	J2(4)	26

E	RE0	Entrada Analógica o E/S Digital Externa	J1 (6)	8
	RE3	Entrada digital externa para *reset*	SW2	1

U1	—	Puerto para conexión de Voltaje (+)	VDD	11
	—	Puerto para conexión de Voltaje (-)	VSS	12
	Clk1	Puerto para conexión de Cristal oscilador	Osc1	13
	—	Puerto para conexión de Voltaje (-)	VSS	31
	—	Puerto para conexión de Voltaje (+)	VDD	32

Tabla 7: Distribución de recursos en la placa madre

3.2 Diseño para la adquisición de datos

"El proceso de conversión analógico/digital, de una señal, permite al diseño en cuestión interactuar con las señales simuladas de los sensores, que en nuestro caso particular son: de la posición de la válvula del acelerador, sensor de velocidad, pedal del freno, posición de la dirección, sensor de impacto y por último del cinturón de seguridad.

Para obtener una señal digital que represente aproximadamente una analógica, esta última debe ser muestreada y cuantificada. Ambos procesos determinan en qué grado la señal digital obtenida representa la información contenida en la analógica que le dio origen."[41]

[41] FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Ecuador.

3.2.1 Digitalización de señales

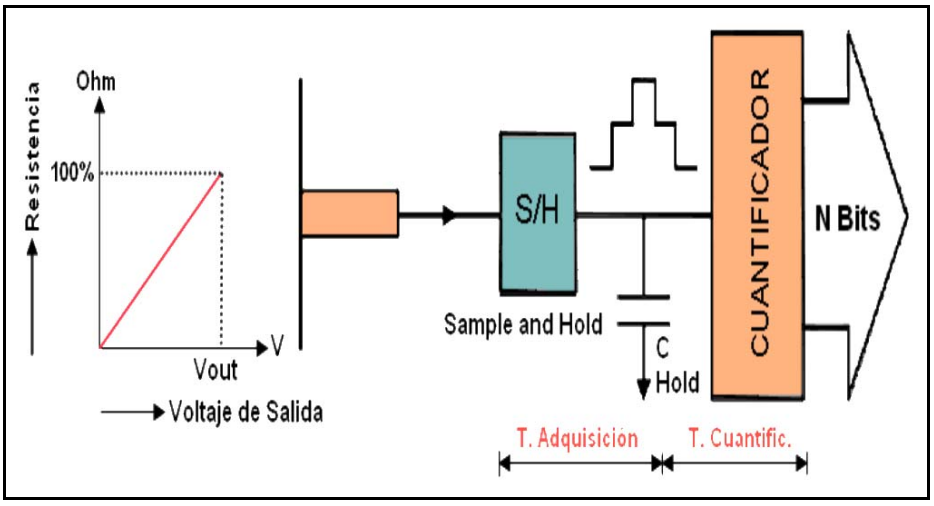


Figura 68: Proceso de digitalización de señales

Después del proceso de digitalización se espera que la señal digital represente fielmente a la analógica. Es imposible lograr que en un 100% la señal digital represente a la continua original, porque al digitalizar una señal se cometen errores. El objetivo del conversor A/D (analógico a digital) es adquirir muestras de la señal analógica a la entrada del puerto A y convertirlas en códigos digitales.

3.2.2 Muestreo y cuantificación

Un conversor A/D es el dispositivo encargado de realizar los dos procesos necesarios para digitalizar una señal: muestreo y cuantificación. (Véase capítulo I convertidor A/D).

- Muestreo**:** En la entrada analógica se carga un capacitor con un valor muestreado *Sample and Hold* (muestra y retiene) a partir de una señal analógica y la almacena en un valor *Hold*.
- Cuantificación**:** Discretiza las amplitudes en un código binario generando la señal digital. El cuantificador aproxima por defecto o exceso un valor (código).

Cada una de las muestras se cuantifica y codifica en un número binario.

3.2.2.1 Procesamiento de señales de la caja negra

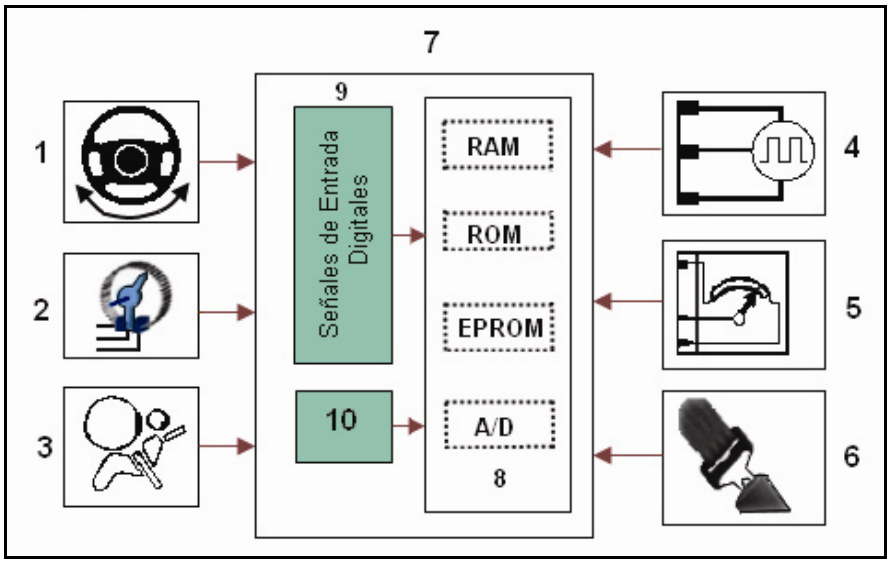


Figura 69: Procesamiento de señales de la caja negra

1) Sensor de ángulo de giro del volante de dirección.
2) Sensor de pedal del freno.
3) Sensor de impacto.
4) Sensor de velocidad del vehículo.
5) Sensor de aceleración.
6) Pulsante del cinturón de seguridad.
7) Conjunto de la caja negra.
8) Microcontrolador PIC18F4550.
9) Preparación de señales hacia la caja negra.
10) Señales de entrada analógicas.

3.3 MPLAB

MPLAB-IDE es un software gratuito que se encuentra disponible en la página web www.microchip.com. Es una plataforma de desarrollo integrada bajo Windows, con múltiples prestaciones, que permite escribir el programa para los PIC en lenguaje ensamblador (*assembler*) o en C (el compilador C es aparte), crear proyectos, ensamblar o compilar, simular el programa y finalmente programar el componente, si se cuenta con el programador adecuado. Incorpora todas las utilidades necesarias para la realización de cualquier proyecto y, para los que no dispongan de un emulador, el programa permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria RAM y EEPROM de usuario como la memoria de programa, según se fueran ejecutando las instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

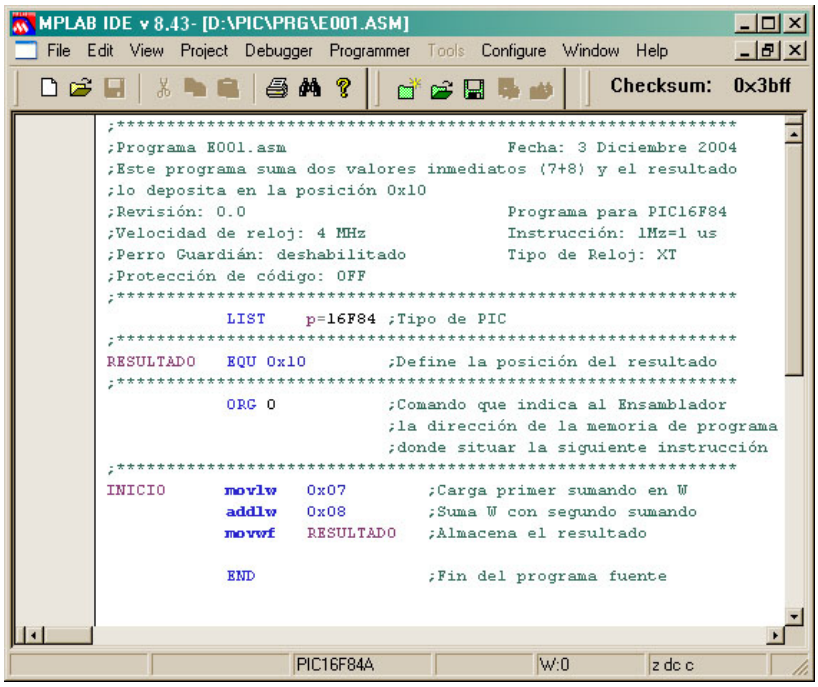


Figura 70: Aspecto del programa con el editor en primer plano.

Fuente: TOBOSO, Emilio. 2010. MPLAB – IDE v8.43 [en línea]. España. ‹http://perso.wanadoo.es/pictob/mplab.htm#mplab_ide_v8.43›. [consulta 17 de junio 2010]. Pp. 2

Partes de MPLAB-IDE:

- Editor: Editor incorporado que permite escribir y editar programas u otros archivos de texto.
- *Proyect manager*: Organiza los distintos archivos relacionados con un programa en un proyecto. Permite crear un proyecto, editar y simular un programa. Además crea archivos objetos y permite bajar archivos hacia emuladores (MPLAB-ICE) o simuladores de hardware (SIMICE).
- Simulador: Simulador de eventos discretos que permite simular programas con ilimitados *breakpoint*, examinar/modificar registros, observar variables, tiempos y simular estímulos externos.
- Ensamblador: Genera varios tipos de archivos objetos y relacionados, para programadores Microchip y universales.
- Linker: Permite unir varios archivos objetos en uno solo, generados por el ensamblador o compiladores C como MPLAB-C18 o compiladores de terceros.
- Programador: Puede trabajar con varios tipos de programadores. El usuario debe seleccionar con cual trabajará, haciendo *click* en opción *Programmer/ Select programmer*, se pueden seleccionar 4 programadores distintos:

  - PICSTART Plus.
  - MPLAB ICD 2.
  - MPLAB PM 3.
  - PRO MATE II."[42]

[42] TOBOSO, Emilio. 2010. MPLAB – IDE v8.43. [en línea]. España. <http://perso.wanadoo.es/pictob/mplab.htm#mplab_ide_v8.43>. [consulta 26 de junio 2010]. Pp. 2, 4

3.3.1 Programación del PIC

Ya se había comentado que por su gran extensión el programa no puede ser incluido en el presente trabajo, pero daremos detalle de todas las funciones que se han ido utilizando así como el trabajo que estas realizan en el programa, su ordenamiento dependerá de las necesidades del usuario:

3.3.1.1 Funciones del programa MPLAB

- Init_Ports( ): Inicializa los puertos del microcontrolador y especifica entradas analógicas y digitales.
- char BUFFER[512]: Contiene el buffer de 512 Bytes, que es utilizado para leer y escribir en la tarjeta de memoria MMC. Debido a una limitante tecnológica por la utilización de un microcontrolador de 8 bits, se tuvo que modificar el archivo m18f4550.lkr que especifica la distribución de los bancos de memoria.
- MMC_Init( ): Inicializa la tarjeta MMC/SD llevándola a un estado de reseteo, este comando responde con 1, si se reinicio exitosamente la tarjeta. En caso de que falle, no estará operativa la tarjeta y no se podrá ni leer, ni escribir en ella.
- MMC_Read_Block (*unsigned long addr*): Función que lee un bloque de 512 bytes en la dirección especificada, la dirección de inicio de lectura es 0x00 00 00 00, e incrementa cada vez 0x200 (512 bytes). Es preciso siempre especificar direcciones múltiplos de 512 bytes. Es decir si se inicia la lectura en la dirección 0x200 las siguientes serán 0x400, 0x600, 0x800, 0xA00, 0xC00, hasta el rango máximo que dependerá según el tamaño de la memoria. Esta función responde con 1, si se leyó correctamente.
- MMC_Read_Block2 (*unsigned char a, unsigned char b, unsigned char c, unsigned char d*): Cumple la misma función que la función de arriba, pero el direccionado se especifica en 4 bytes. Es empleada cuando la tarjeta se encuentra en modo DEVICE. Esta función responde con 1, si se leyó correctamente.
- MMC_Write_Block (*unsigned long addr*): Función que escribe un bloque de 512 bytes, en la dirección especificada. Esta función responde con 1, si se

escribió correctamente.

- MMC_Write_Block2 (*unsigned char a, unsigned char b, unsigned char c, unsigned char d*): Cumple la misma función que está arriba, pero el direccionado se especifica en 4 bytes. Esta función responde con 1, si se escribió correctamente.
- MMC_Read_CID (*unsigned char *buffer*): Lee el registro CID de la tarjeta MMC/SD.
- Init_TMR0(): Inicializa el *timer* 0, para trabajar como un *timer* de 16 bits, con aviso de desbordamiento por interrupción.
- Init_ADC(): Inicializa el módulo ADC, con parámetros básicos de operación.

3.3.1.2 Diagrama de flujos del programa en MPLAB

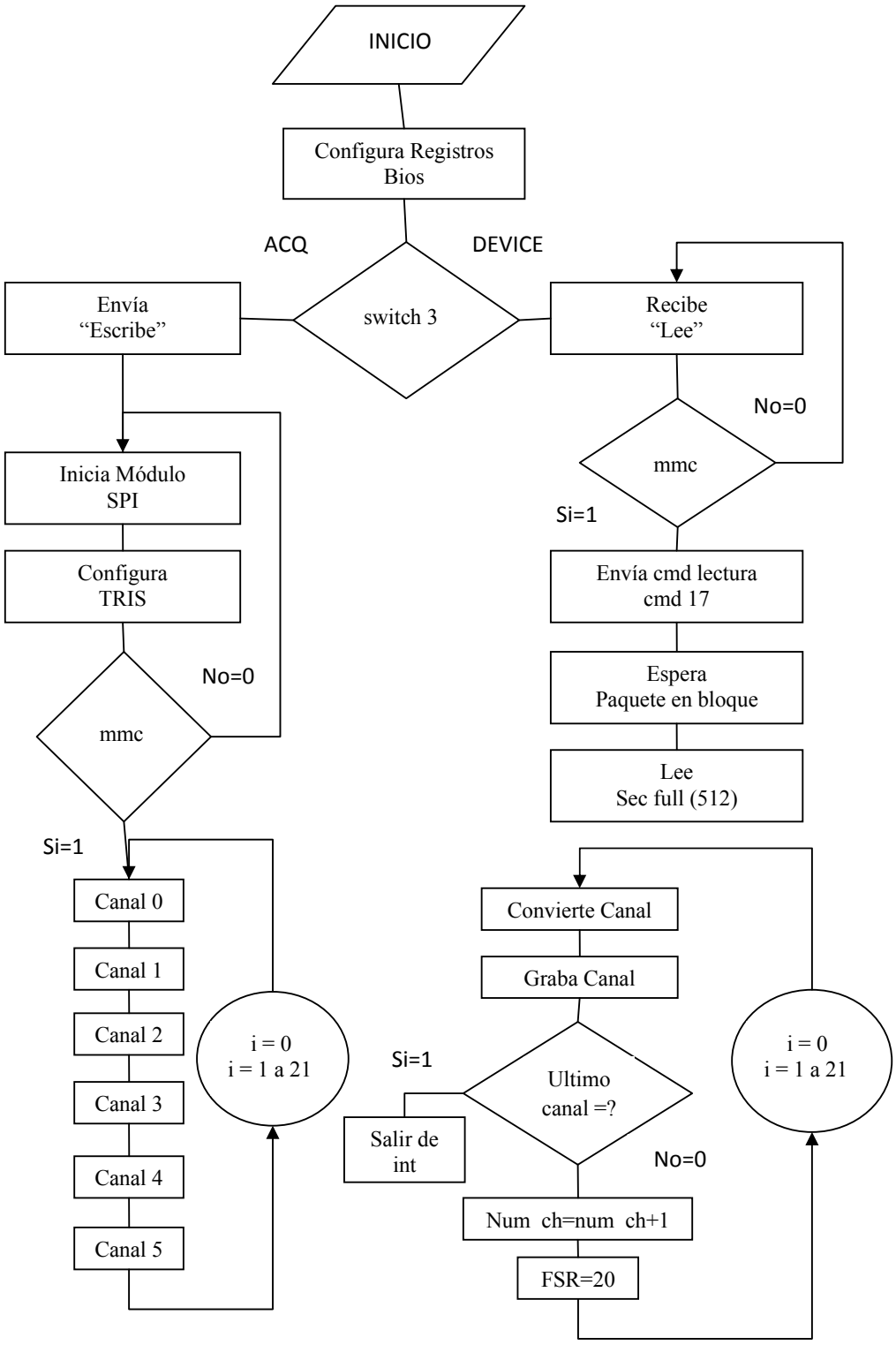


Figura 71: Diagrama de flujos MPLAB

3.3.2 Descripción del programa

La caja de adquisición posee un microcontrolador Microchip PIC 18F4550, corriendo a una frecuencia máxima de 48Mhz. Se puede seleccionar el modo de operación de la caja seleccionando el switch SW3 (*figura: 48*), en "modo ACQ", la caja adquirirá datos empleando los 6 canales (analógicos y digitales) que posee con una resolución real de 10 bits por canal.

Todos los canales son de 0 – 5V compatibles, excepto en el sensor de velocidad que opera con una señal de 0 – 12v, por lo que es preciso acoplar la señal si se tienen otros tipos de niveles a los especificados.

Los datos adquiridos son guardados en una tarjeta de memoria MMC/SD, los datos son almacenados en la tarjeta MMC/SD a una tasa de 10 muestras por segundo, por cada canal.

En modo DEVICE la caja funcionará como un dispositivo esclavo que será comandado a través de un computador personal, empleando comunicación USB 2.0, según el comando enviado la caja responderá y si es el caso del comando se leerá o escribirá información en la tarjeta de memoria MMC/SD, empleando el protocolo de comunicación SPI a 1,5Khz.

3.3.3 Diagrama de organización de una memoria MMC/SD

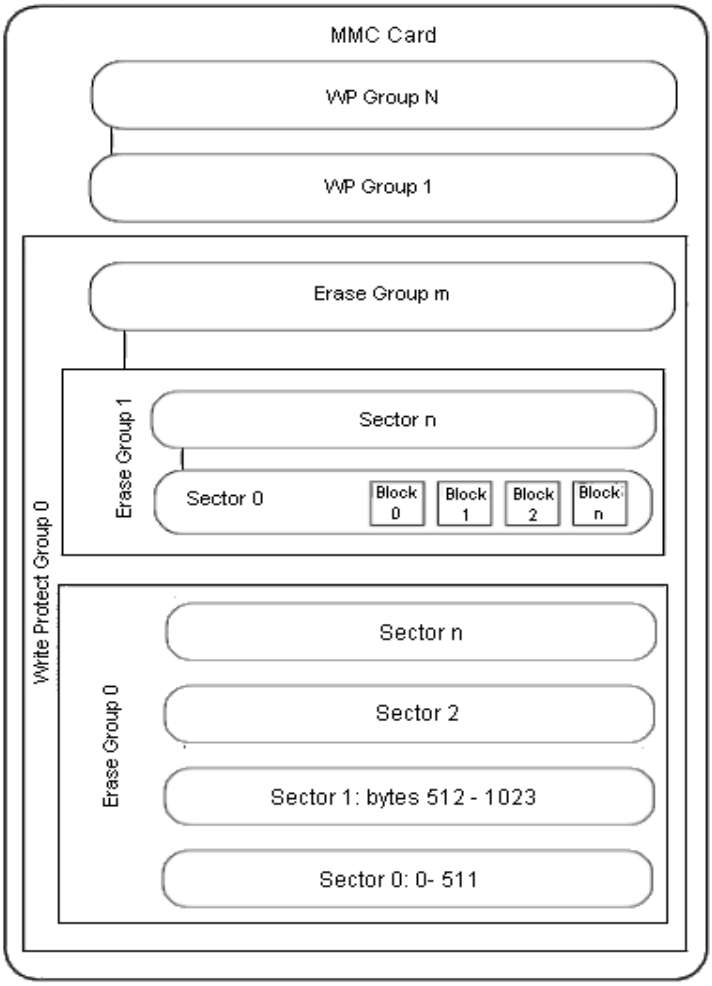


Figura 72: Organización de una memoria MMC/SD

Fuente: SANDISK. 2003. SanDisk security card [en línea]. USA. ‹http://alumni.cs.ucr. edu/~amitra/sdcard/ProdManualSDCARDv1.9.pdf›. [consulta 2 de julio 2010]. Pp. 14

Una memoria MMC/SD contiene varios sectores de 512 bytes, el número de sectores depende del tamaño de la memoria. Por ejemplo una memoria de 64MB contiene aproximadamente 125000 sectores de 512 Bytes.

3.4 MATLAB

MATLAB es el nombre abreviado de *MATrix LABoratory*. Al pasar de los años fue complementado y reimplementado en lenguaje C. Actualmente la licencia es propiedad de *MathWorks Inc*. Es un programa interactivo para computación numérica y visualización de datos. Es ampliamente usado por Ingenieros de Control en el análisis y diseño, posee además una extraordinaria versatilidad y capacidad para resolver problemas en matemática aplicada, física, química, ingeniería, finanzas y muchas otras aplicaciones. Como caso particular puede también trabajar con números escalares tanto reales como complejos, con cadenas de caracteres y con otras estructuras de información más complejas. Está basado en un sofisticado software de matrices para el análisis de sistemas de ecuaciones. Permite resolver complicados problemas numéricos sin necesidad de escribir un programa. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones.

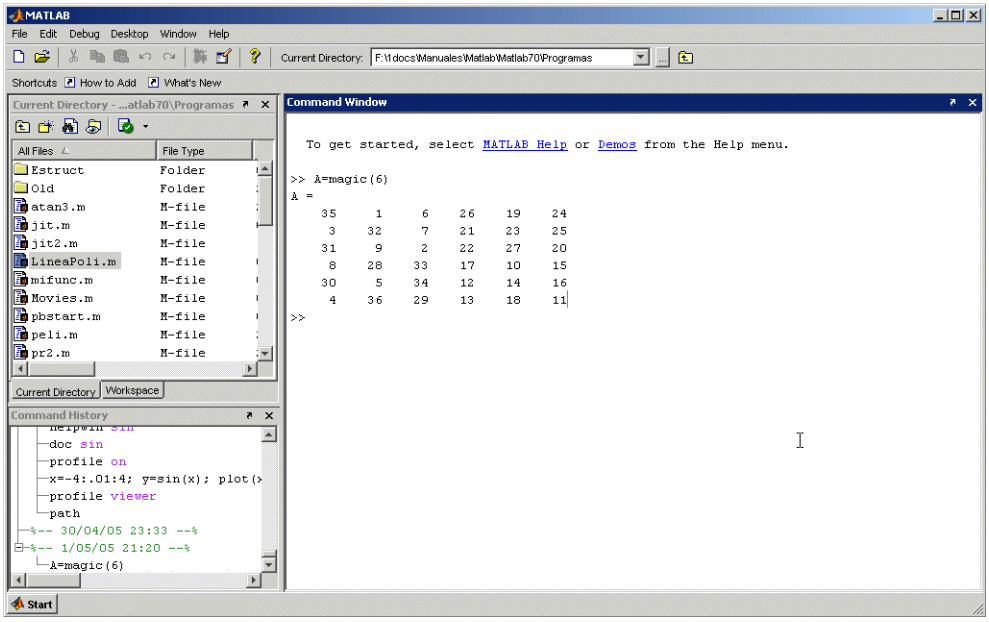


Figura 73: Ventana inicial de MATLAB 7.0.

Fuente: GARCIA, Javier. RODRÍGUEZ, José. VIDAL, Jesús. 2005. Aprenda Matlab 7.0 [en línea]. España. ‹http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/matlab70primero.pdf›. [consulta 9 de julio 2010]. Pp.4

3.4.1 El entorno de trabajo de MATLAB

"El entorno de trabajo de MATLAB es muy gráfico e intuitivo, similar al de otras aplicaciones profesionales de Windows. Las componentes más importantes del entorno de trabajo de son las siguientes:

- El Escritorio de MATLAB (*MATLAB Desktop*), que es la ventana o contenedor de máximo nivel en la que se pueden situar (*to dock*) las demás componentes.
- Las componentes individuales, orientadas a tareas concretas, entre las que se puede citar:
  - La ventana de comandos (c*ommand window*),
  - La ventana histórica de comandos (c*ommand history*),
  - El espacio de trabajo (w*orkspace*),
  - La plataforma de lanzamiento (l*aunch pad*),
  - El directorio actual (c*urrent directory*),
  - La ventana de ayuda (h*elp*)
  - El editor de ficheros y depurador de errores (*Editor & Debugger*)
  - El editor de vectores y matrices (a*rray Editor*).
  - La ventana que permite estudiar cómo se emplea el tiempo de ejecución (*Profiler*).

Otro de los puntos fuertes de MATLAB son los gráficos. A título de ejemplo, se puede teclear la siguiente línea y pulsar *intro*:

>> x=-4:.01:4; y=sin(x); plot(x,y), *grid*, *title*('Función seno(x)') ."[43]

[43] GARCIA, Javier. RODRÍGUEZ, José. VIDAL, Jesús. 2005. Aprenda Matlab 7.0 [en línea]. España. <http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/matlab70primero.pdf>. [consulta 11 de julio 2010]. Pp. 3, 6

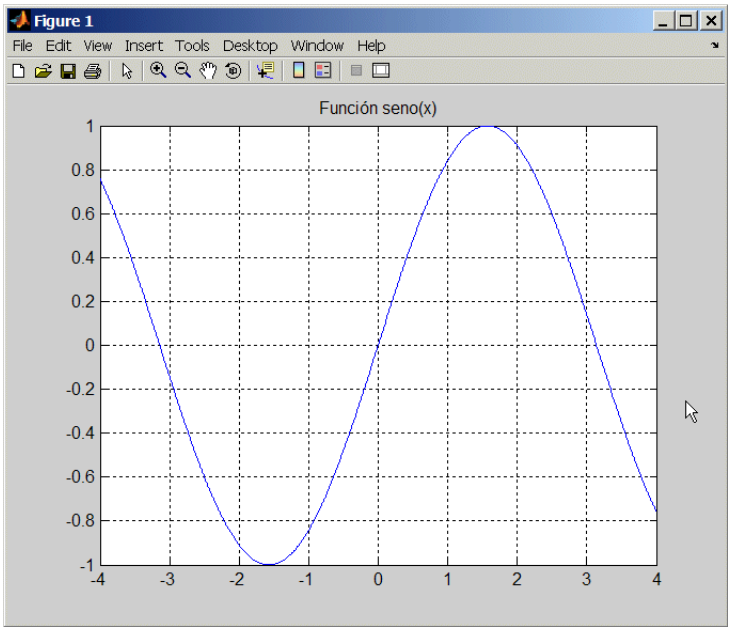


Figura 74: Función seno.

Fuente: GARCIA, Javier. RODRÍGUEZ, José. VIDAL, Jesús. 2005. Aprenda Matlab 7.0 [en línea]. España. ‹http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/ matlab70primero.pdf›. [consulta 11 de julio 2010]. Pp.6

3.4.2 Comunicación USB desde MATLAB con la tarjeta

Para la lectura de los datos es preciso ocupar otra herramienta digital, en este caso el programa seleccionado es MATLAB, el cual con una simple programación se puede configurar para poder revisar los datos guardados en la memoria de datos.

En nuestro caso en particular hemos diseñado dos script que nos ayudaran a dar lectura de los datos, los hemos denominado de la siguiente manera:

- RESET_buffer_MMC.m
- MMC_card.m

Para dar inicio con estos solo se necesita conectar la tarjeta al computador mediante un puerto USB y seguir con los siguientes pasos:

1. Inicializar el programa MATLAB desde el escritorio del computador.
2. En cada reinicio de recopilación de datos ejecutar primero el script: RESET_buffer_MMC.m desde MATLAB
3. Ejecutar desde MATLAB el script: MMC_card.m
4. Seleccionar [1] para inicializar la memoria MMC/SD
5. Según se desee seleccionar opciones 2-5.

Para leer varios sectores de memoria, de la tarjeta MMC, seleccionar [3], iniciar con la dirección de memoria 0x 00 00 06 00, la dirección aumenta en múltiplos de 512 bytes. Es decir la siguiente dirección sera 0x800, 0xA00, etc. El historial de todos los datos, por canal se encuentra en Canal0, Canal1, Canal2, Canal3, Canal4, Canal5. Estos son estáticos es decir que cada vez que se cierre MATLAB se perderán y será necesario iniciar desde el paso 1, para adquirir los datos y analizarlos.

3.4.3 Diagrama de flujos del programa en MATLAB

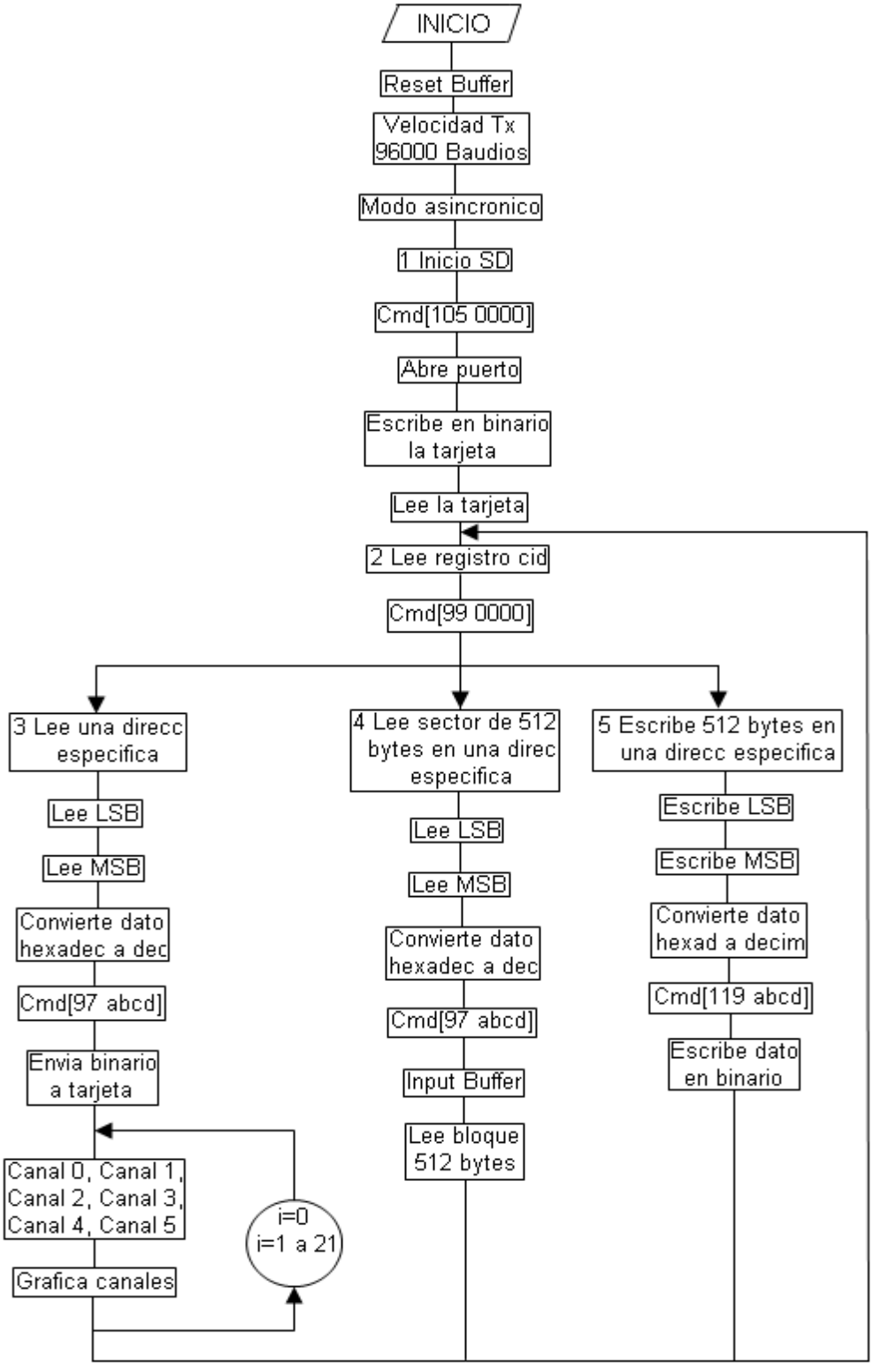


Figura 75: Diagrama de flujos MATLAB

3.5 Pruebas de adquisición de datos

Para reforzar lo que se ha expuesto en este trabajo, procederemos a realizar dos pruebas prácticas en el vehículo Volkswagen Polo sobre el correcto funcionamiento de la adquisición de datos en nuestra tarjeta de almacenamiento, también mostraremos las gráficas que generan estos datos visualizados tanto en un osciloscopio como en el computador para comparar su semejanza, una de estas pruebas se la realizará en un sensor analógico escogido al azar (sensor de aceleración) y la otra en un sensor de carácter digital que por cuestión de costos el elegido es el cinturón de seguridad debido a lo elevado que implicaría probarlo en un sistema *airbag*.

3.5.1 Sensor de aceleración

La señal que obtuvimos del TPS es del tipo analógica, (*figura: 76*), la misma que será transformada a señal digital por medio de un convertidor analógico/digital (A/D), para luego ser almacenada en la memoria del nuestro dispositivo. Para visualizar la forma de onda primero colocamos un osciloscopio a la salida de la señal y se pudo ver la forma de onda en ralentí del TPS, que se muestra a continuación:

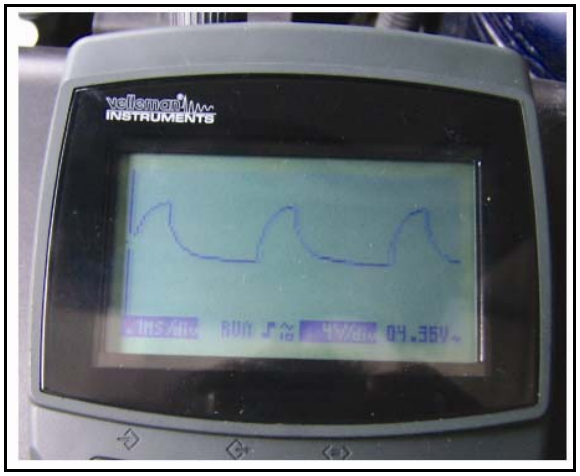


Figura 76: Forma de onda del sensor TPS (Foto autores)

Seguido tenemos la *tabla 8,* en la que se especifica cada uno de los pines del TPS, así como los voltajes obtenidos.

DESCRIPCION DE PINES DEL SENSOR TPS						
1	2	3	4 (GND)	5	6 (GND)	
4,77 V	5 V	3,27 V	0	3,27 V	0	Encendido en Ralentí
4,76 V	5 V	3,27 V	0	3,27 V	0	Apagado con switch en ON

Tabla: 8

En la pantalla del computador se presenta el resultado de la adquisición de datos de la señal del sensor Tps, la misma que se la realizó en el programa MATLAB. La *figura 77* muestra que la curva y su inclinación son muy similares a lo mostrado en el osciloscopio. Probando que los datos que se obtienen son reales.

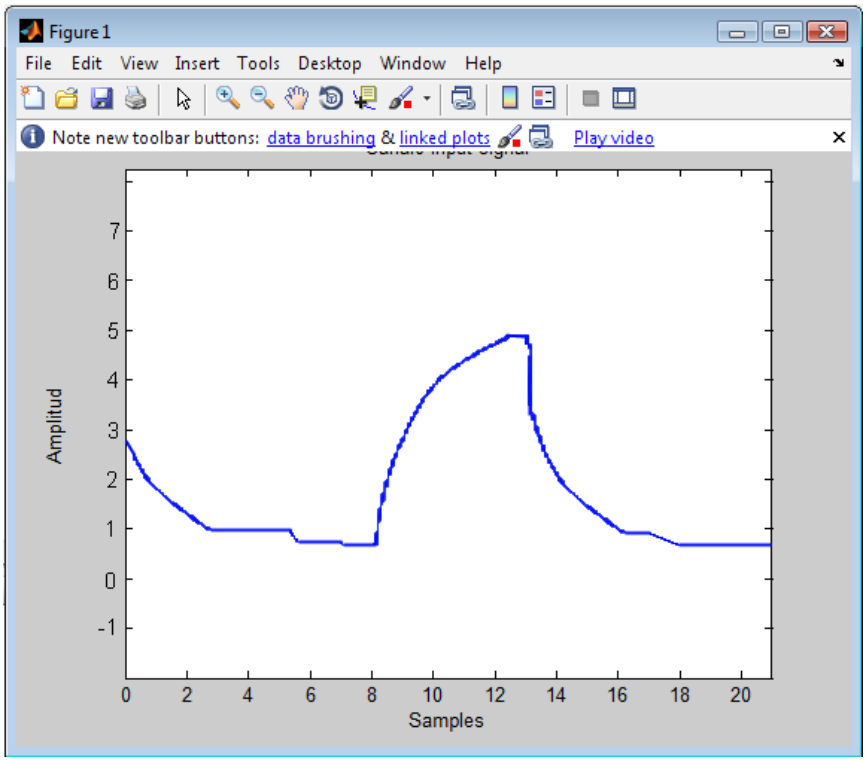


Figura 77: Forma de onda del sensor de aceleración vista en MATLAB

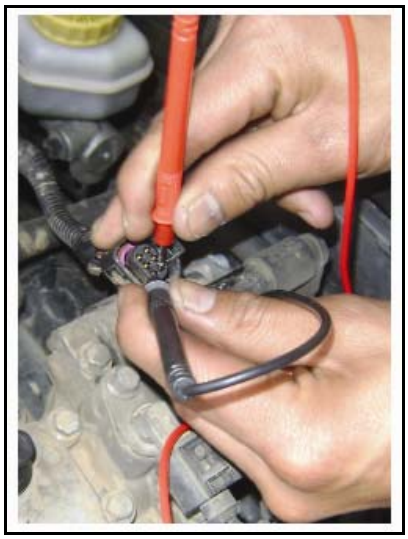


Figura 78: Medición de los pines del TPS (Foto autores)

3.5.2 Pulsante del cinturón de seguridad

La señal que obtuvimos del pulsante del cinturón en el vehículo es de tipo digital, (*figura: 79*), esta no necesita ser transformada para ser almacenada en la memoria de nuestro dispositivo. Para visualizarla colocamos un osciloscopio a la salida de la señal y se pudo ver la forma de onda que en este caso es solo un pulso que varía siempre y cuando el conductor abroche y desabroche su cinturón, esta variación es la que se muestra a continuación:

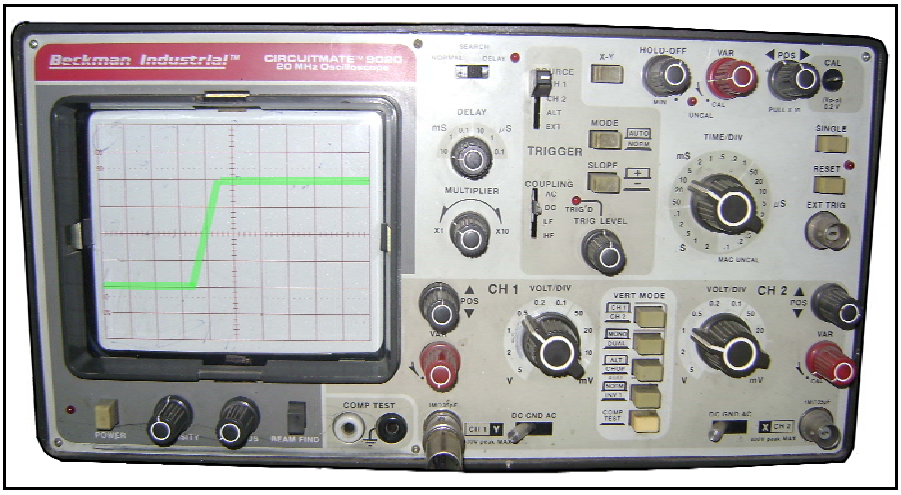


Figura 79: Señal del cinturón de seguridad en el osciloscopio. (Foto autores)

Ahora se muestra la misma variación vista por el usuario en el ordenador:

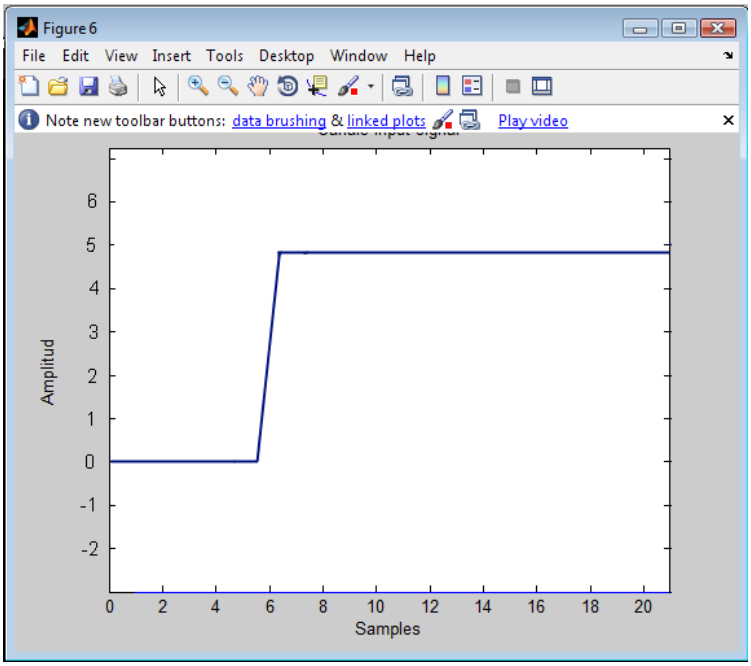


Figura 80: Señal del cinturón de seguridad del vehículo, vista en el programa MATLAB

Al ver en la pantalla del computador el resultado de la adquisición de datos de la señal del sensor Tps, es fácil comparar con la del osciloscopio y verificar la credibilidad del proyecto en cuestión de dar el mismo resultado.

3.6 Conclusiones

- Para la realización del software es indispensable un previo conocimiento del hardware que se ocupa en cuanto a lo que tiene que ver a sus alcances y limitaciones para evitar un desempeño ralentizado del mismo.
- El colocar el circuito electrónico de la memoria de forma externa a la caja negra facilita su mantenimiento.
- El detallar cada pin y su función permite un entendimiento más global del microcontrolador y su funcionamiento.
- La conversión de una señal continua a un formato digital involucra dos procesos fundamentales: muestreo y cuantificación de la señal.
- MPLAB es un software gratuito recomendado por Microchip que brinda grandes beneficios al momento de programar por su compatibilidad con el PIC 18F4550.
- Cuando se programa en MPLAB es preciso tener un diagrama de flujos que sirva de guía por la cantidad de líneas de programación que puede llegar a tener en cada rutina y subrutina de este.
- Para evitar que en el prototipo se siga grabando sin control, se añadió un seguro, por lo que para grabar es necesario mantener siempre aplastado el *switch* 3, caso contrario el dispositivo se mantendrá en modo DEVICE.
- Las memorias son grandes matrices internamente, por lo que para visualizar su contenido en el computador es necesario ingresar una dirección específica de donde se grabo.
- MATLAB permite interactuar mediante USB entre el computador y la memoria de datos, haciendo más fácil visualizar las gráficas.
- Se vio la necesidad de agregar opciones en el programa hecho en MATLAB, con la finalidad de poder utilizar la memoria para ingresar otros datos extras.
- Al momento de la adquisición se debe tener cuidado de no dejar entradas fluctuantes, para que no haya interferencias entre estos.
- El sensor TPS posee 6 pines pero solo se ocupará el de señal que se conectará en el dispositivo de almacenamiento.

CONCLUSIONES Y RECOMENDACIONES

La parte central del proyecto es el microcontrolador que debe ser seleccionado con mucha cautela para que sus características estén de acorde a las necesidades, en este punto debemos tener en cuenta el tipo de comunicación, el lenguaje de programación y el protocolo que utilizaremos, también incumbe el cálculo de la velocidad necesaria considerando que se ocupa en varios sensores de forma simultánea. El armado del circuito necesario para el funcionamiento viene recomendado por la empresa fabricante, sin embargo, se hacen varias modificaciones para aumentar o disminuir el desempeño de ciertas características del mismo. Esto se puede lograr, modificando el hardware en los capacitores, resistencias, oscilador, etc. El microcontrolador posee la conversión analógica/digital de forma interna y para esto es necesaria una programación simple que direccione esta función, recordando que por defecto esta será a 5v, en caso de necesitar convertir un voltaje diferente habrá que dar una señal de referencia, caso contrario el microcontrolador podría quemarse.

La memoria del dispositivo debe tener mayor protección por ser la parte más importante del sistema. Por esta razón como una forma extra de protección se diseñó su circuito independiente al de la tarjeta madre, para facilitar su mantenimiento y extracción. El trabajar con SD o micro SD permite una gran versatilidad de manipulación, dándonos la posibilidad de extraer la memoria y colocarla en un computador, o de interactuar con todo el sistema mediante conexión USB.

Los sensores que se han ido recomendando en el transcurso del tema fueron seleccionados de acuerdo a rangos de medición que den valores más exactos de lo ocurrido en el accidente. Para esto se evitó al máximo de no modificar el diseño de los ya en funcionamiento en el vehículo Volkswagen que estamos ocupando. Acotando que se incluyó potenciómetros en nuestro proyecto para sustituir sensores del vehículo por la facilidad de construcción y por su similar funcionamiento, pero se debe tener en cuenta que para medir datos reales en la práctica se necesitan señales de sensores con mayor complejidad y exactitud, no se deben modificar los originales o sustituir por los mencionados en el tema. Los recubrimientos externos de la caja negra deberán ser adecuados para evitar la pérdida de la grabación, teniendo en

cuenta que tanto un calor extremo como un impacto podría dañar o deformar sus componentes y comprometer la información recopilada.

Para evitar los molestos ruidos eléctricos que pueden variar el resultado de los datos obtenidos se recomienda no dejar puertos fluctuantes y realizar la respectiva impresión de las tarjetas. Además revisar la correcta conexión en cuanto al *pinout* y líneas de continuidad antes de alimentar con voltaje, con esto se evita daños en elementos tanto de la tarjeta como del vehículo.

En la programación del sistema es necesario seguir las recomendaciones del fabricante y elegir una ruta de programación antes de direccionar el *pinout* interno, también se debe recordar que al ir avanzando en las líneas de programación se debe ir colocando guías que nos ayuden a recordar que función cumple determinada rutina o subrutina.

El trabajar con MPLAB nos beneficia en la compatibilidad con el microcontrolador, siendo este un producto de microchip nos permite además reprogramar el microcontrolador cuantas veces sea necesario, dando mucha comodidad en el momento de hacer pruebas de funcionamiento. Se implementó MATLAB por la facilidad de representar gráficamente lo que está sucediendo en cada sensor y por la opción de utilizar la comunicación USB que proporciona la obtención de datos sin desmontar la caja negra, haciendo sencilla la forma de recuperar los fundamentos grabados.

Si se desea hacer pruebas con el prototipo se recomienda tener cuidado con el amperaje del vehículo que podría llegar a quemar componentes de la tarjeta, si no está debidamente protegida, también se recomienda conectar un sensor a la vez, fijándose en la ubicación del pin que se ha venido dando para evitar confusiones al momento de interpretar las señales, recordar además que tipo es, ya que varia de digital a analógica y también se modifica la magnitud.

BIBLIOGRAFIA

REFERENCIAS BIBLIOGRÁFICAS

- ANGULO, José María. 2003. Microcontroladores PIC: diseño práctico de aplicaciones. Tercera Edición. McGraw Hill. España.
- CAMPOS, Guillermo. 2008. Sistemas de diagnóstico del ABS. Curso de Graduación. Cuenca, Ecuador.
- CARDENAS, Javier. [200-]. Técnico en mecánica y electrónica automotriz, Tomo 2. Codesis Editorial. Colombia.
- CASCAJOSA, Manuel. 2005. Ingeniería de vehículos. Sistemas y cálculos. Segunda Edición. Alfaomega Grupo Editor. México.
- EDIBOSCO. 2002. Curso de Electrónica II: Componentes y circuitos básicos de la micro electrónica. Editorial Edibosco. Ecuador.
- FLORES, Marcelo. 2009. Curso de microcontroladores CEE. Cuenca, Ecuador.
- FONT, José. 2001. Tratado sobre automóviles. Alfaomega grupo editor. España.
- JURGEN, Heinz. 2005. Sistemas para la estabilidad del vehículo. Primera Edición. Alemania.
- GIL, Hermógenes. 2002. La electrónica en el automóvil. Grupo editorial CEAC. España.
- LEHMANN, Stefan. 2008. Microcontroladores PIC: prácticas de programación. Primera edición. Ediciones Marcombo. España.
- MANDADO, Enrique. 2006. Autómatas programables: entorno y aplicaciones. Primera edición. Editores Paraninfo. España.
- MILLAN, A. 2004. Determinación de los parámetros mecánicos de una máquina de inducción mediante mediciones en una máquina de corriente continua acoplada a su eje. Venezuela.

- PALLAS, Ramón. 2007. Sensores y acondicionadores de señal. 4ta Edición. Editores Marcombo. España.
- PESANTEZ, Henry. 2008. Inyección de gasolina. Cuenca, Ecuador.
- RUEDA, Jesús. 2006. Manual técnico de fuel inyección. Tercera Edición. Diseli Editores. Ecuador.
- TENESACA, Paúl. 2007. Diseño e implementación de un sistema de diagnóstico para ESP con microcontrolador. Cuenca, Ecuador.

REFERENCIAS ELECTRÓNICAS

- ALEGSA. 2009. Definición de security digital [en línea]. Argentina. ‹http://www. alegsa.com.ar/Dic/secure%20digital.php›.
- ALVES, David. 2008. Microcontroladores [en línea]. España. ‹http://www. monografias.com/trabajos12microco/microco.shtml#MERCA›.
- BLACK, Jason. 2009. Optoacopladores [en línea]. USA. <http://www.scribd. com/doc/5516426/Optoacopladores>.
- CAMPOS, Jorge. 2008. Caja negra [en línea]. México. ‹http://www.autohoy. net/accesorios-para-autos/para-que-sirve-la-caja-negra-en-los-autos.html›.
- ELECTRÓNICA-BÁSICA.COM. 2008. Caja negra después de un accidente [en línea]. México. <http://www.electronica-basica.com/caja-negra-de-avion.html>.
- GARCIA, Javier. RODRÍGUEZ, José. VIDAL, Jesús. 2005. Aprenda Matlab 7.0 [en línea]. España. ‹http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab 70/matlab70primero.pdf›.
- GOMEZ, Eddy. 2008. Características de las memorias [en línea]. México. ‹http://www.mitecnológico.com/Main/CaracteristicasMemorias›.
- GOMEZ, José. 2006. Fotónica aplicada a la informática [en línea]. USA. ‹http://www.personales.upv.es/jogomez/fai/tema08.html›.

- GONZÁLEZ, Guillermo. 2003. Proyecto Instrumento de Medida, parte 2 [en línea]. España. <http://www.arantxa.ii.uam.es/~gdrivera/labetcii/curso0203/pract2.htm>.
- HERNANDEZ, Leonardo. 2009. Tarjeta micro secure digital (SDHC) HP [en línea]. Venezuela. <https://h10025.www1.hp.com/ewfrf/wc/document?Docname=c01664451&tmp_task=prodinfoCategory&lc=es&dlc=es&cc=bo&product=39 34937&lang=es>.
- HERRADON, Rafael. 2006. Curso de introducción a los sistemas de radio comunicaciones móviles [en línea]. España. ‹http://www.Ocw.upm.es/teoria-de-la-senal-y-comunicaciones-1/radiocomunicación/contenidos/presentaciones/radiocomunicación07.pdf›.
- JAMAICA, Arturo. 2009. Pulsante del cinturón de seguridad [en línea]. Argentina. <http://walhez.com/2008/04/campana-usatu-cinturon-de-seguridad>.
- MARQUEZ, Danny. 2008. Sensores [en línea]. España. ‹http://www.mecanicavirtual.org/sensores3.htm›.
- MICROCHIP. 2006. Microchip PIC18f4550 [en línea]. USA. ‹http://www.microchip.com/downloads/en/DeviceDoc/80278a.pdf›.
- MONOGRAFIAS.COM. 2009. Tipos de memorias [en línea]. México. ‹http://www.monografias.com/trabajos12/microco/microco.shtml#MERCA›.
- SANDISK. 2003. SanDisk security card [en línea]. USA. ‹http://www.alumni.cs.ucr.edu/~amitra/sdcard/ProdManualSDCARD v1.9.pdf›.
- TELLEZ, Carlos. 2008. Optoelectrónica [en línea]. Argentina. <http://www.webelectronica.com.ar/news10/nota014/optoelectronica2.htm>.
- TOBOSO, Emilio. 2010. MPLAB – IDE v8.43 [en línea]. España. ‹http://www.perso.wanadoo.es/pictob/mplab.htm#mplab_ide_v8.43›.
- VOLKSWAGEN. 2009. Acelerador electrónico Volkswagen [en línea]. Ecuador. <http://www.volkswagen.com.ec/vwcms/master_public/virtualmaster/es_ec/comunidad_vw/innovation/0/sicherheitspedale.index.html>.

ANEXO 1

```
/********************************************************************
 *            Microchip USB C18 Firmware Versión 1.2
* File Name:          main.c
 * Dependencies:      See INCLUDES section below
 * Processor:         PIC18
 * Compiler:          C18 3.11+
 * Company:           Microchip Technology, Inc.
* Software License Agreement
* Author              Date                Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Rawin Rojvanit      11/19/04    Original.
 * Rawin Rojvanit      08/14/07    A few updates; added #if defined
********************************************************************/
/** I N C L U D E S **********************************************/
#include <p18cxxx.h>
#include "system\typedefs.h"                // Required
#include "system\usb\usb.h"                 // Required
#include "io_cfg.h"                         // Required
#include "system\usb\usb_compile_time_validation.h" // Optional
#include "user\user.h"                      // Modifiable
#include "user\mmc.h"
#include "delays.h"

/** C O N F I G U R A T I O N ************************************/
#if defined(PIC18F4550_PICDEM_FS_USB) // Configuration bits for PICDEM FS
USB Demo Board
#pragma config PLLDIV   = 5         // (20 MHz crystal on PICDEM FS USB board)
#pragma config CPUDIV   = OSC1_PLL2
#pragma config USBDIV   = 2         // Clock source from 96MHz PLL/2
#pragma config FOSC     = HSPLL_HS
#pragma config FCMEN    = OFF
#pragma config IESO     = OFF
#pragma config PWRT     = OFF
#pragma config BOR      = ON
```

```
#pragma config BORV     = 3
 #pragma config VREGEN   = ON    //USB Voltage Regulator
#pragma config WDT      = OFF
#pragma config WDTPS    = 32768
#pragma config MCLRE    = ON
#pragma config LPT1OSC  = OFF
#pragma config PBADEN   = OFF    // #pragma config CCP2MX   = ON
#pragma config STVREN   = ON
#pragma config LVP      = OFF         // #pragma config ICPRT    = OFF
#pragma config XINST    = OFF        // Extended Instruction Set
#pragma config CP0      = OFF
#pragma config CP1      = OFF         // #pragma config CP2      = OFF
                                      // #pragma config CP3      = OFF
#pragma config CPB      = OFF        // #pragma config CPD      = OFF
#pragma config WRT0     = OFF
#pragma config WRT1     = OFF      // #pragma config WRT2     = OFF
                                      // #pragma config WRT3     = OFF
#pragma config WRTB     = ON       // Boot Block Write Protection
#pragma config WRTC     = OFF      // #pragma config WRTD     = OFF
#pragma config EBTR0    = OFF
#pragma config EBTR1    = OFF      // #pragma config EBTR2    = OFF
                                      // #pragma config EBTR3    = OFF
#pragma config EBTRB    = OFF
#elif defined(PIC18F87J50_FS_USB_PIM)
      #pragma config XINST    = OFF         // Extended instruction set
      #pragma config STVREN   = ON        // Stack overflow reset
      #pragma config PLLDIV   = 3           // (12 MHz crystal used on this board)
      #pragma config WDTEN    = OFF               // Watch Dog Timer (WDT)
      #pragma config CP0      = OFF               // Code protect
      #pragma config CPUDIV   = OSC1              // OSC1 = divide by 1 mode
      #pragma config IESO     = OFF          // Internal External (clock) Switchover
      #pragma config FCMEN    = OFF               // Fail Safe Clock Monitor
      #pragma config FOSC     = HSPLL
      #pragma config WDTPS    = 32768     // #pragma config WAIT     = OFF
                                          // Commented choices are
```

```
                                        // #pragma config BW       = 16
                                        // only available on the
                                        // #pragma config MODE     = MM
                                        // 80 pin devices in the
                                        // #pragma config EASHFT   = OFF
                                    // #pragma config PMPMX    = DEFAULT
                                    //  #pragma config ECCPMX   = DEFAULT
#pragma config CCP2MX   = DEFAULT
//#elif defined(YOUR_BOARD)
//#pragma config ...
#else
#error Not a supported board (yet), make sure the proper board is selected in
usbcfg.h, and if so, set configuration bits in __FILE__, line __LINE__
#endif

/** V A R I A B L E S ********************************************/
#pragma udata

/** P R I V A T E  P R O T O T Y P E S ***************************/
static void InitializeSystem(void);
void USBTasks(void);

/** V E C T O R  R E M A P P I N G *******************************/
extern void _startup (void);        // See c018i.c in your C18 compiler dir
#pragma code _RESET_INTERRUPT_VECTOR = 0x000800
void _reset (void)
{
   _asm goto _startup _endasm
}
#pragma code
/** D E C L A R A T I O N S **************************************/
#pragma code

/******************MAIN*******************************************/
void main(void)
{
```

```
  InitializeSystem();
  if(PORTBbits.RB3 == 0){ //ACQ or DEVICE
       CONFIG_MMC();     //Setup MMC card for read/write operation
  }
  while(1)
  {
        if(PORTBbits.RB3 == 0){
              ACQ2MMC();                    //ACQ
       }else{
             USBTasks();        // USB Tasks
       ProcessIO();       // See user\user.c & .h
       }
  }//end while
}//end main
static void InitializeSystem(void)
{
//On the PIC18F87J50 Family of USB microcontrollers, the PLL will not power up and be enabled
//by default, even if a PLL enabled oscillator configuration is selected (such as HS+PLL).
//This allows the device to power up at a lower initial operating frequency, which can be
//advantageous when powered from a source which is not gauranteed to be adequate for 48MHz
//operation.  On these devices, user firmware needs to manually set the OSCTUNE<PLLEN> bit to
//power up the PLL.
       #if defined(__18F87J50)||defined(__18F86J55)|| \
       defined(__18F86J50)||defined(__18F85J50)|| \
       defined(__18F67J50)||defined(__18F66J55)|| \
       defined(__18F66J50)||defined(__18F65J50)
unsigned int pll_startup_counter = 600;
OSCTUNEbits.PLLEN = 1;  //Enable the PLL and wait 2+ms until the PLL locks before enabling USB module while(pll_startup_counter--);
WDTCONbits.ADSHR = 1;
ANCON0 = 0xFF;                  // Default all pins to digital
ANCON1 = 0xFF;                  // Default all pins to digital
```

```
WDTCONbits.ADSHR = 0;       // Select normal SFR locations
 #elif defined(PIC18F4550_PICDEM_FS_USB)
ADCON1 |= 0x0F;                 // Default all pins to digital
#else
#error Double Click this message.  Please make sure the InitializeSystem() function
correctly configures your hardware platform.
#endif
   #if defined(USE_USB_BUS_SENSE_IO)
   tris_usb_bus_sense = INPUT_PIN;   // See io_cfg.h
   #endif
   #if defined(USE_SELF_POWER_SENSE_IO)
tris_self_power = INPUT_PIN;
 #endif
mInitializeUSBDriver();                 // See usbdrv.h
UserInit();                             // See user.c & .h
}//end InitializeSystem
void USBTasks(void)
{
   USBCheckBusStatus();             // Must use polling method
   if(UCFGbits.UTEYE!=1)
      USBDriverService();           // Interrupt or polling method
   #if defined(USB_USE_CDC)
   CDCTxService();
   #endif
}                                    // end USBTasks

/** EOF main.c *************************************************/
```

ANEXO 2

```
/********************************************************************
*               Microchip typedefs.h
* FileName:             typedefs.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       7/21/04     Original.
* Rawin Rojvanit       08/14/07    A few updates; added #if defined
********************************************************************/
  struct
  {
    WORD Word0;
    WORD Word1;
  };
  struct
  {
    byte v[4];
  };
} DWORD;
#define LOWER_LSB(a)    ((a).v[0])
#define LOWER_MSB(a)    ((a).v[1])
#define UPPER_LSB(a)    ((a).v[2])
#define UPPER_MSB(a)    ((a).v[3])
typedef void(*pFunc)(void);
typedef union _POINTER
{
  struct
  {
    byte bLow;
    byte bHigh;
    //byte bUpper;
  };
  word _word;                   // bLow & bHigh
```

```
                                //pFunc _pFunc;
                                // Usage: ptr.pFunc(); Init: ptr.pFunc = &<Function>;
    byte* bRam;                 // Ram byte pointer: 2 bytes pointer pointing
                                // to 1 byte of data
    word* wRam;                 // Ram word poitner: 2 bytes poitner pointing
                                // to 2 bytes of data
    rom byte* bRom;              // Size depends on compiler setting
    rom word* wRom;
    //rom near byte* nbRom;     // Near = 2 bytes pointer
    //rom near word* nwRom;
    //rom far byte* fbRom;      // Far = 3 bytes pointer
    //rom far word* fwRom;
} POINTER;
typedef enum _BOOL { FALSE = 0, TRUE } BOOL;
#define OK      TRUE
#define FAIL    FALSE
#endif //TYPEDEFS_H
```

ANEXO 3

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.0
* FileName:         usb.h
* Software License Agreement
* Author                Date                Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
********************************************************************/
#ifndef USB_H
#define USB_H
#include "autofiles\usbcfg.h"
#include "system\usb\usbdefs\usbdefs_std_dsc.h"
#include "autofiles\usbdsc.h"
#include "system\usb\usbdefs\usbdefs_ep0_buff.h"
#include "system\usb\usbmmap.h"
#include "system\usb\usbdrv\usbdrv.h"
#include "system\usb\usbctrltrf\usbctrltrf.h"
#include "system\usb\usb9\usb9.h"
#if defined(USB_USE_CDC)
#include "system\usb\class\cdc\cdc.h"
#endif
#endif //USB_H
```

ANEXO 4

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.2
* FileName:             io_cfg.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
 * Rawin Rojvanit      05/14/07    Added pin mapping for PIC18F87J50
********************************************************************/
#ifndef IO_CFG_H
#define IO_CFG_H

/** I N C L U D E S **************************************************/
#include "autofiles\usbcfg.h"

/** T R I S **********************************************************/
#define INPUT_PIN           1
#define OUTPUT_PIN          0
#if defined(PIC18F4550_PICDEM_FS_USB)

/** U S B ************************************************************/
#define tris_usb_bus_sense  TRISAbits.TRISA1    // Input
#if defined(USE_USB_BUS_SENSE_IO)
#define usb_bus_sense       PORTAbits.RA1
#else
#define usb_bus_sense       1
#endif
#define tris_self_power     TRISAbits.TRISA2          // Input
#if defined(USE_SELF_POWER_SENSE_IO)
#define self_power          PORTAbits.RA2
#else
#define self_power          1             // Used by USBStdGetStatusHandler() in usb9.c
#endif                                    // External Transceiver Interface
```

```
#define tris_usb_vpo       TRISBbits.TRISB3        // Output
#define tris_usb_vmo       TRISBbits.TRISB2        // Output
#define tris_usb_rcv       TRISAbits.TRISA4    // Input
#define tris_usb_vp        TRISCbits.TRISC5    // Input
#define tris_usb_vm        TRISCbits.TRISC4    // Input
#define tris_usb_oe        TRISCbits.TRISC1    // Output
#define tris_usb_suspnd    TRISAbits.TRISA3    // Output

/** L E D ********************************************************/
#define mInitAllLEDs()     LATD &= 0xF0; TRISD &= 0xF0;
#define mLED_1             LATDbits.LATD0
#define mLED_2             LATDbits.LATD1
#define mLED_3             LATDbits.LATD2
#define mLED_4             LATDbits.LATD3
#define mLED_1_On()        mLED_1 = 1;
#define mLED_2_On()        mLED_2 = 1;
#define mLED_3_On()        mLED_3 = 1;
#define mLED_4_On()        mLED_4 = 1;
#define mLED_1_Off()       mLED_1 = 0;
#define mLED_2_Off()       mLED_2 = 0;
#define mLED_3_Off()       mLED_3 = 0;
#define mLED_4_Off()       mLED_4 = 0;
#define mLED_1_Toggle()    mLED_1 = !mLED_1;
#define mLED_2_Toggle()    mLED_2 = !mLED_2;
#define mLED_3_Toggle()    mLED_3 = !mLED_3;
#define mLED_4_Toggle()    mLED_4 = !mLED_4;

/** S W I T C H **************************************************/
#define mInitAllSwitches()  TRISBbits.TRISB4=1;TRISBbits.TRISB5=1;
#define mInitSwitch2()     TRISBbits.TRISB4=1;
#define mInitSwitch3()     TRISBbits.TRISB5=1;
#define sw2                PORTBbits.RB4
#define sw3                PORTBbits.RB5

/** P O T ********************************************************/
```

```
#define mInitPOT()     TRISAbits.TRISA0=1;ADCON0=0x01;ADCON2=0x3C;

/** S P I : Chip Select Lines ***********************************************/
#define tris_cs_temp_sensor TRISBbits.TRISB2    // Output
#define cs_temp_sensor      LATBbits.LATB2
#define tris_cs_sdmmc       TRISBbits.TRISB3      // Output
#define cs_sdmmc            LATBbits.LATB3

/** S D M M C ***************************************************************/
#define TRIS_CARD_DETECT    TRISBbits.TRISB4    // Input
#define CARD_DETECT         PORTBbits.RB4
#define TRIS_WRITE_DETECT   TRISAbits.TRISA4    // Input
#define WRITE_DETECT        PORTAbits.RA4
/***************************************************************************/
#elif defined(PIC18F87J50_FS_USB_PIM)

/** U S B *******************************************************************/
// Bus sense pin is RB5 on PIC18F87J50 FS USB Plug-In Module.
// Must put jumper JP1 in R-U position to use bus sense feature.
#define tris_usb_bus_sense  TRISBbits.TRISB5    // Input
#if defined(USE_USB_BUS_SENSE_IO)
#define usb_bus_sense       PORTBbits.RB5
#else
#define usb_bus_sense       1
#endif
#define self_power          0

/** L E D *******************************************************************/
#define mInitAllLEDs()      LATE &= 0xFC; TRISE &= 0xFC;
#define mLED_1              LATEbits.LATE1
#define mLED_2              LATEbits.LATE0
#define mLED_1_On()         mLED_1 = 1;
#define mLED_2_On()         mLED_2 = 1;
#define mLED_1_Off()        mLED_1 = 0;
#define mLED_2_Off()        mLED_2 = 0;
```

```
#define mLED_1_Toggle()     mLED_1 = !mLED_1;
#define mLED_2_Toggle()     mLED_2 = !mLED_2;

/** S W I T C H ************************************************/
#define mInitAllSwitches()  TRISBbits.TRISB4=1;
#define mInitSwitch2()      TRISBbits.TRISB4=1;
#define sw2                 PORTBbits.RB4

/******************************************************************/
//Uncomment below if using the YOUR_BOARD hardware platform
//#elif defined(YOUR_BOARD)
//Add your hardware specific I/O pin mapping here
#else
   #error Not a supported board (yet), add I/O pin mapping in __FILE__, line
__LINE__
#endif
#endif //IO_CFG_H
```

ANEXO 5

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.0
* FileName:             usb_compile_time_validation.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       7/10/04     Original.
********************************************************************/
#ifndef USB_COMPILE_TIME_VALIDATION_H
#define USB_COMPILE_TIME_VALIDATION_H

/** I N C L U D E S **************************************************/
#include "system\typedefs.h"
#include "system\usb\usb.h"

/** U S B  V A L I D A T I O N **************************************/
#if (EP0_BUFF_SIZE != 8) && (EP0_BUFF_SIZE != 16) && \\
   (EP0_BUFF_SIZE != 32) && (EP0_BUFF_SIZE != 64)
#error(Invalid buffer size for endpoint 0,check "autofiles\usbcfg.h")
#endif
#if defined(HID_INT_OUT_EP_SIZE)
   #if (HID_INT_OUT_EP_SIZE > 64)
      #error(HID Out endpoint size cannot be bigger than 64, check
"autofiles\usbcfg.h")
   #endif
#endif
#ifdef HID_INT_IN_EP_SIZE
   #if (HID_INT_IN_EP_SIZE > 64)
      #error(HID In endpoint size cannot be bigger than 64, check
"autofiles\usbcfg.h")
   #endif
#endif
#endif //USB_COMPILE_TIME_VALIDATION_H
```

ANEXO 6

```
/********************************************************************
 *              Microchip USB C18 Firmware Versión 1.0
* FileName:            user.h
* Software License Agreement
* Author                Date                  Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
********************************************************************/
#ifndef USER_H
#define USER_H

/** P U B L I C  P R O T O T Y P E S ********************************/
void UserInit(void);
void ProcessIO(void);
void ACQ2MMC(void);
void CONFIG_MMC(void);
void Init_TMR0(void);
void Init_ADC(void);
void ADC_SCH(unsigned char ch);
#endif //USER_H
```

ANEXO 7

```
/*****************************************************************
///          _____________________
///         /  1 2 3 4 5 6 78     |     <- view of MMC/SD card looking at contacts
///        / 9                    |     Pins 8 and 9 are present only on SD cards
///        |   MMC/SD Card    |
///        |                        |
///            /\/\/\/\/\/\/\/\
*****************************************************************/
#ifndef mmc_H
#define mmc_H

/** D E F I N I T I O N S ***********************************************/
//CONFIGURATION PINOUT SPI / MMC card module
#define SCK       PORTBbits.RB1
#define SDI       PORTBbits.RB0
#define SDO       PORTCbits.RC7
#define SD_CS     PORTBbits.RB2
#define SDI_DIR TRISBbits.TRISB0
#define SCK_DIR TRISBbits.TRISB1
#define SDO_DIR TRISCbits.TRISC7
#define SD_CS_DIR TRISBbits.TRISB2
#define clockSPI() writeSPI(0xFF)             //CHIP SELECT
#define SD_Enable() SD_CS = 0 /* set low to activate SD Card chip select */
#define SD_Disable() SD_CS = 1 /* set high to deactivate SD Card chip select */
//LED DEFINITIONS & STATUS
#define CMD0 0       // Resetea la SD Card.-
#define CMD1 1       // Activa proceso de inicializaci?n de SD Card.
#define CMD9 9       // Lectura registro CSD.-
#define CMD10 10     // Lectura registro CID.-
#define CMD16 16     // Selecciona largo del bloque para lectura/escritura (1 a 512)
#define CMD17 17     // Lectura de un ?nico bloque.
#define CMD24 24     // Escritura de un ?nico bloque.
#define CMD59 59     // Enciende/Apaga CRC.
```

```
#define CMD32 32     // Setea direccion  del primer bloque a borrar.-
#define CDM33 33     // Setea direccion del ultimo bloque en un continuo rango a borrar.-
#define CMD38 38     // Borra todos los sectores entre las direcci?nes establecidas.-
#define BLOCK_SIZE 512 // Tama?o del bloque de lectura/escritura.

/** S T R U C T U R E S ********************************************/
/*typedef unsigned long        LBA; //logic block address
typedef unsigned char   byte;          // 8-bit
typedef unsigned int    word;          // 16-bit
typedef unsigned long   dword;         // 32-bit

/** P R O T O T Y P E S ********************************************/
extern void Init_Ports(void);
extern unsigned char BUFFER[512];
extern unsigned char ResVec[2];
extern int BLOCK;
extern unsigned char MMC_Init(void);
extern unsigned char MMC_Read_Block(unsigned long addr);
extern unsigned char MMC_Read_Block2(unsigned char a, unsigned char b, unsigned char c, unsigned char d);
extern unsigned char MMC_Write_Block(unsigned long addr);
extern unsigned char MMC_Write_Block2(unsigned char a, unsigned char b, unsigned char c, unsigned char d);
extern unsigned char MMC_Read_CID(unsigned char *buffer);
extern unsigned char MMC_Send_Cmd2(unsigned char cmd, unsigned char a, unsigned char b, unsigned char c, unsigned char d);
#endif
```

ANEXO 8

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.2
* FileName:             usbcfg.h
* Software License Agreement
********************************************************************/
#ifndef USBCFG_H
#define USBCFG_H

/** D E F I N I T I O N S ****************************************/
#define MAX_NUM_INT         1   // For tracking Alternate Setting
#define EP0_BUFF_SIZE       8   // Valid Options: 8, 16, 32, or 64 bytes.
                                // There is very little advantage in using
                                // more than 8 bytes on EP0 IN/OUT, so 8 is the
                                // recommended value.
/* Parameter definitions are defined in usbdrv.h */
#define MODE_PP             _PPBM0
#define UCFG_VAL            _PUEN|_TRINT|_FS|MODE_PP
/* Uncomment only the hardware platform that you are using*/
#define PIC18F4550_PICDEM_FS_USB
//#define PIC18F87J50_FS_USB_PIM
//#define YOUR_BOARD
#if defined(PIC18F4550_PICDEM_FS_USB)
//#define USE_SELF_POWER_SENSE_IO
//#define USE_USB_BUS_SENSE_IO
#elif defined(PIC18F87J50_FS_USB_PIM)   //#define USE_USB_BUS_SENSE_IO
                //JP1 must be in R-U position to use this feature on this board
/*If using the YOUR_BOARD selection, uncomment below section as appropriate
for your hardware*/
//#elif defined(YOUR_BOARD)
//#define USE_SELF_POWER_SENSE_IO
//See main.c and MCHPFSUSB Firmware User's Guide
//#define USE_USB_BUS_SENSE_IO
//(DS51679) for more details about these features.
#else
```

```
 #error Not a supported board (yet), See __FILE__, line __LINE__, or double click on this text.
//See above commented section.  You need to select the features your hardware will be using.
#endif

/** D E V I C E  C L A S S  U S A G E ******************************/
#define USB_USE_CDC  *  MUID = Microchip USB Class ID  * Used to identify which of the USB classes owns the current  * session of control transfer over EP0  */
#define MUID_NULL            0
#define MUID_USB9            1
#define MUID_HID             2
#define MUID_CDC             3
#define MUID_MSD             4

/** E N D P O I N T S  A L L O C A T I O N ************************/
/*See usbmmap.c for an explanation of how the endpoint allocation works*/
/* CDC */
#define CDC_COMM_INTF_ID       0x00
#define CDC_COMM_UEP           UEP2
#define CDC_INT_BD_IN          ep2Bi
#define CDC_INT_EP_SIZE        8
#define CDC_DATA_INTF_ID       0x01
#define CDC_DATA_UEP           UEP3
#define CDC_BULK_BD_OUT        ep3Bo
#define CDC_BULK_OUT_EP_SIZE   64
#define CDC_BULK_BD_IN         ep3Bi
#define CDC_BULK_IN_EP_SIZE    64
#define MAX_EP_NUMBER          3          // UEP3
#endif //USBCFG_H
```

ANEXO 9

```
/********************************************************************
*               Microchip USB C18 Firmware Version 1.0
* FileName:             usbdefs_std_dsc.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
********************************************************************/
/********* * USB Definitions: Standard Descriptors**********************/
#ifndef USBDEFS_STD_DSC_H
#define USBDEFS_STD_DSC_H
/** I N C L U D E S **************************************************/
#include "system\typedefs.h"

/** D E F I N I T I O N S ********************************************/
/* Descriptor Types */
#define DSC_DEV     0x01
#define DSC_CFG     0x02
#define DSC_STR     0x03
#define DSC_INTF    0x04
#define DSC_EP      0x05
/*********************************************************************
#define _EP01_OUT   0x01
#define _EP01_IN    0x81
#define _EP02_OUT   0x02
#define _EP02_IN    0x82
#define _EP03_OUT   0x03
#define _EP03_IN    0x83
#define _EP04_OUT   0x04
#define _EP04_IN    0x84
#define _EP05_OUT   0x05
#define _EP05_IN    0x85
#define _EP06_OUT   0x06
#define _EP06_IN    0x86
```

```
#define _EP07_OUT   0x07
#define _EP07_IN    0x87
#define _EP08_OUT   0x08
#define _EP08_IN    0x88
#define _EP09_OUT   0x09
#define _EP09_IN    0x89
#define _EP10_OUT   0x0A
#define _EP10_IN    0x8A
#define _EP11_OUT   0x0B
#define _EP11_IN    0x8B
#define _EP12_OUT   0x0C
#define _EP12_IN    0x8C
#define _EP13_OUT   0x0D
#define _EP13_IN    0x8D
#define _EP14_OUT   0x0E
#define _EP14_IN    0x8E
#define _EP15_OUT   0x0F
#define _EP15_IN    0x8F
/* Configuration Attributes */
#define _DEFAULT    0x01<<7        //Default Value (Bit 7 is set)
#define _SELF       0x01<<6        //Self-powered (Supports if set)
#define _RWU        0x01<<5        //Remote Wakeup (Supports if set)
/* Endpoint Transfer Type */
#define _CTRL       0x00           //Control Transfer
#define _ISO        0x01           //Isochronous Transfer
#define _BULK       0x02           //Bulk Transfer
#define _INT        0x03           //Interrupt Transfer
/* Isochronous Endpoint Synchronization Type */
#define _NS         0x00<<2        //No Synchronization
#define _AS         0x01<<2        //Asynchronous
#define _AD         0x02<<2        //Adaptive
#define _SY         0x03<<2        //Synchronous
/* Isochronous Endpoint Usage Type */
#define _DE         0x00<<4        //Data endpoint
#define _FE         0x01<<4        //Feedback endpoint
```

```
#define _IE         0x02<<4              //Implicit feedback Data endpoint

/********** USB Device Descriptor Structure**************************/
typedef struct _USB_DEV_DSC
{
   byte bLength;       byte bDscType;      word bcdUSB;
   byte bDevCls;       byte bDevSubCls;    byte bDevProtocol;
   byte bMaxPktSize0;  word idVendor;      word idProduct;
   word bcdDevice;     byte iMFR;          byte iProduct;
   byte iSerialNum;    byte bNumCfg;
} USB_DEV_DSC;
/*************** USB Configuration Descriptor Structure*****************/
typedef struct _USB_CFG_DSC
{
   byte bLength;       byte bDscType;      word wTotalLength;
   byte bNumIntf;      byte bCfgValue;     byte iCfg;
   byte bmAttributes;  byte bMaxPower;
} USB_CFG_DSC;
/****************** USB Interface Descriptor Structure*******************/
typedef struct _USB_INTF_DSC
{
   byte bLength;       byte bDscType;      byte bIntfNum;
   byte bAltSetting;   byte bNumEPs;       byte bIntfCls;
   byte bIntfSubCls;   byte bIntfProtocol; byte iIntf;
} USB_INTF_DSC;
/***************USB Endpoint Descriptor Structure**********************/
typedef struct _USB_EP_DSC
{
   byte bLength;       byte bDscType;      byte bEPAdr;
   byte bmAttributes;  word wMaxPktSize;   byte bInterval;
} USB_EP_DSC;
#endif //USBDEFS_STD_DSC_H
```

ANEXO 10

```
/********************************************************************
 *              Microchip USB C18 Firmware Version 1.0
* FileName:             usbdsc.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Descriptor specific type definitions are defined in:
 * system\usb\usbdefs\usbdefs_std_dsc.h
********************************************************************/
#ifndef USBDSC_H
#define USBDSC_H

/** I N C L U D E S **************************************************/
#include "system\typedefs.h"
#include "autofiles\usbcfg.h"
#if defined(USB_USE_CDC)
#include "system\usb\class\cdc\cdc.h"
#endif
#include "system\usb\usb.h"

/** D E F I N I T I O N S ********************************************/
#define CFG01 rom struct                        \
{   USB_CFG_DSC           cd01;                 \
   USB_INTF_DSC          i01a00;                \
   USB_CDC_HEADER_FN_DSC   cdc_header_fn_i01a00;  \
   USB_CDC_ACM_FN_DSC      cdc_acm_fn_i01a00;     \
   USB_CDC_UNION_FN_DSC    cdc_union_fn_i01a00;   \
   USB_CDC_CALL_MGT_FN_DSC cdc_call_mgt_fn_i01a00; \
   USB_EP_DSC            ep02i_i01a00;          \
   USB_INTF_DSC           i02a00;               \
   USB_EP_DSC            ep03o_i02a00;          \
   USB_EP_DSC            ep03i_i02a00;          \
} cfg01
```

```
/** E X T E R N S ****************************************************/
extern rom USB_DEV_DSC device_dsc;
extern CFG01;
extern rom const unsigned char *rom USB_CD_Ptr[];
extern rom const unsigned char *rom USB_SD_Ptr[];
extern rom pFunc ClassReqHandler[1];
#endif //USBDSC_H
```

ANEXO 11

```
/********************************************************************
 *             Microchip USB C18 Firmware Version 1.0
* FileName:            usbmmap.h
* Software License Agreement
* Author                Date                  Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
********************************************************************/
#ifndef USBMMAP_H
#define USBMMAP_H
/** I N C L U D E S **************************************************/
#include "system\typedefs.h"
/** D E F I N I T I O N S ********************************************/
/* Buffer Descriptor Status Register Initialization Parameters */
#define _BSTALL     0x04              //Buffer Stall enable
#define _DTSEN      0x08              //Data Toggle Synch enable
#define _INCDIS     0x10              //Address increment disable
#define _KEN        0x20              //SIE keeps buff descriptors enable
#define _DAT0       0x00              //DATA0 packet expected next
#define _DAT1       0x40              //DATA1 packet expected next
#define _DTSMASK    0x40              //DTS Mask
#define _USIE       0x80              //SIE owns buffer
#define _UCPU       0x00              //CPU owns buffer
/* USB Device States - To be used with [byte usb_device_state] */
#define DETACHED_STATE         0
#define ATTACHED_STATE         1
#define POWERED_STATE          2
#define DEFAULT_STATE          3
#define ADR_PENDING_STATE      4
#define ADDRESS_STATE          5
#define CONFIGURED_STATE       6
/* Memory Types for Control Transfer - used in USB_DEVICE_STATUS */
#define _RAM 0
```

```
#define _ROM 1
/** T Y P E S ******************************************************/
typedef union _USB_DEVICE_STATUS
{
  byte _byte;
  struct
  {
    unsigned RemoteWakeup:1;// [0]Disabled [1]Enabled: See usbdrv.c,usb9.c
    unsigned ctrl_trf_mem:1;// [0]RAM     [1]ROM
  };
} USB_DEVICE_STATUS;
typedef union _BD_STAT
{
  byte _byte;
  struct{
    unsigned BC8:1;
    unsigned BC9:1;
    unsigned BSTALL:1;          //Buffer Stall Enable
    unsigned DTSEN:1;           //Data Toggle Synch Enable
    unsigned INCDIS:1;          //Address Increment Disable
    unsigned KEN:1;             //BD Keep Enable
    unsigned DTS:1;             //Data Toggle Synch Value
    unsigned UOWN:1;            //USB Ownership
  };
  struct{
    unsigned BC8:1;
    unsigned BC9:1;
    unsigned PID0:1;
    unsigned PID1:1;
    unsigned PID2:1;
    unsigned PID3:1;
    unsigned :1;
    unsigned UOWN:1;
  };
  struct{
```

```
        unsigned :2;
        unsigned PID:4;              //Packet Identifier
        unsigned :2;
    };
} BD_STAT;                           //Buffer Descriptor Status Register
typedef union _BDT
{
    struct
    {
        BD_STAT Stat;
        byte Cnt;
        byte ADRL;                   //Buffer Address Low
        byte ADRH;                   //Buffer Address High
    };
    struct
    {
        unsigned :8;
        unsigned :8;
        byte* ADR;                   //Buffer Address
    };
} BDT;                               //Buffer Descriptor Table
/** E X T E R N S
**************************************************************/
extern byte usb_device_state;
extern USB_DEVICE_STATUS usb_stat;
extern byte usb_active_cfg;
extern byte usb_alt_intf[MAX_NUM_INT];
extern volatile far BDT ep0Bo;        //Endpoint #0 BD Out
extern volatile far BDT ep0Bi;        //Endpoint #0 BD In
extern volatile far BDT ep1Bo;        //Endpoint #1 BD Out
extern volatile far BDT ep1Bi;        //Endpoint #1 BD In
extern volatile far BDT ep2Bo;        //Endpoint #2 BD Out
extern volatile far BDT ep2Bi;        //Endpoint #2 BD In
extern volatile far BDT ep3Bo;        //Endpoint #3 BD Out
extern volatile far BDT ep3Bi;        //Endpoint #3 BD In
```

```
extern volatile far BDT ep4Bo;          //Endpoint #4 BD Out
extern volatile far BDT ep4Bi;          //Endpoint #4 BD In
extern volatile far BDT ep5Bo;          //Endpoint #5 BD Out
extern volatile far BDT ep5Bi;          //Endpoint #5 BD In
extern volatile far BDT ep6Bo;          //Endpoint #6 BD Out
extern volatile far BDT ep6Bi;          //Endpoint #6 BD In
extern volatile far BDT ep7Bo;          //Endpoint #7 BD Out
extern volatile far BDT ep7Bi;          //Endpoint #7 BD In
extern volatile far BDT ep8Bo;          //Endpoint #8 BD Out
extern volatile far BDT ep8Bi;          //Endpoint #8 BD In
extern volatile far BDT ep9Bo;          //Endpoint #9 BD Out
extern volatile far BDT ep9Bi;          //Endpoint #9 BD In
extern volatile far BDT ep10Bo;         //Endpoint #10 BD Out
extern volatile far BDT ep10Bi;         //Endpoint #10 BD In
extern volatile far BDT ep11Bo;         //Endpoint #11 BD Out
extern volatile far BDT ep11Bi;         //Endpoint #11 BD In
extern volatile far BDT ep12Bo;         //Endpoint #12 BD Out
extern volatile far BDT ep12Bi;         //Endpoint #12 BD In
extern volatile far BDT ep13Bo;         //Endpoint #13 BD Out
extern volatile far BDT ep13Bi;         //Endpoint #13 BD In
extern volatile far BDT ep14Bo;         //Endpoint #14 BD Out
extern volatile far BDT ep14Bi;         //Endpoint #14 BD In
extern volatile far BDT ep15Bo;         //Endpoint #15 BD Out
extern volatile far BDT ep15Bi;         //Endpoint #15 BD In
extern volatile far CTRL_TRF_SETUP SetupPkt;
extern volatile far CTRL_TRF_DATA CtrlTrfData;
#if defined(USB_USE_CDC)
extern volatile far unsigned char cdc_notice[CDC_INT_EP_SIZE];
extern volatile far unsigned char cdc_data_rx[CDC_BULK_OUT_EP_SIZE];
extern volatile far unsigned char cdc_data_tx[CDC_BULK_IN_EP_SIZE];
#endif                                          #endif //USBMMAP_H
```

ANEXO 12

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.2
* FileName:             usbdrv.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit        11/19/04    Original.
 * Rawin Rojvanit       05/14/07    Fixed endpoint definitions
********************************************************************/
#ifndef USBDRV_H
#define USBDRV_H

/** I N C L U D E S **************************************************/
#include "system\typedefs.h"
#include "system\usb\usb.h"

/** D E F I N I T I O N S ********************************************/
/* UCFG Initialization Parameters */
#define _PPBM0      0x00            // Pingpong Buffer Mode 0
#define _PPBM1      0x01            // Pingpong Buffer Mode 1
#define _PPBM2      0x02            // Pingpong Buffer Mode 2
#define _LS         0x00          // Use Low-Speed USB Mode
#define _FS         0x04          // Use Full-Speed USB Mode
#define _TRINT      0x00          // Use internal transceiver
#define _TREXT      0x08           // Use external transceiver
#define _PUEN       0x10          // Use internal pull-up resistor
#define _OEMON      0x40            // Use SIE output indicator
#define _UTEYE      0x80           // Use Eye-Pattern test
/* UEPn Initialization Parameters */
#define EP_CTRL     0x06              // Cfg Control pipe for this ep
#define EP_OUT      0x0C              // Cfg OUT only pipe for this ep
#define EP_IN       0x0A               // Cfg IN only pipe for this ep
#define EP_OUT_IN   0x0E              // Cfg both OUT & IN pipes for this ep
```

```
#define HSHK_EN     0x10              // Enable handshake packet
                                      // Handshake should be disable for isoch
/*****************************************************************
#define OUT         0
#define IN          1
#define PIC_EP_NUM_MASK 0b01111000
#define PIC_EP_DIR_MASK 0b00000100
#define EP00_OUT    ((0x00<<3)|(OUT<<2))
#define EP00_IN     ((0x00<<3)|(IN<<2))
#define EP01_OUT    ((0x01<<3)|(OUT<<2))
#define EP01_IN     ((0x01<<3)|(IN<<2))
#define EP02_OUT    ((0x02<<3)|(OUT<<2))
#define EP02_IN     ((0x02<<3)|(IN<<2))
#define EP03_OUT    ((0x03<<3)|(OUT<<2))
#define EP03_IN     ((0x03<<3)|(IN<<2))
#define EP04_OUT    ((0x04<<3)|(OUT<<2))
#define EP04_IN     ((0x04<<3)|(IN<<2))
#define EP05_OUT    ((0x05<<3)|(OUT<<2))
#define EP05_IN     ((0x05<<3)|(IN<<2))
#define EP06_OUT    ((0x06<<3)|(OUT<<2))
#define EP06_IN     ((0x06<<3)|(IN<<2))
#define EP07_OUT    ((0x07<<3)|(OUT<<2))
#define EP07_IN     ((0x07<<3)|(IN<<2))
#define EP08_OUT    ((0x08<<3)|(OUT<<2))
#define EP08_IN     ((0x08<<3)|(IN<<2))
#define EP09_OUT    ((0x09<<3)|(OUT<<2))
#define EP09_IN     ((0x09<<3)|(IN<<2))
#define EP10_OUT    ((0x0A<<3)|(OUT<<2))
#define EP10_IN     ((0x0A<<3)|(IN<<2))
#define EP11_OUT    ((0x0B<<3)|(OUT<<2))
#define EP11_IN     ((0x0B<<3)|(IN<<2))
#define EP12_OUT    ((0x0C<<3)|(OUT<<2))
#define EP12_IN     ((0x0C<<3)|(IN<<2))
#define EP13_OUT    ((0x0D<<3)|(OUT<<2))
#define EP13_IN     ((0x0D<<3)|(IN<<2))
```

```
#define EP14_OUT    ((0x0E<<3)|(OUT<<2))
#define EP14_IN     ((0x0E<<3)|(IN<<2))
#define EP15_OUT    ((0x0F<<3)|(OUT<<2))
#define EP15_IN     ((0x0F<<3)|(IN<<2))
#define mInitializeUSBDriver()      {UCFG = UCFG_VAL;                   \
                                     usb_device_state = DETACHED_STATE;     \
                                     usb_stat._byte = 0x00;                 \
                                     usb_active_cfg = 0x00;}
#define mDisableEP1to15()       ClearArray((byte*)&UEP1,15); /*
#define mDisableEP1to15()       UEP1=0x00;UEP2=0x00;UEP3=0x00;\
                                UEP4=0x00;UEP5=0x00;UEP6=0x00;UEP7=0x00;\
                                UEP8=0x00;UEP9=0x00;UEP10=0x00;UEP11=0x00;\
                                UEP12=0x00;UEP13=0x00;UEP14=0x00;UEP15=0x00; */
#define mUSBBufferReady(buffer_dsc)                                     \
{                                                       \
   buffer_dsc.Stat._byte &= _DTSMASK;         /* Save only DTS bit */     \
   buffer_dsc.Stat.DTS = !buffer_dsc.Stat.DTS; /* Toggle DTS bit    */     \
   buffer_dsc.Stat._byte |= _USIE|_DTSEN;      /* Turn ownership to SIE */ \
}

/** P U B L I C  P R O T O T Y P E S
********************************************/
void USBCheckBusStatus(void);
void USBDriverService(void);
void USBRemoteWakeup(void);
void USBSoftDetach(void);
void ClearArray(byte* startAdr,byte count);
#endif //USBDRV_H
```

ANEXO 13

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.2
* FileName:             usbctrltrf.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
 * Rawin Rojvanit      08/14/07    Bug fixes.
********************************************************************/
#ifndef USBCTRLTRF_H
#define USBCTRLTRF_H

/** I N C L U D E S **************************************************/
#include "system\typedefs.h"

/** D E F I N I T I O N S ********************************************/
/* Control Transfer States */
#define WAIT_SETUP          0
#define CTRL_TRF_TX         1
#define CTRL_TRF_RX         2
/*********Bug Fix: May 14, 2007 (#F7)**************/
* Short Packet States - Used by Control Transfer Read  - CTRL_TRF_TX */
#define SHORT_PKT_NOT_USED  0
#define SHORT_PKT_PENDING   1
#define SHORT_PKT_SENT      2
/* USB PID: Token Types - See chapter 8 in the USB specification */
#define SETUP_TOKEN         0b00001101
#define OUT_TOKEN           0b00000001
#define IN_TOKEN            0b00001001
/* bmRequestType Definitions */
#define HOST_TO_DEV         0
#define DEV_TO_HOST         1
#define STANDARD            0x00
#define CLASS               0x01
```

```
#define VENDOR            0x02
#define RCPT_DEV          0
#define RCPT_INTF         1
#define RCPT_EP           2
#define RCPT_OTH          3

/** E X T E R N S
***************************************************************/
extern byte ctrl_trf_session_owner;
extern POINTER pSrc;
extern POINTER pDst;
extern WORD wCount;

/** P U B L I C  P R O T O T Y P E S
***************************************/
byte USBCtrlEPService(void);              // Bug Fix - Work around, void->byte
void USBCtrlTrfTxService(void);
void USBCtrlTrfRxService(void);
void USBCtrlEPServiceComplete(void);
void USBPrepareForNextSetupTrf(void);
#endif //USBCTRLTRF_H
```

ANEXO 14

```
/********************************************************************
*               Microchip USB C18 Firmware Versión 1.0
* FileName:             usb9.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       11/19/04    Original.
********************************************************************/
#ifndef USB9_H
#define USB9_H

/** I N C L U D E S **************************************************/
#include "system\typedefs.h"

/** D E F I N I T I O N S ********************************************/
/********* Standard Request Codes* USB 2.0 Spec Ref Table 9-4*********/
#define GET_STATUS  0
#define CLR_FEATURE 1
#define SET_FEATURE 3
#define SET_ADR     5
#define GET_DSC     6
#define SET_DSC     7
#define GET_CFG     8
#define SET_CFG     9
#define GET_INTF    10
#define SET_INTF    11
#define SYNCH_FRAME 12          /* Standard Feature Selectors */
#define DEVICE_REMOTE_WAKEUP    0x01
#define ENDPOINT_HALT           0x00
********************************************************************/
#define mUSBCheckAdrPendingState()
if(usb_device_state==ADR_PENDING_STATE) \
                        {                               \
                           UADDR = SetupPkt.bDevADR._byte;     \
```

```
    if(UADDR > 0)                   \
      usb_device_state=ADDRESS_STATE; \
    else                            \
      usb_device_state=DEFAULT_STATE; \
  }//end if

/** P U B L I C  P R O T O T Y P E S ********************************/
void USBCheckStdRequest(void);
#endif //USB9_H
```

ANEXO 15

```
/*********************************************************************
*           Microchip USB C18 Firmware -  CDC Versión 1.2
 * FileName:            cdc.h
* Software License Agreement
* Author                Date                    Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Rawin Rojvanit       7/21/04     Original.
 * Rawin Rojvanit       6/19/07     Fix bugs in CONTROL_SIGNAL_BITMAP.
********************************************************************/
#ifndef CDC_H
#define CDC_H

/** I N C L U D E S
*****************************************************/
#include "system\typedefs.h"

/** D E F I N I T I O N S
*************************************************/
/* Class-Specific Requests */
#define SEND_ENCAPSULATED_COMMAND   0x00
#define GET_ENCAPSULATED_RESPONSE   0x01
#define SET_COMM_FEATURE            0x02
#define GET_COMM_FEATURE            0x03
#define CLEAR_COMM_FEATURE          0x04
#define SET_LINE_CODING             0x20
#define GET_LINE_CODING             0x21
#define SET_CONTROL_LINE_STATE      0x22
#define SEND_BREAK                  0x23
#define NETWORK_CONNECTION          0x00
#define RESPONSE_AVAILABLE          0x01
#define SERIAL_STATE                0x20
/* Device Class Code */
#define CDC_DEVICE                  0x02
/* Communication Interface Class Code */
```

```
#define COMM_INTF                0x02
/* Communication Interface Class SubClass Codes */
#define ABSTRACT_CONTROL_MODEL      0x02
/* Communication Interface Class Control Protocol Codes */
#define V25TER                   0x01    // Common AT commands ("Hayes(TM)")
/* Data Interface Class Codes */
#define DATA_INTF                0x0A
/* Data Interface Class Protocol Codes */
#define NO_PROTOCOL              0x00    // No class specific protocol required
/* Communication Feature Selector Codes */
#define ABSTRACT_STATE           0x01
#define COUNTRY_SETTING          0x02
/* Functional Descriptors */
/* Type Values for the bDscType Field */
#define CS_INTERFACE             0x24
#define CS_ENDPOINT              0x25
/* bDscSubType in Functional Descriptors */
#define DSC_FN_HEADER            0x00
#define DSC_FN_CALL_MGT          0x01
#define DSC_FN_ACM               0x02    // ACM - Abstract Control Management
#define DSC_FN_DLM               0x03    // DLM - Direct Line Managment
#define DSC_FN_TELEPHONE_RINGER     0x04
#define DSC_FN_RPT_CAPABILITIES     0x05
#define DSC_FN_UNION             0x06
#define DSC_FN_COUNTRY_SELECTION    0x07
#define DSC_FN_TEL_OP_MODES         0x08
#define DSC_FN_USB_TERMINAL         0x09
/* more.... see Table 25 in USB CDC Specification 1.1 */
/* CDC Bulk IN transfer states */
#define CDC_TX_READY             0
#define CDC_TX_BUSY              1
#define CDC_TX_BUSY_ZLP          2      // ZLP: Zero Length Packet
#define CDC_TX_COMPLETING        3
#define mUSBUSARTIsTxTrfReady()     (cdc_trf_state == CDC_TX_READY)
#define mCDCUsartRxIsBusy()         CDC_BULK_BD_OUT.Stat.UOWN
```

```
#define mCDCUsartTxIsBusy()        CDC_BULK_BD_IN.Stat.UOWN
#define mCDCGetRxLength()          cdc_rx_len
#define mUSBUSARTTxRam(pData,len)   \
{                                   \
   pCDCSrc.bRam = pData;          \
   cdc_tx_len = len;              \
   cdc_mem_type = _RAM;           \
   cdc_trf_state = CDC_TX_BUSY;    \
}
#define mUSBUSARTTxRom(pData,len)   \
{                                   \
   pCDCSrc.bRom = pData;          \
   cdc_tx_len = len;              \
   cdc_mem_type = _ROM;           \
   cdc_trf_state = CDC_TX_BUSY;    \
}

/** S T R U C T U R E S ********************************************/
#define LINE_CODING_LENGTH         0x07
typedef union _LINE_CODING
{
   struct
   {
      byte _byte[LINE_CODING_LENGTH];
   };
   struct
   {
      DWORD   dwDTERate;         // Complex data structure
      byte    bCharFormat;
      byte    bParityType;
      byte    bDataBits;
   };
} LINE_CODING;
typedef union _CONTROL_SIGNAL_BITMAP
{
```

```
    byte _byte;
    struct
    {
      unsigned DTE_PRESENT:1;     // [0] Not Present  [1] Present
      unsigned CARRIER_CONTROL:1; // [0] Deactivate   [1] Activate
    };
} CONTROL_SIGNAL_BITMAP;
/* Header Functional Descriptor */
typedef struct _USB_CDC_HEADER_FN_DSC
{
    byte bFNLength;
    byte bDscType;
    byte bDscSubType;
    word bcdCDC;
} USB_CDC_HEADER_FN_DSC;
/* Abstract Control Management Functional Descriptor */
typedef struct _USB_CDC_ACM_FN_DSC
{
    byte bFNLength;
    byte bDscType;
    byte bDscSubType;
    byte bmCapabilities;
} USB_CDC_ACM_FN_DSC;
/* Union Functional Descriptor */
typedef struct _USB_CDC_UNION_FN_DSC
{
    byte bFNLength;
    byte bDscType;
    byte bDscSubType;
    byte bMasterIntf;
    byte bSaveIntf0;
} USB_CDC_UNION_FN_DSC;
/* Call Management Functional Descriptor */
typedef struct _USB_CDC_CALL_MGT_FN_DSC
{
```

```
    byte bFNLength;
    byte bDscType;
    byte bDscSubType;
    byte bmCapabilities;
    byte bDataInterface;
} USB_CDC_CALL_MGT_FN_DSC;

/** E X T E R N S ****************************************************/
extern byte cdc_rx_len;
extern byte cdc_trf_state;
extern POINTER pCDCSrc;
extern byte cdc_tx_len;
extern byte cdc_mem_type;

/** P U B L I C  P R O T O T Y P E S *********************************/
void USBCheckCDCRequest(void);
void CDCInitEP(void);
byte getsUSBUSART(char *buffer, byte len);
void putrsUSBUSART(const rom char *data);
void putsUSBUSART(char *data);
void CDCTxService(void);
#endif //CDC_H
```

ANEXO 16

```
%   MMC card
%   Read/Write a MMC/SD Card
%   SAVE  CHANNEL BUFFERS
disp 'RESET channel buffers ...'
saved_n=1;
m1 = '600';
save('MEMORIA','saved_n','m1');
```

ANEXO 17

```
%   MMC card
%   Read/Write a MMC/SD Card
 %  Decimal Commands:
%   Initialize MMC card = 105
%   Stop = 112
%   Read the first 512 bytes = 114 0 0 0 0
%   Read 512 bytes ( specific address) = 97 0 0 4 0
%   Write 512 bytes (specific address) = 119 0 0 4 0
%   Read CID register = 99
clear all;
clc;
disp '***************************************'
disp '*   Read/Write MMC/SD Card commander   *'
disp '***************************************'
disp 'Command menu:'
disp '  [1] Initialize MMC/SD card'
disp '  [2] Read MMC card CID register'
disp '  [3] Read 512bytes in specified address'
disp '  [4] Read one buffer of 512 bytes in specified address'
disp '  [5] Write 512bytes in specified address'
disp '***************************************'
in = input('Select: ','s');
%USB CDC emulation interface
s = serial('/dev/ttyS4','BaudRate',9600); %230400); %open serial port
s.InputBufferSize = 32768; %setup input buffer
s.ReadAsyncMode = 'continuous';
if (s.Status == 'closed')   %check if open port
   fopen(s);   %open the serial port
end
if in == '1'
   % INITIALIZE MMC CARD
   cmd =[105 0 0 0 0];
   fwrite(s,cmd); %send binario a tarjeta
```

```
    while ( s.BytesAvailable < 0)
    end
    % lee dato de la Tarjeta
    input = fscanf(s,'%c',16)
elseif in == '2'
    % READ CID REGISTER
    cmd =[99 0 0 0 0];
    fwrite(s,cmd); %send binario a tarjeta
    while ( s.BytesAvailable < 0)
    end
    % lee dato de la Tarjeta
    input = fscanf(s,'%c',16)
elseif in == '3'
    % READ 512 bytes from MMC card
    disp 'MMC Card 64MB [0h 3D 09 00 00] max address'
    disp 'Only 512 bytes multiples for address'
    disp '                    MSB       LSB'
    disp 'Setup address in hex [0h 00 00 06 00]'
    a = input('msb[0]: ','s');
    b = input('   [2]: ','s');
    c = input('   [3]: ','s');
    d = input('lsb[4]: ','s');
    a = hex2dec(a);
    b = hex2dec(b);
    c = hex2dec(c);
    d = hex2dec(d);
    cmd =[97 a b c d]; %command
    fwrite(s,cmd); %send binario a tarjeta
    % READ BUFFER AND ASIGN TO EACH CHANNEL
    n = 1;
    while(n <= 21)
       while ( s.BytesAvailable <= 24)
       end
       inputbuffer = fscanf(s,'%c',4); %read data
       CH0(n) = str2num(inputbuffer);
```

```
        inputbuffer = fscanf(s,'%c',4); %read data
        CH1(n) = str2num(inputbuffer);
        inputbuffer = fscanf(s,'%c',4); %read data
        CH2(n) = str2num(inputbuffer);
        inputbuffer = fscanf(s,'%c',4); %read data
        CH3(n) = str2num(inputbuffer);
        inputbuffer = fscanf(s,'%c',4); %read data
        CH4(n) = str2num(inputbuffer);
        inputbuffer = fscanf(s,'%c',4); %read data
        CH5(n) = str2num(inputbuffer);
        n = n+1;
    end
        while(s.BytesAvailable < 0)
        end
        inputbuffer = fscanf(s,'%c',8);
        INFO = str2num(inputbuffer);
        %while ( s.BytesAvailable < 0)
        %end
        %inputbuffer = fscanf(s,'%c',512); %ok..
 load MEMORIA;
n = saved_n;
 for(i=1 : 21)
    Canal0(n) = CH0(i);
    Canal1(n) = CH1(i);
    Canal2(n) = CH2(i);
    Canal3(n) = CH3(i);
    Canal4(n) = CH4(i);
    Canal5(n) = CH5(i);
    n= n+1;
end
disp 'Next MEMORY ADDRESS ...'
z = hex2dec(m1);
z = dec2hex(z+512);
st = [z(1) z(2) z(3)];
m1 = st
```

```
 saved_n = n;
save('MEMORIA','saved_n','m1','Canal0','Canal1','Canal2','Canal3','Canal4','Canal5')
;
   %
   %   PLOTING DATA
   %
   figure(1)
   plot(1:length(Canal0), Canal0);
   axis([0 length(Canal0) 0 1024]);
   title('Canal0 input signal');
   xlabel('Samples');
   ylabel('Amplitud');
  figure(2)
   plot(1:length(Canal1), Canal1);
   axis([0 length(Canal1) 0 1024]);
   title('Canal1 input signal');
   xlabel('Samples');
   ylabel('Amplitud');
   figure(3)
   plot(1:length(Canal2), Canal2);
   axis([0 length(Canal2) 0 1024]);
   title('Canal2 input signal');
   xlabel('Samples');
   ylabel('Amplitud');
   figure(4)
   plot(1:length(Canal3), Canal3);
   axis([0 length(Canal3) 0 1024]);
   title('Canal3 input signal');
   xlabel('Samples');
   ylabel('Amplitud');
   figure(5)
   plot(1:length(Canal4), Canal4);
   axis([0 length(Canal4) 0 1024]);
   title('Canal4 input signal');
   xlabel('Samples');
   ylabel('Amplitud');
```

```
    figure(6)
    plot(1:length(Canal5), Canal5);
    axis([0 length(Canal5) 0 1024]);
    title('Canal5 input signal');
    xlabel('Samples');
    ylabel('Amplitud');
elseif in == '4'
    % READ 512 bytes from MMC card
    disp 'MMC Card 64MB [0h 3D 09 00 00] max address'
    disp 'Only 512 bytes multiples for address'
    disp '                    MSB       LSB'
    disp 'Setup address in hex [0h 00 00 06 00]'
    a = input('msb[0]: ','s');
    b = input('   [2]: ','s');
    c = input('   [3]: ','s');
    d = input('lsb[4]: ','s');
    a = hex2dec(a);
    b = hex2dec(b);
    c = hex2dec(c);
    d = hex2dec(d);
    cmd =[97 a b c d]; %command
    fwrite(s,cmd); %send binario a tarjeta
    % READ BUFFER
    while ( s.BytesAvailable < 0)
    end
    inputbuffer = fscanf(s,'%c',512); %ok..
    inputbuffer
 elseif in == '5'
    % WRITE 512 bytes from MMC card
    disp 'Setup address in hex [0h 00 00 02 00]'
    a = input('msb[0]: ','s');
    b = input('   [2]: ','s');
    c = input('   [3]: ','s');
    d = input('lsb[4]: ','s');
    a = hex2dec(a);
```

```
    b = hex2dec(b);
    c = hex2dec(c);
    d = hex2dec(d);
    cmd =[119 a b c d]; %command
    fwrite(s,cmd); %send binario a tarjeta
    out(1,512) = num2mstr(0);
    data = input('Data: ','s');
    d=cat(2,data,out);
    e=d(1:512);
    e = uint8(e);
    fwrite(s,e); %send binario a tarjeta
  end
%END SELECT
 if (s.Status == 'open')%verifica si esta abierto serial port
    fclose(s);   %sierra serial port
end
delete (s);
disp '!!! Run RESET_buffer_MMC.m to reset the buffers Â¡Â¡Â¡'
```