# UNIVERSIDAD DEL AZUAY

## Facultad de Ciencia Y Tecnología

### Escuela de Ingeniería Mecánica

**Estudio para simular una Red CAN con aplicación en comunicación de dispositivos electrónicos en el automóvil.**

**TRABAJO DE GRADUACION PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECÁNICO AUTOMOTRIZ**


**Autor:**

**Teodoro Paúl Tenesaca Arpi**

**Director:**

**José Fernando Muñoz Vizhñay**


**Cuenca – Ecuador**

**2013**

## DEDICATORIA

**A Dios**.

Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

**A mi amada esposa Fernanda.**

Por su amor, comprensión y paciencia que me permitieron alcanzar mi segunda ingeniería, gracias por ser mi luz cuando mi camino se nublaba. Vida le llevo presente en mi mente y mi corazón.

**A mi padre**.

Por su ejemplo de perseverancia y constancia que ha sido mi constante guía para salir adelante. Gracias a Usted soy el hombre que soy, espero algún día poder parecerme a Usted Padre Santo.

**AGRADECIMIENTOS**

El fin y el comienzo de una nueva etapa de mi vida…Agradezco a mis profesores que durante toda mi carrera universitaria han aportado con su sabiduría en mi formación. A mis profesores de taller que supieron impartirme de valiosos consejos. De igual manera agradecer a mi director Ing. Fernando Muñoz, quien con sus conocimientos y su experiencia ha logrado que pueda terminar este trabajo de grado con éxito. Y por último a mis amigos y compañeros de carrera, con los que he compartido todas mis alegrías y problemas.

Para ellos: Muchas gracias y que Dios los bendiga.

# INDICE DE CONTENIDOS

## CAPÍTULO 2: PROPUESTA DEL FIRMWARE PARA LAS TARJETAS CAN

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ÍNDICE DE ANEXOS

**Estudio para simular una Red CAN con aplicación en comunicación de dispositivos electrónicos en el automóvil.**

## RESUMEN

El presente trabajo de graduación se basó en el estudio para simular una Red CAN con aplicación en comunicación de dispositivos electrónicos en el automóvil. Inicialmente se realizó el estudio de los elementos que constituyen la Red, el intercambio de información entre nodos mediante el control de acceso al medio y el control lógico para la transmisión de los mensajes utilizados por el protocolo, formatos de tramas, manejo de errores, etc. Con esa información se elaboró la propuesta del firmware mediante microcontroladores PIC y el compilador MikroC, para interconectar dos tarjetas mediante Bus CAN. El proyecto incluyó un Script para capturar datos en Matlab y visualizar las señales CANH y CANL de la Red. Se concluyó que, se puede construir un interfaz de comunicación CAN en nuestro medio, cumpliendo los requerimientos de un sistema distribuido.

**Palabras clave:** Red CAN, firmware, microcontroladores PIC, compilador MikroC, Script, Matlab, señales CANH y CANL.

Ing. Hernán Viteri.
**Miembro de la Junta Académica**

Ing. Fernando Muñoz.
**Director**

Sr. Paúl Tenesaca A.
**Autor**

## ABSTRACT

**Study to Simulate a CAN Network with application in communication of electronic devices in the automobile**

The present graduation work was based on the study to simulate a CAN network with application in communication of electronic devices in the automobile. Initially, we carried out the study of the elements that are part of the Network, the exchange of information between nodes through media access control and logic control for the transmission of the messages employed by protocol, frame format, error management, etc. With this information we presented a proposal for the firmware through PIC microcontrollers and MikroC compilers in order to interconnect two cards through CAN Bus. The project included a Script to capture Matlab data and visualize CANH and CANL signals in the Network. We concluded that it is possible to build a CAN communication interface in our environment, in compliance with the requirements of a distributed system.

**Key Words:** CAN Network, firmware, PIC microcontrollers, MikroC compiler, Script, Matlab, CANH an CANL signals.

Ing. Hernán Viteri
**Member of the Academic Board**

Ing. Fernando Muñoz
**Director**

Sr. Paúl Tenesaca A.
**Author**

UNIVERSIDAD DEL AZUAY
DPTO. IDIOMAS

Translated by,
Diana Lee Rodas

**Teodoro Paúl Tenesaca Arpi**
**Trabajo de Graduación**
**Ing. José Fernando Muñoz Vizhñay**
**Febrero, 2013**

# ESTUDIO PARA SIMULAR UNA RED CAN CON APLICACIÓN EN COMUNICACIÓN DE DISPOSITIVOS ELECTRÓNICOS EN EL AUTOMÓVIL

## INTRODUCCIÓN

Hoy en día, dado el continuo aumento del equipamiento eléctrico y electrónico en los vehículos, la arquitectura eléctrica de los automóviles se ha modificado. Anteriormente, los fabricantes automotrices conectaban dispositivos eléctricos y electrónicos en los vehículos utilizando sistemas de cableado punto a punto, aumentando en exceso el peso y el volumen del cableado, eso sin contar con las posibles fuentes de avería que ello supondría. También era difícil compartir la información entre las distintas Unidades de Control Electrónico (UCE´s). En algunos casos fue necesario montar una gran cantidad de sensores similares, cables y conectores adicionales, para comunicar a las diferentes Unidades de Control Electrónico. En estas circunstancias surgió el protocolo de comunicaciones CAN como una red estándar para vehículos, una red multiplexada como solución técnica para interconectar las diferentes unidades de control entre sí, mediante un par trenzado de cables (BUS de datos), de forma que todas las unidades control interconectadas a través del BUS puedan intercambiar y compartir informaciones, reduciéndose de forma significativa el volumen, peso y complejidad de la instalación eléctrica en los vehículos. El protocolo CAN, en la actualidad es el más utilizado para aplicaciones en sistemas de comunicación de datos de vehículos, siendo implementado de forma obligatoria desde el año 2008.

# CAPÍTULO 1
# INTRODUCCIÓN AL ESTUDIO DE LA RED CAN (CONTROLADOR DE ÁREA DE RED)

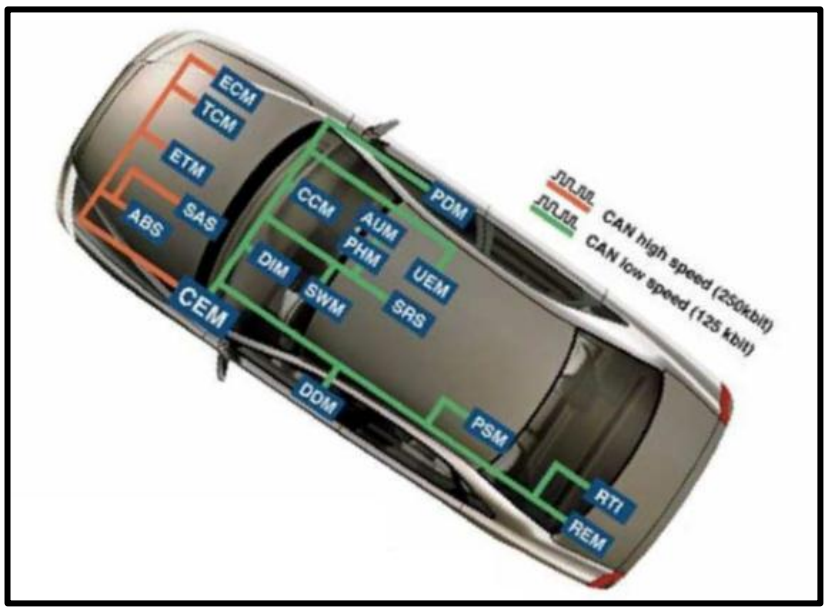



**FIGURA 1.1:** Red CAN

**Fuente:** http://www.c4atreros.com/sistema-can-bus-4863063.html, acceso 05-Abr-2012

CAN o Controller Area Network (Control de Área de Red), fue desarrollado por Bosch en 1985, previo a esto, los fabricantes automotrices conectaban dispositivos electrónicos en los vehículos utilizando sistemas de cableado de punto a punto. Sin embargo, conforme los fabricantes comenzaron a utilizar más y más dispositivos electrónicos en los vehículos, el costo general del vehículo se incrementaba.

Al remplazar el cableado por redes de comunicación en los vehículos, se logró reducir el costo de instalación del mismo, su complejidad, peso del mismo y por supuesto vehículos más seguros. "*CAN, un sistema de bus serial de alta integridad destinado para comunicar dispositivos inteligentes, surgió como la red estándar para vehículos. La industria automotriz adoptó rápidamente CAN y, en 1993, se convirtió en el estándar internacional conocido como ISO 11898. Desde 1994, se han estandarizado varios protocolos de alto nivel a partir de CAN, como CANopen y DeviceNet, y su uso se ha extendido a otras industrias*" (National Instruments, 2012, pág. 1).

## 1.1 Contexto Histórico de CAN

En los años 80 la industria automotriz tenía tres problemas:

- El primero era la demanda de mayor comodidad en los automóviles particulares tales como ventanillas accionadas eléctricamente, asientos con calefacción, control de la temperatura del habitáculo, ajuste de asientos, espejos, equipos de audio , sistema de posicionamiento global controlado por satélite (GPS), etc.
- El segundo era la seguridad en el vehículo: tales como mecanismos y dispositivos encargados de disminuir el riesgo a que se produzca un accidente y entre los cuales tenemos frenos, suspensión, luces y dirección, etc.
- El tercer problema: consumo de combustible, rendimiento, y por la contaminación.

Los tres problemas eran encarados por medio de control electrónico, al principio con una única unidad electrónica de control (ECU), y luego con el agregado de otras. Entonces surge la necesidad de establecer una adecuada comunicación entre las distintas unidades electrónicas de control de dichos procesos.

Una proyección a futuro estimaba que para el año 2005 los automóviles podían contar con 100 microcontroladores especializados. Pero el cableado necesario para establecer la comunicación entre estos microcontroladores sería muy extenso, y además requeriría distintos tipos de cable. "*Es entonces que la compañía Alemana Robert Bosch estudia los buses de campo existentes en los ochenta para implementarlos en los automóviles pero ninguno de ellos cumplía al completo los requerimientos; se decide diseñar y desarrollar un nuevo sistema de comunicaciones industriales que permita enlazar los diversos dispositivos electrónicos instalados en los automóviles, para agregar mayor funcionalidad y reducir el cableado. En el año 1983 Robert Bosch Gmbh inició el proyecto el cual fue dirigido por Siegfried Dais, Martin Litschel y el Dr. Uwe Kiencke.*

En febrero de 1986 Robert Bosch presentó en el congreso de la SAE (Society of Automotive Engineers) la primera versión del protocolo CAN para aplicación en automóviles.

En 1987 INTEL desarrollo el FullCAN-Chip 82526 y poco después Philips Semiconductors también terminó el desarrollo de su primer chip, el 80C200. Ambos constituyen la base para los controladores actuales, con diferencias notables en cuanto a manejo de errores y filtrado de admisión.

En 1992 grupos de fabricantes fundaron la organización "CAN en la automoción" (CiA) promovida por Holger Zeltwanger, sin fines de lucro, con el fin de proporcionar información técnica, de productos y comercialización para el uso del bus CAN; poco tiempo después la CiA publica un artículo técnico el cual describía la capa física del protocolo y recomendaba el uso de transceptores CAN que cumplieran la norma ISO 11898.

En Noviembre de 1993, el protocolo CAN es estandarizado bajo la norma ISO 11898 en la que se define la capa física para velocidades de transferencia de datos de hasta 1 Mbps; un bus con tolerancia a fallos para aplicaciones de baja velocidad hasta 125Kbit/s estandarizado bajo la norma ISO 11519 y luego actualizado por la ISO 11519-2.

En 1995 se publica una mejora a la norma ISO 11898 en la que se añaden 29 bits como identificadores que hasta ese momento eran 11 bits, también se incluyó recomendaciones para la especificación CAN 2.0B.

*"En el año 2000 se define y se desarrolla el protocolo de comunicaciones en tiempo real para CAN denominado TTCAN (Time-Triggered communication of CAN) desarrollado por Demd Mueller Y Tomas Fuehrer (ingenieros de Bosch) junto con expertos de la industria de semiconductores; basado en CAN bajo el estándar ISO 11898-4".* (CALVA CUENCA, 2012, págs. 16,17,18)

## 1.2 Cronología Histórica del Protocolo CAN

***"1983** Inicio del proyecto de Bosch para el desarrollo dentro de una red vehicular.*
***1986** Introducción oficial del protocolo CAN.*
***1987** El primer chip CAN de Intel y Semiconductores Phillips.*
***1991** Especificación de Can 2.0 publicada por Bosch.*

***1991*** *Protocolo de Capa Alta CAN Kingdom CAN-based introducido por Kvaser.*

***1992*** *Establecimiento de usuarios internacionales y grupo de manufactureros de CAN en Automatización.*

***1992*** *Protocolo de Capa de Aplicación Can publicado por CiA.*

***1992*** *Primeros automóviles de Mercedez-Benz usando red CAN.*

***1993*** *Publicación del estándar ISO 11898.*

***1994*** *Primera conferencia internacional de CAN (iCC) organizada por CiA.*

***1994*** *Introducción al protocolo de Dispositivo de Red por Allen-Bradley.*

***1995*** *Publicación de La enmienda de la ISO 11898 (formato extendido del marco).*

***1995*** *Protocolo CANOpen publicado por CiA.*

***2000*** *Desarrollo del protocolo de comunicación tiempo-accionado para CAN (TTCAN).”* (CAN in Automation (CiA), 2012, págs. 1-2).

## 1.3 Necesidad de la Red CAN en el Automóvil

Para el buen funcionamiento de un vehículo, hoy en día varias unidades de control electrónico se encuentran instaladas en él mismo, por tanto la comunicación entre estos dispositivos es indispensable. Así por ejemplo en una transmisión de datos convencional, El módulo de ABS (Sistema de Freno Antibloqueo), El módulo TCS (Sistema de Control de Tracción), El módulo ESP (Sistema de Estabilidad Programable) reciben constantemente señales del sensor de velocidad de las ruedas, al ser una transmisión de datos convencional la comunicación entre cada uno de los módulos y con el sensor es de tipo punto a punto, por consiguiente el número resultante de conexiones punto a punto y la instalación eléctrica es significativa lo que representa un costo elevado y una gran complejidad en la instalación de este tipo de sistemas de seguridad activa.

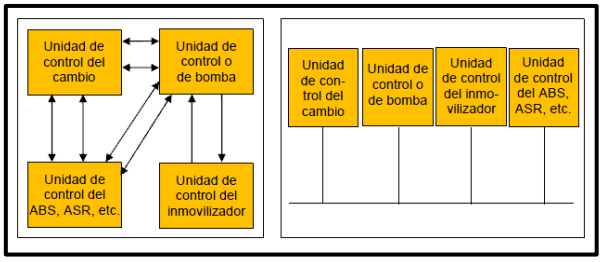



**FIGURA 1.2**: Transmisión de datos convencional en comparación con CAN
**Fuente:** http://www.c4atreros.com/sistema-can-bus-4863063.html, acceso 07-Abr-2012

Mientras que en una Red Multiplexada o Red CAN, las rutas de cableado son cortas, la disposición de información en formato serie y la cantidad de la misma que se mueven en la red, son mayores, además el bus de datos CAN tiene la ventaja particular de que, si falla un componente, el resto del sistema continúa funcionando normalmente, reduciendo en gran medida el riesgo de un fallo total del sistema.

## 1.4 Definición de Can

Can-Bus es un protocolo o lenguaje de comunicación en serie desarrollado por Bosch para el intercambio de información entre unidades de control electrónicas del automóvil, tales como sistemas de gestión del motor, control de luces, aire acondicionado, bloqueo central entre otros, orientados a proporcionar confort y seguridad al conductor.

"*CAN significa Controller Area Network (Red de área de control) y Bus, en informática, se entiende como un elemento que permite transportar una gran cantidad de información*." (ZAPATA Vaca, 2013, pág. 116)

El Bus, en consecuencia, permite compartir información entre las unidades de control conectadas al sistema, provocando que el número de sensores empleados sea menor, así como de la cantidad de cables que componen la instalación eléctrica del vehículo.

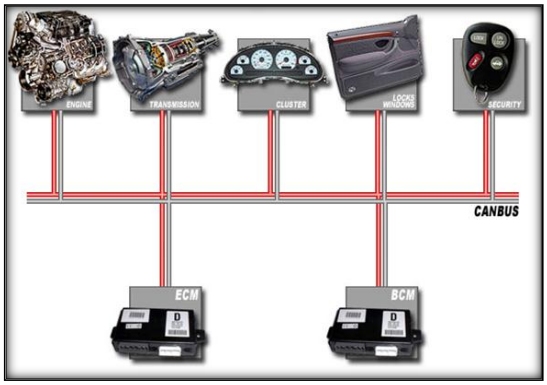



**FIGURA 1.3:** Acoplamiento de las unidades de control mediante CAN Bus
**Fuente:** http://canbuskit.com/what.php, acceso 07-Abr-2012

## 1.5 Campos de aplicación de las Redes CAN en el Automóvil

### 1.5.1 Acoplamiento de unidades de control

Funcionan en tiempo real como por ejemplo: las unidades de control del motor, el control del cambio y el control de estabilidad programable.
Se caracterizan por unas velocidades de transmisión situadas entre 125 kBit/s y 1MBit/s (High-Speed-CAN).

### 1.5.2 Electrónica de carrocería y de confort

Para control y regulación de componentes, por ejemplo: la regulación del aire acondicionado, cierre centralizado y ajuste de los asientos.
Las velocidades de transmisión se sitúan entre 10 KBit/s y 125 125 KBit/s (Low-speed-CAN).

### 1.5.3 Comunicación móvil

Permiten comunicar componentes como el sistema de navegación GPS, celulares, o los equipos de audio con unidades centrales de indicación y mando. El objetivo consiste, agrupar informaciones de estado y control para poder conseguir que la distracción del conductor sea mínima.

Las velocidades de transmisión de los datos se sitúan hasta los 125 kBit/s; no obstante, sin ser posible la transmisión directa de datos de audio o vídeo.

## 1.6 Características del Bus Can

Al ser CAN un protocolo de comunicación serie basado en una topología bus para la transmisión de mensajes en ambientes distribuidos en tiempo real, significa que es idóneo también en aplicaciones de control y automatización industrial, con un alto nivel de seguridad y multiplexación.

### 1.6.1 Normas de Estandarización

Es un protocolo definido por las normas ISO (International Organization for Standardization) específicamente en la ISO 11898, que describe los siguientes estándares aplicables a CAN:

- ***"ISO/DIS 11898-1:*** *Capa de enlace y señalización física.*
- ***ISO/DIS 11898-2:*** *Unidad de acceso al medio de alta velocidad.*
- ***ISO/CD 11898-3:*** *Interfaz de baja velocidad tolerante a fallos.*
- ***ISO/CD 11898-4:*** *Comunicación: Tiempo activado para la comunicación Time triggered*" (Universidad de Oviedo, 2012, pág. 2).

### 1.6.2 Medio de transmisión adaptable

Cableado reducido a comparación de sistemas convencionales (punto a punto) siendo el estándar de transmisión un par trenzado.

### 1.6.3 Estructura definida

"*La información que circula entre las unidades a través de los dos cables (bus) son paquetes de bits (0's y 1's) con una longitud limitada y con una estructura definida de campos que conforman el mensaje*." (ALEPUZ, 2012, pág. 9)

### 1.6.4 Número de nodos

"*Es posible conectar hasta 32 nodos en una sola red CAN, sin necesidad de ningún cambio de software o hardware o en la capa de aplicación*", (CALVA CUENCA, 2012, pág. 19)

Esta característica ofrece a la red una gran flexibilidad y posibilidad de expansión, ya que nuevos nodos pueden ser añadidos a la red sin la necesidad de hacer ningún cambio en el hardware ni en el software existente, pudiendo ser comparable con un sistema plug and play.

### 1.6.5 Garantía de tiempos de latencia

La longitud del mensaje es de 8 bytes como máximo, por lo que la latencia es baja entre transmisión y recepción. El acceso al bus es controlado por el protocolo CARRIER SENSE MULTIPLE ACCES/ COLLISION DETECTION WITH ARBITRATION ON MESSAGE PRIORITY (CSMA/CD with Amp), es decir que la colisión de mensajes es evitada por medio de arbitraje tomando en cuenta la prioridad de los mismos.

### 1.6.6 Optimización del ancho de banda

La optimización del ancho de banda o velocidad a la que se transfieren las tramas de datos por el bus, está en función de la prioridad del mensaje de la trama CAN.

### 1.6.7 Desconexión de nodos defectuosos

"*Si un nodo de la red cae, sea cual sea la causa, la red puede seguir funcionando; de forma contraria también se pueden añadir nodos al bus sin afectar al resto del sistema*". (CALVA CUENCA, 2012, pág. 19)

### 1.6.8 Velocidad flexible

"*ISO define dos tipos de redes CAN; una red de alta velocidad (de hasta 1 Mbps) especificada por la ISO 11898-2 y una red de baja velocidad tolerante a fallos (menor o igual a 125Kbps) especificada por la ISO 11898-3.*" (CALVA CUENCA, 2012, pág. 19)

#### 1.6.8.1 Relación de velocidad distancia

"*La velocidad depende de la distancia hasta un máximo de 1000 metros aunque es posible usar repetidores; la Tabla 1.1 muestra la comparación entre velocidad y distancia para el protocolo CAN*". (CALVA CUENCA, 2012, pág. 19)

| Velocidad (Kbps) | Tiempo de Bit (µs) | Longitud máxima (m) |
|---|---|---|
| 1.000 | 1 | 30 |
| 800 | 1,25 | 50 |
| 500 | 2 | 100 |
| 250 | 7 | 250 |
| 125 | 8 | 500 |
| 50 | 20 | 1.000 |
| 20 | 50 | 2.500 |
| 10 | 100 | 5.000 |

**Tabla 1.1:** Velocidad – Distancia en CAN

**Fuente:** Análisis Protocolar del bus de campo Can, Dr.-Ing. Héctor Kaschel C. Depto. de Ingeniería Eléctrica Universidad de Santiago de Chile, acceso el 10 de junio del 2012

### 1.6.9 Orientado a mensajes

Los mensajes a transmitir desde cualquier nodo o unidad de control en una red CAN no contienen la dirección del nodo emisor ni del nodo receptor. La información se descompone en mensajes asignándoles un identificador único y luego se los encapsula, este identificador indica la prioridad del mensaje. El mensaje de mayor prioridad accede al bus, donde cada nodo receptor decide o no aceptar el mensaje, mientras que los mensajes de menor prioridad se retransmitirán automáticamente en los siguientes ciclos de bus.

### 1.6.10 Recepción por Multidifusión (multicast)

*"Todos los nodos de la red reciben el mismo mensaje al mismo tiempo y pueden acceder al bus de forma simultánea con sincronización de tiempos, realizando el mismo procedimiento de filtrado simultáneamente"*. (COSTALES, 2012, pág. 25)

### 1.6.11 Medio compartido (broadcasting)

Por medio de la red toda la información es enviada de forma simultánea, por lo que los nodos o módulos CAN de recepción habrán de saber si la información les concierne o deben rechazarla.

### 1.6.12 Detección y señalización de errores

CAN posee una gran capacidad de detección de errores, y una alta inmunidad a la interferencia electromagnética. Algunos mecanismos de detección de errores son: monitoreo, chequeo de redundancia cíclica, bits de relleno, chequeo de formato de tramas del mensaje. La probabilidad de error de un mensaje es de 4.7E-11

### 1.6.13 Retransmisión automática de tramas erróneas

Es una característica que aporta integridad a los datos es decir, CAN garantiza que los nodos CAN, recibirán los mensajes. En el caso de mensajes erróneos se identifican tan pronto como el bus esté libre, por lo que cualquier nodo puede detectar un error. Estos mensajes erróneos se descartan y se retransmiten automáticamente.

### 1.6.14 Jerarquía multimaestro

"*CAN es un sistema multimaestro en el cual puede haber más de un maestro (o master) al mismo tiempo y sobre la misma red, es decir, todos los nodos son capaces de transmitir, hecho que permite construir sistemas inteligentes y redundantes*". (CALVA CUENCA, 2012, pág. 20)

## 1.7 Estructura de Capas del Protocolo CAN

El protocolo CAN, en su especificación ISO 11898, describe como la información será transmitida entre los diferentes dispositivos de una red. A su vez emplea el MODELO OSI simplificado, es decir que utiliza solo las dos capas más bajas que son la Capa Física y la Capa de Enlace.

| Capa Enlace de Datos | |
|---|---|
| LLC (Logical Link Control) | Filtrado de mensajes<br>Notificación de sobrecarga<br>Proceso de recuperación |
| MAC (Medium Access Control) | Encapsulamiento/Desencapsulamiento de datos<br>Codificación de tramas (bits de relleno)<br>Arbitrar el acceso al medio<br>Detección de errores<br>Señalización de errores<br>Acuses de recibo<br>Serialización/Deserialización |
| **Capa Física** | |
| Sincronización<br>Codificación/Decodificación de bits<br>Bit timming (temporización de bit) | |
| Características del transmisor/receptor CAN | |

**Tabla 1.2:** Estructura de capas del protocolo CAN (ISO 11898)
**Fuente:** CAN in Automation (CiA), http://www.can-cia.de/index.php?id=161, 07-Abr-2012

### 1.7.1 Capa Física

Es la encargada de la transferencia de bits entre los diferentes nodos que componen la red, define el medio físico o como las señales serán transmitidas a través de él. Esta capa especifica algunos aspectos como:

- Niveles de Señal
- Sincronización.
- Codificación/Decodificación de bits.
- Temporización: tiempos en que los bits se transfieren al bus CAN

#### 1.7.1.1 Medio Físico

El medio físico consiste en un cable de par trenzado y adaptado en los extremos por resistencias que representan la impedancia característica de la línea. En la especificación básica de CAN, la velocidad máxima de transmisión es de 250 Kbps, mientras que en la versión ampliada alcanza velocidades de 1 Mbps.

La longitud máxima es 1 kilómetro. Se permite utilizar los dispositivos puente o los repetidores para aumentar el número de los nodos del bus que pueden ser conectados, o para aumentar la distancia permitida entre los nodos del bus o para proporcionar el aislamiento galvánico.

### 1.7.1.2 Topología bus

Es una red lineal, todos los nodos están conectados a un circuito común (bus). La información viaja por el cable en ambos sentidos y tiene en sus dos extremos una resistencia o elemento de cierre para reducir al mínimo las reflexiones.
Si un computador falla, la comunicación se mantiene, no sucede lo mismo si el bus es el que falla. El tipo de cableado que se usa puede ser un par trenzado.

### 1.7.1.3 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.

Básicamente se presenta tres tipos de ruidos a saber:

- ***"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 impedancia de los cables debe tomarse en cuenta al diseñar el esquema de alumbrado para el sistema de medición, (jamás se debe unir los cables de retorno).*

  ***Solución: "****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*" (PEREZ, 2006, pág. 2).

- **"*Ruido Capacitivo*** *Se produce cuando se establece un campo eléctrico no deseado entre el circuito de medición y alguno próximo.*
  ***Solución:*** *Para disminuir el ruido capacitivo es necesario 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 se observa la conexión adecuada de la guarna, la misma que se ha aterrado solo un extremo*" (PEREZ, 2006, pág. 3).

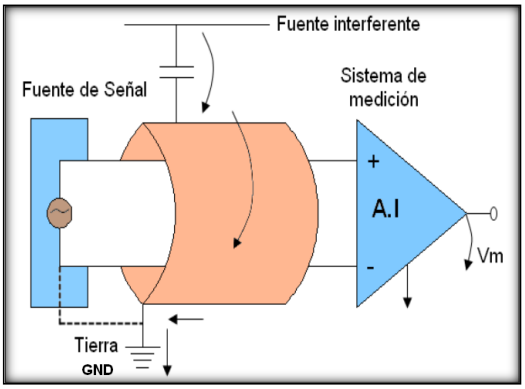



**FIGURA 1.4:** Colocación de la guarda capacitiva, para evitar el Ruido Capacitivo

**Fuente:** Ing. Leonel Pérez, Microcontroladores I, UDA 2006

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.

- "***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.*

  ***Solución:*** *Reducir el área encerrada en el circuito de señal, trenzando los cables que conectan la fuente de señal con el sistema de medición*" (PEREZ, 2006, págs. 4-5).

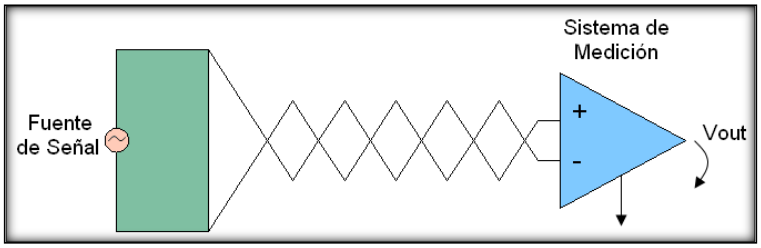



**FIGURA 1.5:** Trenzado para minimizar el campo próximo a un circuito
**Fuente:** Ing. Leonel Pérez, Microcontroladores I, UDA 2006

### 1.7.1.4 Nivel de Señal

El uso de voltajes diferenciales permite que las redes CAN funcionen cuando una de las líneas de señales es separada; con lo que cada nodo conectado al bus interpreta dos niveles lógicos:

- "***Dominante**.: la tensión diferencial (CAN_H-CAN_L) es del orden 2.0 V con CAN_H=3.5V Y CAN_L=1.5V (nominales); equivale al nivel lógico cero.*
- ***Recesivo**: la tensión diferencial (CAN_H-CAN_L) es del orden 0 V con CAN_H=CAN_L=2.5V (nominales); equivale a nivel lógico uno*". (LOPEZ, 2012, pág. 15)

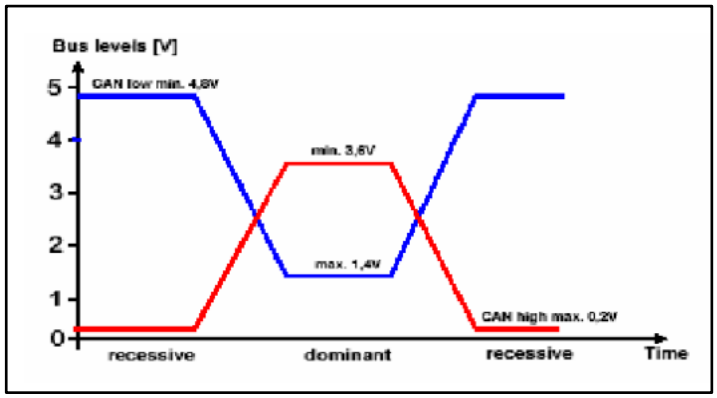



**FIGURA 1.6:** Niveles de tensión (estándar ISO11519).
**Fuente:** www. dspace.epn.edu.ec/bitstream/15000/9136/1/T11576.pdf, 07-Abr-2012

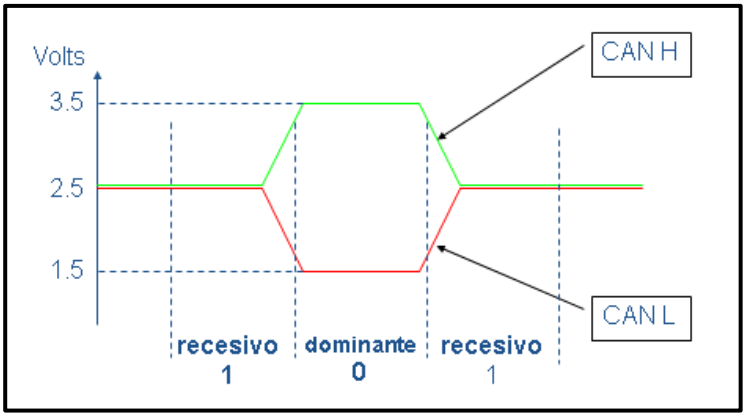



**FIGURA 1.7:** Niveles absolutos de líneas respecto Tierra, estándar (ISO 11898)
**Fuente:** www. dspace.epn.edu.ec/bitstream/15000/9136/1/T11576.pdf, 07-Abr-2012

### 1.7.1.5 Componentes del Medio Físico

#### 1.7.1.5.1 Cableado

"*De acuerdo a los estándares ISO 11898-2 y 3, SAE J2411 e ISO 11992 se emplea cable UTP o STP*" (CALVA CUENCA, 2012, pág. 27), que constituyen la transmisión de línea, y son dos hilos paralelos, trenzados y/o blindados, según los requerimientos electromagnéticos. Los cables se suelen llamar CAN_H en donde el valor de tensión oscila entre 2,75 a 5V y CAN_L entre 0 a 2,25V.

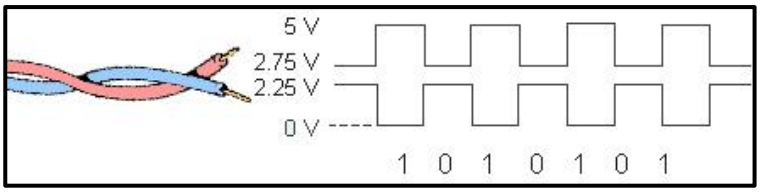



**FIGURA 1.8:** Medio de transmisión eléctrica (par trenzado).
**Fuente:** http://www.arcan.es, 07-Abr-2012

En caso de que se interrumpa la línea CAN High o que se derive a tierra, el sistema trabajará con la señal de CAN Low con respecto a tierra, en el caso de que se interrumpa la línea CAN Low, ocurrirá lo contrario. Esta situación permite que el sistema siga trabajando con uno de los cables cortados o comunicado a tierra.
El sistema queda fuera de servicio únicamente cuando ambos cables se cortan.

#### 1.7.1.5.2 Resistencia de fin de línea

También conocidos como elementos de cierre. "*Son resistencias conectadas a los extremos de los cables H y L. Permiten adecuar el funcionamiento del sistema a diferentes longitudes de cables y número de unidades de control abonadas, ya que impiden fenómenos de reflexión que pueden perturbar el mensaje produciendo errores en la transmisión"* (RUEDA, 2005, pág. 824). Para conseguir esto, estas resistencias deben ser de 120Ω a 1000Ω**,** con una potencia de disipación de 0.25W. Estas resistencias están alojadas en el interior de las unidades de control del sistema por cuestiones de economía y de seguridad de funcionamiento.

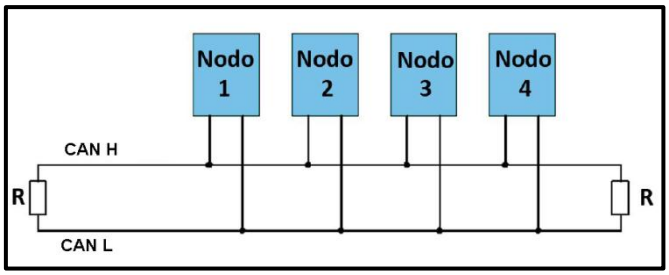



**FIGURA 1.9:** Elementos de Cierre
**Fuente:** www.dspace.epn.edu.ec/bitstream/15000/9136/1/T11576 Visitado 10 Sep. 2012

#### 1.7.1.5.3 Controlador

Permite la conexión física y funcional entre el microprocesador de la unidad de control y el transceptor. El controlador está situado en la unidad de control, trabaja con niveles de tensión muy bajos en el orden de milivoltios y determina la velocidad de transmisión de los mensajes, dependiendo de las necesidades de la transmisión de datos.

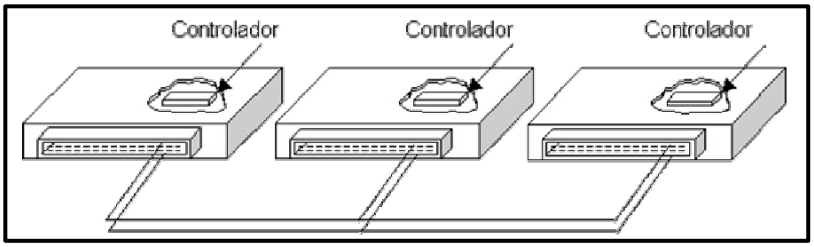



**FIGURA 1.10:** Ubicación del controlador.
**Fuente:** http://www.canbus.galeon.com/electronica/canbus.htm, Visitado 10 Septiembre 2012

#### 1.7.1.5.4 Transceptor (Transmisor-receptor)

El transceptor o transceiver, es un elemento acondicionador de señales en tensión, para lo cual empela amplificadores de señal para que pueda ser utilizada por los controladores, y no modifica el contenido del mensaje. Une el controlador mediante los terminales Rx y Tx con el bus mediante sus terminales CAN_H y CAN_L. El transceptor es un circuito integrado que está situado en cada una de las unidades de control conectadas a la Red, trabaja con una corriente de 500 mA.

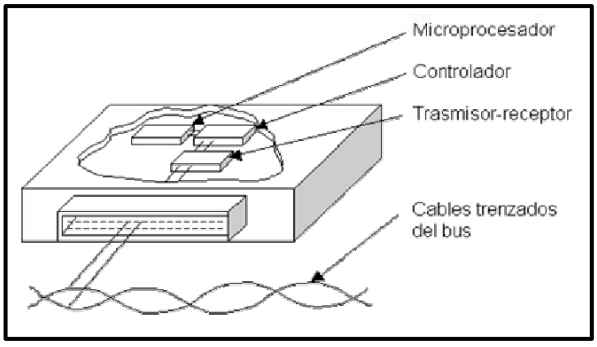



**FIGURA 1.11:** Ubicación del Tranceptor.
**Fuente:** http://www.canbus.galeon.com/electronica/canbus.htm, Visitado 10 Septiembre 2012

La comunicación que provee un transceptor solo puede ser Semiduplex, lo que significa que pueden enviarse señales entre dos terminales en ambos sentidos, pero no simultáneamente, es decir un nodo no podría transmitir si los otros nodos está también transmitiendo porque su transceptor estaría recibiendo en ese momento el mensaje.

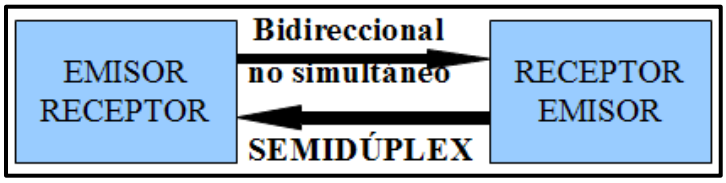



**FIGURA 1.12 :** Sistema de Comunicación Semiduplex del Tranceptor**.**
**Fuente:** http://yeimi-redeslocales.blogspot.com, Visitado 03 febrero 2013

### 1.7.2 Capa de Enlace

La capa de enlace de datos o protocolo CAN de acuerdo al modelo de referencia OSI describe como la información será transmitida entre los diferentes nodos de la RED,

para lo cual realiza el control de acceso al medio, el control lógico para la transmisión de los mensajes, los métodos de confinamiento y detección de errores.

Para un mejor entendimiento de las funciones que realiza la capa de enlace, esta se divide en dos subcapas:

### 1.7.2.1 Control de Enlace Lógico "Logical link Control LLC"

Esta subcapa define las siguientes tareas:

- "***Filtrado de mensajes:*** *el identificador de una trama no especifica el destino de un mensaje solo define su contenido, a través del filtrado, los receptores activos en el sistema determinan si el mensaje deberá ser o no procesado"*. (COSTALES, 2012, págs. 4-5)

- ***"Notificación de sobrecarga:*** *en el caso de que un receptor requiera de un retraso en la transmisión de la siguiente trama, la subcapa LLC emite una trama de sobrecarga. Solo se pueden generar como máximo dos tramas de sobrecarga"* (COSTALES, 2012, págs. 4-5).

- ***Proceso de recuperación:*** *cuando se pierden tramas o se pierde el arbitraje, se presentan errores en la transmisión, el protocolo proporciona la capacidad de retransmisión automática de tramas*". (COSTALES, 2012, págs. 4-5)

### 1.7.2.2 Control de Acceso al Medio "Medium Access Control MAC"

El Control de acceso al medio es la función principal del protocolo CAN, se usa para arbitrar qué nodo tiene el derecho para acceder a transmitir un mensaje en la red. Además impide que dos o más nodos intenten transmitir datos al mismo tiempo.

Las funciones de la subcapa MAC incluyen:

- Encapsulamiento/Desencapsulamiento de datos
- Codificación de tramas (bits de relleno)

- Arbitrar el acceso al medio
- Detección de errores
- Señalización de errores
- Acuses de recibo
- Serialización/Deserialización

La Técnica de acceso al medio empleada por el protocolo CAN se conoce como: Acceso Múltiple por detección de portadora, con detección de colisiones y arbitraje por prioridad de mensaje (CSMA/CD+AMP, Carrier Sense Multiple Access with Collision Detection and Arbitration Message Priority).

El acceso al medio es el conjunto de reglas que definen la forma en que una unidad de control coloca los datos en la red y toma los datos del bus. Una vez que los datos están en la red, los métodos de acceso ayudan a regular el flujo del tráfico de la red.

La técnica se describe de la siguiente forma:

Cada uno de los nodos de la red que quiere transmitir un mensaje comprueba el estado del bus (activo o inactivo) para detectar el tráfico de la red, a esto se le conoce como detección de portadora. Un nodo considera que el bus está libre si el campo de interrupción de la trama en transmisión no ha sido interrumpido por un bit dominante.

Cuando esta condición se cumple dichos nodos transmiten un bit de inicio que es "acceso múltiple". Cada nodo lee el bus bit a bit durante la transmisión de la trama y compara el valor transmitido con el valor recibido; mientras los valores sean idénticos, una vez que el nodo ha trasmitido los datos al bus, ningún otro nodo podrá transmitir datos hasta que éstos hayan llegado a su destino y el bus vuelva a estar libre.

Luego de un período de inactividad, cada nodo de la red, tiene la misma oportunidad de enviar un mensaje. Como la transmisión de datos por parte de los nodos, se da de una forma aleatoria puede darse el caso de que varios nodos empiecen una transmisión de una trama simultáneamente, el conflicto se resuelve por un "Proceso de Arbitraje" no destructivo en el "Campo de Arbitraje" de la trama CAN.

El campo de arbitraje está compuesto por el "Identificador de Trama" y por el "Bit de Solicitud Remota de Transmisión" RTR (Remote Transmission Request) el cual es usado para diferenciar entre una "Trama de Datos" y una "Trama de Solicitud de Datos". En el formato básico de la trama su identificador de trama contiene 11 bits.

El arbitraje no destructivo *se* define mediante una operación idéntica a la de una operación lógica AND, "*Para conseguir un 1 lógico en el bus es necesario que todos los nodos transmitan un 1, mientras que para tener un 0 lógico es suficiente que un solo nodo transmita un 0. Por tanto un nivel 0 es llamado dominante, y un nivel 1 recesivo*". (CAPÍTULO 2, 2012, pág. 2)

De esta forma el bus estará en nivel recesivo mientras se encuentre desocupado (permanece libre solo cuando ningún nodo se encuentra transmitiendo.

.

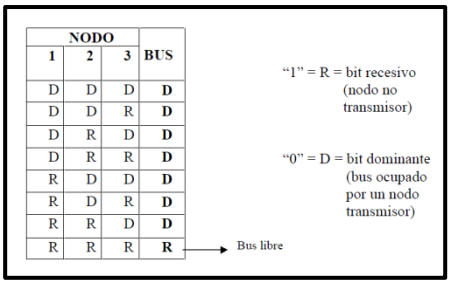



**FIGURA 1.13:** Estados lógicos del bus CAN

**Fuente:** http://repositorio.espe.edu.ec//T-ESPE-012673.pdf, Visitado 08 feb 2012

Para que un nodo inicie una transmisión de una trama debe transmitir un Bit de inicio de trama en estado dominante "SOF" (Start of Frame).

"*Si un nodo quiere iniciar una transmisión de una trama y el bus está ocupado debe esperar a que finalice el "campo de intermisión" es decir esperar un espacio entre tramas de al menos 3 bits recesivos del nodo que está transmitiendo en ese instante, lo que indicará que el bus se encuentra nuevamente libre*". (VERGARA, 2012, pág. 52)

Cuando un nodo transmite un bit recesivo y al mismo tiempo monitorea un bit dominante en el bus detiene su transmisión y se convierte en nodo receptor ya que entiende que un mensaje con mayor prioridad ha accedido al bus y se queda en espera de que el bus se encuentre nuevamente libre.

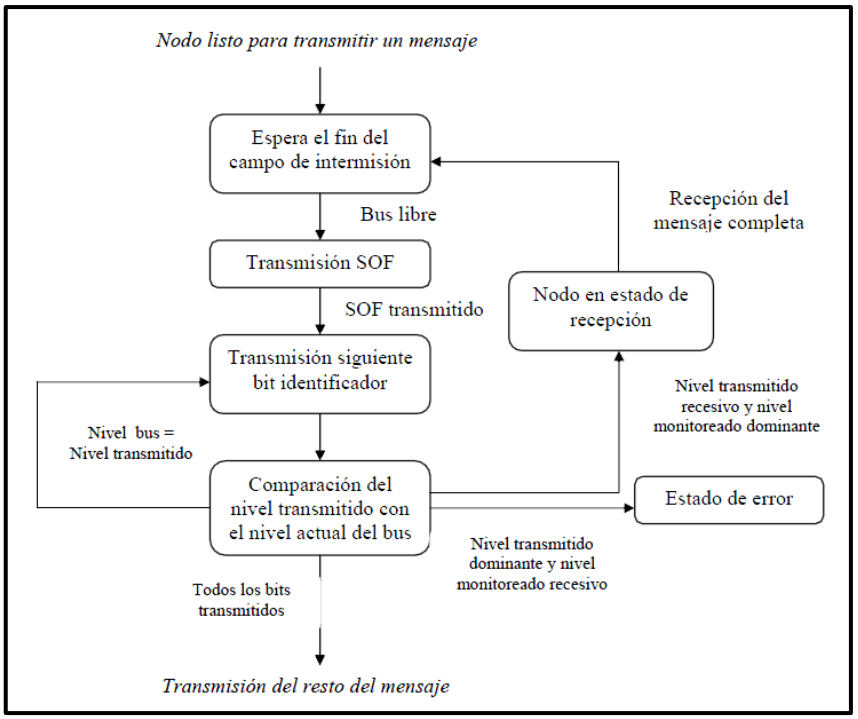



**FIGURA 1.14:** Proceso de arbitraje del bus CAN
**Fuente:** http://repositorio.espe.edu.ec/T-ESPE-012673.pdf, Visitado 08 feb 2012

El nodo que gana el acceso al bus se convierte en el nuevo maestro del sistema hasta que otro nodo gane el acceso al bus, mediante esta fase de arbitraje de bus se asegura que solamente un nodo pueda estar actuando como transmisor en el bus y que a su vez sea el nodo que contiene el mensaje de mayor prioridad.

#### 1.7.2.2.1 Secuencia de un Proceso de Arbitraje del Bus CAN entre tres nodos

Los nodos 1, 2 y 3 inician el proceso de transmisión de tramas de datos al mismo tiempo, durante la transmisión cada nodo emisor comprueba en cada bit dentro del campo de arbitraje si todavía está autorizado para transmitir. El nodo 2 pierde acceso en el bit 5, ocurre lo mismo con el nodo 1 pierde el arbitraje en el bit 2. Bajo estas condiciones el bit RTR "Remote Transmisión Request Frame" define como dominante para la trama de datos al nodo 3 y recesivo para las dos tramas remotas de los dos nodos 1 y 2 resolviendo la colisión; consecuentemente solo el nodo 3 continúa con el proceso de transmisión y recibe acceso al bus al final de la fase de arbitraje y por tanto solo éste nodo puede transmitir su trama sobre el bus.

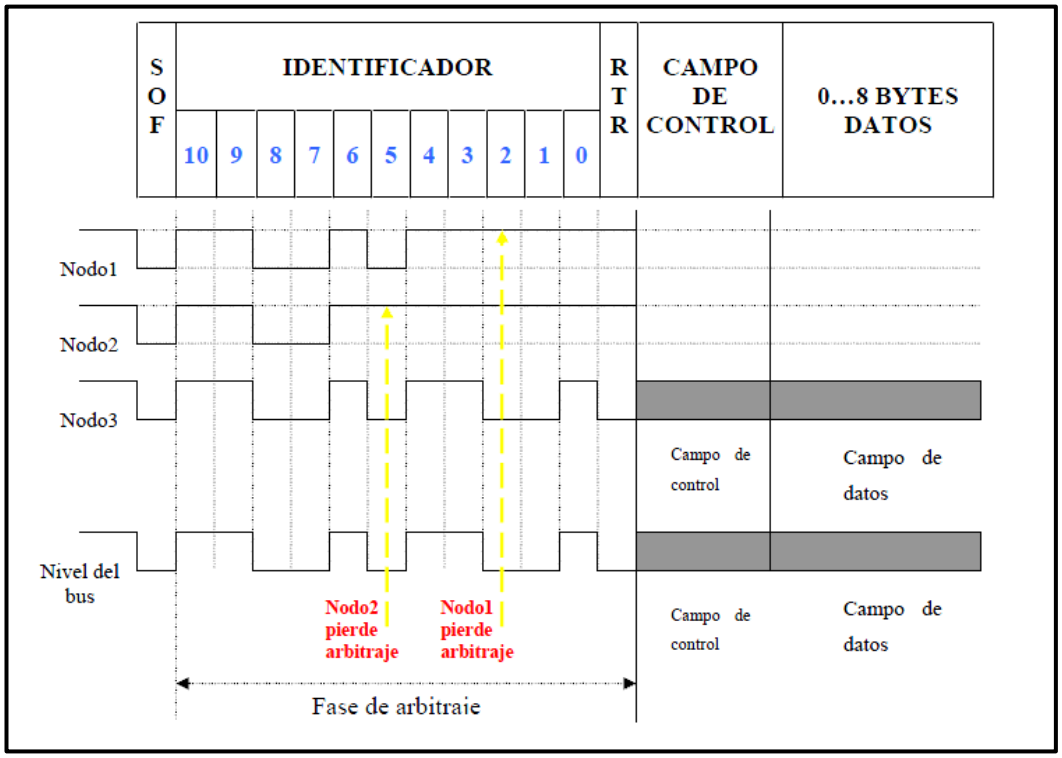



**FIGURA 1.15:** Proceso de arbitraje del bus CAN con 3 nodos simultáneos
**Fuente:** http://repositorio.espe.edu.ec/T-ESPE-012673.pdf, Visitado 12 feb 2012

#### 1.7.2.2.2 Formato de Tramas

El protocolo CAN tiene dos formatos para transmitir datos: El formato de trama estándar, según la especificación CAN2.0A y el formato de trama extendida, según la especificación CAN 2.0B. La diferencia principal entre ellos es la longitud de identificador de mensaje (ID); que en el caso de la trama estándar es de 11 bits (2032 identificadores) y en el caso de la extendida es de 29 bits (más de 536 millones de identificadores).

#### 1.7.2.2.3 Tipos de Trama

El protocolo CAN distingue cuatro tipos de tramas:

##### 1.7.2.2.3.1 Trama de Datos

Es la Información útil que se transmite en "broadcast" a todos los demás nodos, puede incluir entre 0 y 8 Bytes o (0 a 64 bits) de información.
La trama de datos en formato estándar está formada por los siguientes campos de bit:

Inicio de trama (SOF), Campo de arbitraje, Campo de Control, Campo de Datos, Campo CRC, Campo de Acuse de Recibo (ACK) y Campo de Fin de Trama (EOF).

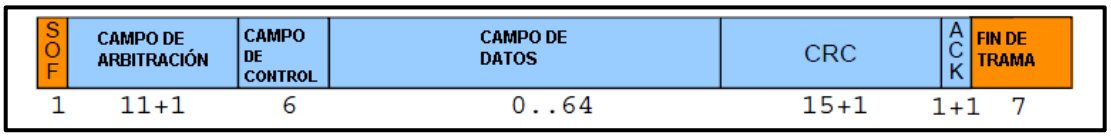



**FIGURA 1.16:** Formato de trama de datos.
**Fuente:** www.dspace.epn.edu.ec/T11576.pdf, Visitado 12 feb 2012

- **Inicio de trama (SOF, Start frame):** Un solo bit dominante, que indica el inicio de la transmisión, su flanco descendente es utilizado por las unidades de control para sincronizarse entre sí.
- **Campo de arbitraje (Arbitration Field):** Formado por 12 bits, Los 11 bit de este campo se emplean como identificador (ID) que permite reconocer a las unidades de control la prioridad del mensaje. Cuanto más bajo sea el valor del identificador más alta es la prioridad del mensaje y 1 bit el RTR (Remote Transmition Request) que indica si el mensaje a transmitir es una trama de datos (RTR=0 dominante) o una trama remota sin datos (RTR=1 recesivo). Una trama de datos siempre tiene una prioridad más alta que una trama remota.
- "***Campo de control (Control Field):*** *Compuesto por 6 bits; 1 bit de IDE (Identifier extension bit) que indica si la trama es estándar (IDE=0) o extendida (IDE=1), 1 bit dominante R0 usado para futuras expansiones del sistema y 4 bits de DLC (Data Lenght Code) que indica el número de bytes a transmitir en el campo de datos*". (CALVA CUENCA, 2012, pág. 37)
- **Campo de datos (data field):** Es la información del mensaje con los datos que la unidad de mando introduce en el Bus. Su tamaño varía entre 0 y 8 bytes (0 y 64bits).
- **Campo de chequeo de errores (CRC, Cyclic redundant code):** Tiene una longitud de 16 bits, 15 bits usados por el receptor para la detección de errores y el último bit (recesivo) que delimita el campo CRC.
- **Campo de aceptación (ACK, aknowledge field)**: Formado por 2 bits ACK-slot y ACK-delimiter que son transmitidos como recesivos. El campo ACK está compuesto por dos bit que son siempre trasmitidos como recesivos (1).

El primero de estos bits se sobre escribe por un bit dominante (0) de reconocimiento transmitido por todas las unidades de control que reciben el mismo CRC, de forma que la unidad de mando que está todavía trasmitiendo reconoce que al menos alguna unidad de mando ha recibido un mensaje escrito correctamente. De no ser así, la unidad de mando trasmisora interpreta que su mensaje presenta un error.

- **Fin de trama (EOF):** Cada trama de datos y remota es delimitada por una secuencia de 7 bits recesivos que indican el final de la trama.
- ***"Espacio entre tramas (IFS):*** *consta de mínimo tres bit recesivos"*. (CALVA CUENCA, 2012, pág. 38)

#### 1.7.2.2.3.2 Trama Remota

Es usada por los nodos receptores para solicitar a un nodo transmisor la transmisión de una trama de datos del mismo identificador de la remota; el nodo que disponga de la información definida por el identificador transmitirá una trama de datos.

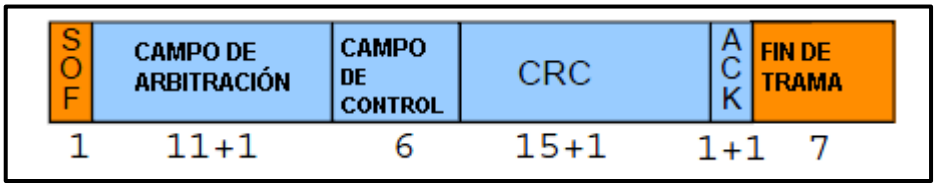



**FIGURA 1.17:** Formato de trama remota.
**Fuente:** www.dspace.epn.edu.ec/T11576.pdf, Visitado 12 feb 2012

El formato es el mismo que la trama de datos pero con el bit RTR recesivo. No incluye datos. El identificador es el del mensaje que se solicita, el campo longitud corresponde a la longitud de ese mensaje.

#### 1.7.2.2.3.3 Trama de Error

Usada para señalizar un error detectado por un nodo del bus ya sea en la transmisión o recepción.
Consiste en dos campos: Bandera de error ("Error Flag") y Delimitador de error.

- **Bandera de Error**

Existen dos formas de representar una bandera de error, la ***Bandera de error activo*** que está formado por seis bits dominantes consecutivos, y la ***Bandera de error pasivo*** que está formado por seis bits recesivos a menos que éstos estén sobrescritos por bits dominantes de otros nodos.

- **Delimitador de Error**

"*Formado por ocho bits recesivos. Después de la transmisión de una bandera de error, el nodo transmite bits recesivos y monitorea el bus hasta que detecta un bit recesivo luego del cual empieza a transmitir siete bits recesivos más. A través de este método el nodo puede determinar si fue el primero en transmitir una bandera de error y con ello detectar una condición de error*". (CALVA CUENCA, 2012, pág. 39)

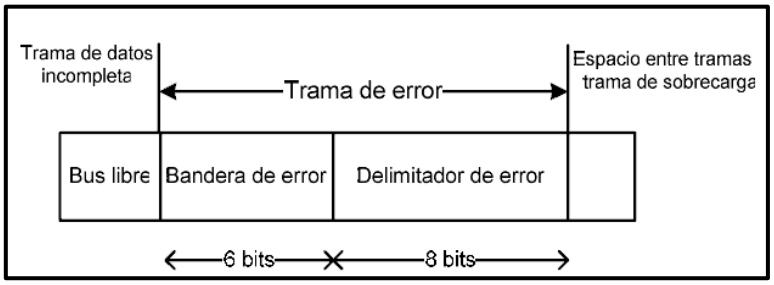



**FIGURA 1.18:** Formato de trama de error.
**Fuente:** www.dspace.epn.edu.ec/T11576.pdf, Visitado 12 feb 2012

### 1.7.2.2.3.4 Trama de Sobrecarga

Es usada para proveer un retardo extra entre dos tramas de datos o de solicitud remota, también sirve para señalar una condición específica de error.
El protocolo CAN permite generar únicamente dos tramas de sobrecarga para retrasar la transmisión de la siguiente trama:

- "*Detección de un bit dominante durante los primeros dos bits del campo de intermisión. La detección de un bit dominante en el tercer bit del campo de intermisión se interpreta como un SOF.*

- *Cuando un receptor detecta un bit dominante en el último bit del campo EOF; o cuando un nodo, receptor o transmisor, detecta un bit dominante en el último bit del delimitador de una trama de error"*. (CALVA CUENCA, 2012, pág. 40)

#### 1.7.2.2.4 Espacio entre Tramas

Separa una trama de la siguiente trama y ha de constar de al menos 3 bits recesivos, esta secuencia de bits se denomina "campo de intermisión". Una vez transcurrida esta secuencia un nodo puede iniciar una nueva transmisión o el bus permanecerá en reposo. Para un nodo en estado de error pasivo la situación es diferente, deberá esperar una secuencia adicional de 8 bits recesivos antes de poder iniciar una transmisión. De esta forma se asegura una ventaja en inicio de transmisión a los nodos en estado activo frente a los nodos en estado pasivo.

#### 1.7.2.2.5 Validación de Tramas

• "***Transmisor:*** *La trama es válida si no existen errores hasta el final del campo EOF.*
• ***Receptor:*** *La trama es válida si no existen errores hasta el siguiente bit después del campo EOF*". (CALVA CUENCA, 2012, pág. 42)

# CAPÍTULO 2

# PROPUESTA DEL FIRMWARE PARA LAS TARJETAS CAN

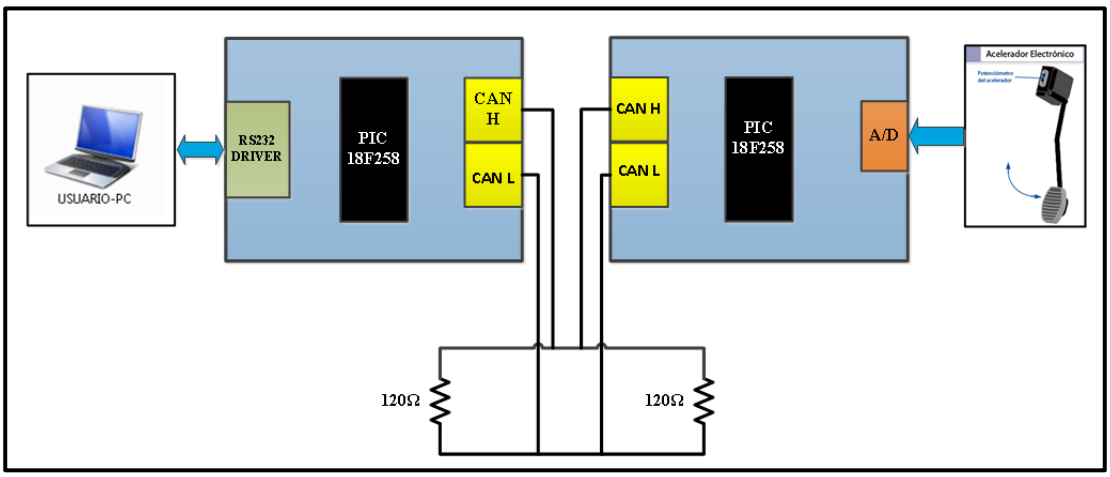



**FIGURA 2.1:** Diagrama de Bloques Tarjetas CAN

El protocolo CAN, en la actualidad es el protocolo más utilizado para aplicaciones en sistemas de comunicación de datos de vehículos, siendo implementado de forma obligatoria desde el año 2008.

Bajo estas condiciones se pretende plantear una propuesta de diseño de una interconexión de dos tarjetas mediante Bus Can, para lo cual se hace uso de dos microcontroladores Pic18F258 que disponen de módulos controladores CAN y dos transceptores microchip MCP2551, así como de componentes electrónicos discretos. Una de las tarjetas actuará como HOST o maestro, y la otra será un nodo esclavo que obtendrá un dato enviado por un dispositivo externo, que en este caso será un potenciómetro que simulara la señal del acelerador electrónico, para enviar luego al dispositivo esclavo mediante bus Can un valor para interactuar con El HOST o maestro el cual leerá el valor y a su vez enviara y graficará la respuesta de este mediante el interfaz gráfico de MATLAB.

## 2.1 Nodo CAN

Una red CAN se compone de una serie de dispositivos conectados mediante un bus serie, denominados nodos.

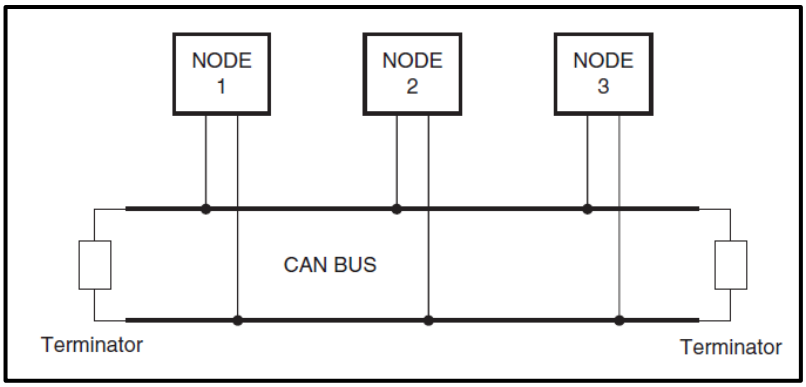



**FIGURA 2.2:** Red CAN.
**Fuente:** http://books.google.com.ec, acceso 14-Dic-2012

La estructura del nodo CAN está compuesta de:

- **Microcontrolador o Host-Processor**: El controlador por sí solo no puede operar una trama CAN, por tanto es necesario de un microcontrolador. El microcontrolador, es el elemento que se encarga de la comprensión de los mensajes recibidos y la elección de los mensajes a enviar.

- **Controlador CAN o CAN-Controller:** Se encarga de la recepción y envío de los mensajes, Arbitraje del Bus y Filtraje de mensajes, Construcción de los formatos del mensaje CAN (generación y decodificación del protocolo). Su función al momento del envío es almacenar la trama a transmitir y enviar los bits de la trama uno a uno. En la recepción se encarga de almacenar datos de la trama bit a bit y una vez completa interrumpe al Host-Processor.
- **Transceptor o Transceiver:** Es el interfaz entre el controlador del protocolo y las líneas físicas del bus. Se encarga de ajustar los niveles lógicos entre el CAN-Controller y el Bus físico.

## 2.2 Tipos de implementación de Nodos CAN

Existen tres tipos de arquitecturas en microcontroladores:

- **Controlador CAN Independiente (Stand-Alone CAN Controller)**

Es una arquitectura formada por:

Un microcontrolador PIC, un controlador CAN como el MCP2510, que introduzca el protocolo CAN, filtros de mensajes, y un transceiver CAN, que es, un transmisor/receptor que puede ser por ejemplo el MCP2551 desarrollado por microchip. Es una arquitectura en donde las capas 1 y 2 del modelo OSI son controladas de forma separada.

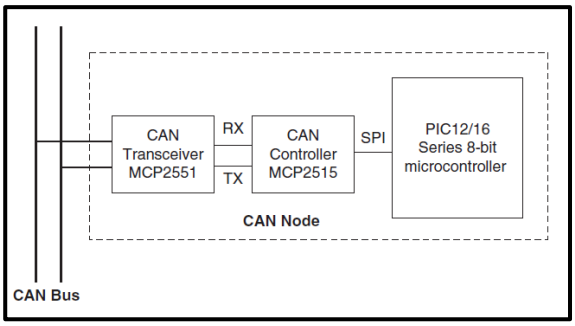



**FIGURA 2.3:** Controlador CAN independiente**.**
**Fuente:** http://www.dspace.epn.edu.ec/T11576.pdf,acceso 06-Dic-2012

- **Controlador CAN Integrado. (Integrated CAN Controller)**

El microcontrolador ya incluye un módulo CAN, es decir que el protocolo y el procesador se encuentran en un solo integrado, el transceiver se sitúa de manera separada y actúa como un controlador de línea, balanceando la señal, es decir, adecuando los niveles de tensión de esta al bus.

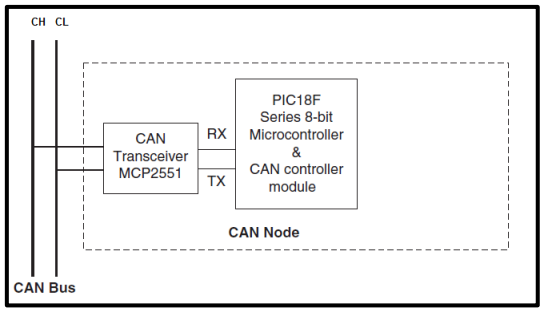



**FIGURA 2.4:** Controlador CAN Integrado
**Fuente:** http://www.dspace.epn.edu.ec/T11576.pdf,acceso 06-Dic-2012

- **Nodo compacto de Chip CAN. (Single-Chip CAN Node)**

El uso de esta arquitectura permite disminuir el tamaño del circuito y problemas de ruido en los circuitos. Es un integrado que incluye en su interior: MICROCONTROLADOR, CONTROLADOR y TRANSCEPTOR, elementos necesarios para llevar cabo las comunicaciones en un entorno CAN.

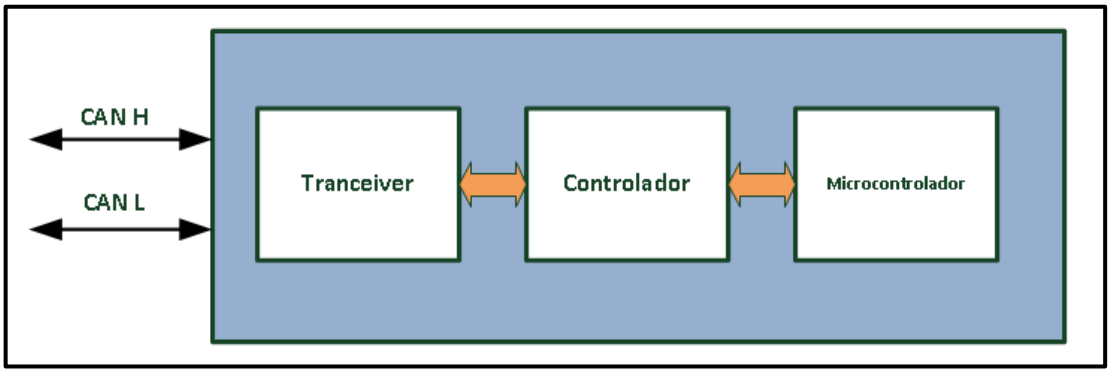



**FIGURA 2.5:** Nodo compacto de Chip CAN.
**Fuente:** http://www.dspace.epn.edu.ec/T11576.pdf,acceso 06-Dic-2012

La comunicación es igual para todas las arquitecturas, la diferencia radica en filtros, capacidad de almacenamiento de las tramas, en el hardware del nodo. Las tres implementaciones incorporan un microcontrolador, ya que este componente representa una herramienta de hardware ideal para el desarrollo de aplicaciones con conexión CAN.

Para el desarrollo del proyecto se empleará la SEGUNDA ARQUITECTURA ya que la separación en dos circuitos integrados es preferible debido a que las funciones que involucran medios físicos y medios lógicos son diferentes, en este caso el controlador CAN tiene la capacidad para filtrar los tipos de mensaje que desee y puede transmitir y recibir mensajes sin ayuda del microcontrolador (debido a que este posee varios buffers), por lo que el controlador le reduce la carga al microcontrolador. El transceiver se situará de manera separada. A más de esto los componentes se encuentran en el mercado local siendo de fácil adquisición y suponen un costo menor.

## 2.3 Microcontrolador Pic

En esta sección se describe el Microcontrolador y su principio de funcionamiento con respecto al módulo del Bus CAN. Los principios son en general aplicables a otros Microcontroladores PIC con módulos CAN.

Un microcontrolador, es capaz de llevar a cabo procesos lógicos que son programados en lenguaje ensamblador o lenguaje C por el usuario, y son introducidos en este a través de un programador. Un microcontrolador integra en una sola pastilla: La Unidad Central de Proceso (UPC), memoria RAM, memoria ROM, EPROM, EEPROM, FLASH, líneas de entrada y salida para comunicarse con el exterior, módulos para el control de periféricos, así como el módulo CAN, entre otros.

*"PIC significa Peripheral Interface Controler es decir un controlador de periféricos*". (SERRA, INTRODUCCIÓN A LA PROGRAMACIÓN Microcontrolador PIC 16F84; Pág1, 2012)

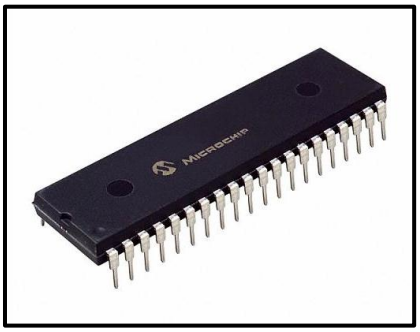


**FIGURA 2.6:** Microcontrolador Microchip PIC18F258**.**
**Fuente:** http://www.didacticaselectronicas.com, acceso 10-Dic-2012

El microcontrolador ejecuta las instrucciones que se encuentran almacenadas en la memoria de programas. Este proceso se genera de manera síncrona con base en un oscilador (OSC), que paso a paso va ejecutando los ciclos de máquina, buscando la instrucción en la memoria, interpretándola y ejecutando las tareas asociadas a la instrucción en particular.

- **Arquitectura Hardware:** posee dos memorias una para datos y otra para instrucciones.

- **Procesador RISC:** posee un conjunto de instrucciones optimizadas para soportar el lenguaje C.
- **Reloj**: Todos los PIC disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia que se utiliza para sincronizar todas las operaciones del sistema. El PIC tiene un oscilador interno incorporado, y en función de la velocidad de trabajo con la que se desee trabajar se utilizará este oscilador.
- **Líneas de entrada/salida (In/Out):** La mayoría de los pines que posee un PIC son de I/O y se destinan a proporcionar el soporte a las señales de entrada, salida y de control.
- **CPU**: Es el elemento que interpreta las instrucciones y procesa los datos en los programas del PIC.
- **Memoria de datos**: Los datos que manejan los programas varían continuamente, y esto exige que la memoria que los contiene debe ser de escritura y de lectura, por lo que la memoria RAM (Random Access Memory) es la más adecuada, aunque sea volátil.

  También se dispone de una memoria de lectura y escritura no volátil, del tipo EEPROM (Electrically-Erasable Programmable Read-Only Memory). De esta forma, un corte en el suministro de la alimentación no ocasiona la pérdida de la información, que está disponible al reiniciarse el programa.
- **Memoria de programas**: El PIC está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. Como este siempre es el mismo, debe estar grabado de forma permanente.

  Existen varios tipos de memoria adecuados para soportar estas funciones, de las cuales en los PIC se utilizan la ROM (Read-Only Memory), OTP (One-Time Programmable) y Flash.
- **Periféricos**: Son todas aquellas unidades a través de las cuales el PIC se comunica con el mundo exterior. En los PIC podemos encontrar por ejemplo: El Módulo ADC (Analog to Digital Converter), comparadores, temporizadores, EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter), USB (Universal Serial Bus): Permite operar con el protocolo USB, CAN (Controller Area Network), CCP/ECCP (Enhanced

Capture/Compare/PWM): Módulo que se puede utilizar como comparador, como capturador o como PWM (Pulse-Width Modulation).

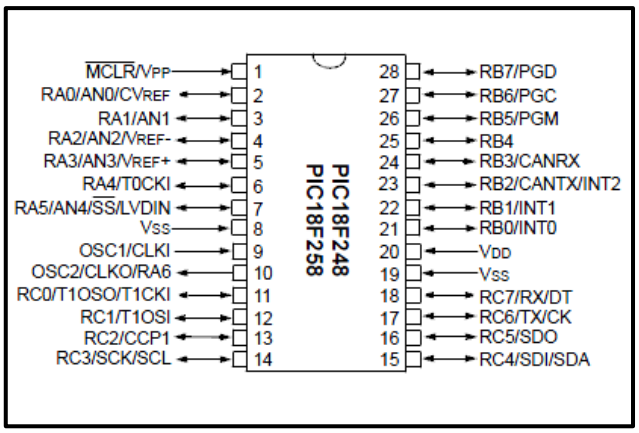



**FIGURA 2.7:** Asignación de Pines del Microcontrolador Microchip PIC18F258**.**
**Fuente**: http://www.microchip.com/datasheetDsPIC18F258, acceso 10-Dic-2012

### 2.3.1 Recursos especiales del PIC micro 18F258

- "*32K flash program memory*
- *1536 bytes RAM data memory*
- *256 bytes EEPROM memory*
- *22 I/O ports*
- *5-channel 10-bit A/D converters*
- *Three timers/counters*
- *Three external interrupt pins*
- *High-current (25mA) sink/source*
- *Capture/compare/PWM module*
- *SPI/I2C module*
- *CAN 2.0A/B module*
- *Power-on reset and power-on timer*
- *Watchdog timer*
- *Priority level interrupts*
- *DC to 40MHz clock input*
- *Wide operating voltage (2.0V to 5.5V)*
- *Power-saving sleep mode*" (IBRAHIM, 2012, pág. 8)

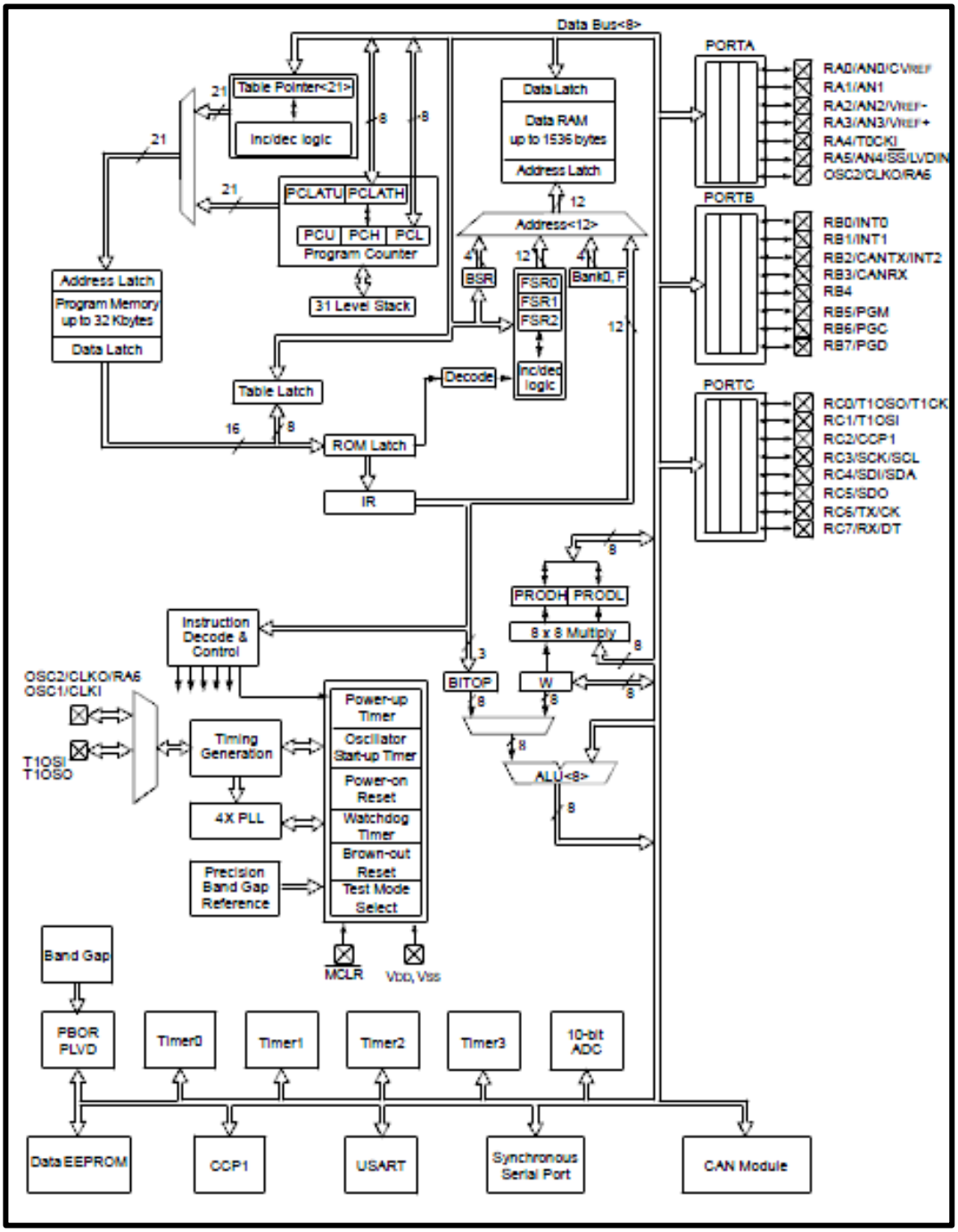



**FIGURA 2.8:** Arquitectura del Microcontrolador Microchip PIC18F258.
**Fuente**: http://www.microchip.com/datasheetPIC18F258, acceso 10-Dic-2012

## 2.4 Módulo CAN del Microcontrolador PIC18F258

Por lo general toda la capa de enlace de datos y la parte superior de la capa física incluyendo la Señalización Física están implementadas en el controlador de comunicación CAN.

CAN se implementa en el microcontrolador como una interfaz serie, utilizado para la comunicación con otros módulos CAN u otros dispositivos del propio microcontrolador.

El módulo CAN está formado por un 'motor de protocolo' (protocol engine) y por un control y almacenamiento de los mensajes. El motor de protocolo CAN maneja todas las funciones encargadas de recibir y transmitir mensajes a través del bus. Además, cualquier mensaje detectado en el bus es sometido a un control de errores y después es filtrado para comprobar si debería ser recibido y almacenado en uno de los registros de recepción, o debería por lo contrario ser rechazado.

| Device | Pins | CAN module | SPI | UART |
|---|---|---|---|---|
| 18F258 | 28 | 1 | 1 | 1 |
| 18F2580 | 28 | 1 | 1 | 1 |
| 18F2680 | 28 | 1 | 1 | 1 |
| 18F4480 | 40/44 | 1 | 1 | 1 |
| 18F8585 | 80 | 1 | 1 | 1 |
| 18F8680 | 80 | 1 | 1 | 1 |

**Tabla 2.3:** Microcontroladores PIC que incluyen módulos CAN.
**Fuente**: http://books.google.com.ec, acceso 17-Dic-2012

## 2.5 Características del Módulo CAN

- *"Implementa el protocolo 'CAN 2.0 A/B' tal y como lo define Bosch, es decir, soporta CAN 1.2, CAN 2.0A, CAN 2.0B Pasivo y CAN 2.0B Activo.*
- *Soporta tramas de datos estándar y extendidas.*
- *Máximo 8 bytes de longitud de datos.*
- *Velocidad de transferencia de datos de hasta 1 Mbit/segundo.*
- *Receptor de doble búfer*

- *6 filtros de aceptación completa (full acceptance filter), tanto para mensajes con identificador estándar como extendido.*
- *2 'máscaras de filtro de aceptación completa' (full acceptance filter masks)*
- *buffers de transmisión con asignación de prioridades específicas y capacidad de aborto (cada búfer puede contener hasta 8 bytes de datos)*
- *Función de 'despertador' (wake-up) programable con filtros paso-bajo integrados.*
- *Modo Loop back programable con operación de 'auto-test' (self-test).*
- *Capacidad de señalización a través de interrupciones por parte de todos los receptores y transmisores de estados de error CAN.*
- *Reloj interno programable.*
- *2 modos de baja potencia: Sleep (dormido) e Idle (libre)"* (PIC18FXX8Data sheet, 2012, pág. 3)

## 2.6 Transceptor (Transceiver)

La parte inferior de la capa 1 o física se encuentra implementada por el transceiver.
El PIC18F258 implementa un controlador CAN integrado, pero el transceptor que convierte las señales TTL del controlador CAN a las de la red CAN y viceversa, no está dentro del propio microcontrolador.
Para poder utilizar el bus CAN del PIC se debe conectar uno de estos transceptores al microcontrolador. El Tranceptor a emplear en nuestro proyecto es el Microchip MCP2551.

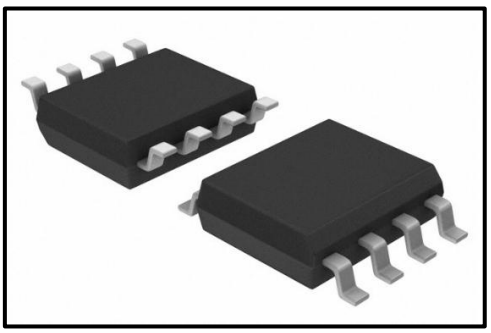


**FIGURA 2.9:** Transceiver SMD Microchip MCP2551
**Fuente:** http://www.digikey.com,acceso 10-Dic-2012

Las características técnicas más significativas del transceiver son las siguientes*:*

- *"Supply Voltage Range:4.5V to 5.5V*
- *N.º de Pines:8*
- *Operating Temperature Range:-40 °C a +85 °C*
- *Número de base:2551*
- *Communication Function:Transceptor CAN*
- *Interfaz de control: CAN*
- *Data Rate Max:1Mbps*
- *Marcador:MCP2551-I/SN*
- *Función CI: Transceptor CAN de alta velocidad*
- *Número genérico CI:2551*
- *Tipo de interfaz: CAN*
- *Line / Bus Driver / Receiver / Transceiver Type: Transceptor CAN*
- *Número de función lógica:2551*
- *N.º of Receivers:1*
- *N.º of Transceivers:1*
- *Operating Temperature Max:85°C*
- *Operating Temperature Min:-40°C*
- *Supply Current:75mA*
- *Tipo de terminación: SMD*
- *Cumple la norma ISO-11898 para la estandarización de la capa física.*
- *Gran inmunidad al ruido"* (Data Sheet MCP2551 & 2, 2012, pág. 4)

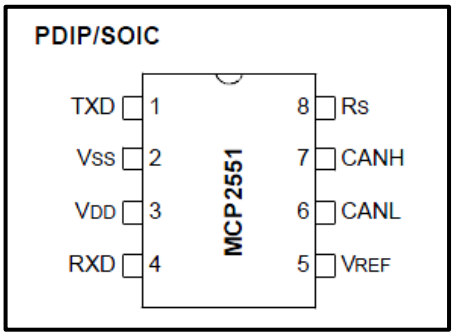



**FIGURA 2.10:** Asignación de Pines del Transceiver Microchip
**Fuente:** http://www.microchip.com/datasheetMCP2551, acceso 10-Dic-2012

### 2.6.1 Conexión a la Red CAN

Las líneas TxD y RxD se conectan directamente a los pines TXCAN y RXCAN del controlador CAN (pines a conectar dentro del microcontrolador). La línea RxD refleja el nivel lógico de la red, un nivel lógico bajo en esta línea significa que en la red CAN hay un nivel recesivo, mientras que un nivel lógico alto refleja un nivel dominante de la red CAN, como ya lo habíamos estudiado en el apartado anterior. A su vez, la línea TxD impone un estado dominante en la red CAN cuando se pone a un nivel lógico alto.

CANH y CANL son las salidas CAN del transceptor, es decir, las que adecuan la señal de entrada del bus al sistema.

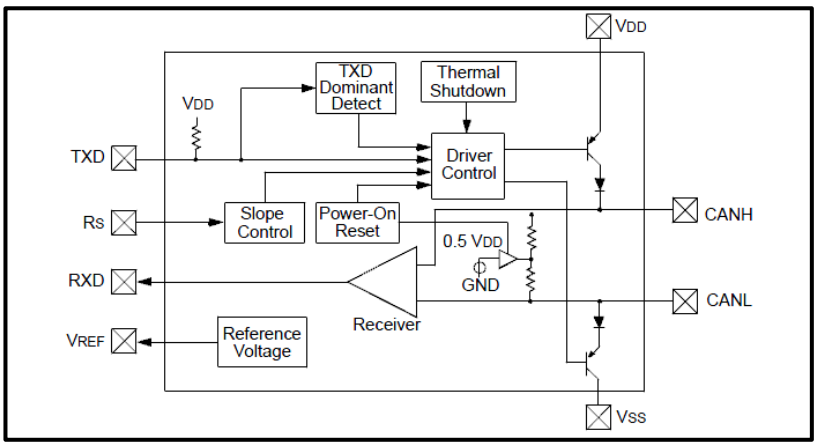



**FIGURA 2.11:** Diagrama de bloques del Transceiver Microchip MCP2551**.**
**Fuente:** http://www.microchip.com/datasheetMCP2551, acceso 10-Dic-2012

La entrada del transceptor VREF proporciona el diferencial de tensión entre CANH y CANL.

El pin Rs permite elegir entre 3 modos de funcionamiento:

- **High Speedmode:** Se consigue conectando el pin Rs a VSS.
- **Slope Control:** Reduce considerablemente las emisiones electromagnéticas disminuyendo los tiempos de subida y de caída de CANH y CANL. Para activarlo, conectaremos una resistencia entre Rs y masa.
- **Standbymode:** Se consigue poniendo a nivel alto el pin Rs. En este modo, el transmisor está apagado, y el receptor funciona ralentizado, es decir, el pin del receptor RXD seguirá operativo, pero de forma más lenta.

## 2.7 Modos de funcionamiento del Controlador CAN

El PIC18F258 tiene 6 modos de operación:

- Modo de inicialización (Initialization mode)
- Modo deshabilitado (Disable mode)
- Modo de funcionamiento normal (Normal operation mode)
- Modo de solo escucha (Listen only mode)
- Modo en bucle (Loopback mode)
- Modo de reconocimiento de errores (Error recognition mode)

### 2.7.1 Modo de inicialización (Initialization mode)

"*En este modo, el módulo no puede transmitir ni recibir datos. Al seleccionarlo, los contadores de error se ponen a 0 y las 'banderas de interrupción' (interruptflags) permanecen inalterables.*" (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005, pág. 12).

### 2.7.2 Modo deshabilitado (Disable mode)

"*Si hay actividad en el bus, el módulo pondrá a 1 los bits del registro WAKIF, mientras que las interrupciones sin ejecutar quedaran pendientes y los contadores de error conservaran su valor.*
*El módulo puede ser programado para aplicar un filtro paso-bajo a la línea de entrada CiRX, mientras el módulo o la CPU estén en modo dormido (Sleep mode). El bit WAKFIL (CiCFG2<14>) es el que activa o desactiva este filtro.* " (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005, pág. 12).

### 2.7.3 Modo de funcionamiento normal (Normal operation mode)

"*El modo de funcionamiento normal, se selecciona con los bits REQOP<2:0> = 000. Al entrar en este modo, el módulo transmitirá y recibirá mensajes del bus a través de los pines CxTX y CxRX*" (ANGULO, Microcontroladores dsPIC. Diseño práctico de aplicaciones, 2006, pág. 10).

### 2.7.4 Modo de solo escucha (Listen only mode)

"*Cuando se activa este modo, el módulo está pasivo. Los buffers de transmisión funcionan como puertos de entrada/salida (I/O), y los pines receptores por su parte, continúan como entradas. El receptor deja de enviar 'banderas de error' (error flags) y 'señales de reconocimiento' (acknowledge signals), y los contadores de error se desactivan. El modo de solo escucha puede ser utilizado para detectar la tasa (velocidad de transmisión) en baudios sobre el bus. Para poder utilizarlo, es necesario que haya al menos dos nodos remotos que se comuniquen el uno con el otro*" (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005, pág. 14)

### 2.7.5 Modo de escucha de todos los mensajes (Listen all messages mode)

"*El módulo es fijado para ignorar todos los errores y poder recibir así cualquier mensaje que se ha enviado a través del bus. En este modo, los datos que están en el buffer de ensamblado de mensajes (message assembly buffer) hasta que ocurre un error, son copiados en el búfer de recepción y pueden ser leídos a través de la interfaz de la CPU*" (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005, pág. 14).

### 2.7.6 Modo loop back (Loop back mode)

"*Cuando el modo loop back se activa, la señal interna de transmisión y la señal interna de recepción se conectan en el límite del módulo, formando una especie de*

*bucle cerrado. Los pines de transmisión y de recepción vuelven a sus funciones de puerto de entrada/salida I/O*" (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005, pág. 15).

## 2.8 Recepción de mensajes

### 2.8.1 Buffers de recepción

El módulo del bus CAN se compone de 3 buffers de recepción. Dos de ellos son el RXB0 y RXB1 y reciben simultáneamente mensajes completos. Existe otro buffers llamado 'Búfer de ensamblado de mensajes' (Message Assembly Buffer o MAB) que se dedica continuamente a escuchar el bus en espera de mensajes entrantes.

### 2.8.2 Filtros de aceptación de mensajes

"*Los filtros y máscaras de aceptación de mensajes se usan para determinar si un mensaje del búfer de ensamblado de mensajes debe ser cargado en alguno de los buffers de recepción.*" (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005, pág. 14).

### 2.8.3 Máscaras de filtro de aceptación de mensajes

Los bits de máscara determinan que bits se han de aplicar en el filtro. Existen 2 máscaras de filtro de aceptación programables, una por cada búfer de recepción. Si un bit de máscara está a cero, entonces este bit será aceptado automáticamente independientemente del bit de filtro.

### 2.8.4 Transmisión de mensajes

El módulo CAN tiene 3 buffers de transmisión, con un tamaño de 14 bytes cada uno. Los ocho primeros bytes son los que puede ocupar como máximo el mensaje

transmitido. El resto son los identificadores estándar, extendidos y otra información de control del mensaje.

### 2.8.5 Errores de transmisión

El modulo CAN detecta los siguientes errores de transmisión:

- Error de ACK
- Error de forma
- Error de bit

Estos errores de transmisión no generan necesariamente una interrupción, pero son indicados por el contador de errores de transmisión.

### 2.8.6 Temporización de bits

Las señales eléctricas del bus sufren alteraciones de distorsión producidas, por las características físicas de la línea, retardos de propagación, y retardos producidos en los propios controladores, también por las posibles fuentes externas de interferencia electromagnética EMC.

Cada controlador CAN depende, normalmente, de un oscilador distinto, lo que provoca diferencias de frecuencia que pueden dar lugar a desfases en el muestreo de tramas de los distintos nodos. Por ello, los controladores siguen un proceso de muestreo y re sincronización orientado a evitar los desajustes debidos a estas eventuales circunstancias.

*"Estos desajustes se manifiestan especialmente en el arbitraje de mensajes, cuando varios nodos transmiten de forma simultánea, y es casi imposible que estén completamente sincronizados; y la frecuencia del oscilador de ambos puede ser ligeramente distinta. Los receptores que se han sincronizado con el primer flanco de bajada detectado, Tienen que ir re sincronizándose sucesivamente a los flancos producidos por el nodo que acabe transmitiendo. CAN utiliza señalización asíncrona y un tipo de codificación NRZ, en la que normalmente 0V representa un '0' lógico, y 5V representa un '1' lógico. Es necesaria, por tanto, una operación de muestreo por parte de los receptores, que se han de re sincronizar periódicamente. Además, la*

*tasa de transmisión (desde menos de 1 kbps hasta 1 Mbps) también es un factor determinante. Todos estos parámetros pueden programarse individualmente y ser ejecutados por el reloj lógico de CAN (Bit Timing Logic*)". (Valencia., 2012, pág. 52)

## 2.9 FIRMWARE DE LAS TARJETAS CAN

Para poder comunicar las Tarjetas CAN se emplea el microcontrolador PIC18F258 de Microchip, y el Firmware como propuesta del mismo se desarrolló con la ayuda del compilador MikroC PRO.

En la siguiente Figura se muestra el nodo de adquisición el cual es el encargado de capturar los datos de un sensor de aceleración (simulado por un potenciómetro pin RA0/AN0) empleando el conversor analógico/digital incorporado del microcontrolador y los transmite a través del bus CAN.

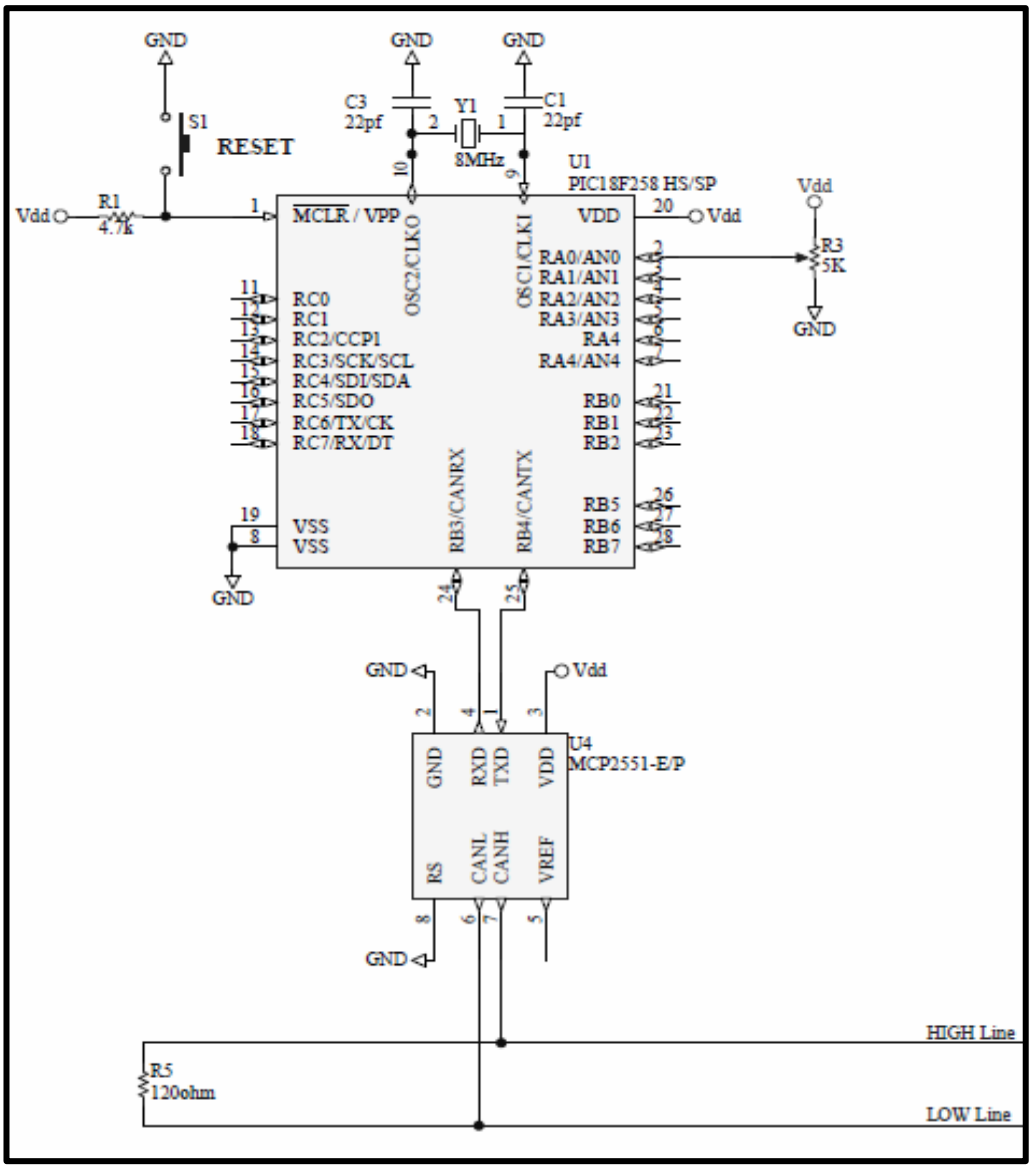



**FIGURA 2.12:** Nodo de Adquisición Esclavo

El nodo maestro de comunicaciones es el encargado de escanear datos en el bus CAN y si recibe datos de algún dispositivo los transmite a un computador por el puerto RS232, y en el computador se podrá observar la variación de aceleración.

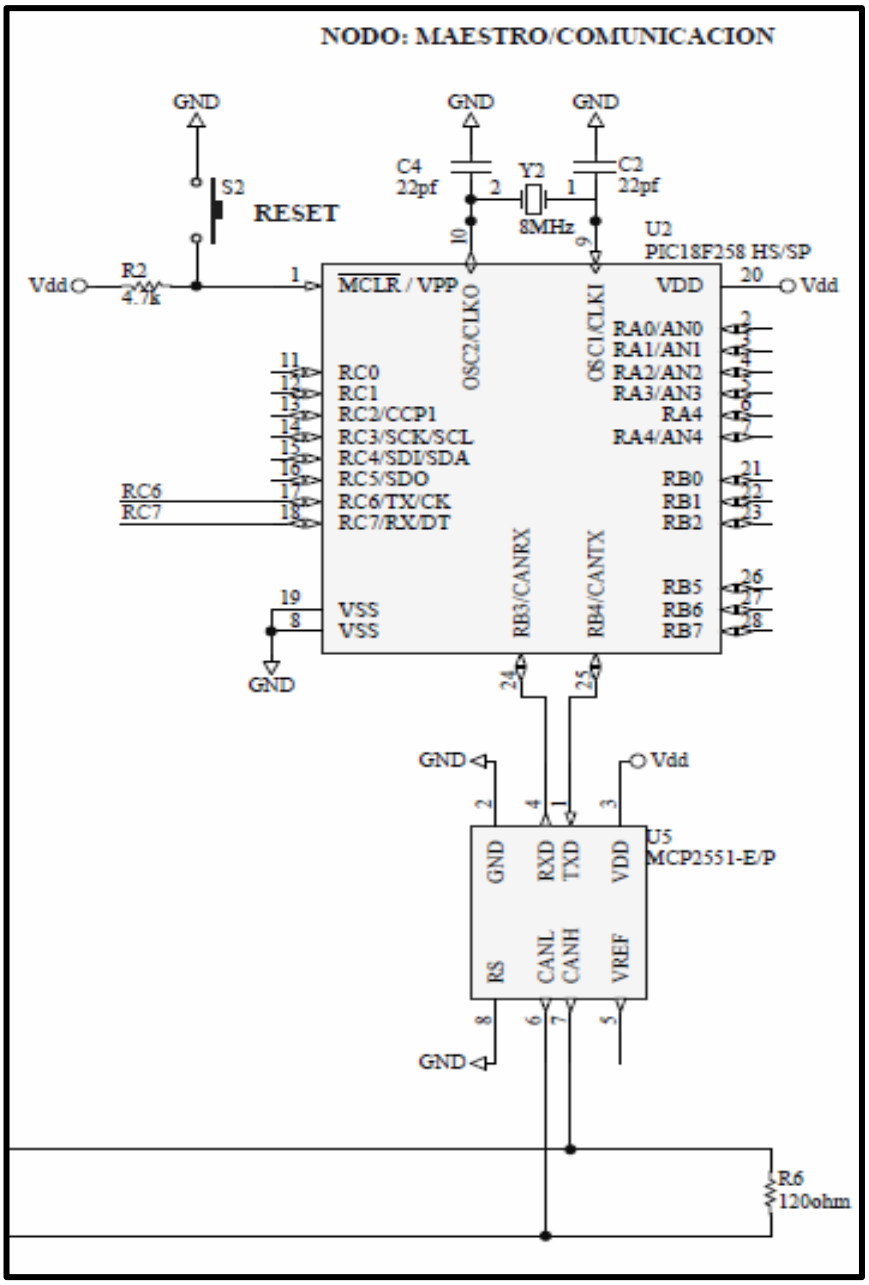



**FIGURA 2.13:** Nodo Maestro**.**

En la siguiente Figura se muestra el adaptador para protocolo RS232 MAX232 utilizado para la comunicación serial entre el nodo maestro y la PC.

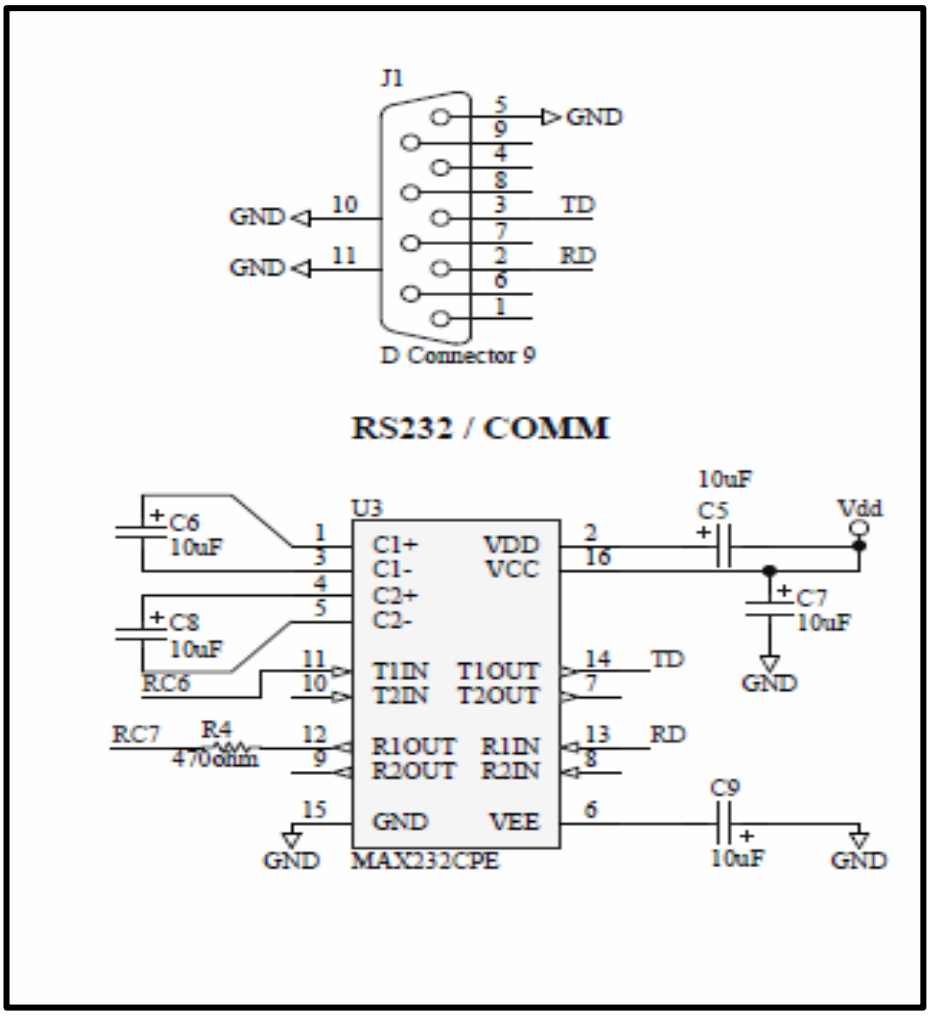



**FIGURA 2.14:** Adaptador para protocolo RS232 MAX232.

### 2.9.1 Firmware del nodo maestro y de comunicación

El puerto RB3/CANRX deberá ser configurado como entrada y el puerto RB2/CANTX configurado como salida, para este proyecto se seleccionó una frecuencia del bus CAN de 100 kbps.

Para el desarrollo del firmware se evaluaron 3 compiladores: ensamblador asm, C18 y MikroC; siendo este último el seleccionado por tener ya escritas las librerías para trabajar con el modulo CAN built-in, ahorrando tiempo en diseño, construcción y aprendizaje.

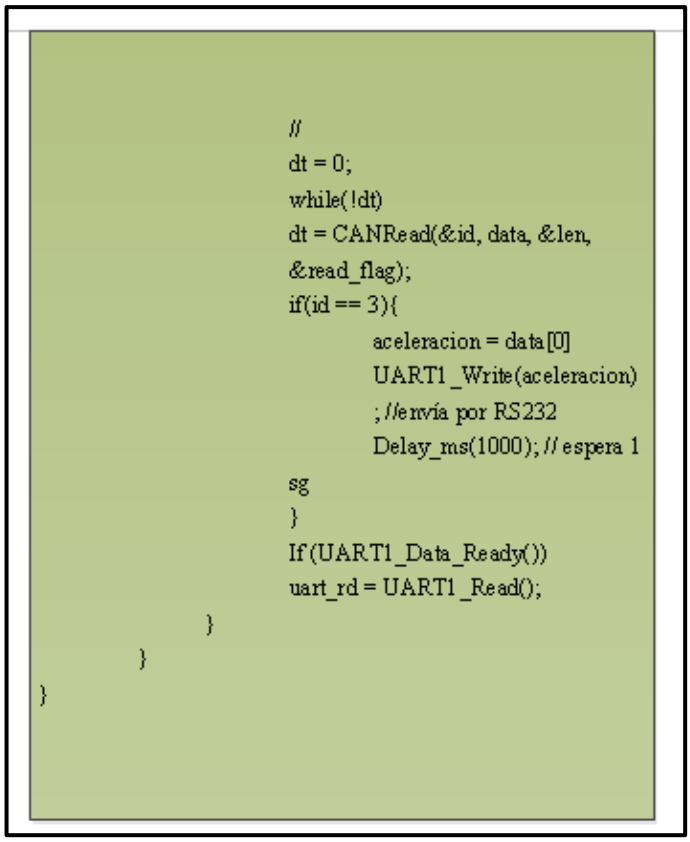
Para inicializar el modulo CAN del PIC18F258 se emplea la función mikroC

```
CANInitialize:
//Parámetros BUS CAN, definidos en el datasheet
SJW = 1;
BRP = 1;
Phase_Seg1 = 6;
Phase_Seg2 = 7;
BRP = 1;
Prop_Seg = 6;

init_flag = CAN_CONFIG_SAMPLE_THRICE &
          CAN_CONFIG_PHSEG2_PRG_ON &
          CAN_CONFIG_STD_MSG &
          CAN_CONFIG_DBL_BUFFER_ON &
          CAN_CONFIG_VALID_XTD_MSG &
          CAN_CONFIG_LINE_FILTER_OFF;

send_flag = CAN_TX_PRIORITY_0 &
          CAN_TX_XTD_FRAME &
          CAN_TX_NO_RTR_FRAME;
          read_flag = 0;
```

```
//
// Inicializa el modulo CAN
//
CANInitialize(SJW, BRP, Phase_Seg1, Phase_Seg2,
Prop_Seg, init_flag);
//
//  Modo configuración modulo CAN
//
CANSetOperationMode(CAN_MODE_CONFIG,
0xFF);
mask = -1; //como poner 0xFFFFFF
//
// Poner todos los bits a 1 MASK1
//
CANSetMask(CAN_MASK_B1, mask,
CAN_CONFIG_XTD_MSG);
//
```

```
// MASK2 bits pone a 1
//
CANSetMask(CAN_MASK_B2, mask,
CAN_CONFIG_XTD_MSG);
//
// ID = 3, filtro B2_F3
//
CANSetFilter(CAN_FILTER_B2_F3,3,CAN_CONFIG_XTD_
MSG);
//
// Modulo CAN en modo normal
//
CANSetOperationMode(CAN_MODE_NORMAL, 0xFF);
//
// Inicializa USART para comunicación RS232 con PC
//
UART1_Init(9600);
```

```
//
// Lee continuamente datos del nodo de aceleracion
//
for(;;)
{
        //Verifica si PC pide dato para transmitir
        If (UART1_Data_Ready()){
                uart_rd = UART1_Read();
                //envia o no dato si = i
                while(uart_rd == "i"){
                        //
                        // Envía un mensaje al nodo de
                        aceleración y pregunta por dato
                        //
                        data[0] = 'A';   //dato a enviar
                        id = 500;                       //
                        Identificador
                        CANWrite(id, data, 1, send_flag); //
                        envía el dato  'A'
                        //
                        // Obtener dato del nodo de
                        aceleración
```



```
                    //
                    dt = 0;
                    while(!dt)
                    dt = CANRead(&id, data, &len,
                    &read_flag);
                    if(id == 3){
                            aceleracion = data[0]
                            UART1_Write(aceleracion)
                            ; //envía por RS232
                            Delay_ms(1000); // espera 1
                    sg
                    }
                    If(UART1_Data_Ready())
                    uart_rd = UART1_Read();
                }
            }
}
```

Donde primero es configurado el modulo, y los valores de las máscaras de los filtros, ambas máscaras mask1 y mask2 son puestas a 1. El filtro 3 para el buffer 2 es puesto con el valor de 3, entonces solo los identificadores que tengan el 3 serán aceptados por el buffer de recepción. El programa se ejecuta indefinidamente en bucle cerrado enviando el dato “A” para adquirir un dato del nodo de aceleración, junto con el identificador id = 500, el nodo de aceleración deberá aceptar un identificador de 500. Si el módulo UART RS232 recibe una “i”, inicia la transmisión de datos a la PC, hasta que reciba una “s” de parar transmisión.

### 2.9.2 Firmware de comunicación o Nodo de Aceleración

El nodo de aceleración es el encargado de adquirir los datos y transmitirlos, la configuración inicial es la misma que en el nodo maestro de comunicación, el filtro receptor deberá contener el 500, de esta manera solo los identificadores con 500 serán aceptados.

```
//
// Configura el convertidor A/D
//
ADCON1 = 0x80;
//
// Parámetros del BUS CAN
//
SJW = 1;
BRP = 1;
Phase_Seg1 = 6;
Phase_Seg2 = 7;
BRP = 1;
Prop_Seg = 6;

init_flag = CAN_CONFIG_SAMPLE_THRICE &
          CAN_CONFIG_PHSEG2_PRG_ON &
          CAN_CONFIG_STD_MSG &
          CAN_CONFIG_DBL_BUFFER_ON &
          CAN_CONFIG_VALID_XTD_MSG &
          CAN_CONFIG_LINE_FILTER_OFF;
```

```
send_flag = CAN_TX_PRIORITY_0 &
CAN_TX_XTD_FRAME &
CAN_TX_NO_RTR_FRAME;

read_flag = 0;

//
// Inicializa modulo CAN
//
CANInitialize(SJW, BRP, Phase_Seg1, Phase_Seg2,
Prop_Seg, init_flag);

//
// ConFigura modulo CAN
//
CANSetOperationMode(CAN_MODE_CONFIG, 0xFF);
mask = -1;
```

```
//
// MASK1 = 1 todos los bits
//
CANSetMask(CAN_MASK_B1, mask,
CAN_CONFIG_XTD_MSG);

//
// MASK2 = 1 todos los bits
//
CANSetMask(CAN_MASK_B2, mask,
CAN_CONFIG_XTD_MSG);

//
// ID = 500, filtro B2_F3
//
CANSetFilter(CAN_FILTER_B2_F3,500,CAN_CONFIG_XT
D_MSG);

//
// Modo de operación normal
//
CANSetOperationMode(CAN_MODE_NORMAL, 0xFF);
```

```
//
// Lee convertidor A/D continuamente
//
for(;;)
{
//
//  Espera hasta que reciba petición de envío
//
dt = 0;
while(!dt) dt = CANRead(&id, data, &len, &read_flag);
if(id == 500 && data[0] == 'A'){
//
// dato valido leer aceleración
//
aceleracion = Adc_Read(0); // lee aceleración
```

```
// envía aceleración a nodo maestro de comunicación
//
data[0] = aceleracion;
id = 3; // Identificador
CANWrite(id, data, 1, send_flag); // envía aceleración
  }
}
```

El puerto serial RS232 puede transmitir un cierto número de caracteres por segundo y no está disponible todo el ancho de banda.
La fórmula para calcular los caracteres por segundo que se pueden transmitir es:

$$Ch/sg = \frac{baud}{11}$$

Para el caso de utilizar 230400 bps

$$Ch/sg = \frac{230400}{11} \approx 20945$$

Se pueden transmitir máximo 20945 caracteres por segundo, por lo que es más que suficiente para nuestro caso ya que el bus CAN funcionara a 100 kbps

### 2.9.3 Script para capturar datos en Matlab

El siguiente script en Matlab sirve para adquirir los valores de aceleración adquiridos por el nodo maestro y poder visualizar en el computador.

```
%UDA – RS232 comunicación
clear all;
clc;
limit =60;
reply = input('Acquisition Time
(sg):','s');
if isempty(reply)
    reply = '0';
end
for i=limit;
  acq(i) = 0;
end
```

```
%RS232
s = serial('COM1','BaudRate',9600);
%crea serial port
s.InputBufferSize = 36000; %configura
buffer entrada
s.ReadAsyncMode = 'continuous';
if (s.Status == 'closed')   %verifica si
está abierto serial port
    fopen(s);   % abre serial port
end

disp('Wait please . . .');
n = str2num(reply);
t = timer('TimerFcn','stat=false;',...
                'StartDelay',n, 'Period',
0.001);
i = 1;
fprintf(s,'i');
start(t)
stat=true;
pause(0.005);
disp('Acquiring data . . .');
```

```
%*********código**********
while(stat==true)
  while ( s.BytesAvailable <= 1)
  end
  %read data
  input = fscanf(s,'%c',1);
  acq(i) = str2num(input);
  %acq(i) = str2num(read_usb(s,4));
  i = i + 1;
end
fprintf(s,'s');
```

```
while (s.BytesAvailable >= 1)
  input = fscanf(s,'%c',1);
  acq(i) = str2num(input);
  i = i + 1;
end

%monitor for your data
disp('Processing data . . .');
figure(1)
  plot(1:length(acq), acq);
  ylabel('Amplitud');
  xlabel('Muestras');
  drawnow

delete(t);
if (s.Status == 'open')%verifica si está
abierto serial port
    fclose(s);   %cierra serial port
end
delete (s);
```

# CONCLUSIONES Y RECOMENDACIONES

- Originalmente el protocolo CAN fue desarrollado por Bosh para aplicaciones en la industria automotriz, pero debido a sus características de robustez, CAN fue adoptado para aplicaciones industriales y de control.
- CAN es un protocolo de comunicaciones basado en una arquitectura de bus para transferencia de mensajes en ambientes distribuidos.
- Una de las características del protocolo es su robustez debido a que posee una alta inmunidad al ruido.
- Existen diferentes sistemas de multiplexado, tales como J1850 (Protocolo normalizado según SAE) que utiliza Chrysler, GM y Ford, Abus Bus de Volkswagen y VAN (Vehicule Area Network) de Renault.
- Un vehículo actual cuenta con alrededor de dos kilómetros de cable que es igual a 50Kg de peso, la solución a este problema es la multiplexación.
- El multiplexado es una aplicación a reducir una determinada cantidad de cable en el vehículo.
- Una red CAN se compone de una serie de dispositivos conectados mediante un bus serie, denominados nodos. La forma de transmisión de dichos nodos es por broadcast, por lo que el nodo debe tener la capacidad de comprender cuales son mensajes útiles.
- En formato de mensajes CAN2.0A del protocolo CAN, es posible tener hasta 2.048 mensajes, mientras que en el formato CAN2.0B, es posible tener hasta 537 millones de mensajes, es quiere decir que no hay limitación de mensajes.
- La separación en dos circuitos integrados se da debido a que las funciones que involucran medios físicos y medios lógicos son tan diferentes que las tecnologías actuales no han sido capaces de integrar su funcionalidad entera en un solo integrado.
- Las limitaciones de las velocidades vienen marcadas por el transceiver que se utilice y por la longitud total es decir de extremo a extremo de la red, según el fabricante el transceiver MCP2551 de Microchip tiene una velocidad máxima de 1Mbps. Así que por este aspecto en nuestra propuesta la comunicación no se verá afectada

- Los microcontroladores PIC suponen un avance significativo en la materia de dispositivos digitales empleados en el control de sistemas. Su reducido tamaño así como sus elevadas prestaciones lo convierten en un elemento indispensable en el campo de los microcontroladores.
- El programa se puede escribir en lenguajes de programación como: "Assembler" (ensamblador) o en C. Para que el PIC pueda entender lo que se escribió es necesario traducir el programa a lenguaje de máquina (1's y 0's), esto se hace con un compilador.
- Aumentar la frecuencia del reloj implica disminuir el tiempo de ejecución de las instrucciones, pero compromete un incremento de la temperatura.
- Se puede decir que el firmware funciona como el nexo entre las instrucciones que llegan al dispositivo desde el exterior y sus diversas partes electrónicas.
- En el protocolo CAN la estructura más importante es la estructura de datos, la cual es manejada enteramente por el usuario, y está formada por 11 bits y si es la extendida 29 bits.
- El controlador CAN es el encargado de las tramas de sobrecarga.
- La Velocidad es un limitante en cuanto a longitud de la red, aunque en este trabajo no influye como para tenerlo en cuenta. Según el estándar, la velocidad máxima que puede alcanzar el bus CAN es de 1Mbps y esta se puede alcanzar hasta con una longitud de red de 40 metros.
- Para fines prácticos y teniendo en cuenta que la distancia es pequeña entre las tarjetas CAN, la velocidad máxima que se puede elegir es de 100 Kbps. Se recomienda esto debido a que el bus estará libre y no soportará elevado tráfico.
- Para la propuesta del firmware se evalúo la tecnología del PIC18F450, pero se descartó el uso de este, debido a que no posee el módulo integrado CAN, esto habría llevado a desarrollar un circuito más complejo. Otra alternativa fue el uso del dsPIC30F4013 pero es un PIC con demasiados recursos por lo que con el uso de este, quedarían demasiados pines sin uso.
- Para programar el PIC se consideró tres compiladores: C18, MPLab, y MikroC; eligiéndose el último debido a que este ya posee un gran número de librerías, lo que facilita el diseño del proyecto ya que las subrutinas están diseñadas, mientras que en MPLab que maneja lenguaje ensamblador se

debe programar cada línea del programa o como se le conoce en el argot de programación “a golpe de instrucciones”.

- La finalidad de las resistencias es evitar que los datos transmitidos sean devueltos en forma de eco desde los extremos de los cables y que se falsifiquen los datos. Se recomienda siempre colocar la resistencia de fin de línea para evitar que la señal transmitida se falsee o distorsione y genere interferencias en forma de onda en la trama o paquetes de datos enviados entre nodos.
- Para trabajar con el módulo CAN, se recomienda configurar los filtros de aceptancia de tal forma que siempre deben coincidir con el ID o identificador.
- Se recomienda siempre colocar lo más cerca el transceiver al PIC micro para no meter ruido a la circuitería.
- Siempre se debe trenzar los cables para reducir al máximo las interferencias que pueden ocasionar circuitos externas.
- En el hardware se recomienda colocar también los cristales cerca del micro con objeto de no introducir ruido al PIC. Se recomienda que los condensadores elegidos vayan del pin OSC1 ó OSC2 a 0V (GND). Para el diseño de la placa se recomienda una frecuencia de oscilación 20 MHz.
- Se recomienda optimizar los recursos del PIC. Todas las acciones a realizar estarán en las funciones de las interrupciones. La idea es que las interrupciones deben quitar carga de trabajo al procesador para que pueda realizar otros procesos. Si no se implementaran estas interrupciones se tendría que estar preguntando en todo momento al procesador por algún dato en concreto. De esta manera se deja libre al procesador, y solo cuando se produce la circunstancia por la que ha sido programada la interrupción se toma el control de este.
- Es muy importante conectar las tierras entre los dispositivos para evitar problemas de masa flotante en la alimentación.

# BIBLIOGRAFIA

## REFERENCIAS BIBLIOGRÁFICAS


- AGUILAR Sergio, ASTUDILLO Pedro, "Diseño de una Caja Negra", Asesor de Tesis, Ing. Paúl Tenesaca A. 2010
- BOSCH Robert. "Los Sensores en el Automóvil." Buenos Aires Argentina 2002.
- BOSCH Robert. "Sistemas para la estabilización del automóvil". Buenos Aires Argentina 2005.
- CAMPOS Guillermo Ing. "PEUGEOT. ABS, TCS, ESP" 2011
- CAN BUS Exercise Book, *"Embedded C Language for the PIC MCU,* DEVELOPMENT KIT", CCS, Wisconsin (2006)
- DOGAN, Ibrahim, "Advanced PIC microcontroller projects in C: From USB to ZIGBEE with the 18F Series" United States of America. 2008.
- ERRASQUIN Jorge Ing. ABS. Buenos Aires, Argentina. Curso de Autotrónica UBA Oct. 2006.
- GIL Hermógenes. "La Electrónica en el automóvil" 2011.
- HUANG, Han-Way, "Pic Microcontroller: An Introduction to Software and Hardware Interfacing", United States of America. 2005.
- PEREZ Leonel Ing. Cuenca, Ecuador. Clases de microcontroladores. UDA 2005-2006.
- SANTANDER Jesús. Manual Técnico de Fuel Injection. 2012.
- SAMBRANO Daniel Ing. ABS. Buenos Aires, Argentina. Curso de Autotrónica UBA Nov.2006.
- WINNER H. Regulación adaptiva de la velocidad de marcha. Buenos Aires Argentina 2005.
- TENESACA Paúl. Diseño e Implementación de un emulador para el sistema de diagnóstico de ESP con microcontrolador. 2007
- TEDESCO, Carlos Francisco, "Ascensores electrónicos y variadores de velocidad". Buenos Aires Argentina 2011.

**REFERENCIAS ELECTRÓNICAS**


CABLES Y CONECTORES. (2012, Diciembre 28). Retrieved from http://www.angelfire.com/wi/ociosonet/15.html

LOS FILTROS PASIVOS DE PRIMER ORDEN. . (2012, Agosto 28). Retrieved from http://www.terra.es/personal2/equipos2/filtros.htm

THE FREE DICTIONARY BY FARLEX. (2012, Diciembre 28). Retrieved from http://www.es.thefreedictionary.com/bit

ALEPUZ, S. (2012, Diciembre 12). *Qué es CAN y cómo funciona el Módulo del microcontrolador Dspic30f4013pdf; Pág. 36.* Retrieved from http://www grupos.emagister.com

ANGULO, J. M. (2005). *MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones.* Buenos Aires: Ediciones Paraninfo. S.A.

ANGULO, J. M. (2006). *Microcontroladores dsPIC. Diseño práctico de aplicaciones.* Buenos Aires: McGraw-Hill.

BOSCH, C. (2012, Diciembre 20). *CAN Specification Version 2.0 can2spec.pdf.* Retrieved from www-micro.deis.unibo.it/~magagni/can2spec.pdf

CALVA CUENCA, J. (2012, Septiembre 04). *Repositorio Digital EPN.* Retrieved from http://bibdigital.epn.edu.ec/handle/15000/2055

CAN in Automation (CiA). (2012, Abril 07). *CAN in Automation*. Retrieved from http://www.can-cai.de/index.php?id=161

CAPÍTULO 2, P. C. (2012, Noviembre 06). *Repositario digital EPN.* Retrieved from dspace.epn.edu.ec/bitstream/15000/8689/9/T10504CAP2.pdf

COSTALES, M. ( 2012, Febrero 08 ). *ANÁLISIS DEL PROTOCOLO CAN E IMPLEMENTACION DE UN PROTOTIPO DE NODOS INTERCONECTADOS POR UN BUS DE COMUNICAIONES CAN.* Retrieved from http:// bibdigital.epn.edu.ec/handle/15000/4194 Visitado

DATA SHEET MCP2551, & 2, P. (2012, Diciembre 14). *Microchip.* Retrieved from http://www.microchip.com/datasheet MCP2551

ECHEVERÍA LÍBANO, R. (2012, Noviembre 14). *BREVES APUNTES DE MATLAB.* Retrieved from Dpto. Ecuaciones Diferenciales y Análisis Numérico: http://personal.us.es/echevarria/documentos/IntroduccionMATLAB.pdf

ENCINAS, D. (2012, Diciembre 15). *Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores.pdf.* Retrieved from ; http://www sedici.unlp.edu.ar.

IBRAHIM, D. (2012). *Advanced PIC microcontroller projects in C: From USB to ZIGBEE with the 18F Series,.* Oregon: Elsevier.

KASCHEL C, H. (2012, Febrero 08). *Cabierta.Uchile.* Retrieved from ANÁLISIS DE LA CAPA FÍSICA DEL BUS DE CAMPO CAN: http://cabierta.uchile.cl/revista/25/articulos/pdf/paper1.pdf

LOPEZ, J. A. (2012, Octubre 07). *Departamento de Ingenieria eléctrica, electrónica y automática.* Retrieved from deeea.urv.cat/public/PROPOSTES/pub/pdf/561pub.pdf - España

MICROCHIP. (2012, Dkiciembre 14). *Language Tool Libraries.* Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/16Bit_Language_Tool_ Libraries_51456d.pdf

NATIONAL INSTRUMENTS, p. 1. (2012, Febrero 08). *National Instruments NI Developer Zone.* Retrieved from http//zone.ni.com/devzone/cda/tut/p/id/7183

Net, C. C. (2012, Septiembre 10). *Electronica - FRBA - Universidad Tecnologica Nacional.* Retrieved from www.electron.frba.utn.edu.ar/materias/95.../Cap8_2009_CAN.pdf

PACHECO, J. (2012, Diciembre 08). *Capítulo 4: Desarrollo de un nodo CAN,Pag3.* Retrieved from http://www.catarina.udlap.capitulo4.pdf;

PEREZ, L. (2006). *Microcontroladores I.* Cuenca: UDA.

PIC18FXX8Data sheet, ,. 3. (2012, Diciembre 15). *Microchip.* Retrieved from http://www.microchip.com/datasheet PIC18F258

RUEDA, J. (2005). RED CAN. In *Manual Técnico de Fuel Injection* (p. 824). Bogotá: Diseli.

SERRA, E. (2012, Diciembre 14). *INTRODUCCIÓN A LA PROGRAMACIÓN Microcontrolador PIC 16F84; Pág1.* Retrieved from ; www.sputnik.epsj23.net/~eserra/elect/pics/pic16f84.html

SERRA, E. (2012, Noviembre 14). *INTRODUCCIÓN A LA PROGRAMACIÓN Microcontrolador PIC 16F84;Pág1.* Retrieved from http://www.sputnik.epsj23.net/~eserra/elect/pics/pic16f84.html

TENESACA, P. (2005). *Control de ESP.* Cuenca: UDA.

UNIVERSIDAD DE OVIEDO, T. 6. (2012, Septiembre 03). *isa.uniovi.es.* Retrieved from http://wwwisa.uniovi.es/sirgo/doctorado/tema6.pdf

VERGARA, I. (2012, Diciembre 06). *Análisis e Investigación sobre las características Fundamentales del Protocolo CANpdf, Pág. 24.* Retrieved from http://www repositorio.espe.edu.ec,.

ZAPATA Vaca, A. F. (2013, Enero 11). *Repositorio Digital ESPE.* Retrieved from http://repositorio.espe.edu.ec/handle/21000/2895

# GLOSARIO DE TERMINOS

**BIT:** Binary digit, unidad de información más pequeña. (The free dictionary by Farlex, 2012)

**BUS:** Es el medio empleado para transportar y distribuir datos entre los distintos dispositivos.

**BYTE:** Unidad de información con dirección, de ocho bits consecutivos. (The free dictionary by Farlex, 2012)

**CABLE STP:** Se refiere a la cantidad de aislamiento alrededor de un conjunto de cables y, por lo tanto, a su inmunidad al ruido, su impedancia es de 150 Ohmios. (Cables y Conectores, 2012)

**CÓDIGO FUENTE:** Es el programa escrito en lenguaje ensamblador al que se le asigna una extensión (*.asm).

**CÓDIGO MÁQUINA:** Fichero proporcionado por el programa ensamblador al que se le asigna una extensión (*.hex).

**CONTROLADOR CAN:** Gestiona y procesa datos a enviar que recibe del microcontrolador, por la línea de I/O del bus.

**DATA FRAME:** Trama de datos del protocolo de datos.

**EUSART**: Utilizado por el protocolo RS232, es un interfaz de entrada salida serie. Con este módulo se puede transformar los datos en serie a paralelo

**FILTRO PASA BAJO:** Un filtro pasa bajo corresponde a un filtro caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas. (Los filtros pasivos de primer orden. , 2012)

**FIRMWARE:** El firmware se considera un hibrido entre el Software y el Hardware, al estar integrado en la parte electrónica, ya que pertenece al Hardware, pero a su vez también es Software ya que proporciona lógica y se establece en un lenguaje de programación, en este caso “C”. (SERRA, INTRODUCCIÓN A LA PROGRAMACIÓN Microcontrolador PIC 16F84; Pág1, 2012)

**INTERFAZ:** Es el medio de comunicación entre dos sistemas diferentes.

**LATENCIA:** Suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red. (ANGULO, Microcontroladores dsPIC. Diseño práctico de aplicaciones, 2006)

**LENGUAJE C:** El lenguaje C es sin duda el más apropiado para la programación de sistemas, pudiendo sustituir al ensamblador en muchos casos. Sin embargo, hay

ocasiones en que es necesario acceder a un nivel más bajo por razones de operatividad e incluso de necesidad (programas que economicen memoria, algoritmos rápidos para operaciones críticas, etc.).

**LENGUAJE DE MAQUINA:** El único lenguaje que entienden los microcontroladores, formado por ceros y unos del sistema binario.

**LENGUAJE ENSAMBLADOR:** Expresa las instrucciones de una forma más natural al hombre a la vez que muy cercana al Microcontrolador, ya que cada una de sus instrucciones se corresponde con otra en código máquina. (IBRAHIM, 2012)

**MIKROC PRO:** Para crear un dispositivo controlado por un microcontrolador, se necesita un programa para compilar y un dispositivo para transmitir el código de la PC al PIC.

**MODELO OSI:** El modelo de referencia utilizado en la actualidad para especificaciones protocolares es el modelo OSI desarrollado por la Organización de Estandarización Internacional (ISO) para apoyar el desarrollo y aplicación de protocolos de comunicación abiertos. (ANGULO, MICROCONTROLADORES AVANZADOS dsPIC. Controladores digitales de señales. Arquitectura, programación y aplicaciones, 2005)

**MULTIMAESTRO:** En redes multimaestro, la técnica de acceso al medio es importante ya que todo nodo activo tiene los derechos para controlar la red y acaparar los recursos. (ALEPUZ, 2012)

**PRIORIDAD:** Orden de sucesión de los mensajes a enviar, según su relevancia en cuanto a la seguridad y su evaluación con respecto al tiempo.

**PROM:** Memoria de sólo lectura programable.

**PROTOCOLO DE DATOS:** Mensaje que se transmite; de estructura estándar en siete campos.

**RAM:** Memoria de acceso aleatorio.

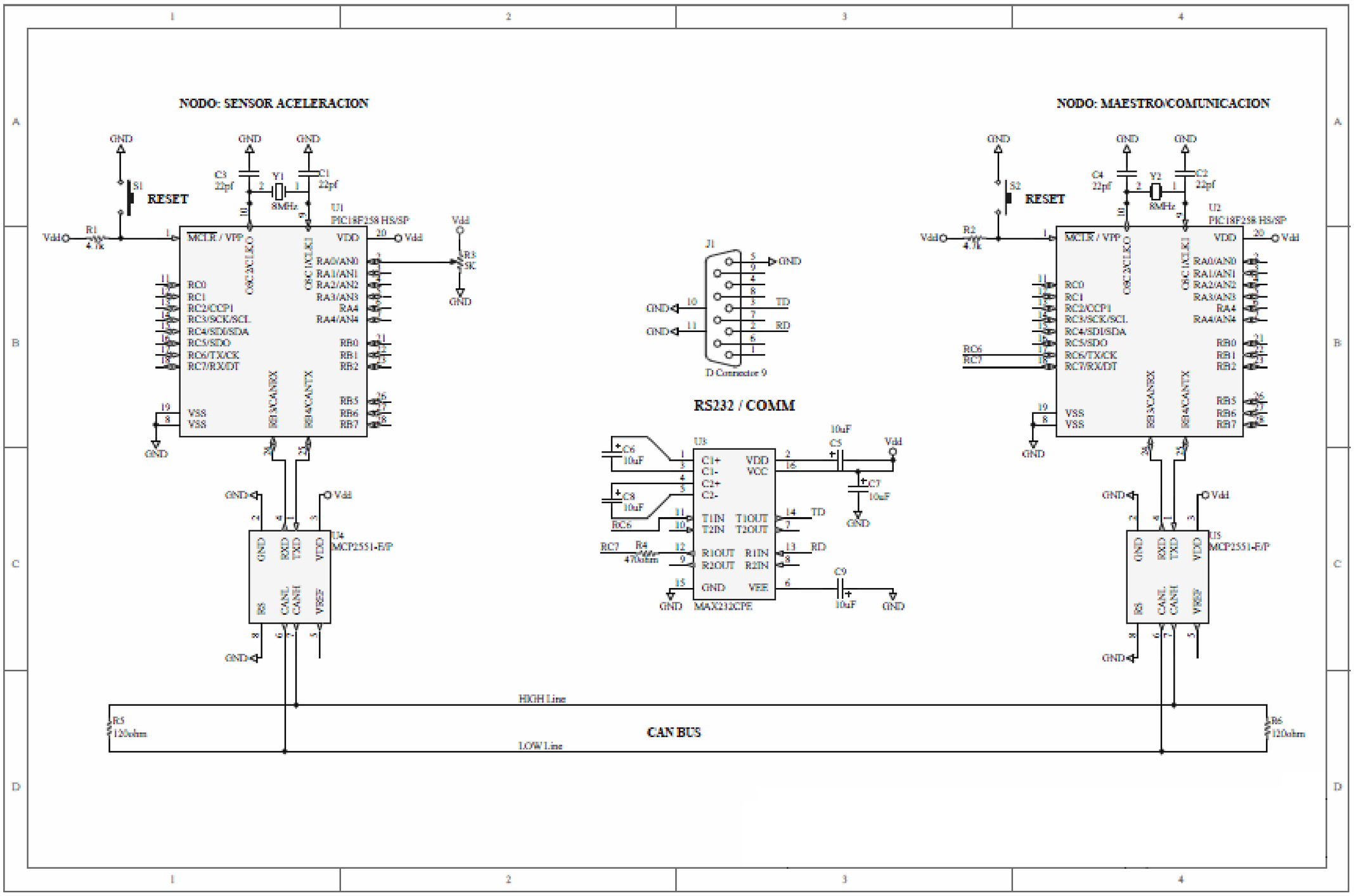
**ROM:** Memoria de sólo de lectura.

**SCRIPT:** Conjunto de comandos escritos en un fichero, que se pueden ejecutar con una única orden. (ECHEVERÍA LÍBANO, 2012)

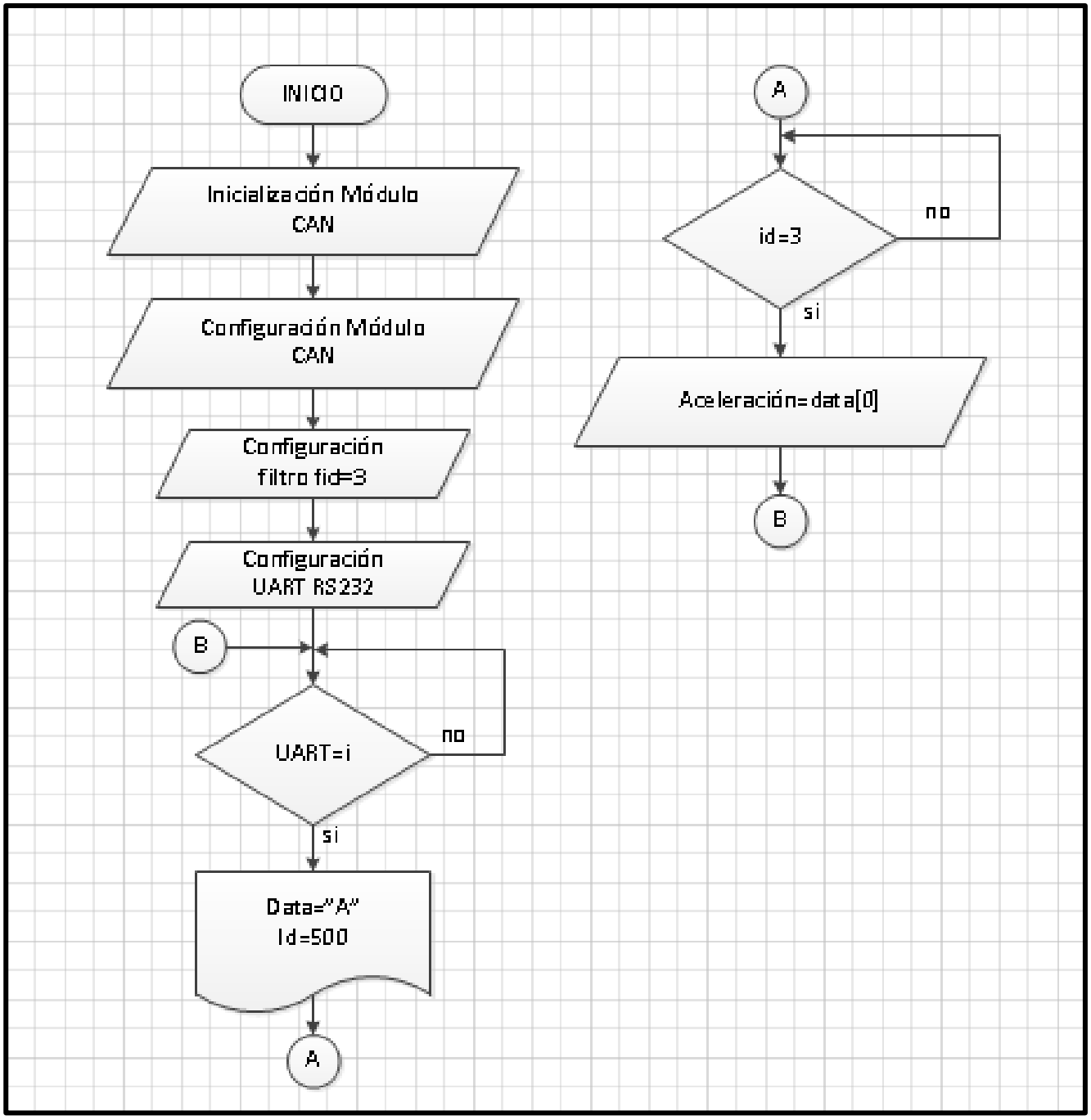
**TRAMA:** Es una unidad de envío de datos es el equivalente de paquete de datos

**TRANSCEPTOR CAN:** Emisor y receptor de señales eléctricas, de transmisor + receptor. (ENCINAS, 2012)

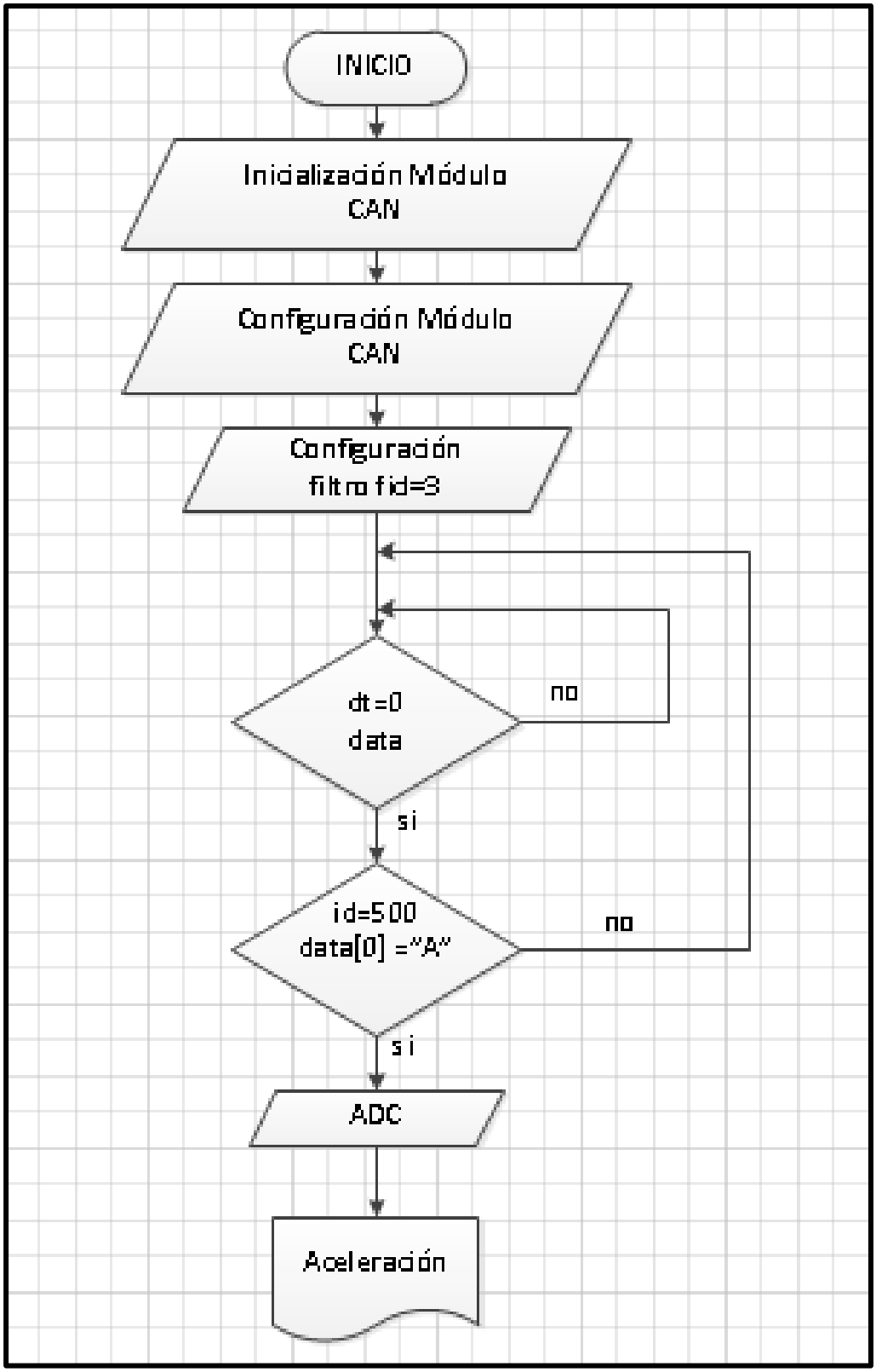
**ANEXO Nro.1:** ESQUEMATICO TARJETAS CAN



**ANEXO Nro.2:** DIAGRAMA DE FLUJO NODO MAESTRO



**ANEXO Nro.3:** DIAGRAMA DE FLUJO NODO ESCLAVO



**ANEXO Nro.4:** DIAGRAMA DE FLUJO COMUNCACION RS232