# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

## ANÁLISIS DEL PROTOCOLO CAN (CONTROLLER AREA NETWORK) E IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTROL DE NODOS INTERCONECTADOS POR UN BUS DE COMUNICACIONES CAN

**PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN**

**MÓNICA TATIANA COSTALES MEDINA**
**tatiana.costales@gmail.com**

**DIRECTOR: ING. FERNANDO FLORES**
**fflores@fie-epn.net**

**Quito, Marzo 2008**

# DECLARACIÓN

Yo, Mónica Tatiana Costales Medina declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Tatiana Costales

# CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Tatiana Costales, bajo mi supervisión.

_____

(Ing. Fernando Flores)

DIRECTOR DEL PROYECTO

# AGRADECIMIENTOS

A Dios por concederme unos padres maravillosos.
A mis padres y hermanos por el soporte brindado.
A quien nunca me abandonó, Diego, gracias por ser mi luz cuando el camino se nublaba.
A mis amigos, a las personas que confiaron en mí, quienes se encontraron presentes y sirvieron de inspiración para nuevas metas.
Al Ing. Fernando Flores, por permitirme continuar este proyecto bajo su dirección.

Finalmente, a los seres anónimos que colaboraron para que este esfuerzo culmine de la mejor manera.

# DEDICATORIA

A Luz y Eduardo, por esa devoción al oficio de ser padres, por el sacrificio y apoyo incondicional brindado durante toda mi existencia.

A Edwin, por ser más que hermano un gran amigo, vivirás en mi corazón siempre.

A Robert, por su bondad y espíritu de lucha.

A Doménica, un angelito que trajo alegría a nuestras vidas.

# CONTENIDO

# ÍNDICE DE FIGURAS

## CAPÍTULO 1: Marco Teórico

## CAPÍTULO 2: Diseño del prototipo

## CAPÍTULO 3: Implementación y pruebas



## CAPÍTULO 4: Costo del prototipo

# ÍNDICE DE TABLAS

## CAPÍTULO 1: Marco Teórico



## CAPÍTULO 2: Diseño del prototipo

## CAPÍTULO 4: Costo del prototipo

# ÍNDICE DE ILUSTRACIONES

## Referidas al Anexo G: Diagramas esquemáticos y PCB de los nodos

# RESUMEN

El presente proyecto expone el análisis del protocolo de comunicaciones CAN, y el desarrollo de un sistema distribuido, implementado en un prototipo de control de nodos interconectados mediante el bus serial asincrónico de comunicaciones CAN.

El análisis del protocolo CAN contempla parámetros como el estudio del intercambio de información en el bus, la manera de arbitrar el acceso al medio de comunicaciones, formatos y tipos de mensajes, manejo de errores, sincronización, entre otras cosas, enfocándonos principalmente en el manejo de la capa física y enlace de datos del bus de comunicaciones CAN.
Adicionalmente, se investiga otros buses de comunicaciones seriales sincrónicos y asincrónicos para aplicaciones en sistemas microprocesados y sobre todo para aquellas inmersas en el ámbito domótico.
También se ilustra características generales de microcontroladores que ofrecen soporte para este bus de comunicaciones tratando de demostrar las grandes capacidades que poseen para realizar tareas complejas gracias al aumento de velocidades de procesamiento y de módulos de comunicación con periféricos externos.

El desarrollo del prototipo contempla dos aspectos, diseño de hardware y diseño de software.
El diseño de hardware comprende todos los elementos electrónicos necesarios para crear tres nodos, un nodo maestro y dos nodos esclavos. Un nodo esclavo contiene un sensor de temperatura y un sensor de movimiento, conectados a un microcontrolador, el otro nodo esclavo contiene un sensor de humo y un sensor magnético, conectados a otro microcontrolador de similares características, el nodo maestro no contiene sensores ya que tiene como función el establecimiento de la comunicación entre los nodos esclavos y un computador personal, quien se encargará del monitoreo remoto de los sensores.
El diseño de software comprende la programación, para microcontroladores, requerida de acuerdo a la funcionalidad de cada uno de los nodos y el desarrollo

de una interface gráfica para el monitoreo remoto de los sensores a través de un computador.

La implementación del prototipo consiste en la construcción de tres circuitos impresos, uno por cada nodo diseñado, el montaje del prototipo y las pruebas a nivel físico y a nivel de capa aplicación a las que fue sometido el mismo.

Finalmente, se especifica el costo del prototipo realizado, incluyéndose coste de diseño, fabricación y montaje del prototipo.
En la actualidad existen múltiples soluciones CAN como lo son interfaces para computador personal, adaptadores USB-CAN, gateways, repetidores, etc., brevemente se examina características básicas y precios de éstas herramientas comerciales.

# PRESENTACIÓN

A principios de los años 80s, la industria automotriz, notó el avanzado incremento en los sensores y actuadores dentro del vehículo y el aumento excesivo de cableado que se generaba a causa de esto. Estos inconvenientes motivaron a la empresa alemana Robert Bosch GMBH a iniciar un proyecto para el desarrollo de una red de interconexión con el fin de lograr una gran funcionalidad y especialmente reducir el cableado dentro de los vehículos. En su búsqueda de una solución nace el sistema de bus serial de comunicaciones CAN (Controller Area Network).
Actualmente la industria automotriz ha evolucionado de tal forma que ahora todos sus dispositivos electrónicos se interconectan a un solo “cerebro” o computador, utilizando este bus de comunicaciones.

La avanzada evolución del protocolo CAN, lo convirtió en un estándar internacional registrado como ISO 11898 e ISO 11519 para control distribuido de aplicaciones en tiempo real cuyos requerimientos sean confiabilidad, robustez y alta tasa de trasferencia.
Una característica importante es que este bus fue diseñado para trabajar en ambientes hostiles o de alta interferencia electromagnética lo que lo impulsó a tener mucha acogida a niveles industriales.
Más allá de la industria automotriz, se está utilizando CAN en automatización de edificios, aeronáutica, control industrial, equipamiento médico, entre otras.
CAN ha sido tan difundido que según predicciones, algún día llegará a ser tan utilizado como el protocolo RS232 en su época, de allí la importancia en el estudio de este bus de comunicaciones.

El presente proyecto propone el estudio del protocolo CAN y la implementación de un prototipo de control de nodos poniendo énfasis en la utilización del bus serial de comunicaciones CAN para la interconexión entre los mismos.

La aplicación presentada consiste de una red de sensores que puede ser monitoreada utilizando un computador personal que tenga disponible un puerto serial.

# CAPÍTULO 1: MARCO TEÓRICO

## 1.1 ANÁLISIS DEL PROTOCOLO CAN

### 1.1.1 RESEÑA HISTÓRICA DEL PROTOCOLO DE COMUNICACIONES CAN[1]

El desarrollo del protocolo CAN empezó cuando se dio continuo incremento de dispositivos electrónicos en los vehículos modernos tales como sistemas de administración de motor, control de luces, aire acondicionado, bloqueo central entre otros, orientados a proporcionar confort y seguridad al conductor. De ahí que fue necesario idear la forma de intercambiar información entre los diferentes sistemas de control y sus sensores.

Anteriormente, este intercambio de información, se realizaba mediante conexiones punto a punto entre los diferentes sistemas, pero las necesidades fueron incrementándose de tal modo que se requería de cuantiosos cables y muchos conectores, este crecimiento desmedido, trajo como consecuencia el aumento en el costo de los materiales y en el tiempo de producción del vehículo.

La solución a este problema fue la conexión de los sistema de control vía bus serial. Con la utilización del bus CAN, se reemplaza el cableado punto a punto por un único bus, esto se logra añadiendo algunas especificaciones CAN al hardware de cada unidad de control para proveer el protocolo de transmisión y recepción de información.

El protocolo CAN fue desarrollado inicialmente para el uso en vehículos por la firma alemana Robert Bosch GMBH. En febrero de 1986 se presentó oficialmente el sistema de bus serial CAN en el congreso de la Sociedad de Ingenieros Automotrices (SAE) celebrado en Detroit – Estados Unidos.

La versión de CAN 11898-1 fue estandarizada por la Organización Internacional de Estandarización (ISO) en el año 1993.

El protocolo CAN hace uso de las capas física y enlace de datos del modelo ISO/OSI. Existen también ciertos protocolos de capas superiores disponibles para CAN como los son CAL, CANOpen, CANKingdom y DeviceNet.

### 1.1.2 CONCEPTOS BÁSICOS[2-5]

En la actualidad el modelo de comunicaciones de datos más utilizado en automatización industrial es el modelo cliente/servidor. El protocolo de comunicaciones CAN se basa en este modelo y en el paradigma que describe la relación entre uno o varios clientes con un servidor. En este modelo se proporciona los datos a través de difusión de mensajes, los clientes pueden o no aceptar dichos mensajes y debido a que el servidor no conoce a sus clientes no requiere de confirmación alguna.

Existen dos tipos de protocolos de comunicaciones de datos:

#### 1.1.2.1 Protocolos orientados a nodos

Este tipo de protocolo está basado en el direccionamiento, cada nodo posee su propia dirección, los mensajes que se transmiten por el bus de comunicaciones contienen la dirección destino y en algunos casos la dirección origen.

#### 1.1.2.2 Protocolos orientados a mensajes

Este tipo de protocolo está basado en identificadores, a cada mensaje se le asigna un identificador único y se lo encapsula en tramas para ser transmitido. Analizando el identificador, los nodos deciden o no aceptar el mensaje.

CAN es un protocolo orientado a mensajes que posee las siguientes características:

- *Prioridad de mensajes:* se define mediante el identificador del mensaje durante el acceso al bus de comunicaciones. El identificador también distingue el contenido del mensaje pero no indica el destino del mismo, solo describe su significado. Cada nodo en la red es capaz de filtrar los mensajes de acuerdo al identificador para decidir si deben actuar o no.
- *Garantía en tiempos de latencia:* la longitud de un mensaje es corta, tiene como máximo 8 bytes de datos por mensaje lo que garantiza una baja latencia entre transmisión y recepción. El acceso al bus es controlado por el protocolo Carrier Sense Multiple Access/ Collision Detection with Arbitration on Message Priority (CSMA/CD with AMP) esto significa que la colisión de mensajes es evitada por medio de arbitraje tomando en cuenta la prioridad de los mismos.

- *Configuración flexible:* en un sistema CAN un nodo no hace ningún uso de la información del sistema de configuración como por ejemplo la dirección física de cada estación, esto hace que el sistema sea flexible, es decir, se puede añadir nodos a la red CAN sin necesidad de ningún cambio de software, hardware o en la capa aplicación.
- *Recepción por multidifusión (multicast) con sincronización de tiempos: t*odos los nodos de la red reciben el mismo mensaje y realizan el mismo procedimiento de filtrado simultáneamente.
- *Robusto en cuanto a consistencia de datos:* todos los receptores verifican la consistencia de la información que es recibida y reconocen un mensaje consistente mediante acuses de recibo.
- *Sistema multimaestro:* cuando el bus está libre, cualquier nodo puede iniciar la comunicación. La unidad con el mensaje de mayor prioridad a ser transmitido gana el acceso al bus.
- *Detección y señalización de errores:* el sistema CAN proporciona alta inmunidad a la interferencia electromagnética, provee además mecanismos de detección de errores como los siguientes:
    - Monitoreo
    - Chequeo de redundancia cíclica
    - Bits de relleno
    - Chequeo de formato de tramas del mensaje

  El mecanismo de detección de errores permite a los nodos detectar errores globales, errores distribuidos de manera aleatoria, ráfagas de errores de longitud menor a 15 y errores de cualquier número impar en un mensaje. La probabilidad de error en un mensaje es de $4.7 * 10^{-11}$.
- *Retransmisión automática de tramas erróneas tan pronto como el bus esté libre*: los mensajes erróneos son identificados por una bandera de modo que cualquier nodo está en capacidad de detectar un error. Los mensajes erróneos se descartan y se retransmiten automáticamente.
- *Distinción entre errores temporales y fallas permanentes en los nodos y desconexión automática de nodos defectuosos.*

### 1.1.3 ESTRUCTURA DE CAPAS DEL PROTOCOLO CAN

El protocolo de comunicaciones CAN, en su especificación ISO 11898, describe como la información será transmitida entre los diferentes dispositivos de una red. En conformidad con el modelo ISO/OSI, el protocolo CAN garantiza transparencia en el diseño y flexibilidad en las implementaciones definiendo únicamente las capas enlace de datos y física.

| **Capa Enlace de Datos** | |
|---|---|
| LLC<br>(Logical Link Control) | Filtrado de mensajes<br>Notificación de sobrecarga<br>Proceso de recuperación |
| MAC<br>(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 1.1:** Estructura de capas del protocolo CAN (ISO 11898)[1]

#### 1.1.3.1 Capa enlace de datos

Se establece los servicios y funciones definidas en la capa de enlace de datos del modelo de referencia ISO/OSI.

##### *1.1.3.1.1 Subcapa Logical Link Control (LLC)*

Esta subcapa define las tareas independientes del método de acceso al medio y proporciona dos tipos de transmisión sin conexión al usuario (LLC User).

[1] Fuente: CAN specification version 2.0 - Robert Bosch GmbH

- *Servicio de transmisión de datos sin reconocimiento:* proporciona al usuario el mecanismo para el intercambio de unidades de datos de servicio de enlace (LSDU, Link Service Data Units) sin el establecimiento de una conexión de enlace de datos. Existen tres tipos de transmisión de datos, punto a punto, multidifusión o difusión.
- *Servicio de transmisión de datos remota sin reconocimiento:* proporciona al usuario el mecanismo para que un nodo remoto transmita sus LDSU's sin el establecimiento de una conexión de enlace de datos.

De acuerdo con los dos tipos de servicios, se definen dos formatos de tramas, de datos LLC y remota LLC. Ambos formatos definen identificadores de 11 bits estándar (CAN 2.0A) y 29 bits extendida (CAN 2.0B).

Las funciones de la subcapa LLC incluyen:

- *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.
- *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.
- *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.

#### *1.1.3.1.2 Subcapa Medium Access Control (MAC)*

La subcapa MAC representa el núcleo del protocolo CAN. Esta subcapa presenta los mensajes recibidos a la subcapa LLC y acepta los mensajes a ser transmitidos desde la subcapa LLC.

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

*Arbitrar el acceso al medio.-* Cuando el bus está libre, cualquier unidad puede transmitir un mensaje. Si dos o más unidades empiezan a transmitir mensajes al mismo tiempo, el conflicto de acceso al bus de comunicaciones es resuelto mediante arbitraje usando un identificador, este mecanismo de control de acceso al medio es conocido como "Carrier Sense Multiple Access/Collision Detection with Arbitration on Message" (CSMA/CD with AMP). Este tipo de arbitraje previene colisiones de mensajes durante la transmisión lo que garantiza que cualquier información no se pierda durante el conflicto. El concepto de prioridad es eficiente en el sentido en el que si dos o más nodos desean transmitir simultáneamente, con el identificador, se asegura que el mensaje de mayor importancia sea transmitido primero sin pérdida de tiempo. Si ocurriera que diferentes tipos de tramas con el mismo identificador intentan ser transmitidas, las tramas de datos predominan sobre cualquier otra. Durante el arbitraje, cada transmisor compara el nivel del bit a ser transmitido con el nivel que es monitoreado en el bus, si esos niveles son iguales a la unidad se puede continuar enviando información. Cuando un nivel de receso es enviado y es monitoreado un nivel dominante[2], entonces la unidad ha perdido el arbitraje y debe retirarse sin enviar ningún bit más.

En la figura 1.1 se muestra cuando tres nodos conectados al bus desean transmitir siguiendo el principio del wired-AND[3]. Cada nodo envía su mensaje con su correspondiente identificador y monitorea el nivel del bus.
A determinado tiempo los nodos 1 y 2 envían un bit identificador dominante. El nodo 2 envía un bit identificador recesivo pero lee que el estado del bus es dominante. Algunos bits después, el nodo 3 pierde el arbitraje contra el nodo 1. Esto significa que el identificador del mensaje del nodo 1 tiene un valor binario

[2] Ver Figura 1.2
[3] Ver Tabla 1.2

más bajo pero con mayor prioridad que los mensajes de los nodos 2 y 3. De esta manera, el nodo con mensaje de más alta prioridad gana el arbitraje.

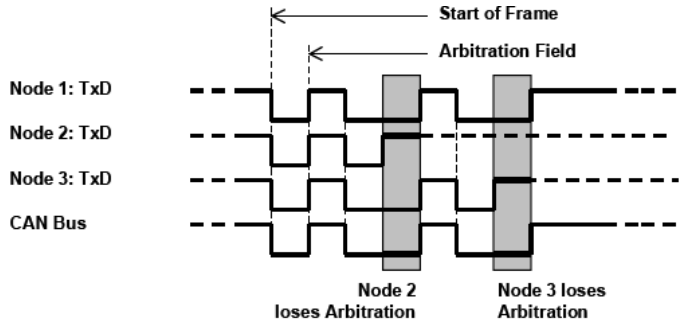



**Figura 1.1:** CSMA/CD con arbitraje mediante priorización de mensajes[4]

### 1.1.3.2 Capa física[1,3-5]

La capa física define como las señales serán transmitidas a través del medio físico. Los servicios de la capa física incluyen:

- Sincronización
- Codificación/Decodificación de bits
- Temporización de bit CAN

#### *1.1.3.2.1 Capa física de acuerdo a ISO 11898*

Bus de comunicaciones

La señalización en el bus es diferencial, de esto se deriva su alta inmunidad al ruido y tolerancia a fallos. La señalización diferencial balanceada reduce aún más el ruido y permite altas velocidades de señalización sobre cables de par trenzado. El hecho de que sea balanceado significa que la corriente fluye en cada línea de transmisión de manera igual pero en dirección opuesta, resultando en un efecto de cancelación de campo lo que es la clave para obtener bajas emisiones de ruido. Para reducir la sensibilidad a la interferencia electromagnética aún más, se puede adicionar a las líneas del bus algún tipo de blindaje o revestimiento.

De acuerdo al estándar, el bus de dos hilos contiene dos señales, CAN_H y CAN_L. El nodo CAN debe proporcionar a la salida el voltaje diferencial:

[4] Fuente: Atmel Microcontrollers for Controller Area Network (CAN) – ATMEL Corporation

*Bit recesivo:* -500mV a 500mV, sin carga.

*Bit dominante:* +1.5V a 3.5V, con carga de 60Ω.

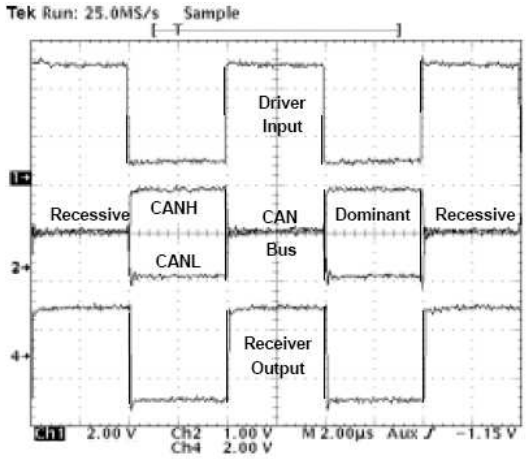



**Figura 1.2:** Estados Dominante y Recesivo del bus[5]

El bus lógico utiliza un mecanismo conocido como “Wired-AND” en el que, los “bits dominantes” (equivalentes al nivel lógico “Cero”) sobrescriben a los “bits recesivos” (equivalentes al nivel lógico “Uno”).

| Nodo 1 | Nodo 2 | Nodo 3 | Bus |
|---|---|---|---|
| d | d | d | d |
| d | d | r | d |
| d | r | d | d |
| d | r | r | d |
| r | d | d | d |
| r | d | r | d |
| r | r | d | d |
| r | r | r | r |

**Tabla 1.2:** Lógica del bus CAN para tres nodos transmitiendo

Solo si todos los nodos transmiten bits recesivos (unos), el bus se encuentra en estado recesivo. Tan pronto como un nodo transmita un bit dominante (cero), el bus se encuentra en estado dominante.

[5] Fuente: Introduction to Controller Area Network - Texas Instruments

Medio físico

*Medio de transmisión eléctrico*

Bus de alambre de par trenzado con resistencias de 120Ω en sus terminaciones, el par trenzado es uno de los medios más comunes y más económicos.

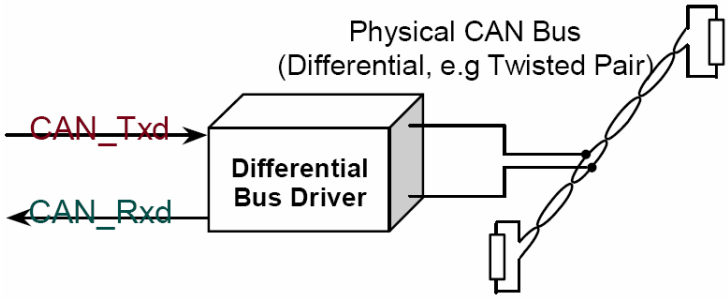



**Figura 1.3:** Medio físico del bus CAN – Par trenzado[6]

*Medio de transmisión óptico*

Dos alambres conductores de señales diferenciales (CAN_L y CAN_H)

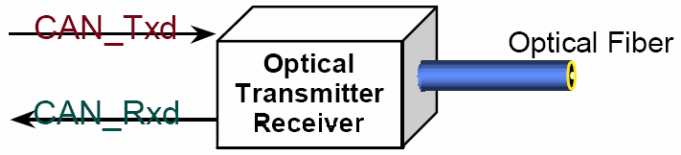



**Figura 1.4:** Medio físico del bus CAN – Fibra óptica[7]

En el caso del medio óptico, los estados recesivo y dominante se pueden representar con “luz apagada” y “luz encendida” respectivamente.

Este tipo de medio es ideal para aplicaciones en ambientes extremadamente ruidosos.

Alcance y número de nodos

De acuerdo al estándar ISO 11898, se puede alcanzar hasta 1Mbps sobre líneas del bus de longitud máxima de aproximadamente 40m (130ft), las líneas deberán contener resistencias de terminación de 120Ω. El número de nodos sobre la línea no tiene límite teórico, pero en la práctica el número total de unidades se ve

[6 - 7] Fuente: CANPRES V2.0 – Siemens

limitado por los tiempos de retardo y las cargas eléctricas en la línea del bus. Con estas consideraciones, para la longitud máxima el número de nodos permitido es 32.

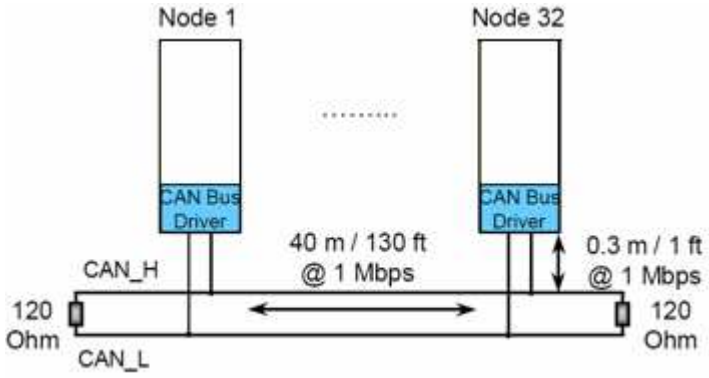



**Figura 1.5:** Medio físico ISO 11898 – CAN de alta velocidad (125 Kbps a 1Mbps)[8]

Se puede alcanzar mayores distancias pero a costa de reducir la velocidad de transmisión, también se puede añadir más de 32 nodos si se colocan repetidores. Para prevenir reflexiones de señal la distancia desde las líneas de transmisión a los nodos no debe exceder 0.3m.

Niveles de señal

De acuerdo al estándar, un bit recesivo es representado por ambas líneas del bus CAN como un nivel de señal de 2.5 V. El voltaje diferencial entre CAN_H y CAN_L es aproximadamente de 0 V. Un bit dominante es representado por 3.5 V en CAN_H y por 1.5 V en CAN_L. El resultado diferencial de estos voltajes para un bit es aproximadamente de 2 V.

En la figura 1.6 se muestra las líneas del bus CAN y los niveles de señal mencionados:

[8] Fuente: CANPRES V2.0 – Siemens

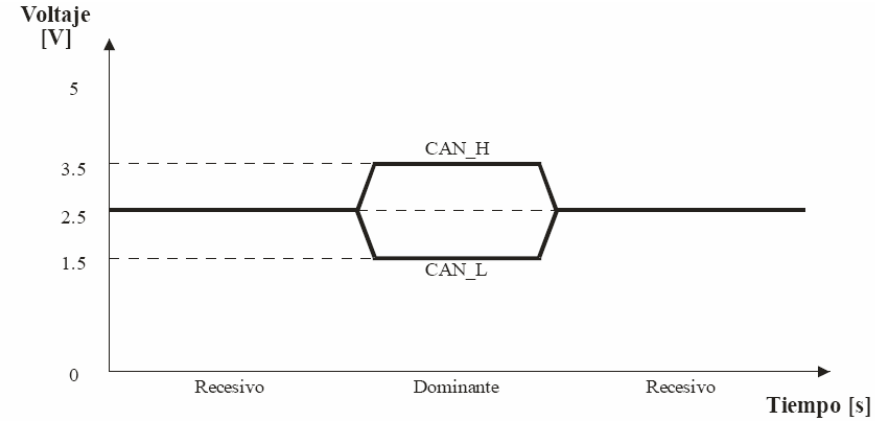



**Figura 1.6:** Niveles de señal en el bus[9]

En la tabla a continuación se resume los estados del bus CAN con sus niveles de voltaje permitidos:

| **Señal CAN** | | **Niveles nominales de voltaje [V]** | | |
|---|---|---|---|---|
| | | CAN_H | CAN_L | Vdif |
| Estado del bus | Dominante | 3.5 | 1.5 | 0.9 - 2.0 |
| | Recesivo | 2.5 | 2.5 | 0 – 0.5 |

**Tabla 1.3:** Posibles estados del bus[10]

Tipo de conector

No existe un estándar para el tipo de conector al bus. Generalmente, cada protocolo de capa aplicación define su conector propietario. Entre los más comunes se incluyen:

- Conector DB9 – Recomendación DS-102
- Conector tipo mini de cinco terminales
- Conector multipolo
- Conector RJ10/RJ45

### *Conector DB9*

Este conector es recomendado por la CiA (CAN in Automation) y es altamente utilizado a nivel industrial.

9 - 10 Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

| Pin | Señal | Descripción |
|---|---|---|
| 1 | - | Reservado |
| 2 | CAN_L | Señal de bus CAN (dominante bajo) |
| 3 | CAN_GND | Señal de tierra (GND) |
| 4 | - | Reservado |
| 5 | CAN_SHLD | Blindaje (opcional) |
| 6 | GND | Tierra (opcional) |
| 7 | CAN_H | Señal de bus CAN (dominante alto) |
| 8 | - | Reservado (línea de error) |
| 9 | CAN_V+ | Fuente de alimentación externa (opcional) |

**Tabla 1.4:** Asignación de terminales del conector DB9[11]

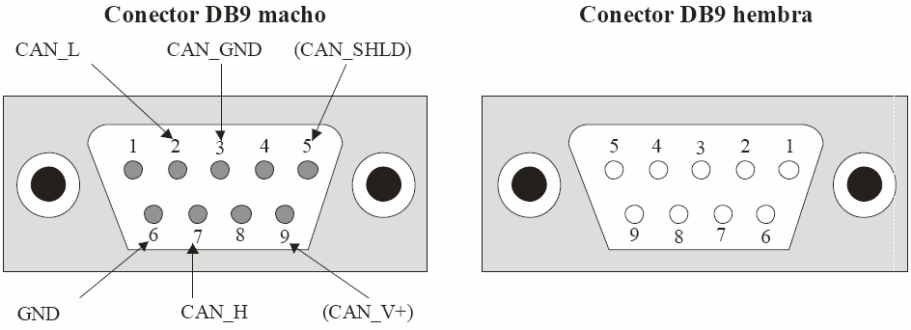



**Figura 1.7:** Asignación de terminales del conector DB9[12]

### *Conector tipo mini de cinco terminales*

Este conector es utilizado por DeviceNet, CANopen y otros fabricantes.

| Pin | Señal | Descripción |
|---|---|---|
| 1 | CAN_SHLD | Blindaje (opcional) |
| 2 | CAN_V+ | Fuente de alimentación externa (opcional) |
| 3 | CAN_GND | Señal de tierra (GND) |
| 4 | CAN_H | Señal de bus CAN (dominante alto) |
| 5 | CAN_L | Señal de bus CAN (dominante bajo) |

**Tabla 1.5:** Asignación de terminales del conector tipo mini[13]

[11] - [12] - [13] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

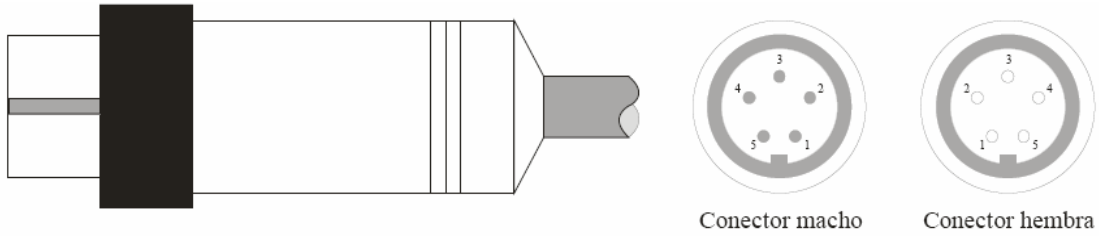



**Figura 1.8:** Asignación de terminales del conector tipo mini[14]

### *Conector multipolo*

Este conector es utilizado por DeviceNet, CANopen y otros fabricantes.

| Pin | Señal | Descripción |
|---|---|---|
| 1 | - | Reservado |
| 2 | GND | Tierra (opcional) |
| 3 | CAN_L | Señal de bus CAN (dominante bajo) |
| 4 | CAN_H | Señal de bus CAN (dominante alto) |
| 5 | CAN_GND | Señal de tierra (GND) |
| 6 | - | Reservado |
| 7 | - | Reservado |
| 8 | CAN_V+ | Fuente de alimentación externa (opcional) |
| 9 | - | Reservado |
| 10 | - | Reservado |

**Tabla 1.6:** Asignación de terminales del conector multipolo[15]

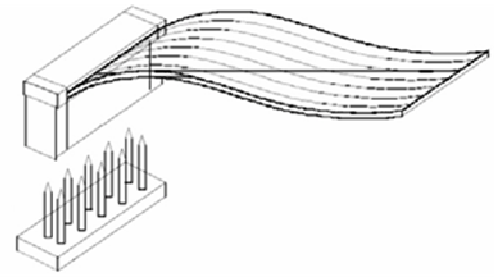


**Figura 1.9:** Asignación de terminales del conector multipolo[16]

[14] - [15] - [16] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

### *Conector RJ10/RJ45*

Estos conectores son utilizados por CANopen y otros fabricantes.

| Pin | Conector RJ10 | | Conector RJ45 | |
|---|---|---|---|---|
| | **Señal** | **Descripción** | **Señal** | **Descripción** |
| 1 | CAN_V+ | Fuente de alimentación externa (op) | CAN_H | Señal de bus CAN (dominante alto) |
| 2 | CAN_H | Señal de bus CAN (dominante alto) | CAN_L | Señal de bus CAN (dominante bajo) |
| 3 | CAN_L | Señal de bus CAN (dominante bajo) | CAN_GND | Señal de tierra (GND) |
| 4 | CAN_GND | Señal de tierra (GND) | - | Reservado |
| 5 | | | - | Reservado |
| 6 | | | CAN_SHLD | Blindaje (opcional) |
| 7 | | | GND | Tierra (opcional) |
| 8 | | | CAN_V+ | Fuente de alimentación externa (op) |

**Tabla 1.7:** Asignación de terminales para los conectores RJ10 y RJ45[17]

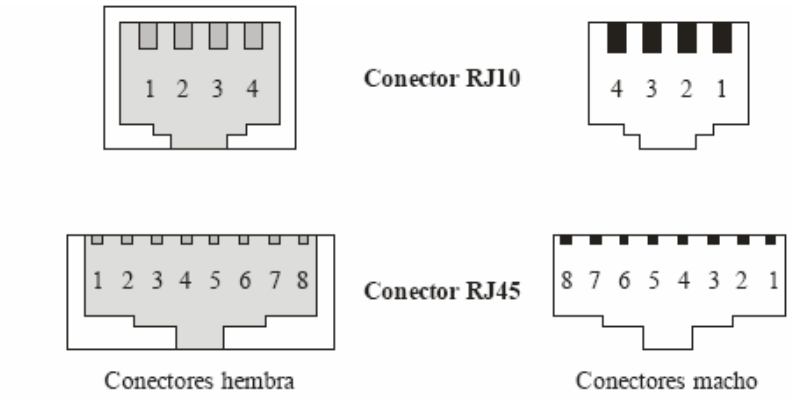



**Figura 1.10:** Asignación de terminales para los conectores RJ10 y RJ45[18]

### 1.1.4 TRANSFERENCIA DE MENSAJES[1]

Existen dos diferentes formatos de mensajes que difieren en la longitud del campo identificador, estos formatos corresponden a las dos versiones del protocolo CAN: CAN estándar y CAN extendido.

**CAN Estándar.-** Fue la primera versión del protocolo CAN, conocida también como ISO1159 (CAN de baja velocidad), soporta aplicaciones bajo los 125 Kbps

[17] - [18] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

con un número de 11 bits de identificador. La segunda versión de CAN fue ISO11898 (CAN de alta velocidad), estandarizada en 1993, también posee 11 bits de identificador pero provee velocidades superiores a la primera versión que oscilan desde 125Kbps a 1Mbps. El estándar ISO 11898 es usualmente conocido como estándar CAN Versión 2.0A y provee $2^{11}$ o 2048 diferentes identificadores de mensaje.

**CAN Extendido.-** Esta es la versión más reciente del protocolo, corresponde a la versión extendida la misma que posee 29 bits de identificador en lugar de 11 bits. Esta versión es conocida como Estándar CAN Versión 2.0B y provee $2^{29}$ o 537 millones de diferentes identificadores de mensaje.

| Nomenclatura | Estándar | Velocidad Máx. | Identificador |
|---|---|---|---|
| Low-Speed CAN | ISO 11519 | 125 Kbps | 11 – bit |
| CAN 2.0A | ISO 11898:1993 | 1 Mbps | 11 – bit |
| CAN 2.0B | ISO 11898:1995 | 1Mbps | 29 – bit |

**Tabla 1.8:** Versiones del protocolo CAN

### 1.1.4.1 Tipos de Tramas

Existen cuatro tipos de tramas: trama de datos, tramas remotas, tramas de error y tramas de sobrecarga.

#### *1.1.4.1.1 Trama de Datos*

La *trama de datos* transporta información desde los transmisores hacia los receptores. Este tipo de trama es la más utilizada. La transmisión de datos se desarrolla de manera autónoma, cuando un nodo emisor (ej. un sensor) envía tramas de datos, un nodo receptor responde a este tipo de trama emitiendo tramas remotas.

## Campos de la trama de datos

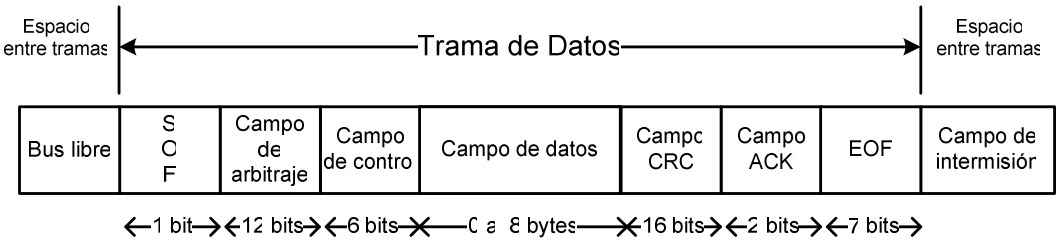



**Figura 1.11:** Campos de la trama de datos[19]

***Inicio de trama (SOF).-*** Este campo determina el inicio de una trama remota o una trama de datos. Consiste de un bit dominante que sincroniza a todos los nodos activos en la red. Este campo es el mismo para una trama estándar (CAN 2.0A) o una trama extendida (CAN 2.0B).

***Arbitraje.-*** Este campo determina la prioridad de un mensaje cuando dos o más nodos se encuentran en contienda para el acceso al bus de comunicaciones.

En el formato estándar, el campo de arbitraje contiene un identificador de 11 bits y un bit de petición de transmisión remota RTR (Remote Transmission Request). En el caso de una trama de datos, el bit RTR es dominante. El bit menos significativo se transmite al final y los siete bits más significativos no pueden ser todos recesivos.

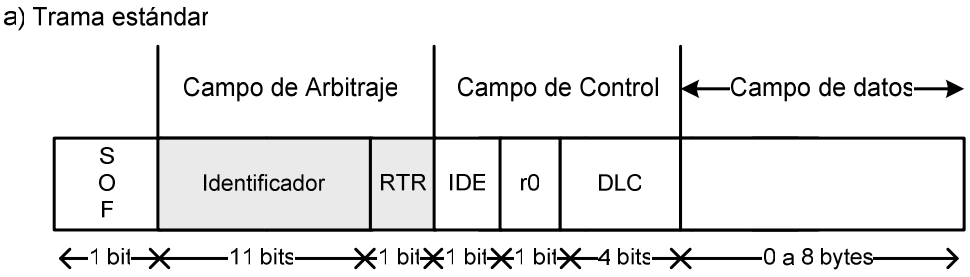



**Figura 1.12:** Trama de datos estándar[20]

[19] - [20] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

En el formato extendido, el identificador contiene 29 bits, se tiene un bit de petición de transmisión substituta SRR (Substitutte Remote Request), un bit de extensión de identificador IDE (Identifier Extension) y un bit de petición de transmisión remota RTR.

El identificador se divide en dos partes, la primera parte consta de 11 bits y se denomina base (base ID) y corresponde al identificador del formato de trama estándar, la segunda parte consta de 18 bits y se la conoce como identificador extendido (extended ID).

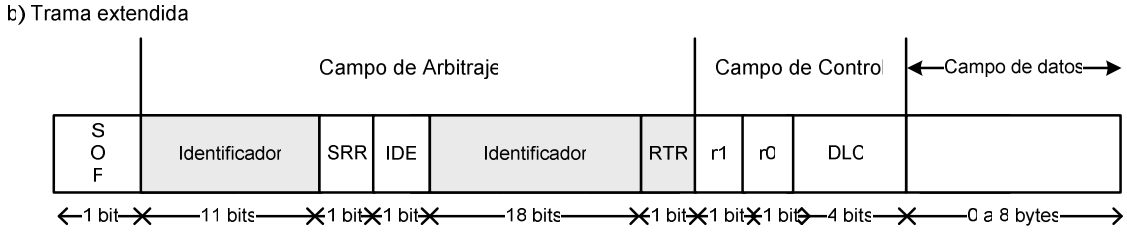



**Figura 1.13:** Trama de datos extendida[21]

Tanto para el formato de trama estándar como para el extendido, el bit RTR es dominante y el bit SRR es recesivo, de esta manera, se evitan posibles colisiones entre ambos tipos de trama que posean el mismo valor para el campo base ID, las tramas estándar predominan sobre las tramas extendidas. En el proceso de determinar la trama a ser transmitida, también se involucra al bit IDE, este bit, en una trama estándar, pertenece al campo de control y es un bit dominante y en una trama extendida, pertenece al campo de arbitraje y es un bit recesivo.

***Control.-*** Este campo consta de seis bits: IDE/r1, r0 y cuatro bits para el código de longitud de datos DLC (Data Length Code) el cual indica el número de octetos (de cero a ocho) que contendrá el campo de datos. El bit IDE, sirve para identificar entre dos tipos de tramas, en estado dominante, especifica que la trama es estándar y es el bit que se transmite primero seguido por r0 en nivel dominante, ambos bits se encuentran reservados para futuras aplicaciones.

[21] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

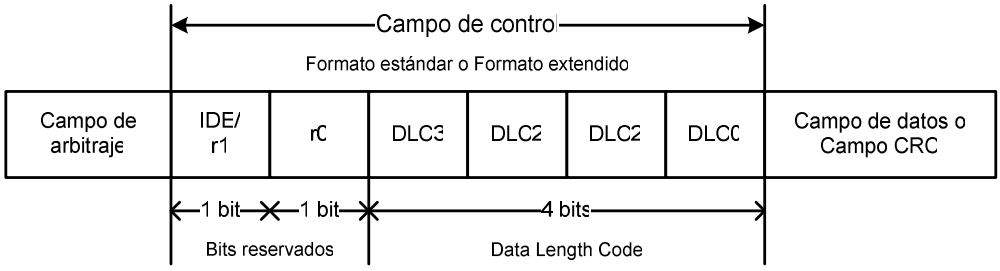



**Figura 1.14:** Campo de control[22]

Codificación para el número de bytes de datos contenida en el campo DLC:

| Bytes de datos | Código de Longitud de Datos | | | |
|---|---|---|---|---|
| | DLC3 | DLC2 | DLC1 | DLC0 |
| 0 | d | d | d | d |
| 1 | d | d | d | r |
| 2 | d | d | r | d |
| 3 | d | d | r | r |
| 4 | d | r | d | d |
| 5 | d | r | d | r |
| 6 | d | r | r | d |
| 7 | d | r | r | r |
| 8 | r | d | d | d |

**Tabla 1.9:** Codificación del campo DLC[23]

El campo DLC de cuatro bits, toma las primeras nueve combinaciones de las dieciséis posibles, las combinaciones restantes se consideran no permitidas. El envío de una combinación no válida alterará la longitud del mensaje y el receptor interpretará que los campos posteriores pertenecen al campo de datos, sin embargo, todos los nodos que reciban este tipo de mensajes están en capacidad de detectar el error utilizando el campo CRC,

***Datos.-*** Este campo contiene la información a ser transmitida por los nodos CAN. La longitud de este campo varía entre cero a ocho bytes. El bit más significativo se transmite primero.

[22] Fuente: CAN Specification 2.0, Part B - CAN in Automation (CiA)
[23] Abreviaturas: d = dominante, r = recesivo Fuente: CAN Specification 2.0, Part B - CAN in Automation (CiA)

***CRC.-*** Este campo contiene 16 bits, de los cuales 15 bits son dedicados para el CRC (Código de Redundancia Cíclica) o secuencia de verificación y un bit delimitador de CRC que no es más que un bit en estado recesivo.

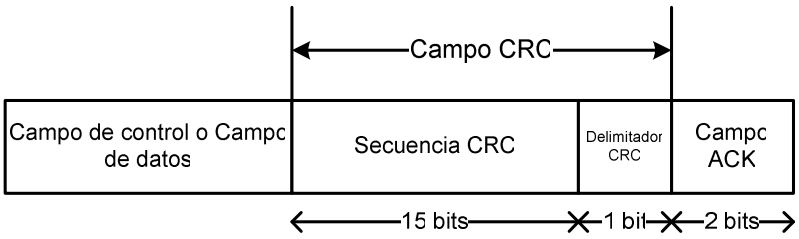



**Figura 1.15:** Campo CRC[24]

El CRC se calcula en base al polinomio $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$.

Los coeficientes que se toman para el cálculo del CRC son el campo delimitador de inicio, arbitraje, control y datos (si existe).

***ACK.-*** Este campo consta de dos bits, ranura ACK y delimitador ACK. Los nodos transmisores, emiten ambos bits en estado recesivo. Los nodos receptores, confirman los mensajes recibidos sin errores, sobrescribiendo el bit ranura ACK con un bit en estado dominante, si no existe dicha confirmación se asume que el mensaje arribó con errores.

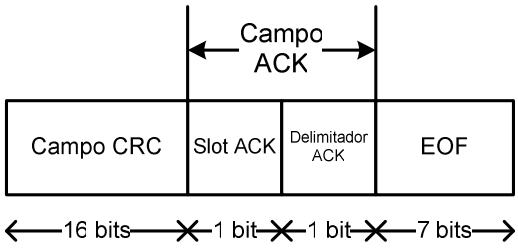



**Figura 1.16:** Campo ACK[25]

***Final de trama (EOF).-*** Este campo consiste de una secuencia de siete bits en estado recesivo, esta bandera especial es utilizada para delimitar el fin de las tramas de datos y remotas.

[24 - 25] Fuente: CAN Specification 2.0, Part B - CAN in Automation (CiA)

Para evitar que esta secuencia sea confundida durante la operación normal de la trama, cada cinco niveles lógicos recesivos consecutivos se introduce un nivel lógico dominante de relleno como violación de bit.

#### *1.1.4.1.2 Tramas Remotas*

La *trama remota* es transmitida por un nodo como respuesta a la petición de una trama de datos, para establecer la correspondencia entre tramas de petición/ respuesta, se emite la trama remota con el identificador correspondiente a la trama de datos a la que está respondiendo.

La trama remota tiene un formato similar a la trama de datos, pero con dos diferencias, la primera es que el tipo de mensaje es explícitamente marcado como trama remota utilizando el bit RTR en estado recesivo en el campo de arbitraje y la segunda es que la trama remota no posee campo de datos. Se debe notar también que el bit DLC debe tener concordancia con el número de bytes de la trama de datos a la que está respondiendo.

Campos de la trama remota

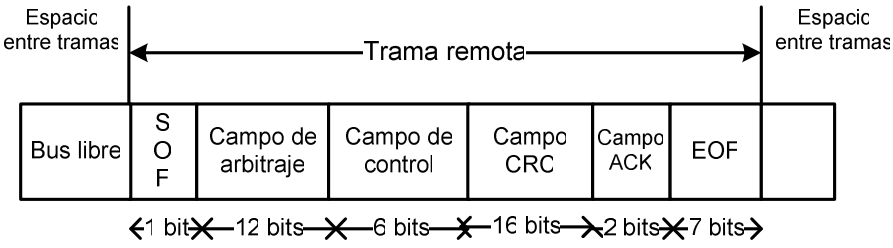



**Figura 1.17:** Campos de la trama remota[26]

#### *1.1.4.1.3 Trama de Error*

La *trama de error* es transmitida cuando un nodo ha detectado un error en un mensaje, esto causa que todos los nodos de la red envíen una trama de error para que el transmisor original retransmita el mensaje que originó el error.

[26] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

Cuando se detecta errores en la transmisión de una trama de sobrecarga y error, también se emite una trama de error.

Campos de la trama de error

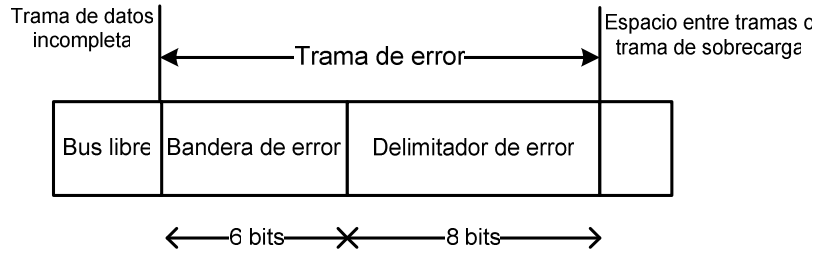



**Figura 1.18:** Campos de la trama de error[27]

La trama de error se divide en dos partes, bandera de error y delimitador de error.

***Bandera de error.-*** De acuerdo al estado del error, existen dos tipos de banderas de error:

- *Error activo:* Cuando un nodo detecta un error en el bus, interrumpe la transmisión de mensajes emitiendo una "bandera de error activo", esta bandera contiene una secuencia de seis bits consecutivos en estado dominante. Todas las estaciones al reconocer esta secuencia, generan tramas de error por sí mismas.

- *Error pasivo:* Cuando un nodo detecta un error en el bus, transmite una "bandera de error pasiva", esta bandera contiene una secuencia de seis bits consecutivos en estado recesivo, estos bits pueden ser sobrescritos por bits dominantes de otros nodos.

***Delimitador de error.-*** Consiste de una secuencia de ocho bits recesivos. Esta secuencia, permite que los nodos en el bus reinicien la comunicación después de que ocurrió un error.

[27] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

#### *1.1.4.1.4 Tramas de Sobrecarga*

La *trama de sobrecarga* es usada para proveer de un cierto retardo a los nodos durante una transmisión de tramas de datos y remotas o para señalizar condiciones de error.

Esta trama tiene un formato similar a la trama de error y es emitida principalmente por los nodos que se encuentran demasiado ocupados. Como máximo, un nodo puede generar hasta dos tramas de sobrecarga secuenciales, para retardar la recepción del siguiente mensaje. Las tramas de sobrecarga pueden ser generadas únicamente durante el espacio entre tramas, no se puede interrumpir la transmisión de una trama de datos o remota. Esta es la principal diferencia con la trama de error, una trama de error puede interrumpir la transmisión de un mensaje.

Las tramas de sobrecarga se emiten luego de detectar las siguientes condiciones de error:

- *Detección de un bit dominante en el tercer bit del campo de intermisión*[28]*:* este error se interpreta como un SOF.

- *Detección de un bit dominante en el último bit en los campos EOF*, delimitador de trama de sobrecarga o delimitador de trama de error.

Campos de la trama de sobrecarga

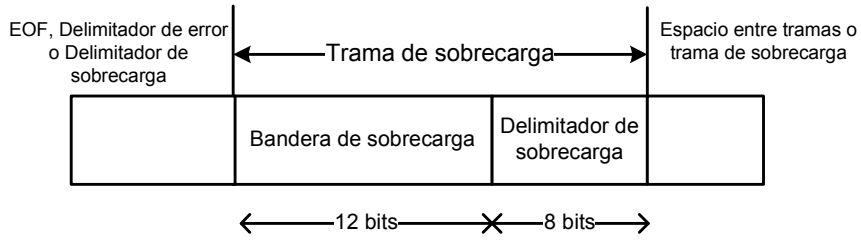



**Figura 1.19:** Campos de la trama de sobrecarga[29]

[28] El campo de intermisión está definido en la página 24.
[29] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

La bandera de sobrecarga posee dos campos, un campo de bandera de sobrecarga y otro campo delimitador de sobrecarga.

***Bandera de sobrecarga.-*** Contiene una secuencia de seis bits dominantes seguidos por una bandera de error activo, esta bandera posee doce bits en estado dominante.

***Delimitador de sobrecarga.-*** Contiene una secuencia de ocho bits recesivos. Esta bandera indica a las estaciones cuando pueden empezar nuevamente la transmisión después de que algún nodo emitió una bandera de sobrecarga.

#### *1.1.4.1.5 Espaciamiento entre tramas*

El *espaciamiento entre tramas* puede ser usado tanto en tramas de formato estándar o extendido y es una secuencia de bits que sirve como separación de tramas precedentes de datos y remotas en una transmisión.

El espaciamiento entre tramas está compuesto de una secuencia de bits en estado recesivo, estos bits determinan una interrupción. Esta interrupción es provista para que los nodos realicen sus procedimientos internos correspondientes, antes de empezar la transmisión de un nuevo mensaje. Después de presentada la interrupción, los nodos CAN se colocan en estado recesivo hasta que la siguiente transmisión empiece.
Las tramas de error y las tramas de sobrecarga no poseen este espaciamiento de tramas, su transmisión es consecutiva.

Campos del espaciamiento entre tramas

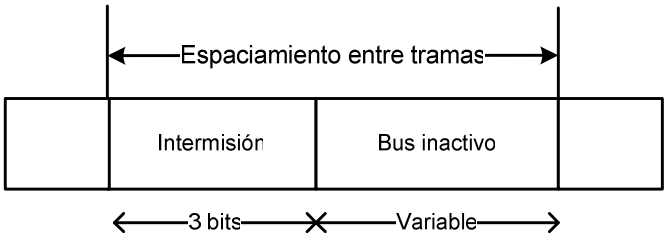



**Figura 1.20:** Campos del espaciamiento entre tramas en un receptor[30]

[30] Fuente: CAN Specification 2.0, Part B - CAN in Automation (CiA)

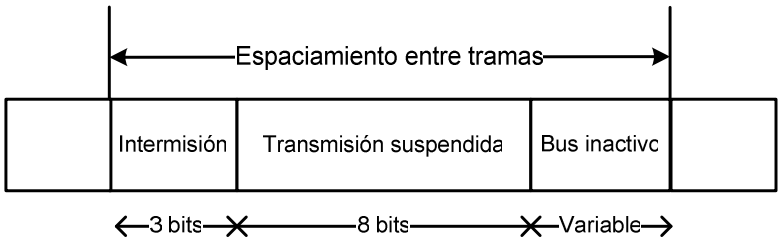



**Figura 1.21:** Campos del espaciamiento entre tramas en un transmisor[31]

El espaciamiento entre tramas consta de tres campos:

***Intermisión.-*** Consiste de tres bits en estado recesivo. Cuando una intermisión ocurre, se indica una condición de sobrecarga y no se admite realizar acciones como transmitir tramas de datos o remotas por ningún nodo hasta que se termine la sobrecarga.

***Bus inactivo.-*** Este campo tiene una longitud arbitraria y permanece en estado recesivo hasta que un nodo inicie la transmisión de un nuevo mensaje.

***Transmisión suspendida.-*** Consiste de una secuencia de ocho bits, este espacio tiene el propósito de proveer un tiempo para inhabilitar la transmisión de los nodos que están en estado de error pasivo.

### 1.1.5 MANEJO DE MENSAJES[2]

El manejo de mensajes implica el proceso de recibir tramas, filtrarlas en base al identificador, y almacenarlas en la memoria del controlador CAN para ser procesadas o descartarlas. Para implementar el proceso de filtrado de mensajes, se registra una máscara en la llamada “memoria intermedia” del controlador CAN la cual, permite seleccionar los grupos de identificadores que serán mapeados con los mensajes recibidos en el buffer. Las funciones de filtrado, son independientes de la versión del protocolo que se utilice, se puede realizar el mismo procedimiento de filtrado tanto para tramas estándar como para extendidas.

[31] Fuente: CAN Specification 2.0, Part B - CAN in Automation (CiA)

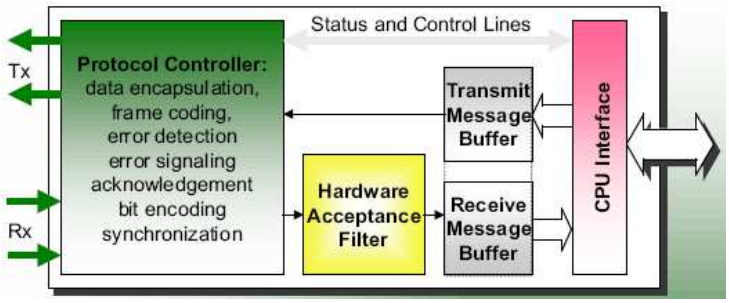



**Figura 1.22:** Arquitectura general de un controlador CAN[32]

De acuerdo a la manera de hacer el filtrado, existen dos tipos de controladores BasicCAN y FullCAN:

### 1.1.5.1 Controlador BasicCAN[1]

Se implementa en hardware una función básica de filtrado y administración de mensajes. Un controlador BasicCAN provee un buffer de transmisión para mensajes salientes y uno o dos buffers de recepción para mensajes entrantes.

El filtro de recepción es simple y permite que tramas con ciertos identificadores se almacenen en el buffer. El controlador es interrumpido para procesar a cada mensaje CAN entrante, la transmisión de un mensaje ocurre de manera similar, se inicia cuando se ha producido un cambio en el buffer de transmisión, esto conlleva a tener un CPU en permanente ocupación, por este motivo los dispositivos CAN que cuenten con este tipo de controladores son utilizados para aplicaciones de bajas velocidades y con pocos tipos de mensajes.

*Arquitectura de un controlador BasicCAN* [33]

El controlador BasicCAN posee un selector de identificador de mensaje (Selector ID-Rx) y una máscara (Máscara ID-Rx), ambos parámetros son definidos por el usuario al momento de configurar la interfaz BasicCAN. Cuando una trama es recibida, se compara su identificador con el Selector ID-Rx y se descarta cuando no coincide con la Máscara ID-Rx, si este proceso es exitoso, el mensaje es almacenado en la memoria intermedia (ID-Rx/Reg. Datos) para que pueda ser procesado por el controlador CAN. En vista de que pueden existir varios mensajes

[32] Fuente: CiA Am Weichselgarten

[33] Ver Figura 1.23

que se filtren exitosamente, la memoria intermedia tiene la capacidad de almacenar más de un mensaje. Cuando un mensaje va a ser procesado, el registro de control (Control) genera una interrupción, para solicitar atención al CPU. De manera similar, cuando existe un desbordamiento, si el controlador no puede atender todas las solicitudes de los mensajes entrantes, se genera una advertencia. Para el proceso de transmisión, se almacena el identificador y los datos del mensaje en el registro de transmisión (Mensaje Tx) y se transmite cuando se gane el acceso al canal.

#### 1.1.5.2 Controlador FullCAN[1]

Se implementan funciones más complejas en hardware para el filtrado y administración de mensajes. Un controlador FullCAN posee una memoria para cada mensaje a ser transmitido o recibido, en la cual se almacena toda la información referente al mensaje como el identificador, bytes de datos, ACK, etc., para que durante la inicialización del dispositivo, el CPU defina el mensaje que será enviado o recibido. Solo si el controlador CAN recibe un mensaje que tenga como identificador alguno de los identificadores almacenados o programados, interrumpirá al CPU, de otro modo se ignora el mensaje. Este tipo de controlador es utilizado para dispositivos con altos requerimientos de velocidad y grandes cantidades de mensajes.

*Arquitectura de un controlador FullCAN* [34]

El controlador FullCAN valida cada trama de manera individual en el selector de identificador, por esta razón no posee una máscara (Máscara ID-Rx) y tiene más de un buffer en donde se almacena cada mensaje, además, se los puede configurar para que actúen en transmisión o recepción lo que añade flexibilidad a esta arquitectura. Cada mensaje recibido o por transmitir, es almacenado por completo en la memoria intermedia, para cada mensaje nuevo recibido, se compara con los mensajes almacenados en la memoria y se sobrescriben si existen uno con el mismo identificador, de esa manera siempre se tiene en el buffer el mensaje más reciente. La carga del CPU disminuye significativamente ya que no se ve obligado a leer la memoria intermedia cuando arriba un nuevo

[34] Ver Figura 1.23

mensaje, esto hace que cuando se reciban varios mensajes con identificadores diferentes en tiempos cortos, el controlador CAN los distribuya y almacene en las diferentes memorias, a diferencia del controlador BasicCAN, este proceso se realiza sin la intervención del CPU. Los dispositivos FullCAN poseen hasta 16 memorias intermedias. En la actualidad muchos controladores FullCAN proveen una característica BasicCAN.

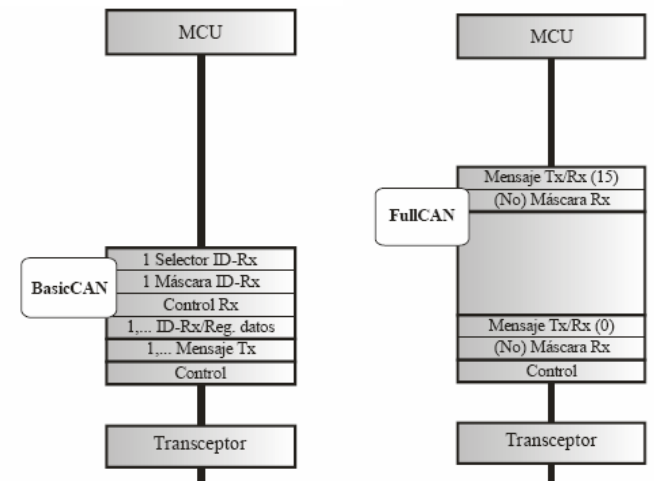



**Figura 1.23:** Arquitectura de un controlador BasicCAN y de un controlador FullCAN[35]

### 1.1.6 VALIDACIÓN DE MENSAJES

Para determinar si una trama no es válida, se debe tomar en cuenta si el nodo está actuando como transmisor o receptor.

***Transmisor:*** Un mensaje es válido para el transmisor, si no existe error hasta el final de la trama (EOF). Si un mensaje tiene error, se llevará a cabo el proceso de retransmisión automática. De acuerdo a la prioridad del mensaje se determina el acceso al bus de comunicaciones tan pronto como sea posible.

***Receptor:*** Un mensaje es válido para el receptor, si no existe error hasta el siguiente bit posterior al final de la trama (EOF). El valor del último bit de la trama no es importante, un valor dominante no determina un error en la trama.

[35] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

### 1.1.7 CODIFICACIÓN DE TRAMAS

El protocolo CAN utiliza el mecanismo de codificación de bit No Retorno a Cero[36] (NRZ) el cual establece un solo nivel de señal, recesivo o dominante, durante todo un tiempo de bit.

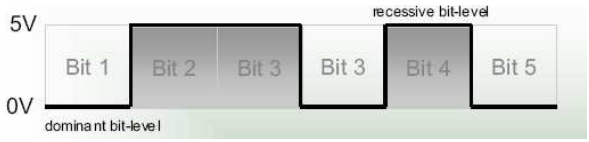



**Figura 1.24:** Codificación de bit No Retorno a Cero (NRZ)[37]

Este tipo de codificación presenta la ventaja de utilizar una frecuencia de operación menor, con respecto a utilizar otros tipos de codificación como Manchester, en donde se presenta una transición a medio tiempo de bit.

En el caso de transmitir muchos bits consecutivos de la misma polaridad, se puede tener problemas de sincronización, es por esta razón que la codificación se ve complementada con el método de inserción de bits (*bits-stuffing*) asegurando que en toda la transmisión no se presenten más de cinco bits consecutivos de la misma polaridad.

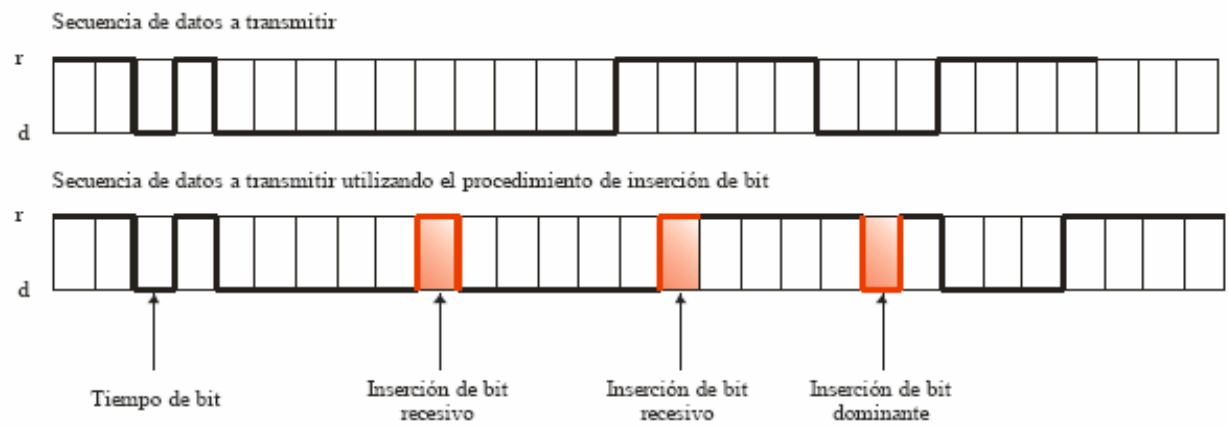



**Figura 1.25:** Mecanismo de inserción de bits[38]

[36] No Retorno a Cero (NRZ).- Tipo de codificación digital en donde el nivel de la señal es siempre positivo o negativo.
[37] Fuente: Interfaz Design Vehicle Busses – University of Tehran
[38] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

Los campos SOF, arbitraje, control, datos y secuencia CRC de cada trama de datos y remota, es codificada con el método de inserción bits, los campos restantes, delimitador CRC, ACK y EOF poseen un formato fijo y no siguen el procedimiento de inserción de bits; de manera similar, las tramas de error y sobrecarga poseen un formato fijo y no se codifican.

### 1.1.8 MANEJO DE ERRORES[5]

El manejo de errores tiene mucha importancia para el correcto desempeño de un sistema. El protocolo CAN permite la detección de errores cuando aparecen en un mensaje, y la retransmisión del mensaje errado tan pronto como sea posible.

Cuando un error es encontrado, el nodo transmite una trama de error para que de acuerdo al tipo de falla, se tomen las acciones apropiadas.

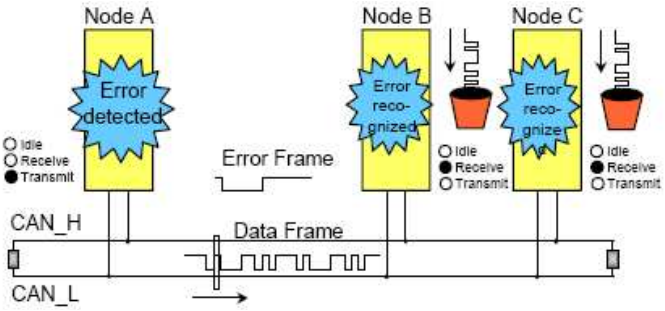



**Figura 1.26:** Manejo de errores en el protocolo CAN[39]

#### 1.1.8.1 Detección de Errores

El protocolo CAN provee un sofisticado mecanismo de detección de errores. Se pueden detectar cinco tipos de errores diferentes:

- Error de bit
- Error de relleno
- Error de CRC
- Error de forma
- Error de ACK

[39] Fuente: CANPRES V2.0 - Siemens

#### *1.1.8.1.1 Error de bit*

Una unidad realiza monitoreo de bits enviando un solo bit a través del bus, un error en el bit es detectado cuando el valor del bit monitoreado es diferente al valor del bit enviado. Una excepción a esta regla ocurre cuando se emite un bit en estado recesivo en el campo de arbitraje o en el slot ACK, en este caso no se genera error cuando se monitorea el bus y se detecta un bit en estado dominante. Un transmisor puede sobrescribir el bit recesivo por un bit dominante para ganar el arbitraje del bus.

#### *1.1.8.1.2 Error de relleno*

Si se detectan seis bits consecutivos con la misma polaridad entre los campos inicio de trama y delimitador CRC, ocurrió un error al aplicar el método de relleno o inserción de bits, esto hace que se genere una trama de error, por lo que el mensaje debe ser retransmitido.

#### *1.1.8.1.3 Error de CRC*

El transmisor calcula un valor para la secuencia de bits desde el campo inicio de trama (SOF) hasta el final del campo de datos y es transmitida en el campo CRC de la trama. El nodo receptor también calcula la secuencia CRC utilizando la misma fórmula y realiza la comparación con la secuencia recibida.

Se presenta un error si el nodo receptor detecta que la secuencia CRC calculada y la secuencia CRC recibida no son las mismas. El nodo receptor descarta el mensaje y transmite una trama de error solicitando retransmisión de la trama errada.

#### *1.1.8.1.4 Error de forma*

Un error de forma es la existencia de uno o más bits no válidos en los campos de la trama que poseen un formato fijo. Si el transmisor detecta un bit dominante en los segmentos: delimitador CRC, delimitador ACK, fin de trama (EOF) o espaciamiento entre tramas, esto se traduce en un error de forma, se genera entonces una trama de error y el mensaje deberá ser retransmitido.

*1.1.8.1.5 Error de ACK*

La verificación de acuse de recibo (ACK), consiste en que el transmisor chequee el campo ACK de un mensaje para determinar si el slot ACK contiene un bit dominante, de ser así, al menos un nodo, sea que le haya interesado o no el mensaje, ha recibido la trama de manera correcta.

Se presenta un error de ACK cuando los nodos receptores monitorean el slot ACK y éste se mantiene en estado recesivo, en ese caso, el mensaje debe ser retransmitido aunque no se haya generado ninguna trama de error.

### 1.1.8.2 Señalización de Errores

El protocolo CAN posee la capacidad de señalizar los errores detectados emitiendo una "bandera de error". Para un nodo que tenga un "error activo" [40] se emite la bandera de error activo, para un nodo que tenga un "error pasivo" [41] se emite la bandera de error pasivo.

Siempre que un nodo detecte un error de bit, de relleno, de forma o de ACK, se transmite una bandera de error en el siguiente bit después del error detectado.

Siempre que se detecte un error de CRC, se transmite una bandera de error después del delimitador de ACK, a menos que una bandera de error haya sido emitida por una condición de error detectada anteriormente.

En condiciones especiales, existe la posibilidad de no detectar errores locales, en ese caso no se emitirá bandera de error.

## 1.1.9 ESTADOS DE ERROR DEL NODO CAN[2]

Cada nodo puede encontrarse en uno de tres estados:

- Error activo
- Error pasivo
- Bus Apagado

[40] Ver Estados de error del nodo CAN pag.32
[41] Ver Estados de error del nodo CAN pag.32

Cada nodo implementa dos tipos de contadores: contador de error de transmisión (TEC) y contador de error de recepción (REC).

Los nodos pasan de uno a otro estado de acuerdo al valor de sus contadores de error internos.

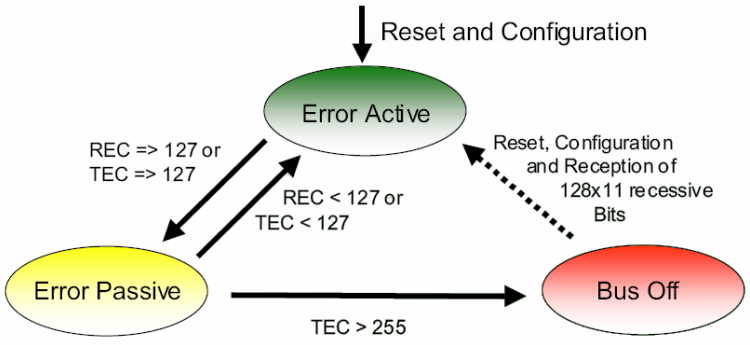



**Figura 1.27:** Estados de error del nodo CAN[42]

### 1.1.9.1 Error activo

En este estado se encuentran los nodos después de haber sido reinicializados, las unidades están en capacidad de transmitir y recibir mensajes, de igual forma puede emitir mensajes de error sin ningún tipo de restricción. En una comunicación, los contadores de error son actualizados de acuerdo a ciertas reglas. Para cada error en la recepción de los mensajes, los contadores de error son incrementados de su valor inicial. En cada transacción exitosa, los contadores de error son disminuidos de su valor inicial. El estado de error activo es válido mientras ambos contadores son menores o iguales a 127.

### 1.1.9.2 Error pasivo

Si los contadores de error del receptor y del transmisor han alcanzado el valor de 128, entonces los nodos pasan al estado de error pasivo. En este estado, los mensajes todavía pueden ser transmitidos y recibidos, sin embargo, después de la transmisión de un mensaje el nodo debe suspender la transmisión, debe esperar 8 tiempos de bit antes de que pueda transmitir otro mensaje. En términos de señalización, en este estado solo se puede transmitir tramas de error pasivas es decir, tramas de error con bits recesivos.

[42] Fuente: CiA Am Weichselgarten

#### 1.1.9.3 Bus apagado

Una característica del protocolo CAN es que los nodos tienen la capacidad de retirarse de la transmisión automáticamente. Cuando los contadores de error excedieron el valor de 255, el bus se coloca en estado apagado. Todas las actividades se detienen y temporalmente la estación no puede participar de la comunicación. Durante este estado, se puede recibir mensajes pero no transmitirlos. Para retornar al estado activo y encerar los valores de los contadores, los nodos deben ser reinicializados.

### 1.1.10 TEMPORIZACIÓN DEL BIT CAN[2]

La flexibilidad para determinar los parámetros de velocidad de transferencia es una característica importante del protocolo CAN, para entender mejor este concepto, se considera lo siguiente:

***Tiempo de bit ($t_b$).-*** Es el tiempo de duración de un bit

***Velocidad de transferencia nominal ($f_b$).-*** Es el número de bits por segundo que un transmisor ideal emite sin resincronización.

***Tiempo de bit nominal.-*** Es la relación inversa entre el tiempo de bit y la velocidad de transferencia nominal.

$$t_b = \frac{1}{f_b}$$

Cada bit nominal en el bus CAN, es dividido en 4 segmentos de tiempo no superpuestos, y son:

- Segmentos de sincronización (sync_sec)
- Segmentos de propagación (prop_sec)
- Buffer de fase del segmento1 (phase_seg1)
- Buffer de fase del segmento2 (phase_seg2)

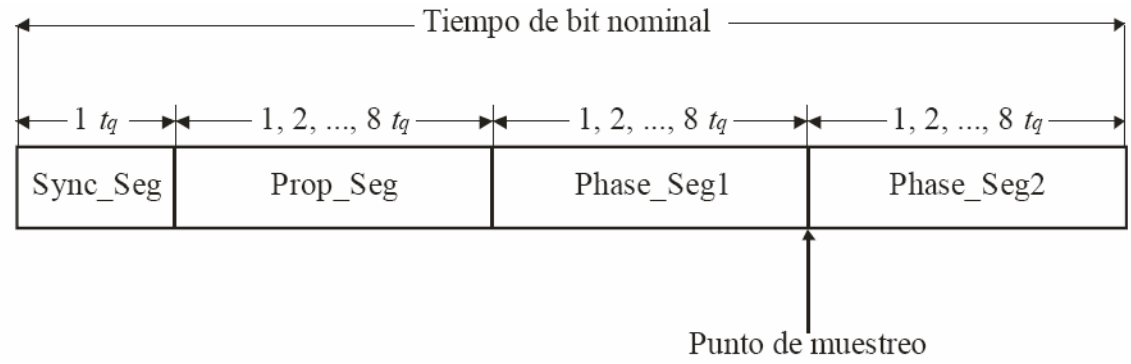



**Figura 1.28:** Segmentos de un tiempo de bit[43]

***Tiempo quantum.-*** Es una unidad básica de tiempo, es de valor fijo y se deriva del período del oscilador. Este valor es programable y puede tomar los valores de 1 a 32.

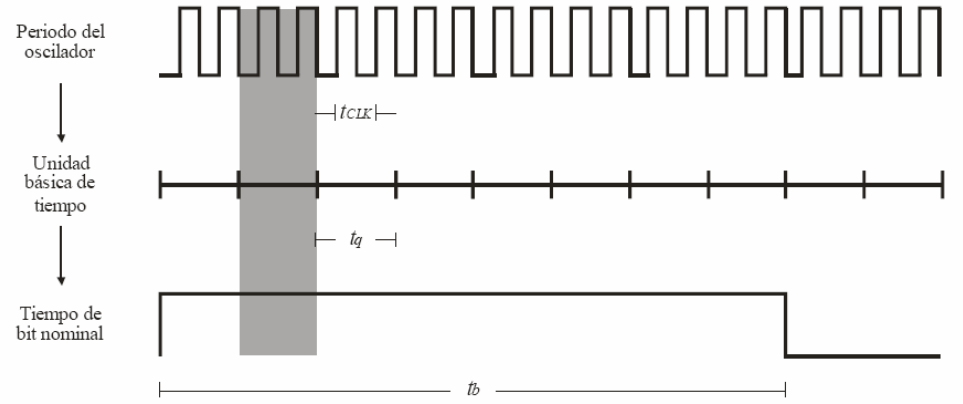



**Figura 1.29:** Principio de derivación del periodo de bit[44]

### 1.1.10.1 Segmentos de sincronización (sync_seg)

Es el primer segmento del tiempo de bit nominal y el cual es utilizado para la sincronización de los nodos en el bus. Su longitud es fija e igual a 1 tiempo quantum.

### 1.1.10.2 Segmentos de propagación (prop_seg)

Es utilizado para compensar el retardo de propagación de la señal en el bus y retardos inherentes a los nodos como los son los retardos del controlador del bus. Su longitud puede tomar los valores enteros de 1 a 8 tiempos quantum.

[43] - [44] Fuente: Desarrollo de un Sistema Educativo para la enseñanza del protocolo de comunicaciones CAN – Universidad Tecnológica de la Mixteca: Carlos Chamú

#### 1.1.10.3 Segmento de fase1 y fase2 (phase_seg1 y phase_seg2)

Estos segmentos, son utilizados para compensar errores de fase y su longitud puede variar para propósitos de resincronización.

La longitud del segmento *phase_seg1* puede tomar los valores enteros de 1 a 8 tiempos quantum.

La longitud del segmento *phase_seg2* puede tomar como valor mínimo el valor del tiempo de procesamiento de información y como valor máximo 8 tiempos quantum.

***Punto de muestreo.-*** Es el instante de tiempo en el cual se lee y se interpreta el nivel del bus para determinar el valor del bit. Se localiza después del segmento phase_seg1.

***Tiempo de procesamiento de la información.-*** Es el segmento de tiempo comenzando desde el punto de muestreo reservado para calcular el nivel subsecuente del bit. Su longitud puede variar de 0 a 2 tiempos quantum.

El total de tiempos quantum en un bit puede ser de 8 hasta 25.

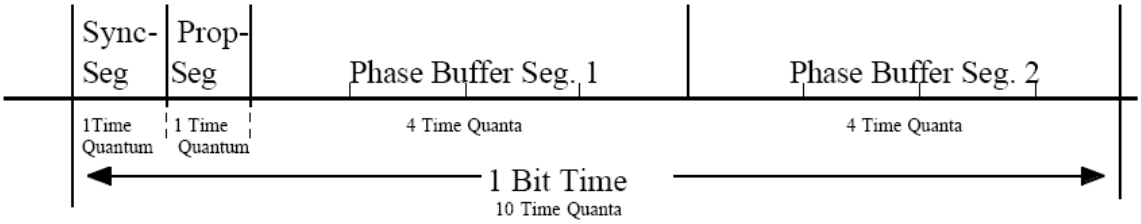



**Figura 1.30:** Definición de un tiempo de bit[45]

### 1.1.11 REQUERIMIENTOS DE SINCRONIZACIÓN[2]

El protocolo CAN maneja la transferencia de mensajes de manera sincronizada. Todos los nodos se sincronizan con el principio de cada mensaje utilizando el campo inicio de trama (SOF), pero este mecanismo no es suficiente. Para

[45] Fuente: CAN Specification 2.0, Part B - CAN in Automation (CiA)

asegurarse del muestreo correcto hasta del último bit, los nodos CAN necesitan resincronización continua a través de toda la trama.

Cada nodo se sincroniza con su oscilador propio, pero en este proceso, pueden ocurrir desplazamientos de fase, para esto el protocolo CAN proporciona mecanismos de compensación.

Previo al estudio de las dos formas de sincronización especificadas por el protocolo CAN, se definirá los siguientes conceptos:

***Error de fase de flanco.-*** Está dado por la posición del flanco relativa al segmento sync_seg medida en tiempo quantum.

La señal de error de fase está definida por:

- e = 0, si el flanco es detectado dentro del segmento
- e > 0, si el flanco es detectado antes del punto de muestreo
- e < 0, si el flanco es detectado después del punto de muestreo del bit previo

***Resincronización.-*** Corresponde al valor programado de tiempos quantum que se suman al segmento phase_seg1 o se reducen al segmento phase_seg2.

#### 1.1.11.1 Tipos de Sincronización

El protocolo CAN, define dos tipos de sincronización:

*1.1.11.1.1 Sincronización al inicio de la trama (Sincronización forzada)*

Ocurre en una transición recesivo-dominante, se obliga al flanco a iniciar con el segmento sync_seg al inicio del bit.


**Figura 1.31:** Sincronización al inicio de la trama[46]

[46] Fuente: CiA Am Weichselgarten

*1.1.11.1.2 Resincronización dentro de la trama (RJW)*

Ocurre cuando el flanco no se encuentra dentro del segmento de sincronización en un mensaje. Uno de los segmentos de la fase se puede alargar (phase_seg1) o se puede acortar (phase_seg2) un determinado valor que depende del error de fase en la señal. El valor máximo que puede variar está definido por el parámetro RJW (Resynchronization Jump Width).

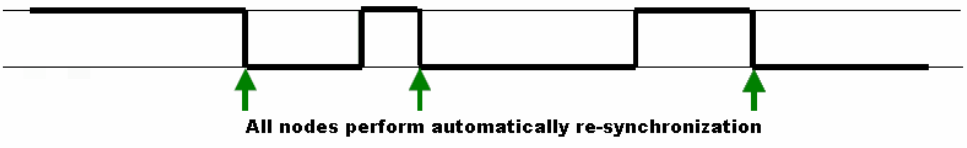



**Figura 1.32:** Re-sincronización automática[47]

Si la magnitud del error de fase es mayor al parámetro RJW, se debe considerar:

- *Si el error de fase es positivo:* el segmento phase_seg1 se prolonga una cantidad igual al valor de RJW (1 a 4 tiempo quantum).
- *Si el error de fase es negativo:* el segmento phase_seg2 se reduce una cantidad igual al valor RJW (1 a 4 tiempo quantum).

### 1.1.11.2 Reglas de sincronización

1. Una única sincronización es permitida dentro de cada tiempo de bit.
2. Un flanco será utilizado para sincronización solo si el valor detectado en el punto previo de muestreo difiere del valor del bus inmediatamente después del flanco.
3. La sincronización debe realizarse siempre que se tenga una transición recesivo-dominante, y cuando se encuentre desocupado el bus.

Los flancos recesivos a dominantes que sigan las reglas 1 y 2, serán usados para resincronización, a excepción de que un nodo se encuentre transmitiendo un bit dominante, no realizará resincronización como resultado de un flanco recesivo a dominante con error de fase positivo, si solo son utilizados para propósitos de resincronización flancos recesivos a dominantes.

[47] Fuente: CiA Am Weichselgarten

## 1.1.12 APLICACIONES

### 1.1.12.1 Industria automotriz

La necesidad de reducir el cableado en los automóviles, fue la motivación del desarrollo del protocolo CAN. Gracias a este protocolo, la industria automotriz ha tenido una notable evolución en los últimos años. La introducción de sistemas de control independiente y altamente integrado a las redes, ha permitido que se añadan funcionalidades en los vehículos tan complejas como el diagnosticar mediante sensores los sistemas de los cuales está compuesto, o tan simple como el procesar la información que se puede recibir al bloquear las puertas o ventanas del automóvil.

Entre los sistemas que comúnmente encontramos en los vehículos actuales se tiene:

- *Sistemas electrónicos inteligentes.-* Conocidos como “smart electronics”, estos sistemas se encargan de enviar y recibir información del exterior del vehículo para informar al conductor y a las unidades de control electrónica del automóvil. Estos sistemas requieren de una comunicación confiable y de gran ancho de banda ya que manejan información referente al control de velocidad, aceleración, desaceleración, frenado, etc., además, también se pueden integrar con dispositivos en el interior del vehículo como bolsas de aire, medidores de tensión en los cinturones de seguridad etc.
- *Frenos inteligentes.-* Compañías como Chrysler, están diseñando en base al protocolo CAN, un sistema de frenado inteligente en el cual, la unidad central del vehículo calcula el frenado exacto mediante parámetros emitidos por sensores tales como la velocidad de los neumáticos o su ángulo de dirección.
- *Control de telemática.-* Consiste en la administración de sistemas de entretenimiento dentro del vehículo mediante buses específicos para audio y video.
- *Sistemas X-by-wire.-* Estos sistemas surgieron a partir del aparecimiento del protocolo CAN, sirven para dar soporte al vehículo tratando de eliminar sistemas mecánicos e hidráulicos basándose únicamente en sistemas electrónicos. El sistema de frenado es un ejemplo de que en la actualidad un vehículo no es totalmente electrónico, si bien cuenta con sistemas como

sensores que emiten información del pedal a la unidad central electrónica, aún se conserva los mismos actuadores hidráulicos o neumáticos para completar el frenado. El propósito de los sistemas X-by-wire es desarrollar un vehículo más seguro, confortable y ecológico.

#### 1.1.12.2 Sistemas domóticos

La adquisición de datos en sistemas domóticos, consiste en analizar la información generada por múltiples sensores en una o varias unidades centrales, basadas en microcontroladores, para tomar acciones de acuerdo a la información procesada.

- *Edificios inteligentes.-* Consiste en conformar redes CAN, instalando nodos en sistemas de calefacción, aire acondicionado, alarmas contra incendios, puertas, ventanas etc., para el control general del estado del edificio.

En la actualidad los lugares de aplicación del protocolo, en el área de la domótica, son incontables, para citar algunos de ellos, se puede tener en supermercados, instituciones, hospitales, edificios, bancos, etc., funciones de oscurecimiento de ventanas, control de acceso a cuartos, acceso de puertas, control del iluminación exterior o interior, sistema de alarma, sistemas rociadores y aspersores, etc.

#### 1.1.12.3 Automatización industrial

En el campo de la industria, muchas fábricas han incursionado en la automatización de su línea de producción, para estas tareas también se puede utilizar el protocolo CAN no solo para optimizar el tiempo de producción si no para mejorar condiciones higiénicas entre otras tareas.

- *Línea de producción.-* Empresas como embotelladoras, tabacaleras, cerveceras, empacadoras de comida, etc., automatizan su línea de producción para obtener reducción de costos de instalación, mejoras en el sistema de mantenimiento y control, etc.

#### 1.1.12.4 Otras aplicaciones

*Sistemas marítimos.-* Ciertas aplicaciones como botes deportivos, embarcaciones y barcos, poseen sistemas automatizados basados en el protocolo CAN, subsistemas con interfaz CAN y dispositivos sencillos con conectividad CAN.

Las embarcaciones actuales poseen sistemas como GPS, radares, sensores de seguridad, etc., todos estos sistemas se encuentran interconectados a través de un bus CAN. Protocolos de capa aplicación como CANKingdom y CANOpen, se utilizan mucho en el mundo de las aplicaciones marítimas ya que ofrecen un equipo naval electrónico con altas prestaciones.

*Sistemas ferroviarios.-* En el año 2002 se propuso la idea de estandarizar un protocolo para sistemas de componentes electrónicos y redes empotradas en vagones de tren, en la actualidad se utiliza para estos fines los protocolos WTB (Wide Train Bus) y MVB (Multiple Vehicle Bus), cabe destacar que el protocolo CAN ha servido de inspiración para estos y otros sistemas.

*Sistemas médicos.-* El protocolo CAN ha sido utilizado para el diseño de instrumentos de diagnóstico médico que poseen la capacidad de supervisar simultáneamente un gran número de muestras para detectar la presencia de microorganismos o bacterias, estas muestras son monitoreadas mediante sensores de presión, temperatura entre otros, optimizando con esto, las tareas en un laboratorio clínico.

## 1.2 BUSES DE COMUNICACIÓN UTILIZADOS PARA APLICACIONES CON MICROCONTROLADORES

### 1.2.1 CONCEPTOS BÁSICOS[9]

Un bus de comunicaciones es un conjunto de conductores eléctricos que pueden tomar la forma de pistas metálicas en el caso de tarjetas impresas o simplemente tomar la forma de un conjunto de cables a través de los cuales se transfiere datos. En un bus, todos los nodos reciben la información a manera de difusión, los nodos a los que no vaya dirigida esta información simplemente la ignoran.

Los buses se encuentran presentes en casi todos los sistemas de comunicaciones, dentro de un computador personal por ejemplo, la información

entre los diferentes dispositivos de hardware como los son disco duro, memoria flash, monitor, teclado, etc. viaja a través de buses.

De acuerdo a las posibilidades de comunicación a través de un bus, existen dos tipos de buses:

- Paralelo
- Serial

#### 1.2.1.1 Bus paralelo

La comunicación a través de este bus se realiza mediante bits simultáneos. Este tipo de bus permite transportar datos a gran velocidad, aunque para ello se necesite un gran número de líneas, lo que resulta en una desventaja frente a los buses seriales.

#### 1.2.1.2 Bus serial

La comunicación a través de este bus es realizada bit por bit. Este bus permite obtener mayores alcances y en ciertos casos grandes velocidades.

Existen periféricos que no necesitan ser muy rápidos, en estos casos se utiliza buses seriales, este tipo de buses minimizan el número de conductores, pines y tamaño del circuito integrado, esto se traduce en reducción de costos en la fabricación de dispositivos, es por esto que se cuenta con muchos periféricos seriales presentes en la actualidad.

En vista de la gran importancia de los buses seriales, se pondrá mayor énfasis en el estudio de los mismos.

### 1.2.2 CLASIFICACIÓN DE LOS BUSES SERIALES PARA SISTEMAS EMBEBIDOS[10,11]

Un sistema embebido es un conjunto de dispositivos utilizados para controlar equipos, maquinaria o en general cualquier dispositivo. El término embebido refleja que todos elementos del circuito de control, microcontroladores y sus periféricos, se encuentran en un conjunto integral o en un módulo.

Los buses seriales son ampliamente utilizados en este tipo de sistemas ya que permiten la comunicación entre un microcontrolador y sus periféricos, o mejor aún, permiten la interconexión entre microcontroladores.
Por la importancia que tiene los buses seriales en la industria de los microcontroladores, se hará un análisis de los más comunes disponibles en el mercado.

De acuerdo a la forma de transmisión de la información, los buses seriales se clasifican en:

- Asincrónicos
- Sincrónicos

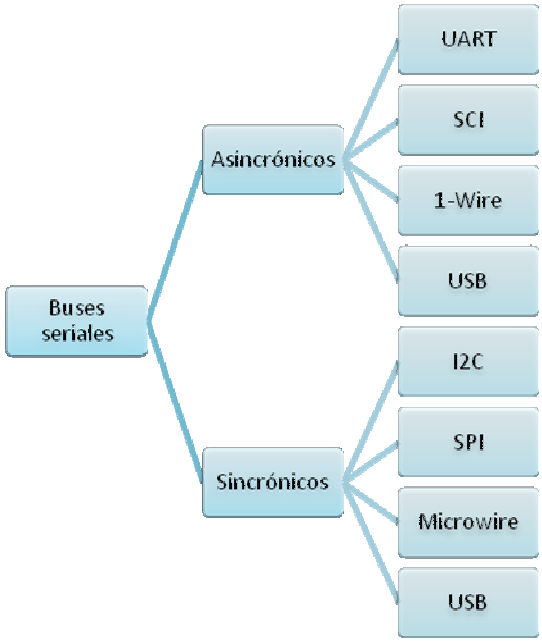



**Figura 1.33:** Clasificación de los buses seriales

### 1.2.2.1 Buses Asincrónicos

#### *1.2.2.1.1 Buses UART y SCI*

*Generalidades*

*UART (Universal Asynchronous Receiver Transmitter)* es un bus de comunicaciones serial, asincrónico, full-duplex, comúnmente utilizado conjuntamente con otros estándares para comunicación seriales como el EIA RS-232. Generalmente un UART se implementa como circuito integrado o parte de

uno y se lo incluye en microcontroladores para lograr obtener una comunicación serial entre un computador y un dispositivo periférico o entre microcontroladores.

Muchos circuitos integrados modernos incluyen un UART que también puede comunicarse sincrónicamente, estos se denominan USART.
La versión más reciente de este bus, introduce mejoras como calibración de velocidad, capacidad de auto-wake-up cuando el microcontrolador se encuentra en estado de reposo, etc., esta nueva versión de denomina EUSART (Enhanced USART).
Un UART tiene la capacidad de transmitir y recibir datos seriales usando diferentes velocidades. Las velocidades estándar varían desde 110 hasta 921600bps.

*SCI (Serial Communication Interfaz)* es un bus serial asincrónico, full-duplex, posee muchas similitudes con el bus UART, por lo tanto, se analiza de manera general el bus SCI, considerando que la diferencia más notable entre ambos es el número de bits de datos enviados en la trama.

*Señales del bus*

- RxD es la línea utilizada por un dispositivo para recibir información.
- TxD es la línea utilizada por un dispositivo para enviar información.

*Forma de operación*

Por convención se envía un bit *start* para marcar el inicio de una trama, este bit es enviado en nivel bajo, seguido de los correspondientes bits de datos, que en el caso de SCI será de 8 a 9 bits y para UART será de 5 a 8 bits, enviando el bit menos significativo primero, un bit de paridad opcional y uno o uno y medio bits de parada o *stop*. El bit de parada tiene polaridad opuesta al estado de la línea de datos, sirve para marcar el final de la trama, además provee un retardo antes de que se envíe el siguiente carácter lo que ayuda a la resincronización entre dispositivos.

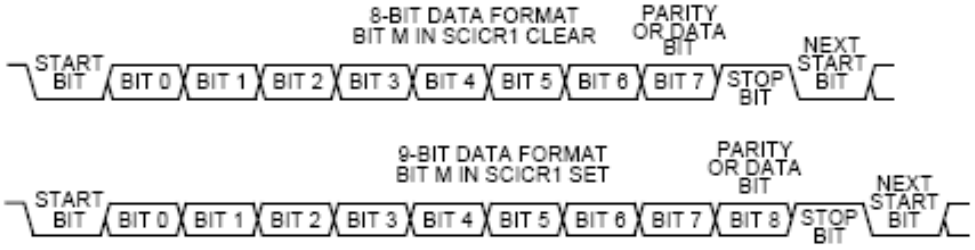



**Figura 1.34:** Formato de datos del bus SCI[48]

*1.2.2.1.2 Bus 1-Wire*

*Generalidades*

El *bus 1-wire* es un bus serial asincrónico desarrollado por Dallas/Maxim con el propósito de simplificar los sistemas de redes de control utilizando un único cable de par trenzado en comunicaciones bidireccionales, esto se logra enviando por el mismo cable tanto las señales de datos como la de alimentación.

Las redes 1-Wire son del tipo maestro/esclavo, el control de acceso al medio es tarea exclusiva del maestro, no se autoriza a ningún nodo a transmitir sin que el maestro lo haya solicitado.

Debido a que los dispositivos 1-Wire existentes en el mercado son unidireccionales, una aplicación típica requiere de dos hilos, uno para cada dirección de flujo de datos, sin embargo aún sigue considerándose una ventaja con respecto a otros buses ya que en aplicaciones que utilicen I2C y SPI, se requiere mínimo tres hilos. La mayoría de dispositivos 1-Wire pueden trabajar en dos velocidades, 15kbps y 111kbps.

*Señales del bus*

Los sistemas 1-Wire utilizan señales convencionales CMOS/TTL y la transmisión de bits se realiza mediante asignación de ranuras de tiempo, similar a codificar la señal por el método de amplitud de pulso.

[48] Fuente: SCI Block Guide V02.05 - Motorola

*Forma de operación*

El maestro inicia una secuencia de bits con un *reset,* el cual consiste en colocar al bus en un nivel bajo por un tiempo de 480µs para que los esclavos conectados al bus, respondan mediante pulsos de presencia de 60µs de duración. Para escribir la lógica de uno, el maestro pone en bajo el bus por 15µs o menos, para escribir la lógica cero, el maestro pone en alto al bus por 60µs aproximadamente. Para recibir datos, el maestro reinicia el bus poniéndolo en bajo el bus durante 15µs o menos. Si el esclavo transmisor quiere enviar un uno, no hace nada y el bus inmediatamente se recupera poniendo el nivel alto, si quiere enviar un cero, pone el bus en bajo durante 60µs.

No se requiere reloj del sistema ya que los dispositivos 1-Wire poseen sus propios osciladores internos que se sincronizan con el flanco negativo del maestro.
Si la presencia de pulsos es detectada, durante el acceso, se le asigna al esclavo una dirección única que utiliza el maestro para controlar la transferencia de información de cada uno y para la selección de esclavos. Una vez seleccionado el esclavo específico, el maestro envía y recibe datos de él, mientras tanto, todos los otros esclavos ignoran esta comunicación hasta que el maestro reinicie el bus.

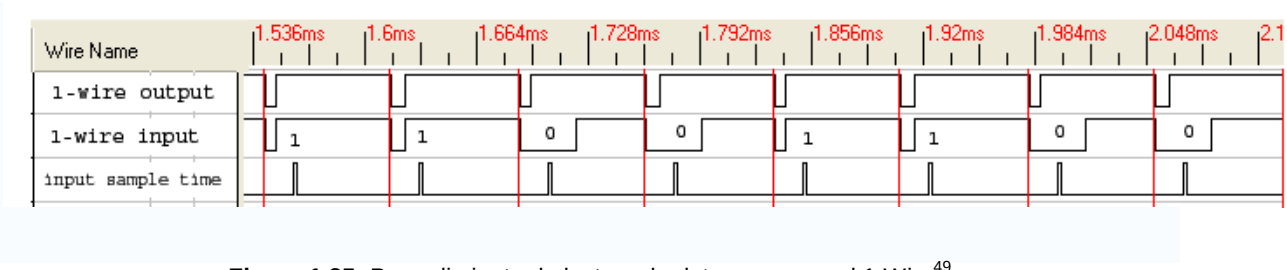



**Figura 1.35:** Procedimiento de lectura de datos en una red 1-Wire[49]

En la memoria ROM de cada esclavo, se almacena una dirección de dispositivo única que contiene 8bytes divididos en tres secciones: el primer byte almacena un código de 8bits que identifica al tipo de dispositivo, los siguientes 6bytes almacenan la dirección específica programable de cada dispositivo, el último byte contiene un código de redundancia (CRC) con un valor basado en la información

[49] Fuente: 1-Wire – Wikipedia

contenida en los 7 primeros bytes, esto permite al maestro determinar si una dirección fue leída sin error.
Se tienen disponibles 48bits para la dirección, por tanto es poco probable tener conflicto de nodos con direcciones duplicadas.
Se tiene un algoritmo para recobrar direcciones de cada dispositivo en el bus de manera que se tiene un inventario de dispositivos confiable.

Para buscar un dispositivo, el maestro envía por difusión un comando de enumeración, seguido de una dirección, después de escuchar la dirección, el esclavo a quien le corresponda responde al maestro.

### 1.2.2.2 Buses Sincrónicos

#### *1.2.2.2.1 Bus I2C*

*Generalidades*

Con el propósito de simplificar las interconexiones entre diferentes dispositivos como memorias o microcontroladores con sus periféricos; LCD, convertidores A/D o I/O de propósito general, Philips Semiconductor desarrolló el protocolo de comunicaciones *$I^2C$ (Inter-Integrated Circuits).* I2C define un bus serial sincrónico de comunicaciones que alcanza velocidades de hasta 100Kbps en modo estándar, 400Kbps en modo rápido (Versión 1.0) y hasta 3.4Mbps en modo de alta velocidad (Versión 2.0).

El número de dispositivos que se puede conectar a este bus teóricamente no tiene límites, pero se debe tener en cuenta que la capacidad máxima total de los nodos no supere los 400pF[50]. En la práctica, se pueden conectar hasta 8 dispositivos idénticos en el mismo bus. Los valores de las resistencia de polarización, puede ser entre 1.8KΩ a 47KΩ[51] aunque los más comunes son 1.8KΩ y 10KΩ.

[50 - 51] Fuente: The $I^2C$-Bus especification – Philips Semiconductor

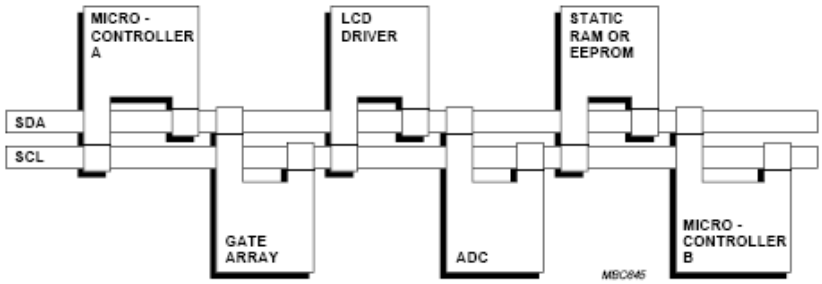



**Figura 1.36:** Interconexión de dos microcontroladores utilizando bus $I^2C$.[52]

*Señales del bus*

Para el intercambio de información, este bus utiliza dos líneas bidireccionales a través de las cuales viajan los datos e información de sincronización.

- *SDA (System Data)* es la línea por la que se transporta los datos de manera serial.
- *SCL (System Clock)* es la línea que transporta los pulsos de reloj para la sincronización.

Tanto SDA como SCL, son líneas de tipo drenaje abierto y deben ser polarizadas a positivo mediante una resistencia de "pull-up" cuando están inactivas, esto permite lograr una estructura en bus en donde se pueden conectar en paralelo múltiples entradas y salidas.

*Forma de operación*

La comunicación en el bus se basa en la lógica maestro/esclavo en donde sólo los maestros emiten la señal de reloj por tanto pueden iniciar la comunicación. Puede existir en el bus más de un maestro, en ese caso la configuración se denomina multimaestro, esto es posible gracias a un proceso de arbitraje.

Se puede dar inicio al intercambio de información, cuando el bus se encuentre en la condición de *bus libre*, en este estado, ambas líneas SDA y SCL se encuentran inactivas, es decir, en estado lógico alto. Un maestro informa a los esclavos del comienzo de la comunicación con la condición *start* o inicio en la cual la línea SDA toma un estado bajo mientras que la línea SCL permanece en alto. En este momento empieza la transferencia de información. Cuando finaliza la

[52] Fuente: The $I^2C$-Bus especification – Philips Semiconductor

comunicación, el maestro informa de esta situación mediante la condición de *stop* o parada en la cual las líneas SDA y SCL toman un estado lógico alto o de bus libre.

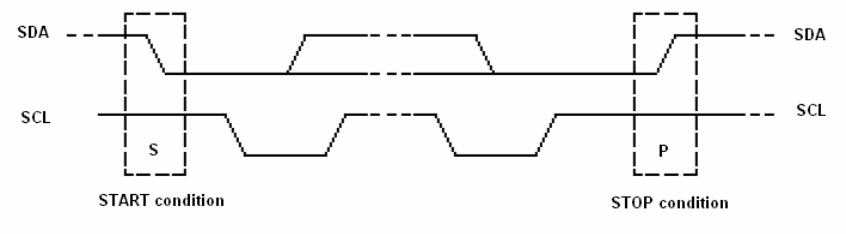



**Figura 1.37:** Condiciones start y stop[53]

El bus transmite tramas de 8 bits. Después de la transmisión de cada trama, se genera un reconocimiento (ACK) que consiste en colocar la línea SDA en un nivel lógico bajo durante el transcurso del noveno pulso del reloj.

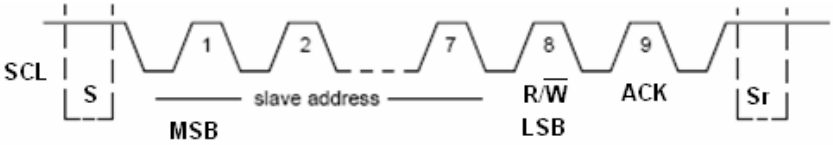



**Figura 1.38:** Formato de trama enviada por el bus I2C[54]

Los siete primeros bits corresponden a la dirección de acceso al dispositivo de los cuales 4 bits son fijos preestablecidos por el fabricante y 3 bits son variables. La dirección 00 corresponde a la dirección de acceso general, todos los dispositivos responden a esa dirección. El octavo bit después de la dirección, corresponde al bit de lectura/escritura (R/W), este bit indica el tipo de operación que se va a realizar, si el bit está configurado en 1L, el maestro lee la información proveniente de un esclavo, si el bit está en 0L, el maestro escribe información para un dispositivo esclavo determinado.

#### *1.2.2.2.2 Buses SPI y Microwire*

*Generalidades*

*SPI (Serial Peripheral Interface)* fue desarrollado por Motorola como un estándar de comunicaciones utilizado para la interconexión serial, sincrónica a tres hilos

[53] Fuente: The $I^2C$-Bus especification – Philips Semiconductor
[54] Fuente: The $I^2C$-Bus especification – Philips Semiconductor

(full-duplex) de microcontroladores con sus periféricos u otros microcontroladores. Este bus puede alcanzar velocidades de hasta 1Mbps.

*Microwire* es un subconjunto del bus SPI, fue desarrollado por National Semiconductor Corporation como un bus serial sincrónico utilizado para los mismos propósitos que el bus SPI, es por esta razón que tiene el mismo número de líneas pero con diferente notación en cada una y la forma de operación es análoga.

*Señales de los buses*

Ambos buses posee tres líneas para el intercambio de información y n líneas para la selección del dispositivo esclavo.

| Señales SPI | Señales Microwire | Descripción |
|---|---|---|
| *MOSI* (*Master Out Slave In)* | *SO (Serial Output)* | Línea unidireccional por la que se transmite datos desde el maestro hacia el esclavo |
| *MISO* (*Master In Slave Out)* | *SI (Serial Input)* | Línea unidireccional por la que se transmite datos desde el esclavo hacia el maestro |
| *SCKL (System Clock)* | *SK (Serial Clock)* | Línea por la que se transmite la señal del reloj |
| *SS (Slave Select)* | *CS (Chip Select)* | Línea que permite seleccionar distintos dispositivos |

**Tabla 1.10:** Analogía de las señales de los buses SPI y Microwire[55]

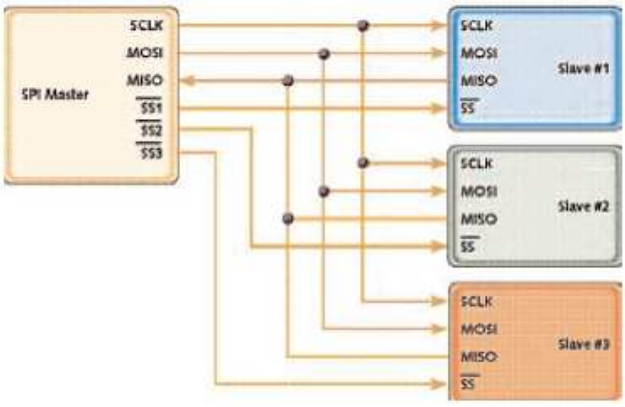



**Figura 1.39:** Configuración de dispositivos usando SPI[56]

[55 - 56] Fuente: Debugging Low-speed Serial Buses in Embedded System Designs - Tektronix

*Forma de operación*

El bus SPI es muy similar a I2C, tiene la misma lógica maestro/esclavo en donde la señal de reloj es generada por el maestro, sin embargo tener dos líneas de datos para entrada y salida independientes, hace que este bus sea mucho más sencillo que I2C.

Los dispositivos esclavos son seleccionados utilizando una línea *slave select*, usualmente es una línea dedicada para cada dispositivo.

SPI define cuatro modos de reloj de acuerdo a la polaridad (CPOL), determinada por el nivel del reloj (0L o 1L), y a la fase (CPHA), determinada por la ubicación del flanco del reloj (positivo o negativo), esto es importante ya que, la manera de leer y escribir los datos varía de acuerdo al modo del reloj de la siguiente manera:

- Cuando CPOL=0 el nivel del reloj es cero
  - Para CPHA=0, lectura de datos en flanco positivo (transición de bajo a alto), escritura de datos en flanco negativo (transición de alto a bajo).
  - Para CPHA=1, lectura de datos en flanco negativo, escritura de datos en flanco positivo.
- Cuando CPOL=1 el nivel del reloj es uno
  - Para CPHA=0, lectura de datos en flanco negativo, escritura de datos en flanco positivo.
  - Para CPHA=1, lectura de datos en flanco positivo, escritura de datos en flanco negativo.

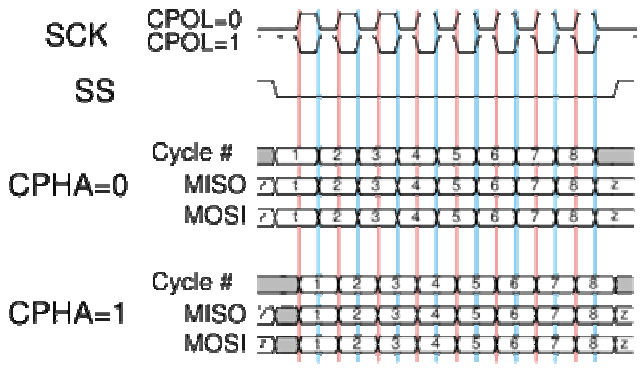



**Figura 1.40:** Diagrama de polaridad de reloj y fase en SPI.[57]

[57] Fuente: Serial Peripheral Interfaz Bus - Wikipedia

Este mecanismo, añade mayor flexibilidad en la comunicación entre maestros y esclavos, pero para que los dispositivos puedan comunicarse, deben estar configurados en el mismo modo de reloj, en nivel y fase.
Muchos dispositivos tienen la capacidad de trabajar en más de un modo respondiendo a valores específicos en la configuración de control.
La información puede ser transmitida en bloques múltiplos de 8, generalmente se emiten bloques de 8 o 16 bits.
El funcionamiento de Microwire, es similar al del bus SPI. Microwire define un único modo de reloj determinado por los valores de CPOL=0 y CPHA=0.
La información transmitida corresponde a 8 bits.

#### *1.2.2.2.3 Bus USB*

*Generalidades*

*USB (Universal Serial Bus)* fue desarrollado por IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC como un estándar de bus serial para la interconexión de dispositivos a un computador personal. USB es un bus que puede operar tanto en modo sincrónico como asincrónico, consiste en un maestro y múltiples dispositivos esclavos, conectados en una estructura de árbol utilizando concentradores especiales. Este bus soporta hasta 127 dispositivos, incluyendo concentradores, conectados a un solo maestro, en un límite de hasta 5 niveles.

El estándar USB incluye la transmisión de energía eléctrica al dispositivo conectado, ciertos concentradores proveen esta energía para dispositivos que requieren de potencias mínimas.
El propósito del diseño del USB fue el mejorar las capacidades plug-&-play de muchos dispositivos, permitiendo a estos ser conectados o desconectados sin necesidades de reiniciar el sistema.
El bus utiliza 4 hilos para la transferencia de señalización y alimentación. La señalización ocurre en sólo 2 hilos en cada segmento punto a punto.
USB soporta tres velocidades:

- ***Low Speed*** Velocidad de operación de 1.5 Mbps.
- ***Full Speed*** Velocidad de operación de 12 Mbps.

- ***Hi-Speed*** Velocidad de operación de 480 Mbps.

*Señales del bus*

El estándar USB utiliza NRZI (Non Return to Zero Inverted) como sistema de codificación de datos.

Las señales son transmitidas vía cables de par trenzado y consiste de dos señales:

- *D*+ transmite niveles altos de señal que oscilan entre 2.8 a 3.6 voltios.
- *D*- transmite niveles bajos de señal que oscilan entre 0.0 a 0.3 voltios.

Estas señales usualmente operan juntas y utilizan un sistema de señalización diferencial half-duplex para contrarrestar los efectos del ruido electromagnético en líneas extensas.

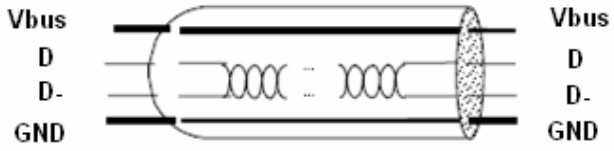



**Figura 1.41:** Cable USB[58]

*Forma de operación*

El maestro realiza un sondeo a los dispositivos, para controlar la transferencia de los datos. Una transacción está definida por tres paquetes. Un paquete USB que contiene el tipo y la dirección de transacción, la dirección del dispositivo y un número que es conocido como “token packet”. En una transacción determinada, los datos pueden viajar desde el maestro a los esclavos o desde los esclavos al maestro, la dirección de la transferencia de información está especificada en “token packet”. Una fuente puede emitir paquetes que contengan o no datos, estos serán respondidos con un “handshake packet” si la transferencia fue exitosa.

[58] Fuente: Universal Serial Bus Specification Revision 2.0 - VA

El modelo de transferencia utilizada por el bus USB, consiste en tener canales lógicos asociados o *pipes*. Los pipes son conexiones desde el maestro a la entidad lógica en el dispositivo llamada punto final o *endpoint*. Existen dos tipos de pipes: flujos y mensajes. Un flujo de datos no tiene una estructura definida, los mensajes de datos si la tienen.

Cuando un dispositivo es conectado por primera vez, el maestro lo enumera, lo reconoce y de ser necesario carga el controlador del mismo. La enumeración consiste en la asignación de una dirección única para poder sondear a los dispositivos según el modelo round-robin[59].

Una característica importante del bus USB es el soporte de transferencia isócrona, los datos isócronos son continuos y creados en tiempo real. Este tipo de información es sensitiva al retardo de entrega. Una típica aplicación de datos isócrona es la voz.

USB es considerado un estándar robusto debido a muchos atributos que implementa, estos atributos son:

- Garantiza integridad de señal.
- Protección CRC sobre los campos de control y de datos.
- Manejo de errores mediante hardware o software
- Detección automática de dispositivos.
- Auto recuperación usando temporizadores para pérdida o errores en los paquetes.
- Control de flujo para datos y control.
- Pipes de datos y control para asegurar independencia de funciones.

USB define 4 tipos de paquetes: token, delimitador de inicio, datos y handshake. La transmisión de datos se inicia a partir del bit menos significativo. El tamaño de los paquetes varía de acuerdo a la velocidad de operación en la que está trabajando el bus. Todos los paquetes inician con un campo de sincronización (SYN) que contiene 8 bits de longitud para low/full speed y 32 bits para high speed.

[59] Round-robin.- Sistema de contienda en donde todos los participantes se enfrentan entre ellos un número constante de oportunidades

### 1.2.2.3 Buses para aplicaciones domóticas

#### *1.2.2.3.1 X10*

*X10* fue la primera tecnología que surgió con el propósito de conseguir un control de los equipos domésticos mediante señales digitales. Este protocolo de comunicaciones, fue diseñado por Pico Electronics, para permitir transmitir información sobre las líneas de transmisión eléctrica (220V o 110V) a muy baja velocidad, alrededor de 60bps. La transmisión binaria en X10, se realiza mediante ráfagas de pulsos de 120Khz, éstos se sincronizan con el cruce por cero de la señal de la red eléctrica. Un uno binario se representa por un pulso de 120Khz durante 1ms, un cero binario se representa por la ausencia de pulso.

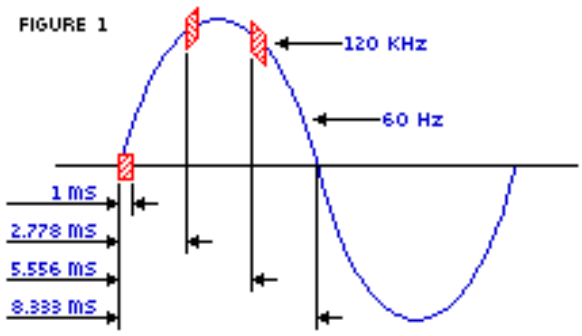



**Figura 1.42:** Método de comunicación en X10.[60]

La trama X10 contiene tres campos:

- Código de inicio
- Código de casa (letras A - P), permite seleccionar grupos de dispositivos.
- Código numérico (1 - 16) o código de función, son comandos como encender, apagar, apagar todas las unidades, etc.

Se necesitan 11 ciclos de red para obtener la transmisión completa de una trama. Para aumentar la confiabilidad del sistema, cada trama es transmitida dos veces con intervalos de 3 ciclos entre trama.

[60 - 61] Fuente: How X10 Works - Smart Home Systems

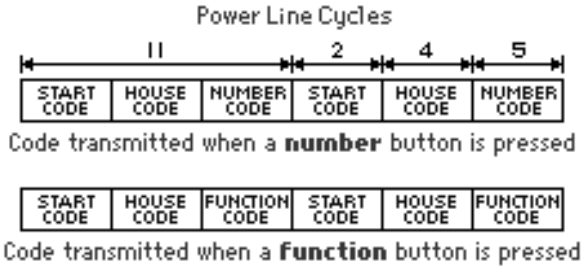



**Figura 1.43:** Formato de trama X10 (1)[61]

Dentro de cada trama, los bits correspondientes a los códigos de casa y de número o función, son transmitidos alternadamente en cada semiciclo positivo o negativo de la onda senoidal. La regla para transmitir señales binarias sobre la sinusoide consiste en emitir o no señales cada semiciclo de la onda.

Para un uno binario, existe señal en el primer semiciclo, en el segundo semiciclo no debe existir señal, para el caso del cero, no existe señal en el primer semiciclo y en el segundo semiciclo existe señal.

La secuencia de bits del código de inicio es 1110, permanece fija para todas las tramas y no sigue la regla de los semiciclos.

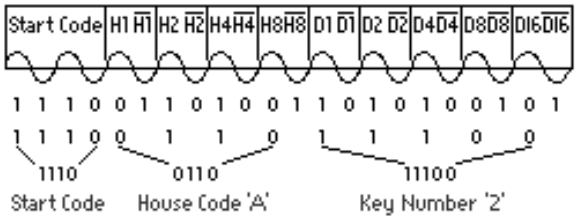



**Figura 1.44:** Formato de trama X10 (2)[62]

La siguiente tabla muestra algunos de los códigos válidos en mensajes X10:

[61 - 62] Fuente: How X10 Works - Smart Home Systems

| Códigos de Función | Comando | Descripción del comando |
|---|---|---|
| 00001 | All Units Off | Apagar todos los módulos correspondientes a este código de casa |
| 00011 | All Units On | Encender todos los módulos correspondientes a este código de casa |
| 00101 | On | Encienda un módulo |
| 00111 | Off | Apague un módulo |
| 01101 | All Lights Off | Apagar todas las luminarias correspondientes a este código de casa |

**Tabla 1.11:** Comandos básicos del protocolo X10[63]

X10 define tres tipos de dispositivos, transmisores, receptores y transmisores-receptores. Los transmisores pueden direccionar hasta 256 receptores. Cada receptor le permite al usuario asignar una dirección. Se puede tener varios receptores con la misma dirección pero con funciones diferentes.

#### *1.2.2.3.2 Bus UPB*

*Universal Powerline Bus* es un bus de comunicaciones diseñado para aplicaciones de control en ámbitos residenciales y comerciales. Similar al bus X10, el método de comunicación del bus UPB, consiste en enviar información digital codificada sobre las líneas de corriente alterna utilizando pulsos eléctricos.

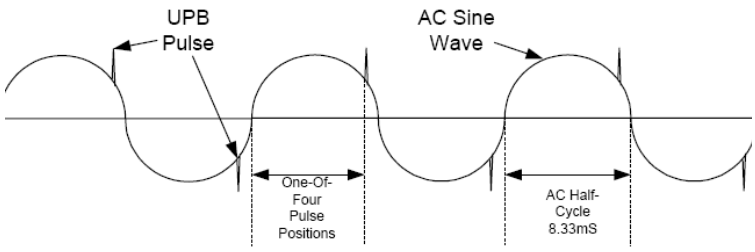



**Figura 1.45:** Método de comunicación mediante pulsos UPB.[64]

La generación de los pulsos se realiza mediante la carga de capacitores a alto voltaje y la descarga de los mismos sobre la línea de AC a un determinado

[63] Fuente: X-10 To UPB Migration Document - Powerline Control Systems
[64] Fuente: The UPB System Description - Powerline Control Systems

tiempo. Esos pulsos son detectados por dispositivos UPB quienes se encargan de decodificar los mensajes.

Se permite un pulso UPB en cada semiciclo de la onda de 60Hz AC. La generación del pulso a un determinado tiempo puede tomar cuatro posiciones correspondientes a la combinación “00”, “01”, “10” y “11”. Este método permite codificar datos de acuerdo a la posición relativa del pulso, mejor conocido como modulación por posición de pulso. Cada pulso UPB permite codificar 2 bits de información con lo cual se puede alcanzar velocidades de 240bps.

Similar a X10, las transmisiones en UPB son sincronizadas con el cruce por cero de la línea de corriente alterna. El método de comunicación de UPB varía notablemente con el de X10 y no existe ningún inconveniente en que ambos sistemas se comuniquen por la misma línea AC simultáneamente.

En el protocolo UPB, un dispositivo puede ser asignado de entre 255 posibles redes y puede tener uno de 250 identificadores de unidad, en total se tiene 63.750 posibilidades para cada dispositivo.

*Formato del paquete UPB*

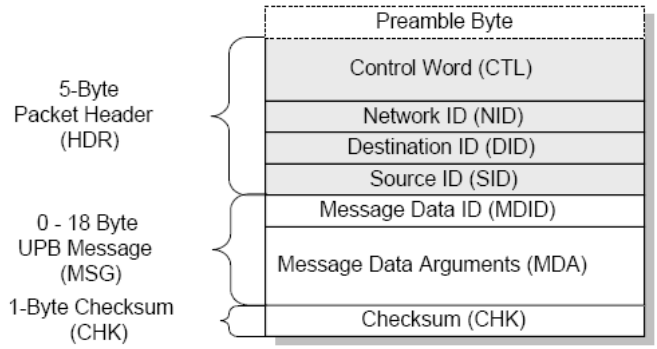



**Figura 1.46:** Formato del paquete UPB[65]

Un paquete UPB contiene los siguientes campos:

- Preámbulo de 1 byte correspondientes a la secuencia 10010110.
- Encabezado de 5 bytes divididos en:
  - Palabra de control de 2 bytes.

[65] Fuente: The UPB System Description - Powerline Control Systems

  - ID de red de 1 byte.
  - ID de destino de 1 byte
  - ID de fuente de 1 byte
- Datos de longitud variable entre 0 y 18 bytes
- Suma de comprobación de 1 byte

El preámbulo permite la sincronización de los dispositivos.

El encabezado indica el tipo de información que lleva el paquete, así como su tamaño, como será recibido, el destino y su origen.

La palabra de control indica el tamaño del paquete, solicitud de retransmisión, conteo de transmisión entre otras opciones.

El identificador de red es usado para separar un grupo de dispositivos en una red, similar a una red virtual, el equivalente en X10 es el código de casa. Se define un identificador de red especial con valor 0, denominado ID global de red, este paquete es aceptado por todas las redes.

El ID de destino sirve para dirigirse a uno o a varios dispositivos. Cada dispositivo acepta un paquete UPB sólo si va dirigido a ellos. También se define un valor 0 para el ID de destino, llamado broadcast ID, este paquete es aceptado por todos los dispositivos.

El ID de fuente determina el dispositivo que originó un paquete.

Los datos o mensajes UPB (MSG) son de longitud variable de 0 a 18 bytes, y contiene la información útil del paquete.

La suma de comprobación CHK se utiliza como verificador de integridad del paquete, es aplicado a todos los campos excepto al preámbulo.

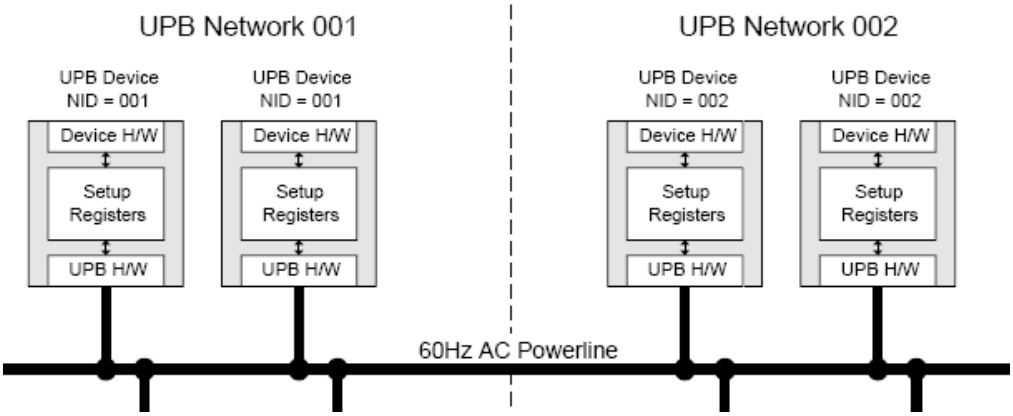



**Figura 1.47:** Modelo de sistema UPB[66]

La siguiente tabla muestra algunos comandos válidos en mensajes UPB:

| MDID | Comando | Descripción del comando |
|---|---|---|
| 00100010 | Goto | Solicita al dispositivo(s) ir a un nivel específico a una velocidad específica |
| 00100100 | Fade stop | Solicita al dispositivo(s) parar y permanecer en el nivel actual |
| 00110000 | Report state | Solicita al dispositivo un reporte de su estado actual. |
| 00110001 | Store state | Almacenar el estado actual del dispositivo. |
| 10000110 | Device state report | Reporte del estado actual de un dispositivo. |

**Tabla 1.12:** Comandos básicos de control del protocolo UPB[67]

## 1.3 MICROCONTROLADORES CON SOPORTE CAN

### 1.3.1 SOLUCIONES ATMEL PARA REDES CAN[28]

ATMEL ha desarrollado un sistema completo para brindar soluciones a redes CAN con microcontroladores flash y transceivers de alta velocidad. ATMEL ha denominado a esta familia como CANary.

Todos los microcontroladores CAN tienen un conjunto de características comunes:

[66] Fuente: The UPB System Description - Powerline Control Systems

[67] Fuente: X-10 To UPB Migration Document - Powerline Control Systems

- Memoria flash
- Memoria EEPROM
- Memoria RAM
- Capacidad de programación en aplicación asistida por un bootloader
- El mismo motor CAN para simplificar migración de software
- Capacidad de bajo voltaje (2.7 V máximo)

Entre los años 2000 y 2003 se introdujeron microcontroladores CAN de 5-MIPS basado en un núcleo C51. En el año 2004 se lanzó al mercado el primer microcontrolador AVR CAN con un rendimiento de 16-MIPS y un conjunto de memorias y periféricos necesarios para aplicaciones de alto rendimiento.

**Microcontroladores 5 MIPS**

En esta categoría, ATMEL ofrece 3 microcontroladores: T89C51CC01, T89C51CC02 y AT89C51CC03. Con 5 MIPS, 5 voltios y memoria flash de 16KBytes a 64Kbytes, estos microcontroladores cubren una gran cantidad de aplicaciones de tipo esclavo, tales como sensores de proximidad, etc.

Para aplicaciones que requieren voltajes bajos (2.7 V min), los tres microcontroladores se mantienen en 3.3 MIPS.

#### 1.3.1.1 Microcontrolador T89C51CC01

Este fue el primer microcontrolador CAN creado por ATMEL. Posee 32KBytes de memoria Flash y 1.2Kbytes de memoria RAM, 8 canales para convertidores A/D de 10 bits, 32 líneas de I/O y 3 temporizadores. Maneja hasta 15 mensajes CAN, adecuado para aplicaciones sencillas. Una característica importante es su capacidad de reprogramación asistida utilizando bootloader, esto es muy útil ya que permite programar el microcontrolador utilizando el puerto serial. Su capacidad de 2Kbytes en memoria EEPROM es adecuada para almacenar configuraciones requeridas por protocolos de capa aplicación.

#### 1.3.1.2 Microcontrolador T89C51CC02

Con 16KBytes de memoria flash, 0.5KBytes de memoria RAM, puede manejar hasta 4 mensajes CAN. Este microcontrolador es adecuado para aplicaciones que requieran un mínimo de dispositivos esclavos DeviceNet o CANOpen.

#### 1.3.1.3 Microcontrolador AT89C51CC03

Este microcontrolador tiene muchas similitudes con el T89C51CC01, maneja el mismo número de mensajes CAN, es decir 15, tiene la misma capacidad en memoria EEPROM 2Kbytes, la única diferencia entre ambos microcontroladores es que en esta versión, se añade más capacidad en memoria RAM, 2.2Kbytes, en memoria Flash, 64Kbytes y un puerto SPI.

### Microcontroladores 16 MIPS

ATMEL ha introducido controladores CAN con arquitectura RISC AVR. El primer controlador CAN creado de este tipo tiene gran capacidad en memoria y procesamiento de 16 MIPS, esta familia de controladores son conocidos como CANary.

#### 1.3.1.4 Microcontrolador AT90CAN128

Este fue el primer microcontrolador CAN con arquitectura AVR. Resulta adecuado para aplicaciones CAN más exigentes.

Con 128Kbytes de memoria Flash, 4Kbytes de memoria RAM y 4Kbytes de memoria EEPROM, ofrece 16 MIPS y forma parte de la familia CANary de Atmel.

Las principales características que tienen los microcontroladores de esta familia son:

*Motor CAN:* Se ha incluido mejoras en el software del motor CAN para eliminar riesgos de sobrecarga en las tramas, además se ha incluido un buffer que puede almacenar múltiples mensajes.

*Programación en aplicación:* Atmel ha desarrollado para el T89C51CC01 un pequeño programa “bootloader” que puede ser usado para reprogramar el microcontrolador utilizando el puerto serial. Empresas como DeviceNet o

CANopen, solicitaron que esta característica se adapte a sus dispositivos, esta sugerencia fue acogida en el AT90CAN128, en donde se incluyó una memoria flash de hasta 8Kbytes para manejar programas de arranque propietarios.

*Tamaño de código:* Algunas aplicaciones necesitan mayor espacio para almacenar protocolos CAN de capa aplicación, por esta razón se aumentó la capacidad de la memoria flash a 128Kbytes.

*Tamaño de datos:* Aplicaciones más exigentes requieren de mayor espacio en la memoria RAM, para solucionar esto se incluye una memoria de 4Kbytes en RAM.

*Seguridad CAN:* Se ofrece un mecanismo para que el controlador principal monitoree al receptor, utilizando un enlace serial de alta velocidad, de modo que se determine si la trama llegó sin errores. Se puede usar el puerto serial SPI para interconectar dos microcontroladores AT90CAN128. El AT90CAN128 incluye además otras características muy comunes como convertidores A/D, temporizadores y líneas de I/O de propósito general.

### 1.3.1.5 ATA660 Transceiver de Alta Velocidad

Típicamente, cada nodo en un sistema CAN debe tener un dispositivo que convierta las señales digitales generadas por el controlador CAN a señales adecuadas para ser transmitidas sobre el cableado del bus (salidas diferenciales), para esto, Atmel ofrece el ATA6660 como transceiver CAN que puede ser usado con cualquier controlador Atmel que no posea interfaz CAN.

| | **8051 Architecture** | | | **AVR 8-Bit RISC** |
|---|---|---|---|---|
| | **T89C51CC01** | **T89C51CC02** | **T89C51CC03** | **AT90CAN128** |
| ***Introduced in*** | Y2000 | Y2001 | July 2003 | February 2004 |
| ***MIPS*** | 5 | 5 | 5 | 16 |
| ***Flash Program/Boot*** | 32KB/2KB | 16KB/2KB | 64KB/2KB | 128KB/up to 8KB |
| ***EEPROM*** | 2Kx8 | 2Kx8 | 2Kx8 | 4K |
| ***RAM*** | 1.2Kx8 | 0.5Kx8 | 2.2Kx8 | 4K |
| ***Power Fall Detect*** | No | No | Yes | Yes |
| ***CAN Controller*** | 15 message objects | 4 message objects | 15 message objects | 15 message objects |
| ***SPI*** | No | No | Yes | Yes |

| | | | | |
|---|---|---|---|---|
| ***Supply*** | 3 to 5.5 | 3 to 5.5 | 3 to 5.5 | 2.7 to 5.5 |
| ***ADC*** | 10-bit, 8 channels | 10 bit, 8 channels | bit, 8 channels | 10 bit, 8 channels |
| ***PCA*** | 5 channels | 2 channels | 5 channels | No |
| ***Timers*** | Timers 0/1/2 | Timers 0/1/2 | Timers 0/1/2 | Timers 0/1/2/3 |
| ***UART*** | 1 | 1 | 1 | 2 |
| ***Port*** | Port 0/1/2/3 | Port 1/2/3 | Port 0/1/2/3 | Port A/B/C/D/E/F/G |
| ***Bootloader*** | UART/CAN | UART/CAN | UART/CAN | UART/CAN |

**Tabla 1.13:** Microcontroladores Atmel CAN[68]

### 1.3.2 SOLUCIONES MICROCHIP PARA REDES CAN

Microchip ha incursionado de manera más acelerada en el mundo de las redes CAN, en la actualidad ofrece 16 microcontroladores con interfaz CAN, todos estos dispositivos se agrupan en la familia PIC18FXXX, todos poseen similares características, la variación entre unos y otros radica en el tamaño de la memoria flash, EEPROM, líneas de entradas y salidas, convertidores A/D, temporizadores, etc., pero en general poseen las mismas interfaces; EUSART, SPI, MI$^2$C y CAN versión 2.0B.

#### 1.3.2.1 Familia PIC18FXX8X

Debido a la gran semejanza en el diseño de estos microcontroladores, a continuación se menciona ciertas características poniendo énfasis en los detalles referentes al manejo del protocolo CAN.

*Modos de administración de potencia:* Se tienen disponibles 3 modos:

- Run, CPU y periféricos encendidos
- Idle, CPU apagada y periféricos encendidos
- Sleep, CPU y periféricos apagados

*Oscilador flexible:* Soporta 4 modos de cristal, dos modos externos RC, sobre los 4MHz y dos modos externos de reloj, sobre los 40MHz. Además posee un bloque de oscilador interno de 8 frecuencias seleccionables desde 31KHz hasta 8MHz.

[68] Fuente: Atmel Microcontrollers for Controller Area Network (CAN) – ATMEL Corporation

*Programación en aplicación:* Programación vía interfaz serial en el circuito, utilizando dos pines, similar al bootloader de Atmel.

*Memoria de datos:* La memoria de datos varía dependiendo el modelo del microcontrolador, se tienen memorias desde 768 hasta 3328 bytes.

*Memoria de programa:* Al igual que la memoria de datos, el tamaño de la memoria flash varía de acuerdo al modelo de microcontrolador, se tienen memorias desde 16Kbytes hasta 48Kbytes. El tamaño de la memoria EEPROM también varía, se tienen memorias desde 256 bytes hasta 1024 bytes.

*Periféricos:* Se dispone de tres interrupciones externas, algunos modelos poseen módulos comparadores de señales, módulos SPI, $I^2C$, USART mejorado (EUSART), soporte para interfaz RS-485, RS-232 y LIN 1.3, convertidores A/D de 10 y 11 bits de acuerdo al modelo del microcontrolador.

*Módulo Enhanced CAN (ECAN):* Este módulo contiene un controlador de comunicaciones CAN que soporta las versiones 2.0A y 2.0B definidas en la especificación de BOSCH. Las principales características de este módulo son:

- Implementación del protocolo CAN 1.2, CAN2.0A y CAN 2.0B
- Velocidad programable hasta 1Mbps.
- Tres buffers dedicado a transmisión con priorización
- Dos buffers dedicados a recepción con priorización
- Seis buffers programables como recepción/transmisión
- Tres máscaras de aceptación de 29 bits
- Filtros de aceptación de 16 y 29 bits (para identificador estándar y extendido)
- Filtro de datos compatible con dispositivos DeviceNet
- Soporte para manejo automático de tramas remotas
- Características para manejo avanzado de errores
- Funcionalidad wake-up programable y modo de consumo de baja potencia sleep.
- Modo loopback programable para operación de auto comprobación

- Señalización vía interrupciones para todos los estados de error en los transmisores y receptores CAN

*Motor CAN:* El módulo de bus CAN consiste en un motor del protocolo y una serie de buffers de mensajes y control. El motor del protocolo CAN se encarga de manera automática del manejo de todas las funciones de recepción y transmisión de mensajes sobre el bus CAN. Los mensajes son transmitidos cargando el dato apropiado en los registros. Los estados de error son verificados en base a la lectura de los registros. Cualquier mensaje detectado en el bus CAN es verificado para determinar si existen errores, luego se procesa el mensajes mediante filtrado y se almacena en uno de los dos buffers de recepción.

| Device | Program Memory | | Data Memory | | I/O | 10-bit A/D (ch) | CCP/ ECCP (PWM) | MSSP | | EUSART | Comp | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | SPI | Master $I^2C$ | | | |
| PIC16F2480 | 16K | 8192 | 768 | 256 | 25 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC16F2580 | 32K | 16384 | 1536 | 256 | 36 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC16F4480 | 16K | 8192 | 768 | 256 | 25 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |
| PIC16F4580 | 32K | 16384 | 1536 | 256 | 36 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |

**Tabla 1.14:** Microcontroladores Microchip PIC18F2480/2580/4480/4580[69]

| Device | Program Memory | | Data Memory | | I/O | 10-bit A/D (ch) | CCP/ ECCP (PWM) | MSSP | | EUSART | Comp | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | SPI | Master $I^2C$ | | | |
| PIC16F2585 | 48K | 24576 | 3328 | 1024 | 28 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC16F2680 | 64K | 32768 | 3328 | 1024 | 28 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC16F4585 | 48K | 24576 | 3328 | 1024 | 44 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |
| PIC16F4680 | 64K | 32768 | 3328 | 1024 | 40/ 44 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |

**Tabla 1.15:** Microcontroladores Microchip PIC18F2585/2680/4585/4680[70]

[69] Fuente: PIC18F2480/2580/4480/4580 Data Sheet – Microchip
[70] Fuente: PIC18F2585/2680/4585/4680 Data Sheet – Microchip

| Device | Program Memory | | Data Memory | | I/O | 10-bit A/D (ch) | CCP/ ECCP (PWM) | MSSP | | EUSART | Comp | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | SPI | Master $I^2C$ | | | |
| PIC16F2682 | 80K | 40960 | 3328 | 1024 | 28 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC16F2685 | 96K | 49152 | 3328 | 1024 | 28 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC16F4682 | 80K | 40960 | 3328 | 1024 | 40/44 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |
| PIC16F4685 | 96K | 49152 | 3328 | 1024 | 40/44 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |

**Tabla 1.16:** Microcontroladores Microchip PIC18F2682/2685/4682/4685[71]

| Device | Program Memory | | Data Memory | | I/O | 10-bit A/D (ch) | CCP/ ECCP (PWM) | MSSP | | ECAN/ AUSART | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | SPI | Master $I^2C$ | | |
| PIC16F6585 | 48K | 24576 | 3328 | 1024 | 53 | 12 | 1/1 | Y | Y | Y/Y | 2/3 |
| PIC16F6680 | 64K | 32768 | 3328 | 1024 | 53 | 12 | 1/1 | Y | Y | Y/Y | 2/3 |
| PIC16F8585 | 48K | 24576 | 3328 | 1024 | 69 | 16 | 1/1 | Y | Y | Y/Y | 2/3 |
| PIC16F8680 | 64K | 32768 | 3328 | 1024 | 69 | 16 | 1/1 | Y | Y | Y/Y | 2/3 |

**Tabla 1.17:** Microcontroladores Microchip PIC18F6585/6680/8585/8680[72]

[71] Fuente: PIC18F2682/2685/4682/4685 Data Sheet – Microchip
[72] Fuente: PIC18F6585/6680/8585/8680 Data Sheet – Microchip

## Otras soluciones CAN de Microchip

### 1.3.2.2 MCP2502X/5X CAN expansión de I/O

Estos dispositivos operan como expansión de I/O seriales para sistemas CAN. Los dispositivos MCP2502X/5X permiten implementar un simple nodo CAN sin la necesidad de utilizar microcontrolador.

Los MCP2502X/5X implementan la versión 2.0B del protocolo CAN, tienen velocidad programable hasta 1Mbps, poseen también un cierto número de periféricos, incluyendo 8 líneas de I/O digitales, 4 canales de 10 bits de A/D y salidas PWM, además, se proveen tres buffers de transmisión, dos buffers de recepción, dos filtros y una máscara.
El dispositivo puede ser configurado vía mensajes a través del bus CAN y también posee la capacidad de programación vía interfaz serial (ICSP).

### 1.3.2.3 MCP2515 Controlador CAN independiente con interfaz SPI

El MCP2515 es un controlador CAN independiente desarrollado para simplificar aplicaciones que requieren de una interfaz con un bus CAN.
El MCP2515 consiste de tres bloques:

1. *Un módulo CAN* que incluye el motor del protocolo CAN, máscaras, filtros y buffers de transmisión y recepción para el manejo de los mensajes sobre el bus CAN.
2. *La lógica de control y registros* que son usados para la configuración del dispositivo y para su operación.
3. *Bloque del protocolo SPI* para permitir interconexión con el controlador del dispositivo.

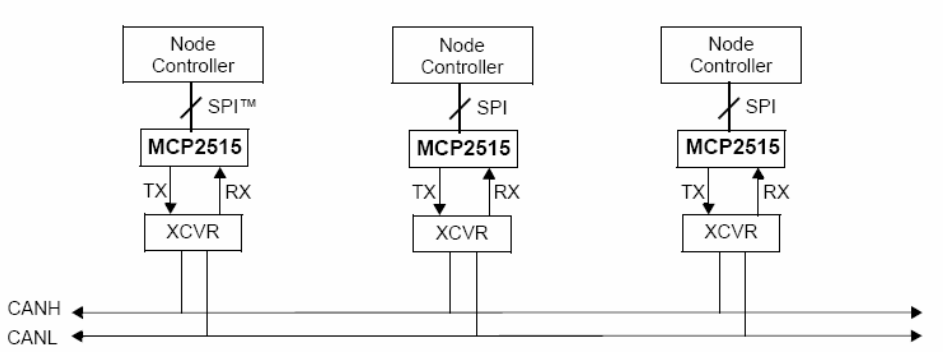



**Figura 1.48:** Ejemplo de implementación de un sistema CAN usando MCP2515[73]

#### 1.3.2.4 MCP2551 Transceiver CAN de alta velocidad

El MCP2551 es un transceiver CAN de alta velocidad tolerante a fallas en los dispositivos que serán conectados en el bus físico. Totalmente compatible con la capa física de ISO 11898, el MCP2551 es altamente inmune al ruido debido a la implementación de bus diferencial. Opera a velocidades de 1Mbps y es configurable para sistemas de 12V y 24V.

Este transceiver, provee protección contra daños producidos en condiciones de corto circuito (voltajes positivos o negativos de baterías) y protección contra altos voltajes. La capacidad máxima de nodos CAN que pueden ser conectados a este dispositivo es de 112.

### 1.3.3 SOLUCIONES MOTOROLA PARA REDES CAN

Motorola ofrece muchas soluciones para redes CAN disponibles en sus microcontroladores de 8, 16 y 32 bits.

Motorola usa tres diferentes modelos de controlador CAN:

- MsCAN – módulo para microcontroladores de 8 y 16 bits
- TouCAN – módulo para microcontroladores de 32 bits
- FlexCAN – módulo para microcontroladores de 32 bits (familia MCF5xxx)

A continuación se analizan las características generales de los controladores CAN presentes en microcontroladores de 8 y 16 bits:

[73] Fuente: MCP2515 Data sheet - Microchip

### 1.3.3.1 Familia MC68HC05XX

Los microcontroladores MC68HC05X4/16/32 fueron los primeros microcontroladores con soporte CAN ofrecidos por Motorola. Son miembros de la familia de microcontroladores de 8 bits MC68HC05. Entre las características generales de estos microcontroladores se encuentran:

- Controlador MCAN (primera versión del MsCAN)
- Dos modos de operación: wait y stop.
- Memoria de programa tipo EEPROM
- Convertidores A/D
- I/O de propósito general distribuidas en 4 puertos.
- Módulo de comunicación serial (SCI)
- Dos temporizadores

*Módulo MCAN:* El módulo MCAN incluye todo el hardware necesario para implementar el núcleo del protocolo CAN definido por la BOSCH. El MCAN provee los mecanismos dedicados al manejo de mensajes, transmisión, recepción y filtrado. La limitación de este módulo es su incapacidad de manejar tramas de sobrecarga.

### 1.3.3.2 Familia MC68HC908GZX

Los microcontroladores MC68HC908GZ8/16/32/48/60 son miembros de la familia de microcontroladores de 8 bits. Todos estos dispositivos poseen una versión mejorada de la una unidad central de procesamiento MC68HC08 y se encuentra disponible con una variedad de módulos, tamaños y tipos de memoria.

Las principales características de los microcontroladores de la familia MC68HC908GZx incluyen:

- Alto rendimiento utilizando arquitectura M68HC08 optimizada para compiladores C.
- Totalmente compatible con microcontroladores de la familia M68HC05.
- Controlador escalable MsCAN08 (scalable controller area network) implementado en base al estándar 2.0B definido por la especificación de BOSCH.

- Memoria de programa tipo flash.
- Programación en aplicación vía interfaz serial.
- Dos modos de operación: wait y stop.
- Módulo serial peripheral interface (SPI)
- Módulo enhanced serial communications interface (ESCI)
- Módulo de 8 bits para de interrupción de teclado.
- Módulo de seguridad
- Módulo de voltaje dual.
- Dos temporizadores
- Canales para convertidor A/D de 10 bits.
- Puerto de 8 bits para teclado.
- I/O de propósito general distribuidas en 5 y 7 puertos.

*Módulo MsCAN08:* El módulo MSCAN08 implementa las versiones 2.0A/B del protocolo CAN como se encuentra definido en la especificación de BOSCH, su velocidad es programable hasta 1Mbps, provee tres buffers para transmisión con priorización, máscara flexible y filtros que soportan identificadores estándar o extendidos, funcionalidad wake-up programable y modo sleep de baja potencia, modo de loop-back para operación de auto comprobación, señalización separada y capacidad de interrupción para todos los receptores CAN y transmisores en estado de error.

#### 1.3.3.3 MC33989 Transceiver CAN de alta velocidad

El MC33989 es un circuito integrado monolítico que combina funciones utilizadas por microcontroladores encontrados comúnmente en automóviles como unidades de control de motor (ECU). Este dispositivo incorpora funciones tales como regular voltaje o monitorear fallas en la circuitería, además tiene implementado una interfaz para bus CAN.

# REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 1

## Direcciones Electrónicas

[1] Carlos Antonio Chamú Morales; "Desarrollo de un sistema educativo para la enseñanza del protocolo de comunicaciones CAN", Universidad Tecnológica de la Mixteca, Huajapan de León - Abril 2005, http://jupiter.utm.mx/~tesis_dig/9567.pdf

[2] Robert Bosch GmbH, Stuttgart, Germany, April 5 1995; "CAN specification version 2.0", http://www.algonet.se/~staffann/developer/can2spec.pdf

[3] Am Weichselgarten 26, D-91058 Erlangen; "Controller Area Network Can-Cia.org", http://can.marathon.ru/projects/canschool-02/RU-CAN02.pdf

[4] Steve Corrigan; Texas Instruments; "Application Report SLOA101 - August 2002", http://focus.ti.com/lit/an/sloa101/sloa101.pdf

[5] Siemens Microelectronics, Inc. October 98; "CANPRES Version 2.0", http://staticweb.rasip.fer.hr/rip/seminari/can_atlas/can_doc/canpres.pdf

[6] © Kvaser AB; Advanced CAN Solutions, 2007-07-05, http://www.kvaser.com/can/protocol/index.htm,

[7] Infineon Technologies, Inc., August 1999, http://www.microcontroller.com/LearnEmbedded/CAN1_sie/CAN1_files/frame.htm

[8] Guillermo Pallarés Castillo; "Equipo portátil para comunicación con centralitas por bus CAN", Madrid, Universidad Pontificia Comillas, Escuela Técnica Superior de Ingeniería (ICAI), Junio 2005, http://www.iit.upcomillas.es/pfc/resumenes/42a58bea2b984.pdf

[9] J. Artieda, Cybertech 2005; “Buses de comunicación”, http://www.disam.upm.es/cybertech/2005/buscom.pdf
[10] Museo de la Informática y Computación Aplicada, México, 2004; “Buses”, http://mx.geocities.com/pcmuseo/mecatronica/buses.htm

[11] Zator Systems, 2008; “Puerto Serie: La UART”, http://www.zator.com/Hardware/H2_5_1_1.htm

[12] Leroy Davis, December 2007; “SCI Bus Description”, http://www.interfacebus.com/Design_Connector_SCI_Bus.html

[13] Motorola Inc., October 2001; “Serial Communications Interface (SCI) Block Guide02.05”,http://www.ece.unh.edu/biolab/hof/public/S12SCIV2_sci_serial.pdf

[14] Philips Semiconductor, JANUARY 2000; “The I2C Bus Specification Version 2.1, http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf

[15] Eduardo J. Carletti, Enero 2008; “Comunicación – Bus I2C Descripción y funcionamiento”, http://robots-argentina.com.ar/Comunicacion_busI2C.htm

[16] “I2C”, Noviembre 2007, http://es.wikipedia.org/wiki/I%C2%B2C

[17] “SPI”, Enero 2008, http://es.wikipedia.org/wiki/SPI

[18] Daniel Eggert, c961047, DTUsat, System Engineering, January 19, 2002; “Interboard Communication”, http://microsat.sm.bmstu.ru/e-library/ccdh/Hard/Interboard_Communication+ok.pdf

[19] Volker Soffel, MicroController Pros Corporation, 2008; “Synchronous Microcontroller Communication Interfaces: SPI, Microwire and I2C Protocol Formats”, http://www.ucpros.com/work%20samples/Microcontroller%20Communication%20Interfaces%202.htm

[20] National Semiconductor, Application Note 452, Abdul Aleaf, January 1992; “Microwire Serial Interface”, http://www.national.com/an/AN/AN-452.pdf

[21] MAXIM, Application Note 3989, January 2007; “Add Control, Memory, Security, and Mixed-Signal Functions with a Single Contact”, http://pdfserv.maxim-ic.com/en/an/AN3989.pdf

[22] “USB”, Diciembre 2007; http://es.wikipedia.org/wiki/USB

[23] Creators of USB technology, April 2000 ; “USB 2.0 Specification, USB Implementers Forum, Inc.”, http://www.usb.org/developers/docs/

[24] O.E.M. Original Equipment Manufacturer; “X-10 Power Line Carrier Technology”, http://www.smarthome.gr/pdf/PLCtechnology.pdf

[25] Smart Home Systems, Inc. 2008; “X10 Theory”, http://www.smarthomeusa.com/info/x10theory/

[26] Powerline Control Systems, 19201 Parthenia Street, Suite J, Northridge, CA, September 2003; “The UPB System Description Version 1.1”, http://www.smarthomeusa.com/Common/UPB/UPBdescription.pdf

[27] “UPB”, November 2007; http://en.wikipedia.org/wiki/Universal_powerline_bus

**Papers**

[28] Michel Passemard, Atmel Corporation, February 2004; “Atmel Microcontrollers for Controller Area Network (CAN)”, http://atmel.com/dyn/resources/prod_documents/doc4069.pdf

[29] Karl Henrik Johansson, Martin T¨orngren, Lars Nielsen; “Vehicle Applications of Controller Area Network”,

http://www.md.kth.se/RTC/Papers/VehicleApplicationsCan2005.pdf
[30] F. Cortés, S. Gallardo, F. Barrero, S. Toral ; “ Plataforma para el desarrollo de sensores inteligentes y sistemas microprocesados”, Departamento de Ingeniería Electrónica, Escuela Superior de Ingenieros, Universidad de Sevilla, España, http://www.euitt.upm.es/taee06/papers/S7/p81.pdf

[31] Dr.-Ing. Héctor Kaschel C. Ing. Ernesto Pinto L.; “Análisis protocolar del bus de campo CAN”, Facultad de Ingeniería, Depto. de Ingeniería Eléctrica, Universidad de Santiago de Chile,
http://cabierta.uchile.cl/revista/19/articulos/pdf/edu2.pdf,

**Datasheets**

[32] Microchip Technology Inc.

- PIC18F2480/2580/4480/4580 Data Sheet, 2004
- PIC18F2585/2680/4585/4680 Data Sheet, 2007
- PIC18F2682/2685/4682/4685 Data Sheet, 2007
- PIC18F6585/8585/6680/8680 Data Sheet, 2004
- MCP2515 Stand-Alone CAN Controller with SPI Interface, 2005.
- MCP2502X/5X CAN I/O Expander Family Data Sheet, 2007.
- MCP2551 High-Speed CAN Transceiver, 2007.

[33] Motorola Inc.

- MSCAN Block Guide V02.15, 15 JUL 2004

[34] Freescale Semiconductor

- MC68HC05X16 MC68HC05X32 MC68HC705X32
- MC68HC908GZ60 MC68HC908GZ48 MC68HC908GZ32 Data Sheet
- System Basis Chip withHigh-Speed CAN Transceiver MC33989.

# CAPÍTULO 2: DISEÑO DEL PROTOTIPO

## 2.1 SELECCIÓN DEL MICROCONTROLADOR PIC18F2480

Existe un gran número de soluciones CAN disponibles en microcontroladores de diferentes fabricantes, en versiones de 8, 16 y 32 bits, la selección de un determinado producto se realiza de acuerdo al tipo de aplicación y a sus características técnicas.

### 2.1.1 CONCEPTOS BÁSICOS

#### 2.1.1.1 Procesador

El procesador es el corazón del microcontrolador, es la parte encargada del procesamiento de las instrucciones y determina las características del microcontrolador, tanto en hardware como en software. Entre las tareas del procesador, se encuentran el direccionamiento de memoria, ejecución de operaciones y tareas de control.

El procesador o CPU (Unidad Central de Procesamiento), puede ser de tres tipos:

- ***CISC*** *(Computadores de conjunto de instrucciones complejo)*
- ***RISC*** *(Computadores de conjunto de instrucciones reducido)*
- ***SISC*** *(Computadores de conjunto de instrucciones específico)*

#### 2.1.1.2 Arquitectura del procesador

Los procesadores pueden basarse en dos tipos de arquitecturas: Von Neumann y Hardvard.

La ***arquitectura Von Neumann*** se compone de una única memoria principal en la que se comparte el almacenamiento de datos e instrucciones. El acceso a esta memoria se realiza a través de un sistema de buses único (direcciones, datos y control).

La ***arquitectura Harvard*** se compone de dos memorias independientes, una para almacenamiento de instrucciones y otra para almacenamiento de datos. El acceso a ambas memorias es posible a través de sus respectivos sistemas de buses.

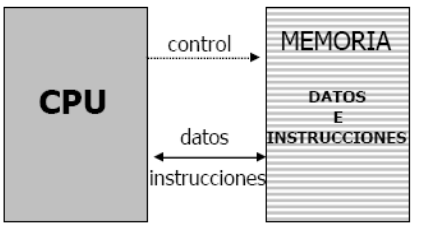



**Figura 2.1:** Arquitectura Von Neumann[74]

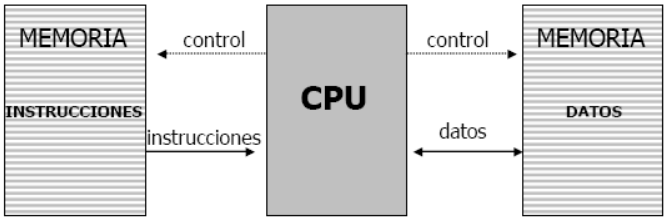



**Figura 2.2:** Arquitectura Harvard[75]

### 2.1.1.3 Tipo de memoria

#### *2.1.1.3.1 Memoria de datos*

Debido a que los datos varían constantemente, esta memoria debe ser apta para soportar operaciones de lectura y escritura. La memoria volátil RAM (Random Access Memory) es adecuada para este tipo de funciones.

En la actualidad, muchos microcontroladores incorporan también como memoria de datos, una memoria no volátil tipo EEPROM para almacenar información en caso de cortes de suministro eléctrico.

#### *2.1.1.3.2 Memoria de programa*

En esta memoria, se almacenan todas las instrucciones del programa. Una vez que el programa sea diseñado, no sufrirá modificaciones, razón por la cual se lo almacena de manera permanente.

Existen algunos tipos de memoria que son adecuados para esta función, entre los se encuentran:

- ***ROM*** *(Read Only Memory),* memoria de tipo no volátil.
- ***EPROM*** *(Erasable Programmable Read OnIy Memory),* memoria que se graba eléctricamente, para su borrado se utiliza luz ultravioleta.
- ***OTP*** *(One Time Programmable),* memoria que puede ser programa una sola vez. Por su bajo costo son ideales para aplicaciones industriales.
- ***EEPROM*** *(Electrical Erasable Programmable Read OnIy Memory),* tanto el grabado como el borrado se realizan eléctricamente.

[74 - 75] Fuente: Introducción a los microcontroladores - M. Asunción Vicente

- ***FLASH***, memoria no volátil de bajo consumo que permite operar a grandes velocidades. Ideales para aplicaciones que serán modificadas continuamente.

#### 2.1.1.4 Líneas de I/O

Las líneas de I/O permiten al microcontrolador comunicarse con periféricos externos que se encuentran bajo su control. Las líneas de I/O manejan la información que reciben y emiten, suelen agruparse en conjuntos de ocho y se denominan puertos.

En la actualidad existen muchos microcontroladores con líneas de I/O que soportan comunicaciones seriales utilizando diversos protocolos como el USB, 1-Wire, I2C, CAN, etc.

#### 2.1.1.5 Reloj principal

Los microcontroladores dependen de un circuito oscilador para generar ondas cuadradas de alta frecuencia, a manera de reloj, este circuito sincroniza todas las operaciones del sistema. El circuito de reloj viene incorporado en el microcontrolador casi en tu totalidad, solo se requiere de pocos componentes exteriores para lograr estabilizar la frecuencia de trabajo, estos componentes son cristales de cuarzo y elementos pasivos como capacitores.

#### 2.1.1.6 Recursos especiales

Las capacidades adicionales que puede ofrecer un microcontrolador son diversas y complementan la potencia y flexibilidad del dispositivo.

Entre los recursos especiales más comunes se tiene:

- Temporizadores
- Perro guardián (watchdog)
- Conversores A/D y D/A
- Comparadores analógicos
- Modulador por ancho de pulso (PWM)
- Puertos de comunicación: UART, USART, I2C, 1-Wire, SPI, USB, CAN, etc.

#### 2.1.1.7 Lenguaje de programación

Los lenguajes de programación en microcontroladores pueden ser: lenguajes de bajo nivel y lenguajes de alto nivel.

La utilización de lenguajes de bajo nivel (lenguaje ensamblador) implica una ejecución muy rápida aún cuando para el programador demanda de una elaboración de programas extensos y difíciles de manipular.

Los lenguajes de alto nivel más empleados son lenguaje C, Basic y Pascal en sus versiones de compiladores e intérpretes para microcontroladores. Los programas realizados con este tipo de lenguajes resultan menos complejos que los programados en bajo nivel, aunque se requiere la utilización de una mayor cantidad de memoria.

##### *2.1.1.7.1 Herramientas de desarrollo*

En el entorno de desarrollo de los microcontroladores, se dispone de tres diferentes herramientas:

- ***Ensamblador,*** es un software que se encarga de traducir el programa escrito en lenguaje de bajo nivel a lenguaje de máquina.
- ***Compilador,*** es un software que se encarga de traducir el programa escrito en lenguaje de alto nivel a lenguaje ensamblador.
- ***Simulador***, es un software capaz de ejecutar en un computador programas realizados para el microcontrolador. Los simuladores pueden ser muy útiles para la depuración de programas previo a la grabación de los mismos.

En la actualidad, existen muchos programas que combinan las tres tareas en un solo software, un ejemplo de esto es PIC Simulator IDE, este programa permite la programación en lenguaje Basic, se puede compilar y ensamblar el programa, además posee algunas interfaces gráficas para la simulación del mismo. Existen otros programas que brindan funcionalidades únicas como el simulador Proteus. Programas como MikroC, MikroBasic entre otros, no solo permiten compilar si no también ensamblar el programa, pero no brindan las funciones de simulador.

## 2.1.2 CONSIDERACIONES TÉCNICAS

Una parte importante en el diseño, es encontrar el microcontrolador adecuado con las características mínimas pero que a la vez satisfaga las necesidades requeridas por la aplicación, de esta manera se puede minimizar el costo del diseño en hardware.

### 2.1.2.1 Requerimientos de la aplicación

La mayor exigencia de la aplicación es el soporte del protocolo CAN para el intercambio de información entre los nodos.

Se controlarán 2 sensores por cada nodo CAN esclavo, por lo tanto no se requiere de un número elevado de líneas de I/O en cada microcontrolador.

La interacción del nodo CAN maestro con el computador personal requiere que el microcontrolador tenga soporte para comunicaciones seriales.

#### *2.1.2.1.1 Especificaciones técnicas del microcontrolador*

Para el desarrollo del proyecto, se ha seleccionado el microcontrolador PIC18F2480 de Microchip, por sus características técnicas, su bajo costo y la enorme cantidad de documentación y soporte para los usuarios. Aunque no ofrece mayores prestaciones, este microcontrolador es adecuado para las necesidades de la aplicación planteada.

El microcontrolador PIC18F2480 ofrece un buen rendimiento debido a su procesador de tipo RISC y su arquitectura Harvard. Utiliza una memoria de programa tipo Flash, este tipo de memoria es más veloz y tolera hasta 100.000[76] procesos de escritura y borrado.

El PIC18F2480 es un microcontrolador de 8 bits cuyas características incluyen:

- Memoria de programa tipo Flash de 16 Kbytes.
- Memoria de datos tipo EEPROM de 256 bytes.
- Memoria de datos tipo RAM de 768 bytes.
- Tres interrupciones externas

[76] Fuente: PIC18F2480 Data Sheet - Microchip

- 25 entradas/salidas
- 8 canales A/D de 10 bits.
- Módulo EUSART con soporte RS-485, RS-232 y LIN1.3.
- Módulo ECAN conforme a la especificación CAN2.0B.

Este microcontrolador se encuentra disponible en encapsulado tipo DIP, a continuación se muestra un diagrama de pines del mismo:

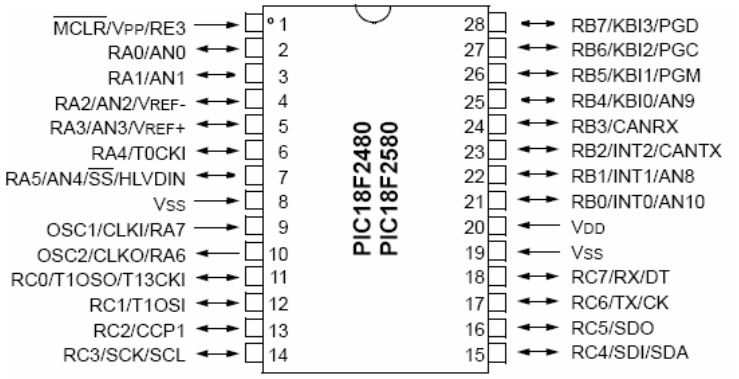



**Figura 2.3:** Diagrama de pines del microcontrolador PIC18F2480[77]

### *2.1.2.1.2 Especificaciones técnicas de los sensores*

El diseño contiene dos nodos esclavos los mismos que tienen a su cargo dos sensores, el nodo esclavo 1 manejará un sensor de movimiento y temperatura, el nodo esclavo 2 manejará un sensor de humo y un sensor magnético.

A continuación se analiza de manera breve ciertas características técnicas de algunos sensores:

#### *2.1.2.1.2.1 Sensores de movimiento*

Los sensores de movimiento son altamente utilizados en control domótico permiten controlar la iluminación de áreas de corta permanencia o la activación del sistema de seguridad.

[77] Fuente: PIC18F2480/2580/4480/4580 Data Sheet - Microchip

Muchos de los sensores de movimiento tienen integrados dispositivos como sensores de luminosidad, receptores infrarrojos, temporizadores y controladores horarios.

Sensores de movimiento disponibles en el mercado:

| ***MCPIR-2000*** | |  |
|---|---|---|
| *Amplitud* | 100º | |
| Alimentación | 3.6V DC | |
| *Frecuencia* | 433.92MHz | |
| *Alcance* | 18 m | |
| *Descripción* | Detector infrarrojo pasivo inalámbrico de doble lente con transmisor incorporado. Detección por contador de doble pulso. Conmutador para configurar zonas de detección. Baja corriente en modo de espera para ahorrar energía. Led indicador de batería baja y botón para pruebas de funcionamiento | |
| *Precio* | $ 56 | |

**Tabla 2.1:** Características generales del detector MCPIR-2000[78]

| ***Rokonet Comet PIR Pet Immune*** | |  |
|---|---|---|
| *Amplitud* | 90ºC | |
| *Alimentación* | 12V DC | |
| *Alcance* | 12 m | |
| *Descripción* | Modelo con inmunidad a perros de hasta 20Kg. Compensación de temperatura. Contador de pulsos seleccionable (1,2,3). Protección contra luz blanca. Prisma óptico. | |
| *Precio* | $ 18 | |

**Tabla 2.2:** Características generales del detector Rokonet Comet PIR Pet Immune[79]

[78] Fuente: Products - General-security.ge
[79] Fuente: Products - Gx-security.co.uk

| ***TALON TLC-15*** | |  |
|---|---|---|
| *Amplitud* | 90,5º | |
| *Alimentación* | 12V DC | |
| *Alcance* | 15 m | |
| *Descripción* | Tres opciones de detección mediante conteo de 1, 2 y 3 pulsos. Lentes protegidos con una cubierta a prueba de golpes. Compensación de temperatura. Incorpora lentes con micro prismas para la detección de una única señal. Adecuando para aplicaciones residenciales. | |
| *Precio* | $ 14.50 | |

**Tabla 2.3:** Características generales del detector TALON TLC-15[80]

### *2.1.2.1.2.2 Sensores de temperatura*

El sensor de temperatura es un dispositivo muy utilizado para tareas de automatización de procesos como lo son elaboración de alimentos, control de invernaderos, calentadores de agua, ventiladores etc.

Sensores de temperatura disponibles en el mercado:

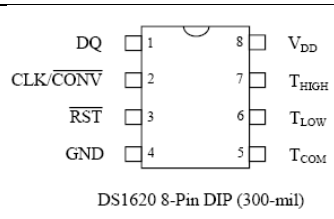
| ***DS1620*** | |  |
|---|---|---|
| *Alimentación* | 2.7V - 5.5V | |
| *Resolución* | 9 bits | |
| *Rango* | - 55ºC a 125ºC | |
| *Descripción* | No requiere de componentes externos. Tres salidas de alarma térmica. Las variables de temperatura definidas por el usuario pueden ser almacenadas en una memoria no volátil. Puede ser programado para ser utilizado en aplicaciones independientes de un microcontrolador. | |
| *Precio* | $ 13.75 | |



**Tabla 2.4:** Características generales del sensor DS1620[81]

[80] Fuente: Products – Crow Electronic Engineering (CEE)
[81] Fuente: http://datasheets.maxim-ic.com/en/ds/DS1620.pdf - Dallas Semiconductor MAXIM

| ***DS18S20*** | |  |
|---|---|---|
| *Alimentación* | 3V – 5.5 V | |
| *Resolución* | 9 bits | |
| *Rango* | -55ºC a 125ºC | |
| *Descripción* | No requiere de componentes externos. Conversión de temperatura a palabra digital en 750ms máximo. Una salida de alarma térmica. Las variables de temperatura definidas por el usuario pueden ser almacenadas en una memoria no volátil de 64 bits. | |
| *Precio* | $ 6 | |

**Tabla 2.5:** Características generales del sensor DS18S20[82]

| ***LM35DZ*** | |  |
|---|---|---|
| *Alimentación* | 4V - 30V | |
| *Rango* | 0ºC a 100ºC | |
| *Factor de escala* | 10mV/ºC | |
| *Descripción* | No requiere de calibración externa. Su salida es proporcional a la temperatura en grado Celcius. | |
| *Precio* | $ 3.25 | |

**Tabla 2.6:** Características generales del sensor LM35DZ[83]

#### *2.1.2.1.2.3 Sensores de humo*

Un objetivo básico de todo sistema de control domótico es el asegurar la vida de las personas y los bienes.

Los sensores de humo pueden ser de varios tipos, los más comunes son los fotoeléctricos.

Cabe notar que los detectores de humo no previenen incendios por tanto es importante tener otro tipo de elementos como extinguidores para complementar la seguridad.

[82] Fuente: http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2815 - Dallas Semiconductor MAXIM
[83] Fuente: http://cache.national.com/ds/LM/LM35.pdf - National Semiconductor

| ***SMD-30*** | |  |
|---|---|---|
| *Tipo* | Fotoeléctrico | |
| *Alimentación* | 9 V DC o 110V AC | |
| *Señal sonora* | 85dB | |
| *Temperatura soportada* | -10ºC a 50ºC | |
| *Descripción* | Mayor rango de detección gracias a las celdas distribuidas en el contorno del dispositivo. Posee un led monitor de funcionamiento y un pulsador de prueba. | |
| *Precio* | $ 26 | |

**Tabla 2.7:** Características generales del sensor SMD-30[84]

| ***HWG-2000*** | |  |
|---|---|---|
| *Tipo* | Fotoeléctrico | |
| *Alimentación* | 9 V DC o 110V AC | |
| *Señal sonora* | 85 dB | |
| *Temperatura soportada* | -5ºC a 40ºC | |
| *Descripción* | Verificación de funcionamiento por medio de un pulsador. Indicador luminoso de batería baja. Sirena. | |
| *Precio* | $18 | |

**Tabla 2.8:** Características generales del sensor HWG-2000[85]

| ***LX98A-D*** | |  |
|---|---|---|
| *Tipo* | Fotoeléctrico | |
| *Alimentación* | 9 V DC o 110V AC | |
| *Señal sonora* | 85 dB | |
| *Temperatura soportada* | -10ºC a 60ºC | |
| *Descripción* | Posee una sirena electrónica que incorpora un zumbador. Verificación de funcionamiento por medio de un pulsador. Indicador luminoso de batería baja. | |
| *Precio* | $ 20 | |

**Tabla 2.9:** Características generales del sensor LX98A-D[86]

[84] Fuente: Products - Acesuppliers.com
[85] Fuente: Products - Sz-wholesale.com
[86] Fuente: Products - Alibaba.com

#### *2.1.2.1.2.4 Sensores magnéticos*

Los sensores magnéticos se dividen en dos partes, un mecanismo y un imán. Su funcionamiento se basa en el campo magnético. Estos sensores son utilizados para accionar alarmas al abrir puertas o ventanas.

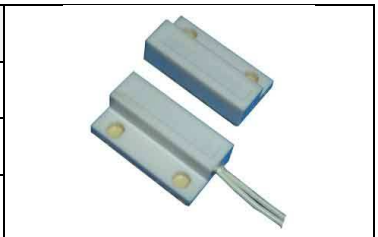
| ***AU-MS-14S*** | |  |
|---|---|---|
| *Voltaje máximo* | 40V DC | |
| *Contacto* | NC (Normalmente Cerrado) | |
| *Descripción* | Recomendado para puertas interiores y marcos de ventanas. | |
| *Precio* | $3.50 | |

**Tabla 2.10:** Características generales del sensor AU-MS-14S[87]

| ***SP-AUMS10R*** | |  |
|---|---|---|
| *Voltaje máximo* | 40 V DC | |
| *Contacto* | NA (Normalmente abierto) | |
| *Descripción* | Adecuado para montaje en marcos de puertas y ventanas. | |
| *Precio* | $ 4 | |

**Tabla 2.11:** Características generales del sensor SP-AUMS10R[88]

## 2.2 DISEÑO DE HARDWARE

El protocolo CAN fue diseñado para brindar comunicaciones confiables para aplicaciones de redes de control. Microchip ofrece varios productos para ser utilizados en aplicaciones CAN, estos productos incluyen microcontroladores de 8 y 16 bits que integran una interface CAN, controladores CAN independientes, expansiones de I/O CAN y transceivers CAN.

La versatilidad de las soluciones CAN permiten satisfacer las necesidades de una gran variedad de aplicaciones, las mismas que pueden ser creadas en base al concepto de un diseño en particular.

[87] Fuente: Products – DirectIndustry.com
[88] Fuente: Products - Onlinesecurity.com

### 2.2.1 DISEÑO DE UNA RED CAN

Una red CAN consiste de varios nodos CAN implementados en hardware, que en conjunto con software programado en los microcontroladores, pueden enviar y transmitir mensajes que contienen información útil para otros nodos de la red utilizando el bus CAN.

Las opciones de diseño de una red CAN son múltiples, cada una varía de acuerdo a los elementos que conforman el nodo, a continuación se plantea tres ejemplos de diseño de nodos CAN.

#### 2.2.1.1 Diseño de un nodo CAN utilizando un microcontrolador, un controlador CAN y un transceiver CAN

Diseñar un simple nodo utilizando un microcontrolador CAN que posee periféricos extras, una gran capacidad de memoria RAM, etc., puede resultar muy costoso, si la aplicación no es muy exigente, en estos casos se puede optar por utilizar otras soluciones CAN más económicas.

A continuación en la figura 2.4 se muestra el esquema de un nodo CAN simple, este nodo integra un microcontrolador (PIC12F875), un controlador CAN independiente con interfaz SPI (MCP2515) y un transceiver CAN (MCP2551).

El controlador CAN realiza las funciones de emisión y recepción de mensajes CAN, este dispositivo se conecta al microcontrolador utilizando la interfaz SPI, de esta manera se puede añadir una interfaz CAN a cualquier microcontrolador utilizando el MCP2510.
Los cinco pines usados por el microcontrolador para la conexión SPI, pueden ser recuperados utilizando los pines de transmisión y recepción del controlador CAN, como entradas y salidas de propósito general.

El MCP2551 es un transceiver CAN de alta velocidad que sirve de interfaz entre el bus CAN y el controlador CAN. El MCP2551 implementa la capa física ISO11898 requerida para aplicaciones CAN, su tarea es convertir las señales lógicas

generadas por el controlador, a señales diferenciales en el bus. Las señales diferenciales son menos propensas a perturbaciones eléctricas.

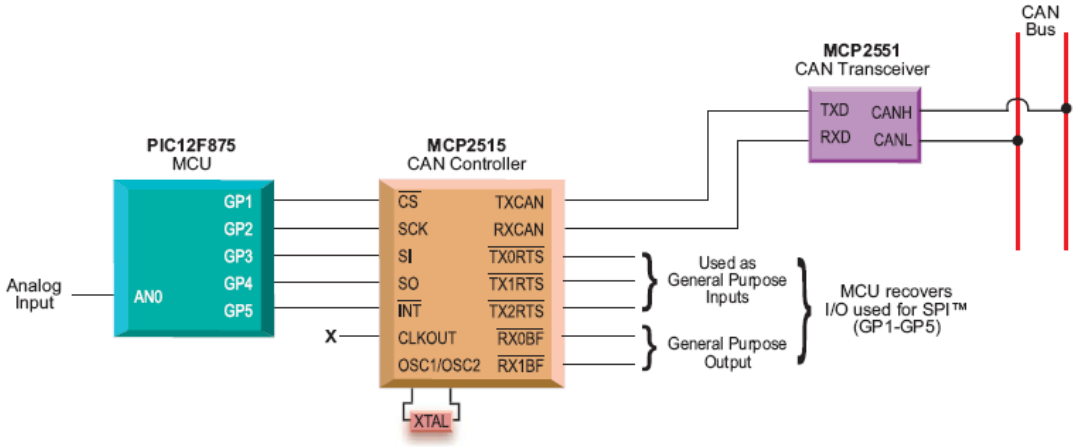



**Figura 2.4:** Diseño de un nodo CAN utilizando un microcontrolador de propósito general, un controlador CAN y un transceiver CAN.[89]

El MCP2551 protege al controlador CAN de anomalías eléctricas tales como variaciones de voltaje, cortos circuitos y transiciones ocurridas en el bus.

### 2.2.1.2 Diseño de un nodo CAN utilizando un microcontrolador CAN y un transceiver CAN

Este esquema de diseño es adecuado para aplicaciones que requieran nodos con cierto grado de inteligencia, capaces de tomar decisiones ante eventuales sucesos. Cada nodo CAN está conformado por un microcontrolador con interface CAN, lo que elimina la necesidad de añadir el controlador CAN MCP2515.

Cada nodo CAN está compuesto de un microcontrolador con interface CAN y un transceiver MCP2551. Este tipo de diseño podría resultar costoso, debido a que este microcontrolador en comparación con otros de características técnicas inferiores es más caro y en ciertas ocasiones es demasiado sofisticado para aplicaciones no muy exigentes, sin embargo, el costo se compensa con el hecho de no utilizar un elemento adicional en el diseño (controlador CAN independiente), lo que resulta en la disminución de la complejidad en el circuito impreso.

[89] Fuente: Interface Products Design Guide - Microchip

En la figura 2.5 se muestra la conexión básica de este tipo de diseño.

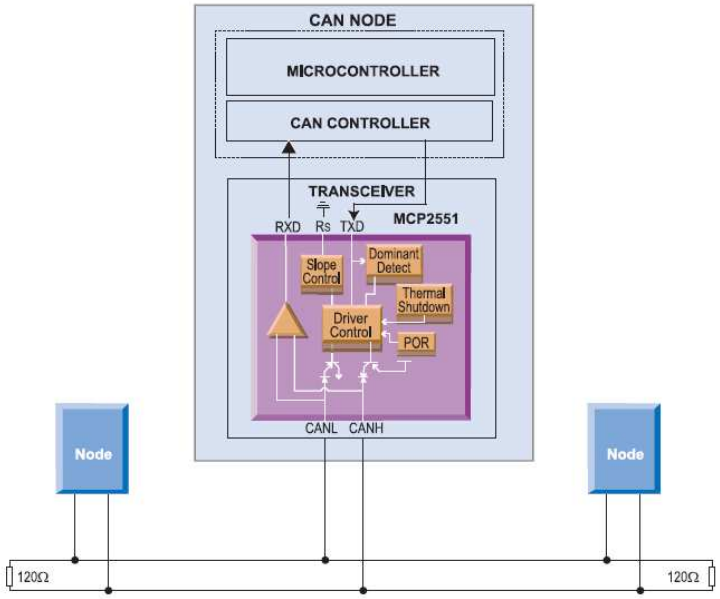



**Figura 2.5:** Diseño de un nodo CAN utilizando un microcontrolador CAN y un transceiver CAN[90]

#### 2.2.1.3 Diseño de un nodo CAN utilizando una expansión de I/O CAN y un transceiver CAN

El MCP25050 es un dispositivo que opera como un sistema de expansión de I/O CAN. Este dispositivo incluye algunos periféricos como 8 I/O digitales, 4 convertidores A/D de 10 bits y 2 canales PWM de 10 bits. El MCP25050 permite enviar y recibir mensajes automáticamente cuando una entrada cambia de estado o cuando el canal analógico excedió el umbral preestablecido. Para la comunicación con el MCP25050, se reservan tres bits (más un bit RTR o un bit ID para la dirección) del campo identificador/arbitraje del mensaje CAN, esto permite al nodo maestro comunicarse o controlar a la expansión de I/O CAN a través del bus CAN.

El MCP25050 es ideal para redes sencillas, de bajo costo, en las que los requerimientos no incluyan nodos inteligentes. El bajo costo se debe a que el MCP25050 posee características técnicas inferiores a las de un microcontrolador CAN y por consiguiente se encuentra disponible en el mercado a un precio más asequible.

[90] Fuente: Interface Products Design Guide - Microchip

El esquema mostrado en la figura 2.6, muestra una red de 4 sensores, utilizando un único microcontrolador, el mismo que actuará de nodo maestro. Nótese que al igual que en los otros diseños se necesita de la presencia de un transceiver MCP2551 para adaptar las señales entregadas por el controlador al bus CAN.

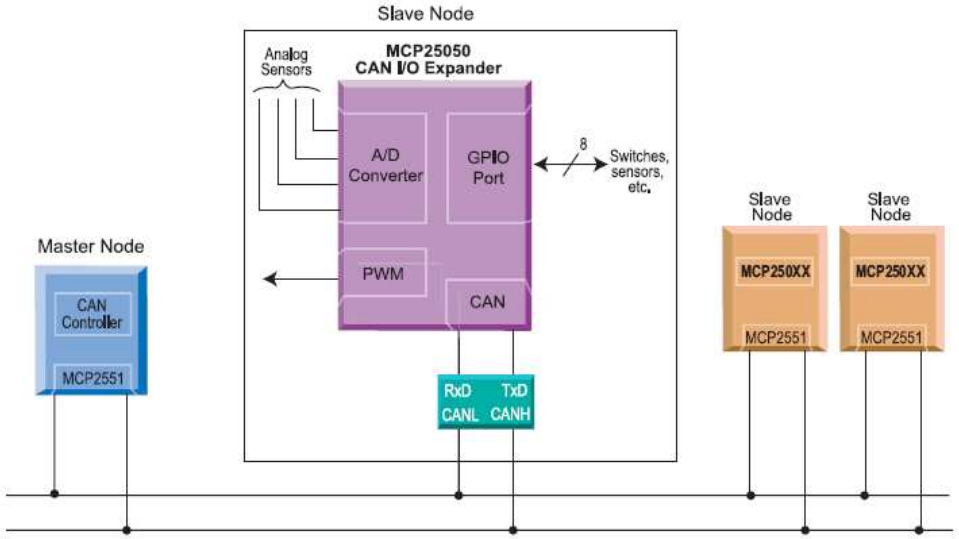



**Figura 2.6:** Diseño de un nodo CAN utilizando una expansión I/O CAN y un transceiver CAN.[91]

### 2.2.2 SELECCIÓN DEL DISEÑO

Debido a que la aplicación se remite a un prototipo, los requerimientos de precisión, sensibilidad y calibración en los sensores no son mandatorios. Posterior al análisis de los sensores existentes en el mercado, se han seleccionado opciones adecuadas en cuanto a características técnicas y a costos. Las soluciones escogidas se detallan en la parte de *diseño de los nodos esclavos*[92].

Cualquiera de las opciones de diseño para la red CAN propuestas es totalmente funcional y válida para el propósito del prototipo, sin embargo, se optó por el diseño de una red CAN utilizando microcontroladores con interface CAN (PIC18F2480) y transceivers CAN (MCP2551). El motivo por el cual se optó por este diseño fue el desarrollar un nodo CAN inteligente y disminuir el número de

[91] Fuente: Interface Products Design Guide - Microchip
[92] Ver página 92.

elementos requeridos en el prototipo, este objetivo se logra utilizando microcontroladores integrados a los sensores.

A continuación la figura 2.7 muestra la configuración sugerida para el prototipo, la cual consiste de tres nodos sobre un bus CAN, uno de los nodos actuará de maestro y los dos restantes actuarán de esclavos. El nodo maestro se conecta a un computador personal utilizando el puerto serial a través de la interfaz EUSART del microcontrolador, y cada nodo esclavo tendrá bajo su control dos sensores.

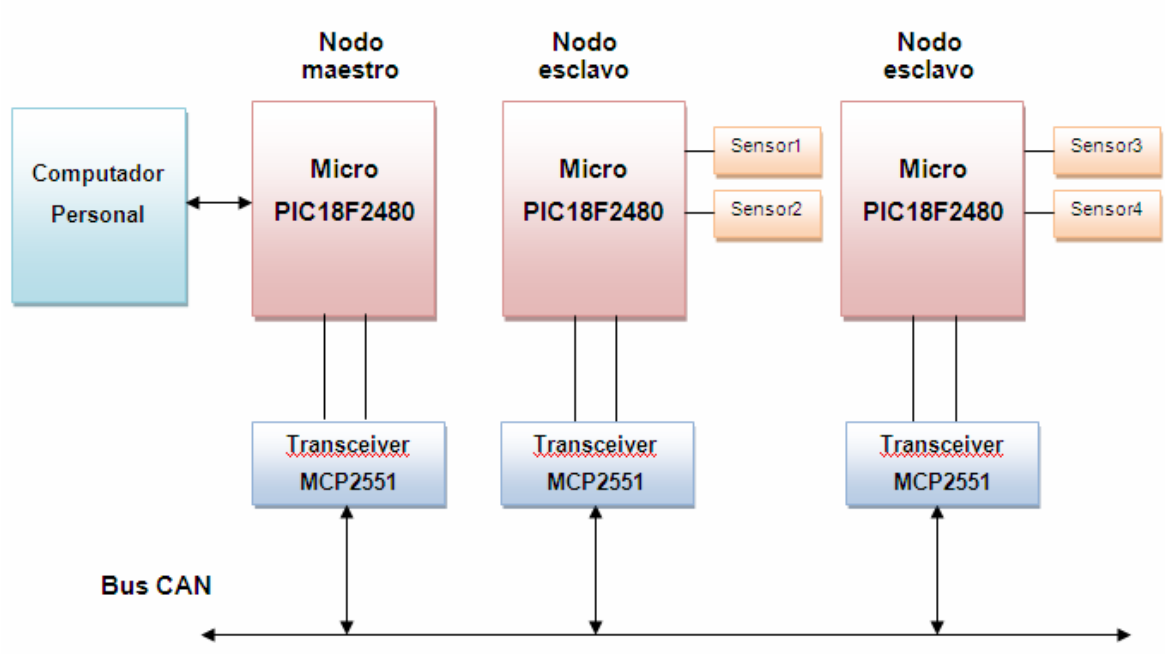



**Figura 2.7:** Diagrama general del prototipo

### 2.2.2.1 Diseño del nodo maestro

El nodo maestro consiste de un microcontrolador PIC18F2480 y un transceiver MCP2551. Este nodo actuará como gateway entre los esclavos y el computador personal, por esta razón incluye un MAX232.

Debido a la naturaleza del nodo maestro, no tendrá a su cargo ningún tipo de sensor, este nodo se encargará únicamente del control de los esclavos, enviando y recibiendo mensajes a través del bus.

#### *2.2.2.1.1 MAX232*

Para acoplar los niveles de voltaje utilizados por el puerto serial con niveles TTL utilizados por el microcontrolador, se utiliza el circuito integrado MAX232, éste dispositivo se encarga de enviar señales digitales generadas por el microcontrolador al computador personal y viceversa.

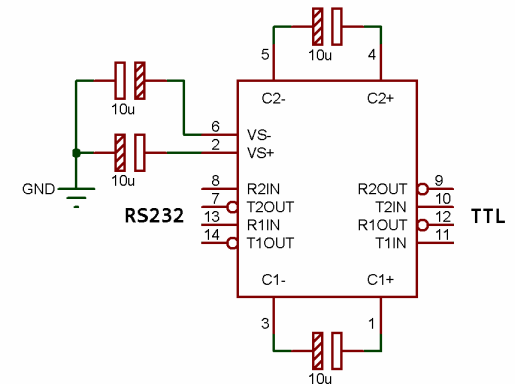



**Figura 2.8:** Diagrama de conexión del MAX232 al microcontrolador[93]

| Elemento | No. Pin | Observación |
|---|---|---|
| **PIC18F2480** | 1 | $\overline{MCLR}$ Conectado al circuito de reset |
| | 9 | OSC1 Conectado a un terminal del cristal de 8MHz y a un terminal de un capacitor de 22pF. |
| | 10 | OSC2 Conectado a un terminal del cristal de 8MHz y a un terminal de un capacitor de 22pF. |
| | 17 | TX conectado a la entrada TTL del Max232 |
| | 18 | RX conectado a la salida TTL del Max232 |
| | 19 | VSS conectado a tierra. |
| | 20 | VDD conectado a 5V. |
| | 23 | CANTX conectado a un resistencia de pull-up de 10KΩ |
| | 24 | CANRX conectado a un resistencia de pull-up de 10KΩ |
| **MCP2551** | 1 | TXD conectado al pin CANTX del microcontrolador |
| | 2 | GND conectado a tierra. |
| | 3 | VCC conectado a 5V. |
| | 4 | RXD conectado al pin CANRX del microcontrolador |
| | 6 | CANH conectado al bus y a una terminación de 120Ω |
| | 7 | CANL conectado al bus y a una terminación de 120Ω |
| | 8 | Rs conectado a tierra por medio de una resistencia de 10Ω. |

**Tabla 2.12:** Elementos y distribución de pines en el nodo maestro[94]

[93] Fuente: MAX220/222/232A/233A/242/243 Data Sheet – Maxim

*2.2.2.1.2 Circuito de reset*

Es importante proveer un mecanismo externo para reinicializar al microcontrolador. El circuito de reset es simple, consiste en utilizar una resistencia y un switch conectados al pin $\overline{MCLR}$ del PIC18F2480. Este circuito funciona a manera de disparador, colocando al pin en un nivel bajo permitiendo así un reset instantáneo.

Microchip recomienda utilizar una resistencia R<40KΩ para no violar las especificaciones eléctricas requeridas por el microcontrolador.

En el prototipo se utilizó una resistencia de 10KΩ como muestra en la figura:

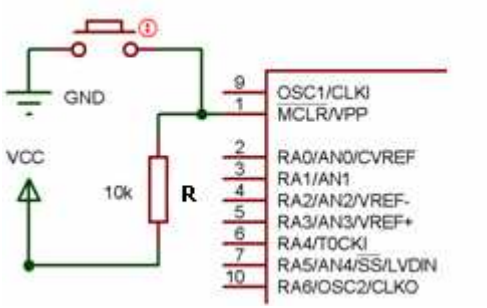



**Figura 2.9:** Circuito de reset utilizado en todos los nodos[95]

El diagrama esquemático así como el PCB del nodo maestro se pueden visualizar en el Anexo G correspondiente a los diagramas esquemáticos y PCB de los nodos.

**Diseño de los nodos esclavos**

Cada nodo esclavo tiene a su cargo dos sensores y el diseño de cada uno dependerá de los mismos.

**2.2.2.2 Diseño del Nodo esclavo 1**

El nodo esclavo 1 consiste de un microcontrolador PIC18F2480 y un transceiver MCP2551, además este nodo, tiene a su cargo un sensor de temperatura LM35DZ, un sensor de movimiento TALON TLC-15 y un buzzer.

[94] Fuente: PIC18F2480 Data Sheet – Microchip, MCP2551 Data Sheet – Microchip
[95] Fuente: PIC18F2480 Data Sheet – Microchip

*2.2.2.2.1 Sensor de temperatura LM35DZ*

El LM35DZ es un sensor de temperatura de precisión, cuyo voltaje de salida es proporcional a la temperatura en grados Celcius (ºC).

Este sensor tiene un rango de operación limitado entre 0ºC y +100ºC y no requiere de ninguna calibración externa para proveer exactitudes típicas de ±1/4ºC a temperaturas ambiente y ±3/4ºC a temperaturas límites de 0ºC a 100ºC.

Se presenta en diferentes encapsulados. El LM35DZ utilizado en el prototipo es de encapsulado TO-92, éste tiene forma de transistor de 3 patas.


**Figura 2.10:** Distribución de pines del sensor LM35DZ[96]

El LM35DZ produce una salida de tensión lineal equivalente a 10mV/ºC y funciona con un rango de alimentación comprendido entre 4 y 30 voltios.

Este sensor se conecta a uno de los convertidores análogo/digital del microcontrolador para que éste procese la medida.

*2.2.2.2.2 Sensor de movimiento TALON TLC-15*

El sensor TALON TLC-15 es un detector de proximidad infrarrojo pasivo que incorpora lentes para una mejor captura, posee una cubierta a prueba de golpes y tiene ajuste de sensibilidad en modo dual: conteo de un solo pulso o conteo de múltiples pulsos, total inmunidad a ruido ambiental y un alcance de 15m.

**Figura 2.11:** Sensor TALON TLC-15[97]

[96] Fuente: LM35/LM35A/LM35C/LM35CA/LM35D Precision Centigrade Temperature Sensors – Data Sheet National Semiconductor
[97] Fuente: Products - Crow Electronic Engineering (CEE)

El TLC-15 está basado en la tecnología SMD (Dispositivos de Montaje en Superficie), que consiste en circuitos integrados de silicio montados en dispositivos de diminutas dimensiones.

| Elemento | No. Pin | Observación |
|---|---|---|
| **PIC18F2480** | 1 | $\overline{MCLR}$ Conectado al circuito de reset. |
| | 2 | RA0/AN0 conectado al pin 2 del sensor LM35DZ. |
| | 9 | OSC1 Conectado al terminal de un cristal de 8MHz y al terminal de un capacitor de 22pF. |
| | 10 | OSC2 Conectado al terminal del cristal de 8MHz y al terminal de un capacitor de 22pF. |
| | 19 | VSS conectado a tierra. |
| | 20 | VDD conectado a 5V. |
| | 21 | RB0/INT0 conectado a una resistencia de pull-up de 4,7 KΩ y conectado al sensor de movimiento. |
| | 23 | RB2/INT2/CANTX conectado a un resistencia de pull-up de 10KΩ |
| | 24 | RB3/CANRX conectado a un resistencia de pull-up de 10KΩ |
| | 28 | RB7/PGD conectado a la base de un transistor 2N3904 por medio de una resistencia de 2,2KΩ. Este pin controla la activación del buzzer. |
| **MCP2551** | 1 | TXD conectado al pin CANTX del microcontrolador |
| | 2 | GND conectado a tierra. |
| | 3 | VCC conectado a 5V. |
| | 4 | RXD conectado al pin CANRX del microcontrolador |
| | 6 | CANH conectado al bus. |
| | 7 | CANL conectado al bus. |
| | 8 | Rs conectado a tierra por medio de una resistencia de 10Ω. |
| **LM35DZ** | 1 | Conectado a VCC |
| | 2 | Conectado a tierra por medio de una resistencia de 100Ω. Este pin corresponde al voltaje de salida proporcional a la temperatura medida, por lo que se conecta a una entrada analógica del microcontrolador, en este caso se utilizó la entrada RA0/AN0. |
| | 3 | Conectado a GND |

**Tabla 2.13:** Elementos y distribución de pines en el nodo esclavo 1[98]

[98] Fuente: PIC18F2480 Data Sheet – Microchip, MCP2551 Data Sheet – Microchip, LM35D Precision Centigrade Temperature Sensors Data Sheet - National Semiconductor

El sensor de temperatura está conectado al pin 2 (AN0) correspondiente al primero de ocho canales analógico/digitales que posee el microcontrolador. Cada canal tiene una resolución de 10 bits ($2^n = 1024$ valores posibles) lo que aumenta la sensibilidad de la medición es decir se utiliza un registro de 10 bits para digitalizar la señal detectada, se debe considerar que el convertidor tiene un error de 1LSB (Least Significant Bit), lo que significa que aunque el LM35DZ tenga una salida estable, siempre se obtendrá una leve diferencia en la medición.

Usando los 10 bits y una referencia de 5V, el convertidor A/D tiene una sensibilidad:

$$Sensibilidad = \frac{V_{ref}}{2^n - 1} = \frac{5V}{2^{10} - 1} = \frac{5V}{1023} = 4.8875mV$$

Para obtener el valor de temperatura se utilizó la formula:

$$T^o = \frac{Sensibilidad \times read\_adc()}{Factor\ de\ escala}$$

$$T^o = \frac{\frac{5V}{1023} \times read\_adc()}{10mV/C}$$

$$T^o = 0.48875 \times read\_adc()$$

Para efectos de estabilidad se tomaron y promediaron 20 muestras.

El sensor de movimiento es controlado por el microcontrolador a través del pin 21 habilitando la interrupción INT0.

El diagrama esquemático así como el PCB del nodo esclavo1 se pueden visualizar en el Anexo G correspondiente a los diagramas esquemáticos y PCB de los nodos.

### 2.2.2.3 Diseño del Nodo esclavo 2

El nodo esclavo 2 consiste de un microcontrolador PIC18F2480 y un transceiver MCP2551, este nodo, tiene a su cargo un sensor de humo, un sensor magnético y un buzzer.

#### *2.2.2.3.1 Sensor de humo LX98A-D*

Este sensor es de tipo fotoeléctrico, detecta humo provocado por quemar objetos de diferentes tipo, no trabaja en condiciones de viento ya que el humo se esparce e impide detectarlo. Otra condición para que no trabaje el detector es la presencia de objetos que obstaculicen el humo.

El detector trabaja con una fuente de 9V DC o con 110V AC.
En el prototipo se utilizó una batería de 9V y se incluyó un switch para el apagado del sensor.

**Figura 2.12:** Sensor de humo LX98A-D[99]

#### *2.2.2.3.2 Sensor magnético AU-MS-14S*

Este sensor basa su funcionamiento en el campo magnético lo cual le permite actuar como conmutador.

**Figura 2.13:** Conmutador magnético AU-MS-14S[100]

[99] Fuente: Product list – Alibaba.com
[100] Fuente: Product list – DirecIndustry.com

| Elemento | No. Pin | Observación |
|---|---|---|
| PIC18F2480 | 1 | $\overline{MCLR}$ Conectado al circuito de reset |
| | 9 | OSC1 Conectado a un terminal del cristal de 8MHz y a un terminal de un capacitor de 22pF. |
| | 10 | OSC2 Conectado a un terminal del cristal de 8MHz y a un terminal de un capacitor de 22pF. |
| | 19 | VSS conectado a tierra. |
| | 20 | VDD conectado a 5V. |
| | 21 | RB0/INT0 conectado a tierra a través de una resistencia de 100Ω y a un terminal del sensor magnético. |
| | 22 | RB1/INT1 conectado a la salida del LM358 |
| | 23 | RB2/INT2/CANTX conectado a un resistencia de pull-up de 10KΩ |
| | 24 | RB3/CANRX conectado a un resistencia de pull-up de 10KΩ |
| | 28 | RB7/PGD conectado a la base de un transistor 2N3904 por medio de una resistencia de 2,2KΩ. Este pin controla la activación del buzzer. |
| MCP2551 | 1 | TXD conectado al pin CANTX del microcontrolador |
| | 2 | GND conectado a tierra. |
| | 3 | VCC conectado a 5V. |
| | 4 | RXD conectado al pin CANRX del microcontrolador |
| | 6 | CANH conectado al bus y a una terminación de 120Ω |
| | 7 | CANL conectado al bus y a una terminación de 120Ω |
| | 8 | Rs conectado a tierra por medio de una resistencia de 10Ω. |
| LM358 | 1 | Salida del amplificador operacional A |
| | 2 | Entrada inversora del amplificador A conectado a tierra con un resistencia de 1,1KΩ y a VCC con una resistencia de 4,7KΩ. |
| | 3 | Entrada no inversora del amplificador A conectado a la salida del sensor de humo. |
| | 4 | Tierra (GND) |
| | 8 | Voltaje de referencia positivo V+ correspondiente al voltaje de la fuente es decir 5V. |

**Tabla 2.14:** Elementos y distribución de pines en el nodo esclavo 2[101]

El sensor de humo utiliza un detector fotoeléctrico en circuito integrado con I/O MC145010, el pin de I/O es utilizado para conectar hasta 40 unidades a una señal común, adicionalmente este pin puede ser utilizado para activar luces de

[101] Fuente: PIC18F2480 Data Sheet – Microchip, MCP2551 Data Sheet – Microchip, LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers - National Semiconductor

escape y habilitar alarmas auxiliares. En el prototipo se utilizó este pin para la conexión del sensor de humo con el microcontrolador.

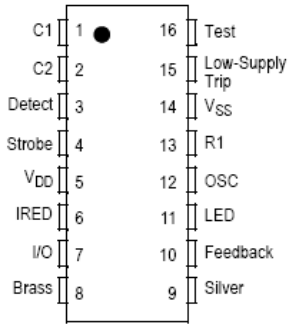



**Figura 2.14:** Diagrama de pines del MC145010[102]

Una vez que la alarma se activa, la señal de salida por este pin se incrementa a 9V. Dado que el microcontrolador no permite niveles de voltaje mayores a 5V, por lo que la señal debía reducirse, por tanto se incluyó un divisor de tensión en el sensor de humo lográndose obtener una salida de 4,27V.

En el prototipo se utilizó el amplificador operacional LM358 debido a que puede ser alimentado con una sola fuente. El LM358 fue utilizado en modo comparador, la entrada no inversora se alimentó con el voltaje obtenido en el sensor de humo, la entrada inversora se conectó a un divisor de tensión en la fuente igual a 0,78V.

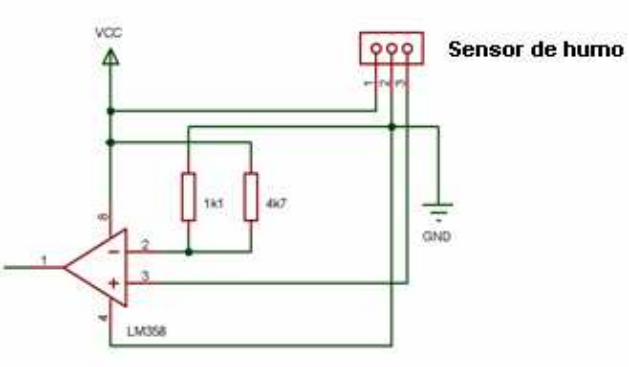



**Figura 2.15:** Esquema de conexión del LM358[103]

[102] Fuente: Freescale Semiconductor - Technical Data
[103] Fuente: LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers - National Semiconductor

La salida del amplificador operacional se conectó al microcontrolador, el mismo que fue configurado para manejar el detector de humo mediante la función de interrupción INT1.

El sensor magnético normalmente conforma un circuito cerrado, debido a la impedancia propia del sensor y a la resistencia de 100Ω colocada a tierra, el voltaje obtenido bajo estas condiciones fue de 4V aproximadamente, al momento de la activación del sensor el circuito se abre. El microcontrolador fue configurado para manejar este sensor utilizando la interrupción INT0.

El diagrama esquemático así como el PCB del nodo esclavo 1 se pueden visualizar en el Anexo G.

En la figura 2.16 se muestra un esquema del prototipo completo. Para ilustrar su funcionamiento se detalla el envío de un mensaje de consulta específico.

Un usuario desea saber el estado del sensor de movimiento, envía desde el computador el mensaje de 5 bytes 2a000[104] al nodo maestro, el nodo maestro envía en modo de difusión a los nodos esclavos el mensaje de 2 bytes 0xa2 0x00[105] a través de los pines 23 y 24, el mensaje llega a las líneas TXD y RXD del transceiver MCP2551 el mismo que se encarga de proveer de la capa física del bus CAN y envía la señal por sus pines 6 y 7 correspondientes a las líneas CAN_L y CAN_H. El mensaje es recibido por todos los nodos y es filtrado para determinar si el mensaje debe ser procesado, de no ser así lo descarta. El nodo esclavo 1 procesa el mensaje y envía al nodo maestro una respuesta. En el caso de haber sido activado el sensor, el nodo esclavo 1 envía al maestro el mensaje de 1 byte 0x31[106] y en el caso de estar desactivado el mensaje 0x30.

La activación y desactivación de los sensores se registran en una variable local al nodo esclavo la cual se mantiene almacenada en espera de una nueva consulta o hasta que se encere explícitamente.

[104] Ver tabla 2.9: Flujo de datos computador personal – nodo maestro
[105] Ver tabla 2.10: Flujo de datos nodo maestro – nodos esclavos
[106] Ver tabla 2.11: Flujo de datos nodo esclavo 1 – nodo maestro

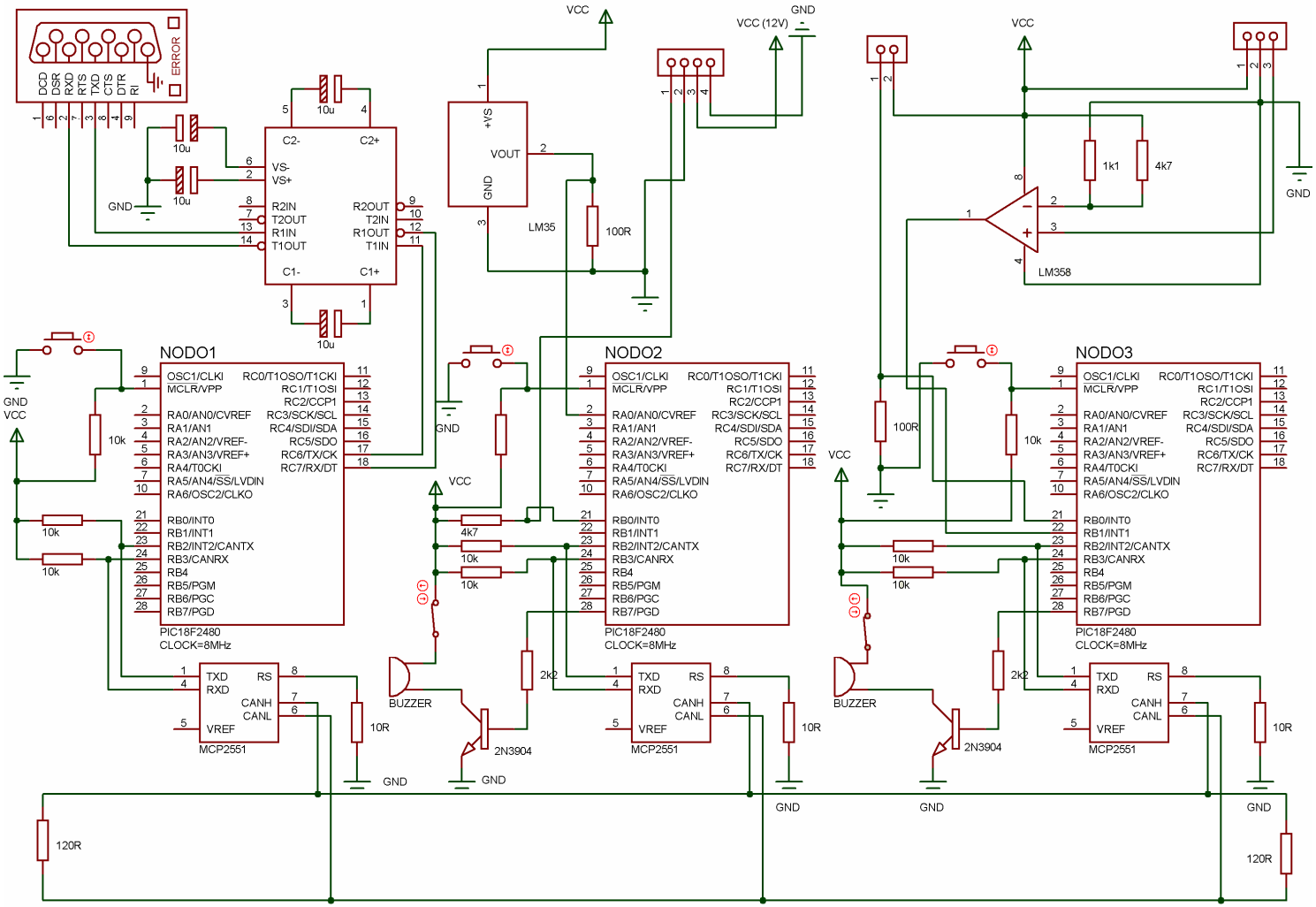



**Figura 2.16:** Esquemático del prototipo completo

## 2.3 DISEÑO DEL SOFTWARE

El sistema a diseñar se divide en cuatro partes:

- **Nodo maestro:** recibe mensajes del computador personal, monitorea los estados de los nodos y emite mensaje hacia computador personal.
- **Nodo esclavo 1:** comanda a los sensores de temperatura y movimiento.
- **Nodo esclavo 2:** comanda a los sensores de humo y magnético.
- **Estación de monitoreo:** emite y recibe mensajes del nodo maestro.

Cada una de estas partes contiene software adecuado para la tarea que realizará.

### 2.3.1 ESQUEMA GENERAL DEL PROTOTIPO

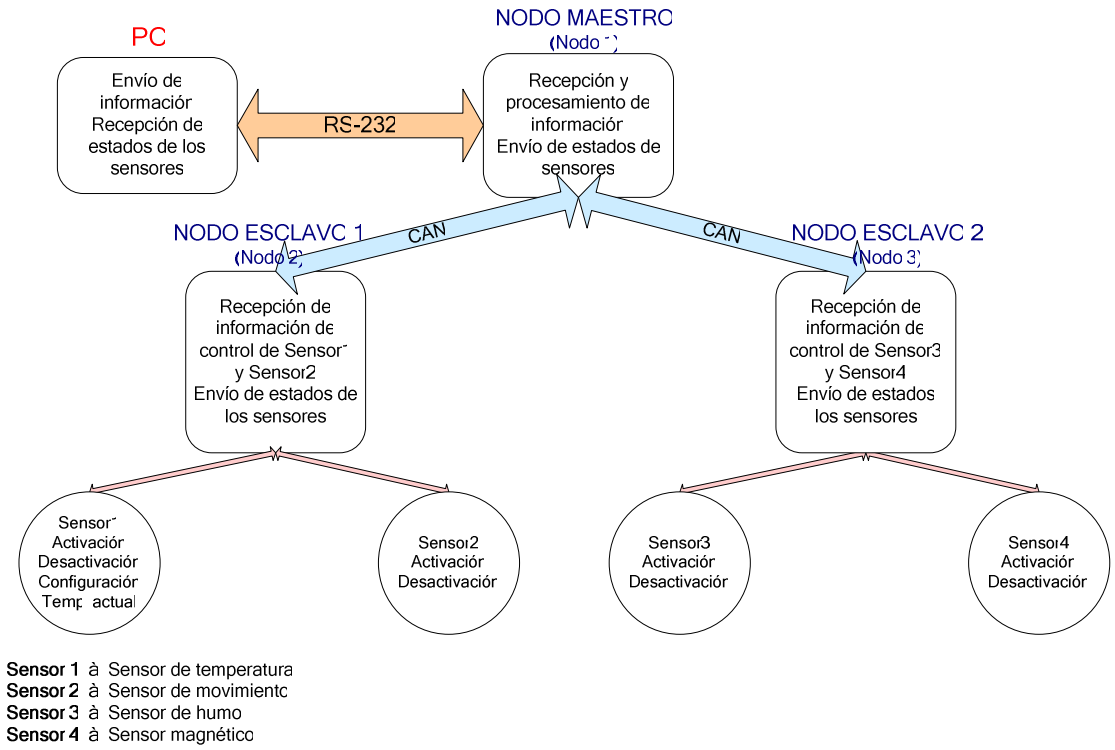



**Figura 2.17:** Diagrama de bloques del prototipo

Las herramientas de desarrollo utilizadas fueron: MikroC 5.0 y Microsoft Visual Studio 2005. El programa MikroC ofrece la librería CAN para facilitar el diseño de sistemas CAN, esta librería contiene funciones de configuración/inicialización y funciones de operación.

Las funciones de configuración/inicialización son:

- **CANSetOperationMode:** Configura el modo de operación del módulo CAN, los modos permitidos son bits, normal, sleep, loop, listen y config.

  *Modo normal:* En este modo el nodo puede enviar y recibir mensajes, aceptar ACK, detectar errores, etc.

  *Modo sleep:* En este modo se deshabilita el módulo CAN, para abandonar este estado se usa interrupciones.

  *Modo loop:* En este modo se envía y recibe mensajes al interior del microcontrolador para verificación del funcionamiento del módulo CAN.

  *Modo listen:* En este modo únicamente se permite la recepción de mensajes, incluyendo los de error, este modo puede ser usando para la detección de velocidad.

  *Modo config:* En este modo se puede configurar filtros y máscaras de recepción, no se puede enviar ni recibir mensajes.
- **CANGetOperationMode:** Retorna el modo de operación del módulo CAN.
- **CANInitialize:** Inicializa el modulo CAN con valores predefinidos de acuerdo al microcontrolador utilizado, los valores configurables son salto de sincronización (SJW), prescaler de velocidad del bit (BRP), segmento de fase 1 (PHSEG1), segmento de fase 2 (PHSEG2) y segmento de propagación (PROSEG) correspondientes a los segmentos de un bit nominal en el bus CAN. La configuración de estas variables y la frecuencia del oscilador del microcontrolador, determinan la velocidad de trasmisión en el bus. Conjuntamente con esta función se debe establecer las banderas de configuración que van a ser utilizadas como son tipo de muestreo, tipo de mensaje (estándar o extendido) entre otras.

  Más adelante se detallará que valores de segmentos fueron utilizados para conseguir que el bus trabaje a una velocidad de 500Kbps a una frecuencia de 8Mhz.
- **CANSetBaudRate:** Configura la tasa de transferencia del bus, dada la complejidad del protocolo no es tan sencillo forzar a trabajar a una tasa determinada, se considera además que en la inicialización del bus, la velocidad ya se establece de alguna forma, por ésta razón ésta función puede ser omitida.

- **CANSetMask:** Configura las máscaras para las funciones de filtraje de mensajes. De acuerdo al microcontrolador, para trasmitir mensajes se tienen tres buffers y para recibir se tiene dos buffers, la máscara indican cual buffer de recepción se utilizará, por tanto se pude configurar hasta dos máscaras.
  Configurar adecuadamente la máscara es importante ya que con esto se controla que el buffer de mayor prioridad (buffer 0) no se sobrecarge y se pierdan los mensajes, de esta forma se hace un balanceo de carga manual.
- **CAN SetFilter:** Permite realizar las funciones de filtraje. Los filtros dependen del buffer de recepción y del identificador del nodo local. Se pueden configurar hasta dos filtros en el buffer 0 y cuatro filtros en el buffer 1, en total seis filtros diferentes. En el filtro también debe especificarse el tipo de mensaje que se aceptará es decir si es estándar o extendido.

| Buffer RXBn | Máscara RXMn | Filtro RXFn |
|---|---|---|
| 0 | 0 | 0 |
| | | 1 |
| 1 | 1 | 2 |
| | | 3 |
| | | 4 |
| | | 5 |

**Tabla 2.15:** Máscaras y filtros permitidos en los buffers de recepción.[107]

Las máscaras son usadas para determinar cuales bits en el identificador del mensaje serán examinadas con los filtros. La tabla de verdad a continuación muestra como un bit en el identificador es comparado con la máscara y filtro, para determinar si un mensaje puede ser almacenado en el buffer de recepción, si la máscara es configurada en 0, todos los bits serán automáticamente aceptados independientemente del filtro de aceptación.

[107] Fuente: Datasheet PIC18F2480 - Microchip

| Mask bit n | Filter bit n | Messenger Identifier bit n001 | Accept or Reject bit n |
|---|---|---|---|
| 0 | X | X | Accept |
| 1 | 0 | 0 | Accept |
| 1 | 0 | 1 | Reject |
| 1 | 1 | 0 | Reject |
| 1 | 1 | 1 | Accept |

**Legend: x**=don't care

**Tabla 2.16:** Filter/Mask Truth Table[108]

A continuación se muestra un diagrama de flujo del proceso de inicialización del módulo CAN para la familia PIC18CXXX.

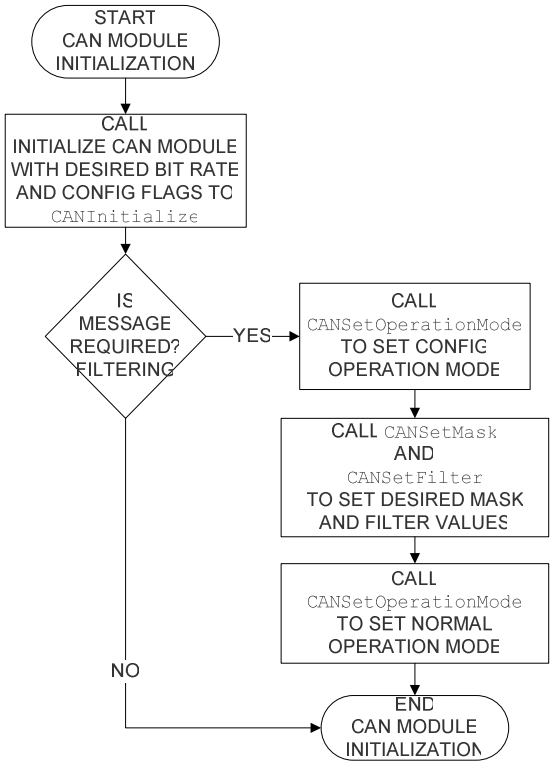



**Figura 2.18:** Procedimiento de inicialización del módulo CAN para la familia PIC18[109]

Las funciones de operación son:

- **CANRead:** Lee el mensaje del buffer de recepción, retorna cero si no existe mensaje. Se debe especificar una referencia al identificador del nodo trasmisor y a la longitud de los datos a ser recibidos.

[108] Fuente: Datasheet PIC18F2480 - Microchip
[109] Fuente: PIC18C CAN Routines in 'C' - Microchip Technology, Inc. (AN738)

- **CANWrite:** Envía el mensaje que se encuentra en el buffer de emisión, si la cola está llena, la función retorna cero. Se debe especificar el identificador del nodo receptor y la longitud del mensaje a ser enviado.

Todas las funciones CAN excepto CANSetBaudRate fueron utilizadas en el diseño de software de los nodos maestro y esclavos.

Otra librería de MikroC utilizada en el diseño fue la librería USART, ésta librería facilita al diseñador el uso del módulo serial para comunicación asincrónica en modo full-duplex. Esta librería contiene cuatro funciones:

- **Usart_Init:** Inicializa el hardware del módulo USART del microcontrolador con la tasa de transferencia deseada, cabe notar que no todas las velocidades son permitidas y dependen de la frecuencia de oscilación utilizada para el microcontrolador que en este caso fue 8MHz.
- **Usart_Data_Ready:** Retorna 1 si existen datos y 0 si no existen datos en el buffer de recepción del módulo USART.
- **Usart_Read:** Retorna el byte recibido si no existe byte retorna 0. Antes de leer el buffer de recepción, es conveniente verificar si existe dato antes.
- **Usart_Write:** Transmite un byte de datos vía USART.

### 2.3.2 CÁLCULO DE LAS VARIABLES DE TEMPORIZACIÓN DEL BIT CAN

El protocolo CAN define un bus serial asincrónico con codificación No Retorno a Cero[110] (NRZ) diseñado para comunicaciones veloces y robustas en ambientes ruidosos tales como la industria automotriz y aplicaciones industriales. El protocolo CAN permite al usuario programar la velocidad del bus, el punto de muestreo en el bit y el número de veces que el bit será muestreado. Con estas características, la red puede ser optimizada de acuerdo a los requerimientos de la aplicación.

Existen relaciones entre los parámetros de temporización del bit, los retardos de propagación en el bus y la frecuencia del oscilador en el sistema, que permiten definir la velocidad a la que viajarán los datos en el bus.

[110] No Retorno a Cero (NRZ).- Tipo de codificación digital en donde el nivel de la señal es siempre positivo o negativo.

El tiempo de bit en el protocolo CAN está constituido de cuatro segmentos no superpuestos, cada uno de estos segmentos son representados por unidades enteras llamada tiempo quantum (TQ).

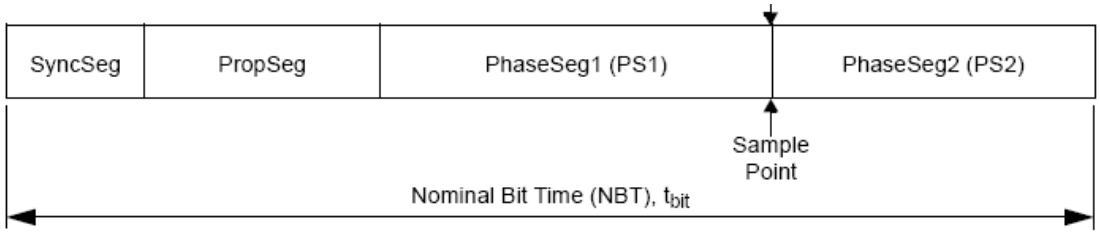



**Figura 2.19:** Segmentos del tiempo de bit CAN[111]

Para determinar la velocidad en el bus, el diseñador necesita establecer los valores de los segmentos del tiempo de bit CAN.

A continuación se describe en la siguiente tabla los rangos mínimos programables requeridos por el protocolo CAN:

| Parámetro | Rango | Observación |
|---|---|---|
| BRP | [1 .. 32] | Define la longitud del tiempo quatum TQ |
| Sync_Seg | 1 TQ | Longitud fija, Sincronización del bus con el reloj del sistema |
| Prop_Seg | [1 .. 8] TQ | Tiempo de compensación para los retardos físicos |
| Phase_Seg1 | [1 .. 8] TQ | Puede ser alargado para propósitos de sincronización |
| Phase_Seg2 | [1 .. 8] TQ | Puede ser acortado para propósitos de sincronización |
| SJW | [1 .. 4] TQ | No debe ser más largo que el segmento de fase 2 |

**Tabla 2.17:** Parámetros para el tiempo de bit CAN[112]

A continuación se procede a calcular los valores de los segmentos del tiempo de bit CAN, partiendo de las siguientes premisas:

- Velocidad de trasmisión: 500 Kbps.
- Frecuencia del oscilador del microcontrolador: 8MHz
- Periodo del bit CAN: 8 a 25 tiempos quantum
- Longitud del cable que conforma el bus CAN: 4m

[111] Fuente: Understanding Microchip's CAN Module Bit Timing - Pat Richards Microchip Technology
[112] Fuente: The Configuration of the CAN Bit Timing - Robert Bosch GmbH, Abt. K8/EIS 6th International CAN Conference 2nd to 4th November, Turin (Italy)

De acuerdo a la especificación del protocolo[113], la longitud del tiempo quantum está definida por:

$$TQ = \frac{2 \times BRP}{f_{osc}}$$

Dado un valor prescaler[114] de velocidad (BRP) igual a 1TQ,

$$TQ = \frac{2}{8MHz} = 250ns$$

De acuerdo a la velocidad propuesta,

$$t_{bit} = \frac{1}{Bit\ rate}$$

$$t_{bit} = \frac{1}{500Kbps} = 2\mu s$$

Los números de tiempos quantum que conforman el tiempo de bit vienen dados por la expresión,

$$num\ TQ = \frac{t_{bit}}{TQ}$$

$$num\ TQ = \frac{2\mu s}{250ns} = 8TQ$$

Se define el parámetro segmento de propagación (Prop_Seg), éste depende de la longitud del cable utilizado y del retardo debido al procesamiento de la información por parte del microcontrolador CAN. De acuerdo a Microchip el retardo introducido por el hardware es aproximadamente 100ns, el retardo introducido por un cable UTP categoría 5e es aproximadamente 5,5ns/m[115].

[113] Fuente: Florian Hartwich, Armin Bassemir, Robert Bosch GmbH, 6th International CAN Conference 2nd to 4th November, Turin (Italy); “The Configuration of the CAN Bit Timing”.
[114] Prescaler.- Divisor de frecuencia configurable o programable en un microcontrolador
[115] Retardo máximo de propagación
en frecuencia de 1 MHz 5,7ns/m
en frecuencia de 10 MHz 5,4ns/m
en frecuencia de 100 MHz 5,3ns/m
Fuente: “Components of structured cabling sytems” - HyperlineSystems

El retardo de propagación se define por,

$$t_{prop} = 2(t_{bus} + t_{micro})$$

$$t_{prop} = 2\left(4m \times \frac{5.5ns}{m} + 100ns\right) = 244ns$$

El retardo de propagación es convertido a tiempos quantum,

$$Prop\_Seg = \frac{t_{prop}}{TQ} = \frac{244ns}{250ns} = 0.952 \approx 1TQ$$

El segmento de sincronización es un valor fijo igual a 1.

Dado que el tiempo total del bit en quantum es igual a 8TQ, el segmento de sincronización es 1TQ y el segmento de propagación es 1TQ, las longitudes de los segmentos de fase 1 y 2 vienen dadas por,

$$Phase\_Seg1 + Phase\_Seg2 = 8TQ - 1TQ - 1TQ = 6TQ$$

El punto de muestreo es el instante de tiempo en el que se lee e interpreta el valor del bit en el bus y se encuentra ubicado al final del segmento de fase 1.

Fabricantes reconocidos como CAN y CANOpen, han fijado el punto de muestreo de la señal en 87.5%[116], de acuerdo a este criterio el segmento de fase 2 (Phase_Seg2) sería,

$$Phase\_Seg2 = 8TQ \times (1 - 0.875) = 1TQ$$

Despejando el valor del segmento de fase 1 (Phase_Seg1) se obtiene,

$$Phase\_Seg1 = 5TQ$$

[116] Fuente: Request form for CANopen Bit Timings –port GmbH - Heinz-Jürgen Oertel http://www.port.de/engl/canprod/sv_req_form.html

De acuerdo a la tabla 2.6, la longitud del salto de sincronización (SJW) no debe exceder el valor del segmento de fase 2 por tanto toma el valor de 1TQ.
Los valores de los segmentos de un tiempo de bit CAN para obtener una velocidad de 500Kbps a una frecuencia de oscilación de 8 MHz sobre un bus de 4 m. se resume en la siguiente tabla:

| Parámetro | Valor |
|---|---|
| BRP | 1 |
| Sync_Seg | 1 TQ |
| Prop_Seg | 1 TQ |
| Phase_Seg1 | 5 TQ |
| Phase_Seg2 | 1 TQ |
| SJW | 1 TQ |

**Tabla 2.18:** Parámetros calculados para una velocidad de 500Kbps y un oscilador de 8MHz.

Estos valores calculados se encuentran dentro del rango permitido por la especificación BOSH y fueron los utilizados en la elaboración de los programas en cada nodo.

La tabla a continuación muestra los parámetros necesarios para diferentes velocidades utilizando un oscilador de 8MHz:

| Bit Rate | Pre-scaler | Number of time quanta | Seg1 (Prop_Seg+Phase_Seg1) | Seg2 | Sample Point |
|---|---|---|---|---|---|
| 500 | 1 | 8 | 6 | 1 | 87.5 |
| 250 | 1 | 16 | 13 | 2 | 87.5 |
| 250 | 2 | 8 | 6 | 1 | 87.5 |
| 125 | 2 | 16 | 13 | 2 | 87.5 |
| 125 | 4 | 8 | 6 | 1 | 87.5 |
| 100 | 2 | 20 | 16 | 3 | 85.0 |
| 100 | 4 | 10 | 8 | 1 | 90.0 |

**Tabla 2.19:** Parámetros de inicialización CAN para un oscilador de 8MHz.[117]

[117] Fuente: Request form for CANopen Bit Timings – Fuente: port GmbH - Heinz-Jürgen Oertel http://www.port.de/engl/canprod/sv_req_form.html

### 2.3.3 SOFTWARE PARA EL NODO 1 (NODO MAESTRO)

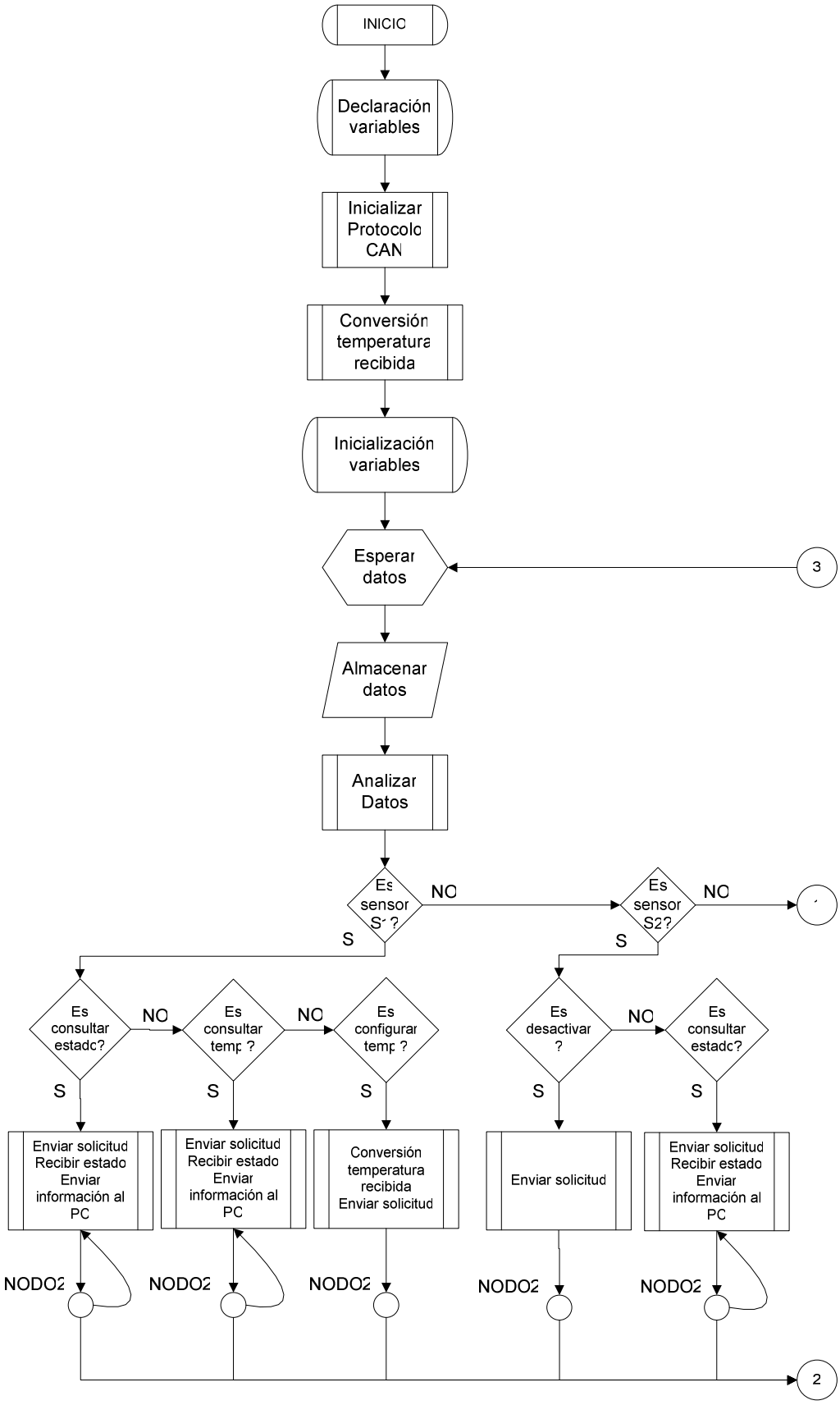
De acuerdo a la funcionalidad del nodo 1, el programa debe almacenar los datos trasmitidos por el computador personal y enviar los mensajes a los nodos.
Las funciones realizadas en el nodo maestro fueron:

- Definir e inicializar variables.
- Configurar pines de entrada y salida
- Establecer modo de configuración para el módulo CAN.
- Configurar variables de inicialización del módulo CAN (SJW, BRP, PHSEG1, PHSEG2, PROSEG y banderas).
- Configurar máscaras y filtros.
- Establecer modo normal para el módulo CAN.
- Inicializar módulo USART a 9600 baudios.
- Almacenar datos recibidos en un arreglo.
- Enviar y recibir información de los nodos según sea el caso.
- Encerar variables utilizadas.
- Finalizar el programa y ejecutarlo nuevamente dentro de un lazo.

El diagrama de flujo a continuación, muestra la distribución del programa, en la inicialización de variables se incluye la configuración del registro ADCON1 para utilizar el convertidor analógico/digital, en el nodo maestro, la función “Inicializar” CAN incluye todas las configuraciones necesarias como son definir filtros y máscaras para el funcionamiento del módulo CAN, la función “Conversión” permite convertir la temperatura recibida de formato ASCII a formato decimal, para realizar operaciones posteriores en el nodo 2.

El código restante se encuentra en un lazo con el propósito de que el módulo USART del microcontrolador se encuentre siempre en espera de datos.

Los datos almacenados se analizan y se define que hacer con ellos en las estructuras switch anidadas de acuerdo al nodo y al sensor específicos.


INICIO
Declaración variables
Inicializar Protocolo CAN
Conversión temperatura recibida
Inicialización variables
Esperar datos
3
Almacenar datos
Analizar Datos
Es sensor S1 ?
NO
Es sensor S2?
NO
S
S
Es consultar estado?
NO
Es consultar temp ?
NO
Es configurar temp ?
Es desactivar ?
NO
Es consultar estado?
S
S
S
S
S
Enviar solicitud Recibir estado Enviar información al PC
Enviar solicitud Recibir estado Enviar información al PC
Conversión temperatura recibida Enviar solicitud
Enviar solicitud
Enviar solicitud Recibir estado Enviar información al PC
NODO2
NODO2
NODO2
NODO2
NODO2
2

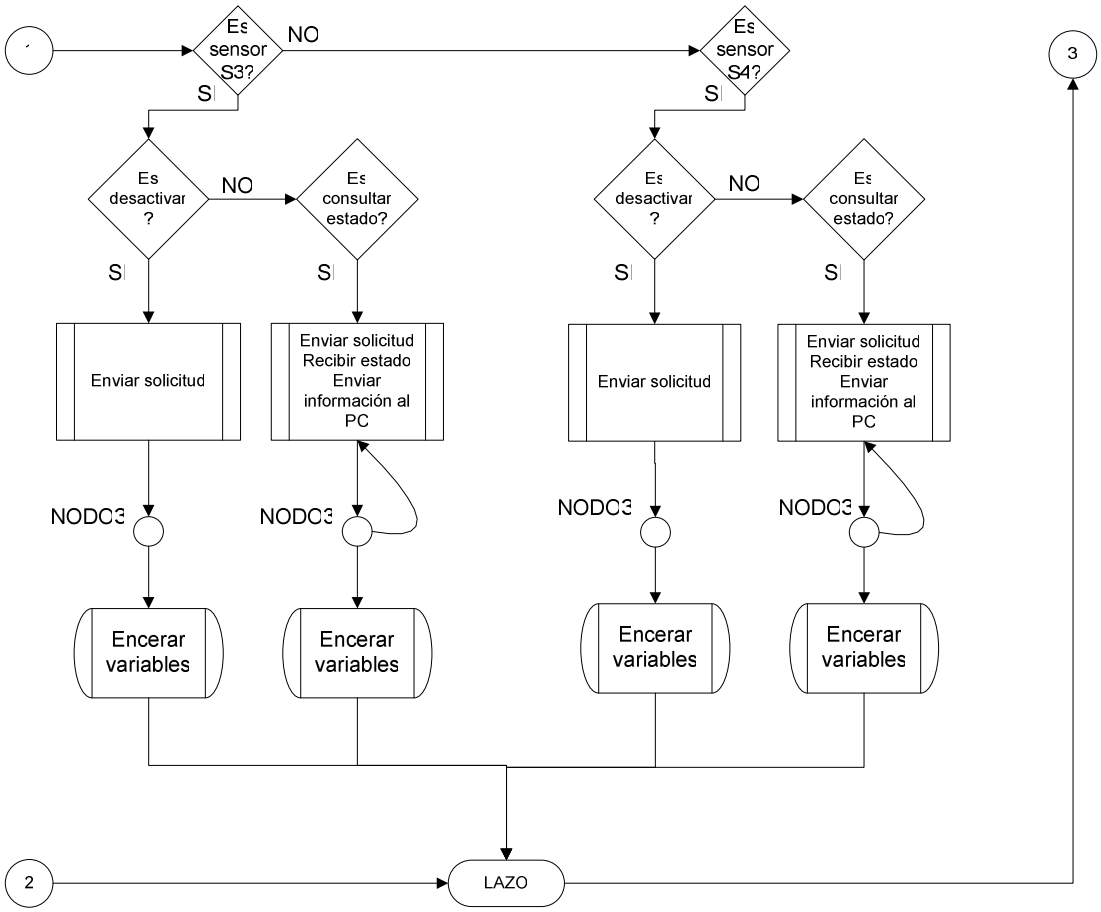



**Figura 2.20:** Diagrama de flujo del Nodo 1

Dependiendo del mensaje de control, los nodos esclavos responden al maestro, desactivan el buzzer o configuran algún parámetro.

Al finalizar se enceran las variables y se continúa en el lazo en espera de más datos provenientes del computador personal.

El protocolo CAN trasmite mensajes de longitud máxima de 8 bytes, en el prototipo se envía mensajes de longitud máxima de 5 bytes con información de configuración de temperatura máxima, temperatura mínima, desactivación del sensor, consulta de la temperatura actual y del estado del sensor.

A continuación se muestran los valores de todos los mensajes posibles que pueden ser enviados desde el computador personal al nodo maestro:

| Destino | Sensor | Mensaje | Observación |
|---|---|---|---|
| **Nodo Esclavo 1** | **Temperatura** | 1a000 | Consultar estado actual del sensor de temperatura |
| | | 1t000 | Consultar temperatura actual |
| | | 1xxxx | Configurar temperatura máxima y mínima |
| | **Movimiento** | 2a000 | Consultar estado actual del sensor de movimiento |
| | | 2d000 | Desactivar sensor de movimiento |
| **Nodo Esclavo 2** | **Humo** | 3a000 | Consultar estado actual del sensor de humo |
| | | 3d000 | Desactivar sensor de humo |
| | **Magnético** | 4a000 | Consultar estado actual del sensor magnético |
| | | 4d000 | Desactivar sensor magnético |

**Tabla 2.20:** Flujo de datos computador personal – nodo maestro

A continuación se muestran los posibles mensajes de control del nodo maestro a los nodos esclavos:

| Destino | Sensor | Mensaje | Observación |
|---|---|---|---|
| **Nodo Esclavo 1** | **Temperatura** | 0x92 0xXX | Configurar temperatura máxima a XX valor |
| | | 0x93 0xXX | Configurar temperatura mínima a XX valor |
| | | 0x94 0x00 | Consultar estado actual del sensor |
| | | 0x95 0x00 | Consultar temperatura actual |
| | **Movimiento** | 0x20 0x00 | Desactivar sensor de movimiento |
| | | 0xa2 0x00 | Consultar estado actual del sensor |
| **Nodo Esclavo 2** | **Humo** | 0x30 0x00 | Desactivar sensor de humo |
| | | 0xb2 0x00 | Devolver estado actual del sensor |
| | **Magnético** | 0x40 0x00 | Desactivar sensor magnético |
| | | 0xc2 0x00 | Devolver estado actual del sensor |

**Tabla 2.21:** Flujo de datos nodo maestro – nodos esclavos

### 2.3.4 SOFTWARE PARA EL NODO 2 (NODO ESCLAVO 1)

De acuerdo a las funciones del nodo 2, el programa debe ser capaz de manejar el buzzer si el sensor de temperatura o el sensor de movimiento se activan. La desactivación remota del buzzer también es posible de modo que el nodo 2 debe recibir mensajes de desactivación, consulta de estado actual de los sensores y configuración de valores límites para el sensor de temperatura, por defecto se han fijado los valores límites en temperatura máxima a 40ºC y temperatura mínima a

5ºC, si la temperatura detectada se encuentra fuera de este rango, el buzzer empezará a sonar. El software en el microcontrolador del nodo 2 debe realizar las siguientes funciones:

- Definir e inicializar variables.
- Establecer modo de configuración para el módulo CAN.
- Configurar variables de inicialización del módulo CAN
- Configurar máscaras y filtros.
- Establecer modo normal para el módulo CAN.
- Configurar pines de entrada y salida
- Habilitar y configurar el convertidor análogo/digital.
- Habilitar y configurar interrupción INT0.
- Actuar de acuerdo al mensaje de control enviado por el nodo maestro.
- Enviar información al nodo maestro según lo solicite.
- Encerar variables utilizadas.
- Finalizar el programa y ejecutarlo nuevamente dentro de un lazo.

A continuación se muestra el diagrama de flujo del software contenido en el nodo 2, se declaran las variables globales que se utilizarán en el programa, para el manejo del sensor de movimiento se habilita la interrupción INT0, en la función “Cálculo de temperatura” se toman 20 mediciones en el canal 0 del convertidor A/D, se promedia y se procesa la información para obtener el valor de temperatura actual, en la función “Estado actual de temperatura” se compara el valor de temperatura actual con la temperatura máxima y mínima configuradas por el usuario o la que se encuentra por defecto.

El código restante se encuentra en un lazo con el propósito de que se repita el programa de manera indefinida.

Se han incluido banderas para controlar a la sirena de modo que si se activa cualquier sensor, el buzzer sonará inmediatamente.

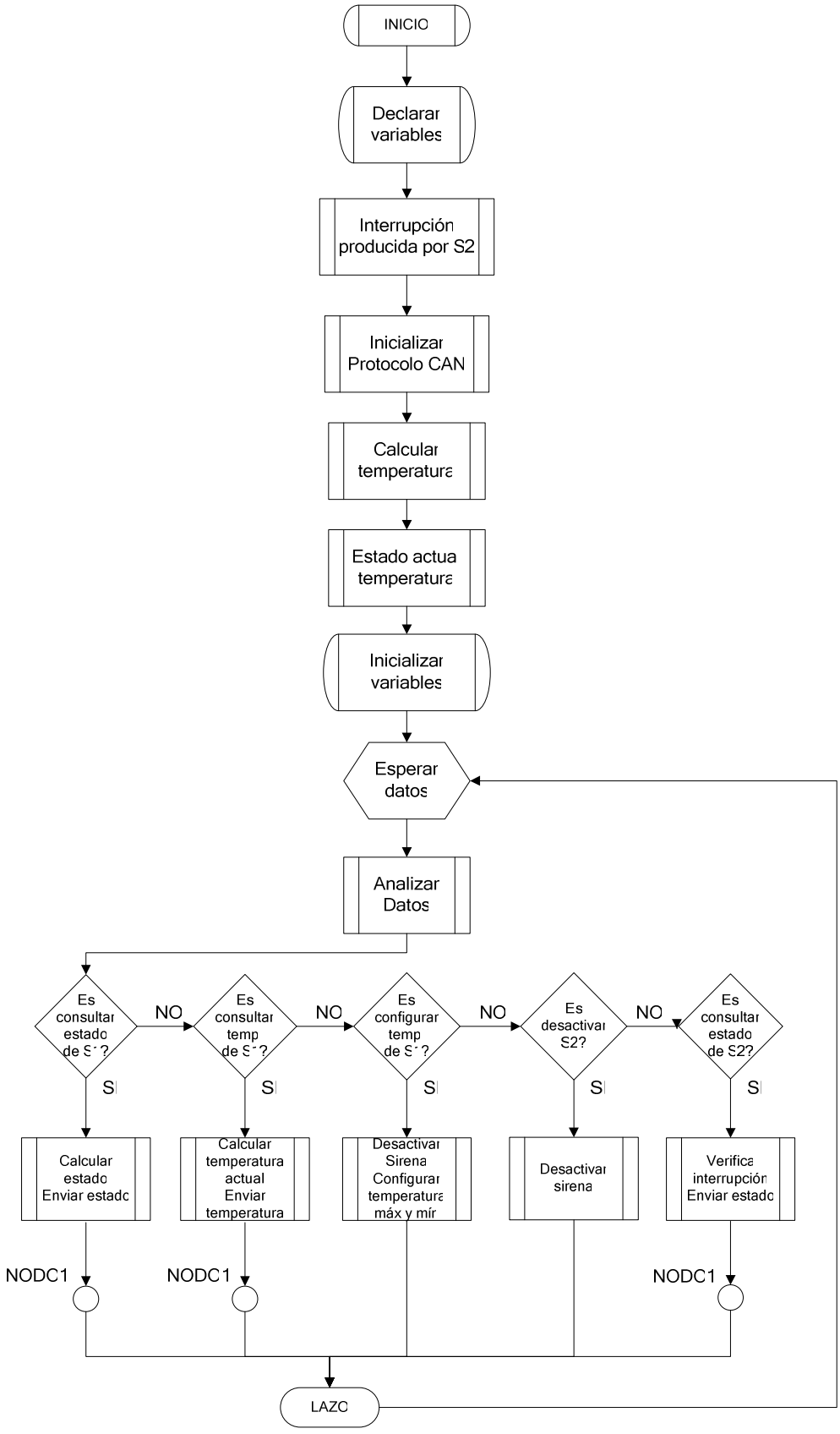



**Figura 2.21:** Diagrama de flujo del Nodo 2

A continuación se detallan los posibles mensajes de respuesta del nodo esclavo 1 al nodo maestro:

| Destino | Sensor | Mensaje | Observación |
|---|---|---|---|
| **Nodo Maestro** | **Temperatura** | 0x30 | Sensor de temperatura desactivado |
| | | 0x31 | Sensor de temperatura activado |
| | | 0xXX | Temperatura actual |
| | **Movimiento** | 0x30 | Sensor de movimiento desactivado |
| | | 0x32 | Sensor de movimiento activado |

**Tabla 2.22:** Flujo de datos nodo esclavo 1 – nodo maestro

### 2.3.5 SOFTWARE PARA EL NODO 3 (NODO ESCLAVO 2)

Similar al nodo anterior, el nodo 3 debe ser capaz de manejar el buzzer si el sensor de humo o el sensor magnético se activan. La desactivación remota del buzzer también es posible, el nodo 3 puede recibir mensajes de control como lo son: consultar estado actual de los sensores y desactivación remota.

Las funciones realizadas en el nodo 3 son:

- Definir e inicializar variables.
- Establecer modo de configuración para el módulo CAN.
- Configurar variables de inicialización del módulo CAN
- Configurar máscaras y filtros.
- Establecer modo normal para el módulo CAN.
- Configurar pines de entrada y salida
- Habilitar y configurar interrupciones INT0 para el sensor magnético e INT1 para el sensor de humo.
- Actuar de acuerdo al mensaje de control enviado por nodo maestro.
- Enviar información al nodo maestro según lo solicite.
- Encerar variables utilizadas.
- Finalizar el programa y ejecutarlo a manera de lazo.

A continuación se muestra el diagrama de flujo del software contenido en el nodo 3.

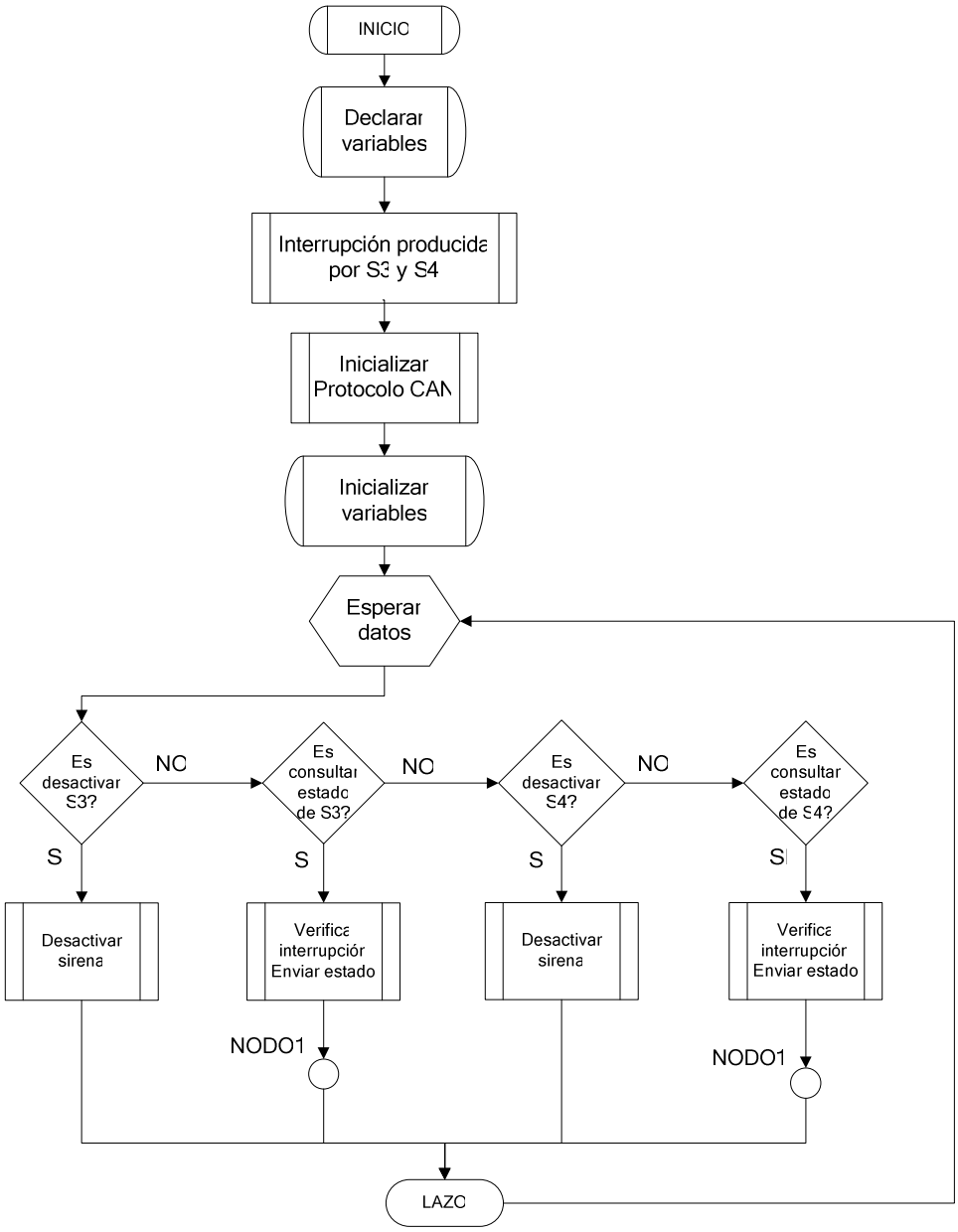



**Figura 2.22:** Diagrama de flujo del Nodo 3

Posibles mensajes de respuesta del nodo esclavo 2 al nodo maestro:

| Destino | Sensor | Mensaje | Observación |
|---|---|---|---|
| **Nodo Maestro** | **Humo** | 0x30 | Sensor de humo desactivado |
| | | 0x33 | Sensor de humo activado |
| | **Magnético** | 0x30 | Sensor magnético desactivado |
| | | 0x34 | Sensor magnético activado |

**Tabla 2.23:** Flujo de datos nodo esclavo 2 – nodo maestro

### 2.3.6 SOFTWARE PARA LA ESTACIÓN DE MONITOREO

La interface diseñada para la estación de monitoreo de los nodos, se realizó utilizando el lenguaje de programación C Sharp (C#) de Microsoft Visual Studio 2005.

Microsoft Visual Studio 2005 integra el nuevo espacio de nombres System.IO.Ports que contiene clases para controlar los puertos seriales del computador. La clase más importante que ofrece es la clase SerialPort, ésta permite el acceso a las propiedades del puerto serie y la comunicación a través del mismo. Al crear un objeto puerto serial, el programador tiene la capacidad de controlar todos los aspectos de la comunicación. La clase SerialPort posee algunas funciones y propiedades para configuración, inicialización y temporización del puerto serial.

Las propiedades y métodos que se utilizaron en el diseño fueron:
*Propiedades:*

- **IsOpen:** Obtiene un valor que indica el estado abierto o cerrado del objeto SerialPort.
- **BaudRate:** Obtiene o establece la velocidad en baudios del puerto serie.
- **DataBits:** Obtiene o establece la longitud estándar de los bits de datos por byte.
- **Parity:** Obtiene o establece el protocolo de comprobación de la paridad.
- **PortName:** Obtiene o establece el puerto de comunicaciones, incluidos por lo menos todos los puertos COM disponibles.
- **StopBits:** Obtiene o establece el número estándar de bits de parada por byte.

Se configuró el puerto serial del computador personal y del microcontrolador con los siguientes valores:
*Nombre del puerto* = Variable (COM1, COM2, COM3, COM4, COM5)
*Velocidad* = 9600 baudios
*Paridad* = Ninguna
*Bits de datos* = 8
*Bits de parada* = Ninguno

*Métodos:*

- **Open():** Abre una conexión con el puerto serial.
- **Close():** Cierra la conexión del puerto y establece el valor de la propiedad IsOpen en false.
- **ReadExisting():** Lee todos los bytes inmediatamente disponibles, basándose en la codificación, en la secuencia y el buffer de entrada del objeto SerialPort.
- **Write(***string***):** Escribe datos en el buffer de salida del puerto serie.

La interface realiza un monitoreo de los nodos basándose en consultas frecuentes de sus estados, para ello se incluyeron temporizadores. La interface consulta cada tres segundos al nodo maestro el estado actual de los nodos y presenta en la pantalla los resultados.

El sistema de monitoreo permite el acceso a dos tipos de usuarios, un *Invitado* y un *Administrador*.


**Figura 2.23:** Interface de registro a la aplicación del prototipo de control de nodos

El usuario Invitado puede realizar un monitoreo básico del sistema, puede observar la temperatura actual, los estados de los nodos y la activación de las alarmas pero no podrá desactivarlas.

A continuación se muestra el diseño de la interface de monitoreo básico, las alarmas generadas se mostrarán en el capítulo de implementación y pruebas.

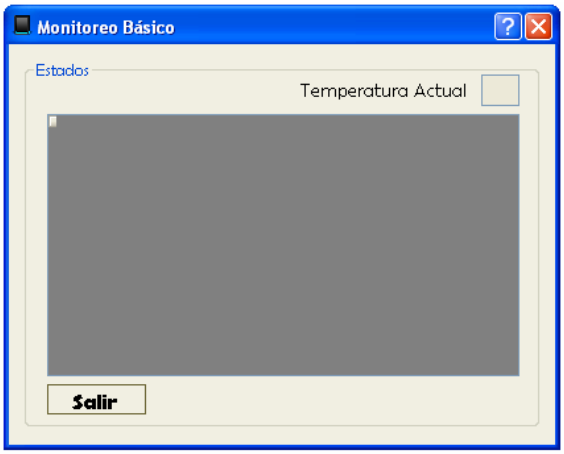



**Figura 2.24:** Diseño de la interface de monitoreo básico

El usuario Administrador puede seleccionar el puerto COM a través del cual se conectará el prototipo al computador, puede visualizar los estados de los sensores, además tiene la opción de desactivar las alarmas y configurar el sensor de temperatura.

A continuación se muestra el diseño de la interface de monitoreo avanzado, al igual que la interface de monitoreo básico, las alarmas generadas se mostrarán en el capítulo de implementación y pruebas.

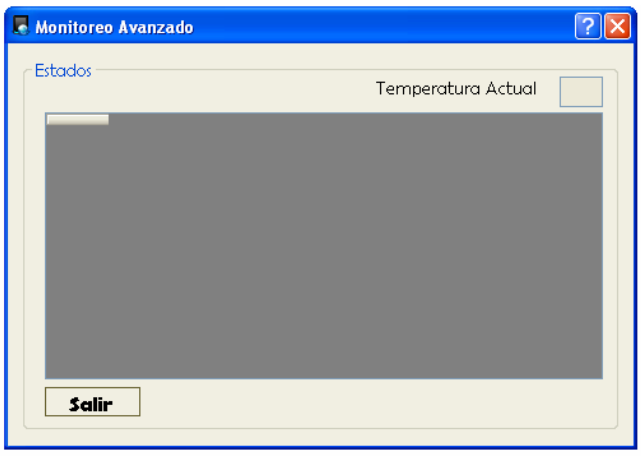



**Figura 2.25:** Diseño de la interface de monitoreo avanzado

Por defecto la temperatura permitida por el sistema se encuentra entre 5ºC y 40ºC, fuera de este rango, se activará la alarma. La manera de desactivar el sensor de temperatura es cambiar el rango de temperatura permitido.


**Figura 2.26:** Interface de configuración del sensor de temperatura

# REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 2

## Direcciones Electrónicas

[1] x-robotics Robotica & µControladores Pic ; “Sensores”, http://www.x-robotics.com/sensores.htm

[2] Mikroelektronika; MikroC - C Compiler for Microchip Microcontrollers User’s Manual,
http://www.mikroe.com/en/compilers/mikroc/pic/download.htm#documentation

[3] Florian Hartwich, Armin Bassemir, Robert Bosch GmbH, 6th International CAN Conference 2nd to 4th November, Turin (Italy); “The Configuration of the CAN Bit Timing”, http://www.semiconductors.bosch.de/pdf/CiA99Paper.pdf

[4] Pat Richards, Microchip Technology Inc. Application Note AN754, 2001; “Understanding Microchip’s CAN Module Bit Timing”,
http://ww1.microchip.com/downloads/en/AppNotes/00754.pdf

[5] PORT CAN and CANopen Products ; “Request form for CANopen Bit Timings”, http://www.port.de/engl/canprod/sv_req_form.html

[6]CAN in AUTOMATION International Users and Manufacturers Group e. V. ; “CANopen Communication Profile Features”,
http://www.nikhef.nl/pub/departments/ct/po/doc/CANopenCiA.pdf

[7] Nilesh Rajbharti, Microchip Technology, Inc. Application Note AN738, 2001; “PIC18C CAN Routines in ‘C’”,
http://ww1.microchip.com/downloads/en/AppNotes/00738b.pdf

[8] Librería MSDN, 2008; “Biblioteca de clases de .Net Framework SerialPort (Miembros)”,

http://msdn2.microsoft.com/es-es/library/system.io.ports.serialport_members(VS.80).aspx

**Datasheets**

[9] Microchip Technology Inc.

- PIC18FXX8 Data Sheet, 2004
- PIC18F2480/2580/4480/4580 Data Sheet, 2004
- MCP2551 High-Speed CAN Transceiver, 2007

[10] National Semiconductor

- LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers
- LM35/LM35A/LM35C/LM35CA/LM35D Precision Centigrade Temperature Sensors

[11] Maxim

- MAX220/222/232A/233A/242/243 +5V-Powered, Multichannel RS-232 Drivers/Receivers

[12] Freescale Semiconductor

- MC145010 Photoelectric Smoke Detector IC with I/O

# CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

## 3.1 IMPLEMENTACIÓN

La implementación consistió en el desarrollo de una interface gráfica encargada de monitorear los estados de los sensores, la construcción de un circuito impreso por cada nodo y el armado de los circuitos correspondientes.

Los circuitos impresos tienen un tamaño de 7cm x 7cm, además se construyó una estructura de madera para la presentación del prototipo.


**Figura 3.1:** Implementación del prototipo

### 3.1.1 IMPLEMENTACIÓN DE CAPA FÍSICA

#### 3.1.1.1 Medio de trasmisión

El protocolo CAN especifica dos medios físicos de transmisión posibles:

- Cable de par trenzado
- Fibra óptica

En el prototipo se utilizaron dos cables de par trenzado UTP cat5e de una longitud de 2m cada uno para la conexión entre los nodos, para la comunicación entre el

nodo maestro y el computador personal también se utilizó el mismo tipo de cable con la misma longitud.

#### 3.1.1.2 Tipo de conector

La especificación no detalla el tipo de conector, en el prototipo se utilizaron conectores tipo DB9 hembra en las terminaciones del cable y DB9 macho en los circuitos impresos. La razón por la que se optó por este tipo de conector es la gran aceptación a nivel industrial, gran parte de módulos CAN de diferentes fabricantes emplean conectores DB9.

**Figura 3.2:** Bus CAN y bus serial para conexión con la PC

#### 3.1.1.3 Alcance y número de nodos

El número de nodos presentes en el prototipo son 3, el alcance total del bus es 4m.


**Figura 3.3:** Implementación del Nodo Maestro


**Figura 3.4** Implementación de los Nodos Esclavo1 y Esclavo2

### 3.1.2 MONTAJE DEL PROTOTIPO

El prototipo se encuentra ensamblado en una estructura de madera, cada circuito impreso se apoya sobre tubos de aluminio y para efectos de protección se cubren con acrílicos transparentes de 3mm de espesor.

Los sensores de movimiento y magnético se encuentran empotrados en la pared de la estructura y el sensor de humo se encuentra fijado al techo.

**Figura 3.5:** Montaje del prototipo

Se han incluido en el prototipo dos switches, uno para controlar el encendido/apagado del prototipo completo ubicado en la parte izquierda junto al

sensor de movimiento, y otro para controlar el encendido/apagado del sensor de humo ubicado en la parte derecha junto al sensor magnético.

## 3.2 PRUEBAS

### 3.2.1 VISUALIZACIÓN DE FORMAS DE ONDA DE LAS SEÑALES SOBRE EL BUS CAN

Las señales diferenciales trasmitidas a través del bus, añaden la característica de inmunidad al ruido y permiten trasmitir a grandes velocidades sobre cables de par trenzado.

Inicialmente, las dos líneas del bus, CAN_H y CAN_L, permanecen en estado recesivo, es decir se encuentra en modo pasivo alrededor de 2,5V. Los estados dominante y recesivo del bus toman valores de ±1 es decir 3,5V en CAN_H y 1,5V en CAN_L, generando una señal diferencial de 2 voltios aproximadamente.

La primera prueba a la que fue sometido el prototipo consistió en conectar dos canales de un osciloscopio a los terminales correspondientes CAN_H y CAN_L de cualquier nodo, al trasmitir mensajes desde el computador, se puede apreciar la forma de onda de la señal que viaja a través del bus CAN, en la misma unidad de tiempo la señal CAN_H y CAN_L son una especie de reflejo una de la otra. Para saber la información exacta que se está enviando por el bus, se puede hacer uso de decodificadores, pero para el nivel práctico que es de lo que se trata un prototipo lo que interesa distinguir es que las dos señales existan y tengan correspondencia con los valores estudiados en el capítulo 1.

A continuación se muestran varias mediciones del las señales obtenidas en el prototipo.

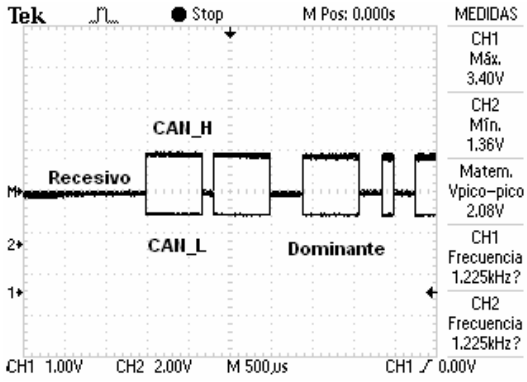



**Figura 3.6:** Medición 1 tomada en el bus CAN

Como se puede observar en la Figura 3.6, el voltaje máximo alcanzado en el bus es de 3,4V, el voltaje mínimo corresponde a 1,36V lo que da como resultado una señal diferencial de 2,08V.

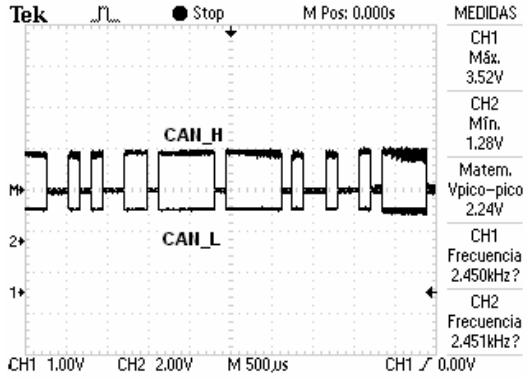



**Figura 3.7:** Medición 2 tomada en el bus CAN

En la Figura 3.7 se muestra otra medición de la señal en el bus CAN, en esta en particular se obtuvo una señal diferencial de 2,24V.

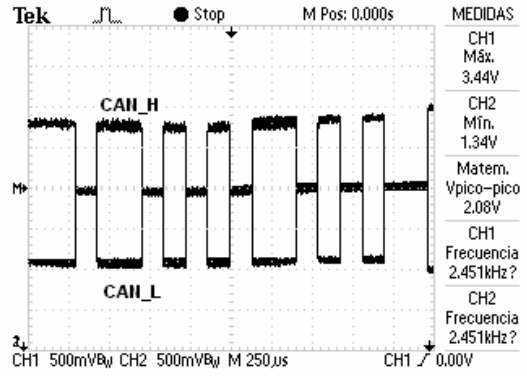



**Figura 3.8:** Medición 3 tomada en el bus CAN

La Figura 3.8 muestra una medición de la señal pero en otra escala, de igual forma se obtuvo una señal diferencial de 2,08V.

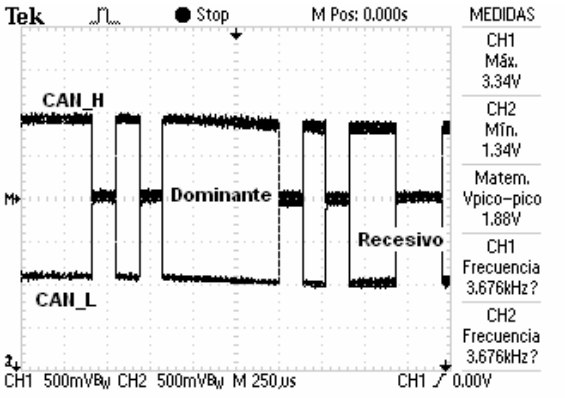



**Figura 3.9:** Medición 4 tomada en el bus CAN

En la Figura 3.9 se obtuvo la señal diferencial de 2V.

Como se pudo observar en todas las mediciones realizadas se obtuvieron valores similares que concuerdan con los valores teóricos estudiados.

### 3.2.2 PRUEBAS DE FUNCIONAMIENTO DEL PROTOTIPO

El funcionamiento del prototipo se basa en la transmisión periódica de señales de control a los nodos y recepción de los estados de los sensores conectados a los mismos.

La comunicación a través del bus CAN es bidireccional, la interface gráfica envía mensajes de control al nodo maestro periódicamente cada tres segundos, en estos mensajes se consulta el estado de cada uno de los sensores, los nodos esclavos reciben este mensajes y responden según sea el caso al nodo maestro, el mismo que comunica esto al computador personal y a través de la interface gráfica, le permite conocer al usuario el estado actual de los sensores.

#### 3.2.2.1 Tiempo de respuesta

El tiempo de respuesta local a los nodos es prácticamente instantánea debido a la conexión directa de los sensores a las líneas de I/O de sus respectivos microcontroladores.

La longitud de cable máxima permitida es de 40m, en el prototipo se utilizaron cables de longitudes de 2m por tanto el retardo introducido por el medio de trasmisión es considerado despreciable.

En la interface gráfica se añadieron retardos en el envío de mensajes al nodo maestro con el propósito de permitirle procesar la información. De acuerdo a las pruebas realizas los retardos adecuados entre mensajes son 200ms.

Incluidos todos los retardos de transmisión, el tiempo que toma en mostrarse en pantalla el estado de un sensor es de tres segundos que corresponde al tiempo que toma la interface en enviar el siguiente mensaje de actualización de estado.

Cabe notar que el buzzer empieza a sonar inmediatamente después de que un sensor se activó. El tiempo de desactivación del buzzer es inmediato, aunque en la pantalla el usuario no nota el cambio sino en algunos cientos de milisegundos después, hasta que la pantalla se refresque.

#### 3.2.2.2 Software de monitoreo

Para proporcionar un entorno de red, se han creado en la interface gráfica dos usuarios, un usuario *Invitado* y un *Administrador*, cada uno con su respectivo nombre de usuario y contraseña.


**Figura 3.10:** Pantalla de presentación del software de monitoreo del prototipo

#### *3.2.2.2.1 Pruebas del prototipo en modo Usuario*

Como se menciona en el capítulo 2, el usuario invitado puede únicamente visualizar los cambios de estados de los sensores, es decir, si un sensor fue activado, un estado de alarma se muestra en la pantalla.

A continuación se muestra algunas pruebas realizadas en modo Usuario:
La primera pantalla en mostrarse es la pantalla de Registro, en esta pantalla se debe seleccionar el usuario *Invitado* con la contraseña correspondiente *user*.


**Figura 3.11:** Pantalla de registro como usuario Invitado

Si la contraseña es errónea, el usuario Invitado no se valida y aparece un mensaje de advertencia que indica que la contraseña es incorrecta.

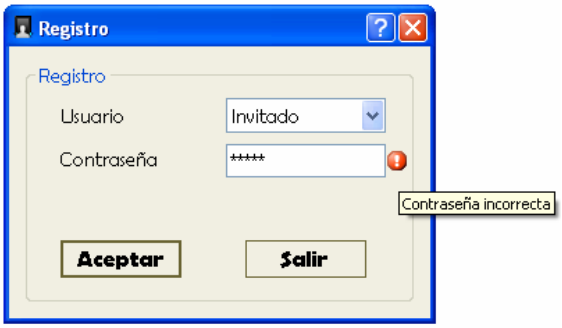



**Figura 3.12:** Pantalla de registro como usuario Invitado con ingreso de contraseña incorrecta

Posterior al registro del usuario la siguiente pantalla en mostrarse es la pantalla de Monitoreo Básico, ésta, muestra información del estado de los nodos y la temperatura actual.

Un sensor se encuentra desactivado cuando no se ha producido ninguna interrupción en el microcontrolador al que se encuentra conectado, las alarmas se

representan con un gráfico con fondo de color blanco en el caso de desactivación y en el caso de activación, se representan con un gráfico superpuesto a una estrella de color rojo.


**Figura 3.13:** Pantalla de Monitoreo Básico sin ningún sensor activado

Cuando un sensor se activa, el gráfico correspondiente cambia como se muestra en las siguientes pantallas:


**Figura 3.14:** Pantallas de Monitoreo Básico con sensores activados


**Figura 3.15:** Pantallas de Monitoreo Básico con sensores activados

### *3.2.2.2.2 Pruebas del prototipo en modo Administrador*

El administrador puede seleccionar el puerto serial (COM) a través del cual el prototipo se va a conectar al computador, además pueden visualizar los estados de los sensores, desactivarlos y configurar el rango de temperatura permitido por el sensor de temperatura, por defecto el rango tolerado fue configurado en 40ºC para la temperatura máxima y 5ºC para la temperatura mínima.

En la pantalla de registro se debe seleccionar el usuario *Administrador* y su contraseña es *admin*. Por defecto se encuentra seleccionado el puerto COM1 para la conexión con el prototipo.


**Figura 3.16:** Pantalla de registro como usuario Administrador

Si se ingresa una contraseña errónea, la pantalla de registro muestra un mensaje de advertencia que indica que la contraseña es incorrecta:

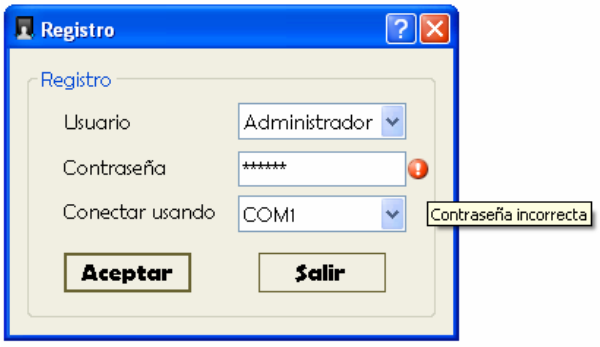



**Figura 3.17:** Pantalla de registro como usuario Administrador con ingreso de contraseña incorrecta

El usuario Administrador tiene acceso a la pantalla de Monitoreo Avanzado en donde tiene los privilegios de configuración del sensor de temperatura y la desactivación remota de los sensores.


**Figura 3.18:** Pantalla de Monitoreo Avanzado con sensores desactivados


**Figura 3.19:** Pantalla de Monitoreo Avanzado con sensores activados

La temperatura ambiente detectada por el sensor en un momento determinado es de 11ºC. Como se muestra en la Figura 3.20, el sensor de temperatura es configurado a 10ºC como temperatura máxima y 5ºC como temperatura mínima, ésta configuración produce que el buzzer suene de manera inmediata.


**Figura 3.20:** Pantalla de configuración de Sensor de Temperatura


**Figura 3.21:** Pantalla de Monitoreo Avanzado con sensor de temperatura activado

La desactivación del sensor se realiza cambiando los valores permitidos de temperatura en la misma pantalla de configuración del sensor, o reiniciando el microcontrolador del nodo 2 para que se tomen los valores de temperatura por defecto.


**Figura 3.22:** Pantalla de Monitoreo Avanzado con sensores activados

### 3.2.3 LIMITACIONES DEL PROTOTIPO

Debido a que el prototipo se desarrolló en un ambiente de pruebas sin mayores variaciones, sin interrupciones eléctricas, el ambiente permaneció a una temperatura considerada constante, sin cambios bruscos, en general no se estableció ruido cercano al bus por lo que se puede suponer que el prototipo adoptó un comportamiento adecuado.

#### 3.2.3.1 Ausencia de suministro eléctrico

Para el funcionamiento del sistema, se hace uso de una fuente de 5V y una fuente de 12V, en caso de no tener a disposición estas fuentes no funcionará el sistema. Para corregir esto se puede añadir baterías de 5 V y 12 V a manera de respaldo de la fuente que se está utilizando normalmente.

#### 3.2.3.2 Sensibilidad del sensor temperatura

El sensor de temperatura debe tomar 20 mediciones, procesarlas y enviarlas al nodo maestro, estas operaciones pueden tener diferencias de ±1 grado en la temperatura debido a aproximaciones erróneas o inexactitudes en la toma de medidas en el microcontrolador.

#### 3.2.3.3 Sensibilidad del sensor de movimiento

El sensor de movimiento fue configurado a la mayor sensibilidad, es decir, el conteo de un único pulso activa el buzzer conectado al microcontrolador.

#### 3.2.3.4 Sensibilidad de la interface de administración

La interface de administración de sensores envía mensajes de control al nodo maestro en intervalos de tres segundos, en ciertos casos coincide que el usuario desactiva algún sensor en el mismo instante en que la interface envía las peticiones, esto ocasiona que el sensor no se desactive ese instante. Para corregir esto, el usuario debe asegurarse de que el buzzer deje de sonar inmediatamente después de dar clic en el botón de desactivación, de no ser así debe dar un clic adicional.

#### 3.2.3.5 Ausencia de conexión del prototipo al computador

En el caso de no poseer conexión entre el computador y el nodo maestro, las pantallas de monitoreo básico y monitoreo avanzado no muestran ningún tipo de información y se despliega en pantalla un mensaje de error informando al usuario la ausencia de conexión. Esto se corrige conectado el prototipo al computador.

#### 3.2.3.6 Conexión del prototipo a un puerto COM inexistente

El usuario Administrador tiene la opción de seleccionar el puerto con el que se va a conectar el prototipo con la PC, en caso de seleccionar un puerto que se encuentre utilizado por otra aplicación, se despliega un mensaje de error, de igual manera si se escoge un puerto COM que no posea el computador. Para corregir esto se puede verificar la existencia y disponibilidad de los puertos.

# CAPÍTULO 4: COSTO DEL PROTOTIPO

## 4.1 CÁLCULO DEL COSTO DEL PROTOTIPO IMPLEMENTADO

Al costo total del prototipo se lo ha desglosado en los costos parciales de cada uno de los nodos, además se incluye el costo del diseño, el mismo que se fijó en $5 / hora de trabajo.

| Referencia | Cantidad | Detalle | V. Unitario | V. Total |
|---|---|---|---|---|
| NODO 1 (Maestro) | 1 | Microcontrolador PIC18F2480 | 16,5 | 16,5 |
| | 1 | Transceiver CAN MCP2551 | 3,25 | 3,25 |
| | 1 | MAX 232 | 2,5 | 2,5 |
| | 1 | Zócalo 28 pines | 0,22 | 0,22 |
| | 1 | Zócalo 8 pines | 0,09 | 0,09 |
| | 1 | Zócalo 16 pines | 0,15 | 0,15 |
| | 4 | Capacitores 10 µF | 0,08 | 0,32 |
| | 3 | Resistencia 10 KΩ | 0,02 | 0,06 |
| | 1 | Resistencia 10 Ω | 0,02 | 0,02 |
| | 1 | Resistencia 120 Ω | 0,02 | 0,02 |
| | 1 | Pulsante (reset) | 0,15 | 0,15 |
| | 2 | Conector DB9 (macho) | 0,75 | 1,5 |
| | 1 | Cristal 8 Mhz | 0,6 | 0,6 |
| | 2 | Capacitor 22 pF | 0,08 | 0,16 |
| | 1 | Borne 2p | 0,35 | 0,35 |
| | 1 | Circuito impreso (7x7) | 22 | 22 |
| | 1 | Cable serial 2m. | 2 | 2 |
| | | | **Subtotal 1** | **49,89** |
| NODO 2 (Esclavo 1) | 1 | Microcontrolador PIC18F2480 | 16,5 | 16,5 |
| | 1 | Transceiver CAN MCP2551 | 3,25 | 3,25 |
| | 1 | LM35DZ | 3,25 | 3,25 |
| | 1 | Sensor movimiento | 14,5 | 14,5 |
| | 1 | Transistor 2N3904 | 0,08 | 0,08 |
| | 1 | Zócalo 28 pines | 0,22 | 0,22 |
| | 1 | Zócalo 8 pines | 0,09 | 0,09 |
| | 3 | Resistencia 10 KΩ | 0,02 | 0,06 |

| | | | | |
|---|---|---|---|---|
| | 1 | Resistencia 10 Ω | 0,02 | 0,02 |
| | 1 | Resistencia 4,7 KΩ | 0,02 | 0,02 |
| | 1 | Resistencia 2,2 KΩ | 0,02 | 0,02 |
| | 1 | Resistencia 100 Ω | 0,02 | 0,02 |
| | 1 | Pulsante (reset) | 0,15 | 0,15 |
| | 2 | Conector DB9 (macho) | 0,75 | 1,5 |
| | 1 | Cristal 8 Mhz | 0,6 | 0,6 |
| | 2 | Capacitor 22 pF | 0,08 | 0,16 |
| | 1 | Buzzer | 0,7 | 0,7 |
| | 1 | Switch dos estados | 0,35 | 0,35 |
| | 4 | Borne 2p | 0,35 | 1,4 |
| | 1 | Circuito impreso (7x7) | 22 | 22 |
| | 1 | Cable serial 2m. | 2 | 2 |
| | | | **Subtotal 2** | **66,89** |
| **NODO 3 (Esclavo 2)** | 1 | Microcontrolador PIC18F2480 | 16,5 | 16,5 |
| | 1 | Transceiver CAN MCP2551 | 3,25 | 3,25 |
| | 1 | Sensor magnético | 3,5 | 3,5 |
| | 1 | Sensor de humo | 20,54 | 20,54 |
| | 1 | Amp. Operacional LM358 | 0,35 | 0,35 |
| | 1 | Transistor 2N3904 | 0,08 | 0,08 |
| | 1 | Zócalo 28 pines | 0,22 | 0,22 |
| | 2 | Zócalo 8 pines | 0,09 | 0,18 |
| | 3 | Resistencia 10 KΩ | 0,02 | 0,06 |
| | 1 | Resistencia 10 Ω | 0,02 | 0,02 |
| | 1 | Resistencia 120 Ω | 0,02 | 0,02 |
| | 3 | Resistencia 4,7 KΩ | 0,02 | 0,06 |
| | 1 | Resistencia 2,2 KΩ | 0,02 | 0,02 |
| | 1 | Resistencia 100 Ω | 0,02 | 0,02 |
| | 1 | Resistencia 1 KΩ | 0,02 | 0,02 |
| | 1 | Pulsante (reset) | 0,15 | 0,15 |
| | 1 | Conector DB9 (macho) | 0,75 | 0,75 |
| | 1 | Cristal 8 Mhz | 0,6 | 0,6 |
| | 2 | Capacitor 22 pF | 0,08 | 0,16 |
| | 1 | Buzzer | 0,7 | 0,7 |
| | 1 | Switch dos estados | 0,35 | 0,35 |
| | 1 | Switch bilateral cuadrado | 0,4 | 0,4 |
| | 2 | Borne 2p | 0,35 | 0,7 |
| | 1 | Borne 3p | 0,45 | 0,45 |

| | | | | |
|---|---|---|---|---|
| | 1 | Circuito impreso (7x7) | 22 | 22 |
| | 1 | Cable serial 2m. | 2 | 2 |
| | | | **Subtotal 3** | **73,1** |
| | 150 | Diseño por hora de trabajo | 15 | 2250 |
| | | | **Subtotal 4** | **2250** |
| | 1 | Software de desarrollo MikroC | 249 | 249 |
| | | | **Subtotal 5** | **249** |
| | 1 | Soporte/Montaje del prototipo | 80 | 80 |
| | | | **Subtotal 6** | **80** |
| | 1 | Fuente de tensión 5V/12V | 15 | 15 |
| | | | **Subtotal 7** | **15** |
| | | | **TOTAL** | **2783,88** |

**Tabla 4.1:** Costo del prototipo implementado

A continuación se detalla un gráfico que muestra el porcentaje de costos de cada rubro para la realización del prototipo implementado:

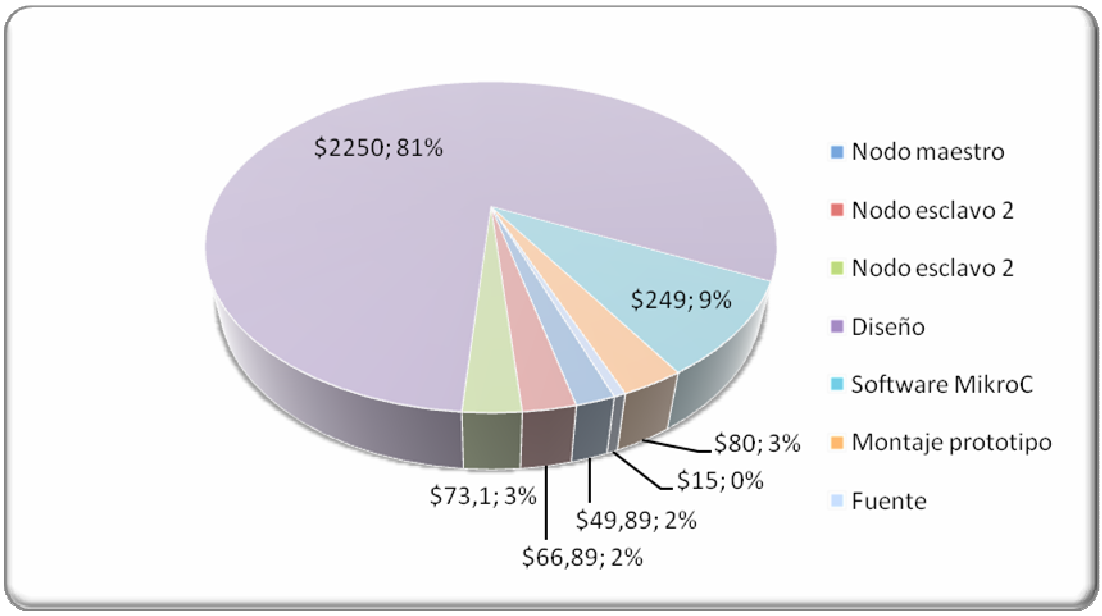



**Figura 4.1:** Porcentaje de costos del prototipo

Como se puede notar, el mayor rubro corresponde al costo del diseño y al software utilizado, por tratarse de un prototipo, la solución puede resultar costosa si se llegara a implementar.

## 4.2 ANÁLISIS DE PRODUCTOS SIMILARES EXISTENTES EN EL MERCADO

Los dispositivos CAN existentes en el mercado se pueden dividir en: herramientas de configuración y análisis, interfaces para PC, adaptadores, repetidores, gateways entre otros, todos estos se encuentran en múltiples variantes de reconocidas marcas como son CAN bus, CANopen y DeviceNet.

### 4.2.1 HERRAMIENTAS DE CONFIGURACIÓN Y ANÁLISIS

Los analizadores CAN facilitan el mantenimiento de un sistema CAN permitiendo el análisis del mismo. Adicionalmente ofrecen opciones de registro de usuarios, interpretación de datos y exhibición de señales recibidas como gráficos lineales.

Se encuentran también a disposición, herramientas de diagnóstico para la red CAN como lo son el CAN-Bus-Tester (CBT) que permite someter a pruebas de capa física al sistema. Este aparato puede ser usado, entre otras cosas, para determinar la relación señal a ruido o la calidad de la señal de los mensajes CAN.

#### 4.2.1.1 Can Analyzer software (CANopen)

Esta herramienta pone a disposición varias funciones de administración. El usuario puede seleccionar el tipo de bus CAN, estándar o extendido y configurar la velocidad de trasmisión de la red.

Con este software, se puede monitorear la carga del bus, además posee algunos LEDs indicadores de estado.

Las tareas que se pueden realizar utilizando este software son:

1. *Network Manager.-* Esta función permite escanear a un nodo específico o a todos para encontrar los nodos disponibles. Con esta función se pueden leer y escribir los registros CAN de los nodos y cambiar los estados (operativo, pre-operativo).

2. *CanOpen Monitor.-* Esta función se encuentra disponible solo para el tipo de formato CAN estándar y permite visualizar todas las tramas CANopen identificando el nodo, su tipo y el dato trasmitido.
3. *Can Monitor.-* Esta función esta disponible para ambos tipos de formatos CAN (estándar y extendido) y permite visualizar las tramas que atraviesan la red identificando el nodo, el dato trasmitido y su equivalente ASCII.
4. *Can Sender.-* Esta función se encuentra disponible para ambos tipos de formatos CAN (estándar y extendido) y permite enviar tramas específicas por el bus.


**Figura 4.2:** Pantalla principal del Can Analyzer[118]

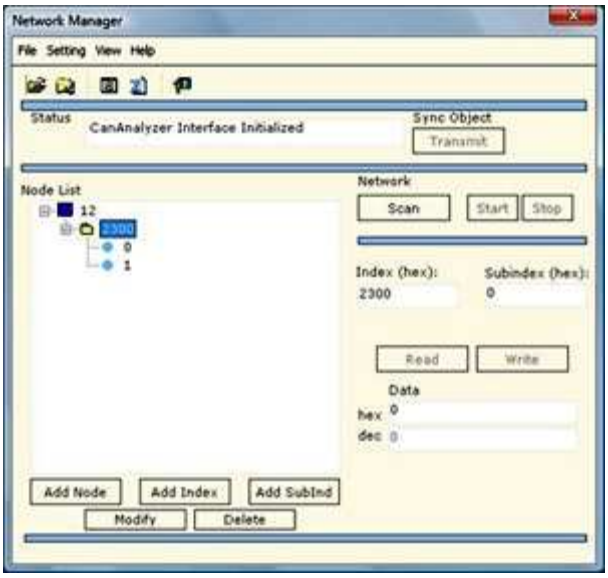



**Figura 4.3:** Pantalla de la opción Network Manager[119]

[118 - 119] Fuente: ADFweb.com /Home/Products/CAN Analizer

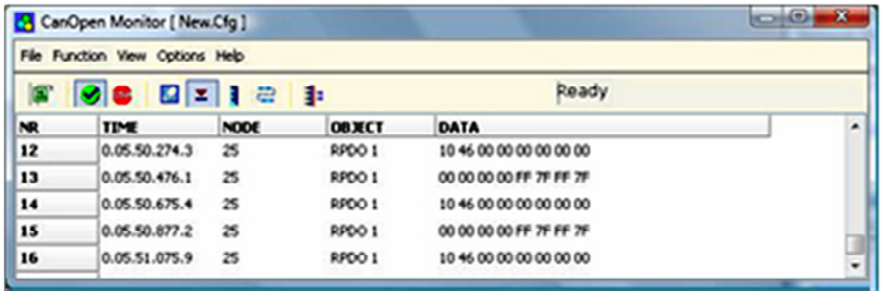



**Figura 4.4:** Pantalla de la opción CanOpen Monitor[120]

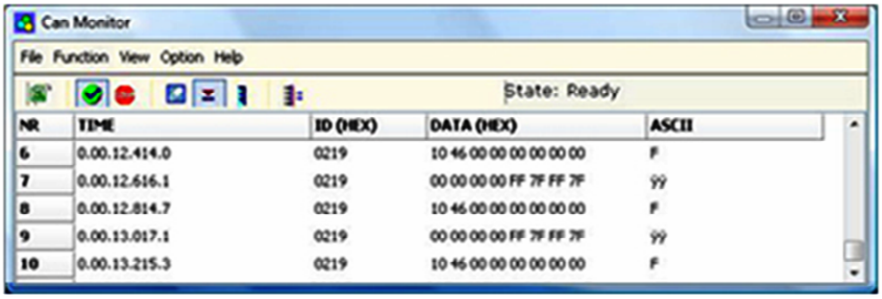



**Figura 4.5:** Pantalla de la opción Can Monitor[121]

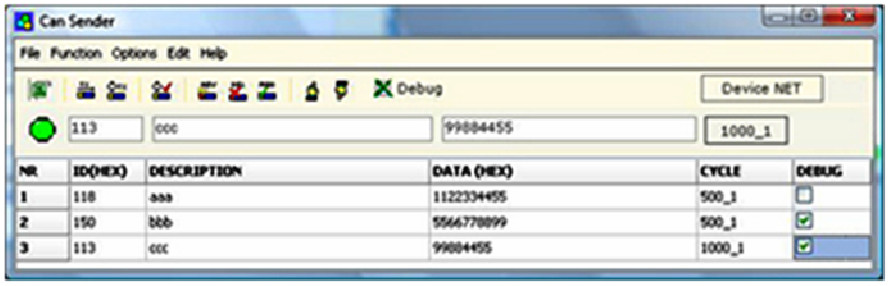



**Figura 4.6:** Pantalla de la opción Can Sender[122]

El precio de este analizador CAN es de $ 287 dólares para la versión básica y $ 355,40 dólares para la versión avanzada[123].

### 4.2.2 INTERFACES PARA COMPUTADOR PERSONAL

Existe una amplia clase de interfaces CAN que ofrecen soluciones para todo tipo de aplicaciones. Entre las interfaces más comúnmente utilizadas se encuentran: CAN-PCI, CAN-ISA, CAN-PC/104, CAN-PCMCIA, CAN-USB, CAN-Bluetooth.

120 - 121 – 122 Fuente: ADFweb.com /Home/Products/CAN Analizer
123 Fuente: http://www.adfweb.com/home/products/CAN_BUS_analyzers.asp

*Soluciones Softing AG*

Softing AG es una empresa que se enfoca principalmente en el desarrollo de tecnologías de comunicación industrial y sistemas de control, comercializando productos y servicios en las áreas de automatización industrial y electrónica automotriz. Los productos y servicios se extienden desde componentes, consultorías, desarrollo y producción de sistemas hasta complejas soluciones de automatización.

Para aplicaciones CAN, Softing integra y vende hardware y software de estándares reconocidos como lo son CANopen y DeviceNet.

#### 4.2.2.1 CANopen

Dentro de las soluciones CANopen, se encuentran disponibles servidores OPC para integrar las diferentes aplicaciones estandarizadas.

Los servidores OPC permiten explorar la red CAN para detectar automáticamente dispositivos instalados. Un servidor de este tipo, es una herramienta muy útil ya que facilita el desarrollo, análisis y configuración de una red CAN.

Las tarjetas CANopen son totalmente configurables y permiten el control y visualización de las tareas.

Las características más importantes de las interfaces CANopen son las siguientes:

- Tarjeta activa maestro con su propio microcontrolador.
- CANopen API.
- El protocolo CANopen es procesado directamente en la tarjeta.
- Uno o dos canales.
- Tiempo de respuesta exacto con resolución de milisegundos.
- Soporte de aplicaciones asincrónicas con tareas y colas de eventos.
- Velocidad de transferencia máxima de 1 Mbps.
- Compatibilidad con Windows XP/2000/NT/ME/98.

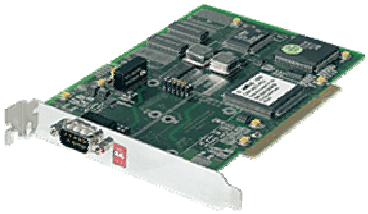
| ***CAN-AC1-PCI*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PCI | |
| *Microcontrolador* | Infineon SAB C165 (16 bits) | |
| *Controlador CAN* | 1 x Philips SJA1000 | |
| *Descripción* | Posee un solo canal, permite filtrado, preprocesamiento, trasmisión y almacenamiento de mensajes CAN. | |
| *Precio* | $ 597 Dólares | |

**Tabla 4.2:** Características generales CAN-AC1-PCI[124]

| ***CAN-AC2-PCI*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PCI | |
| *Microcontrolador* | Infineon SAB C165 (16 bits) | |
| *Controlador CAN* | 2 x Philips SJA1000 | |
| *Descripción* | Posee dos canales, permite filtrado, preprocesamiento, trasmisión y almacenamiento de mensajes CAN. | |
| *Precio* | $ 1012 Dólares | |

**Tabla 4.3:** Características generales CAN-AC2-PCI[125]

| ***CAN-AC1-104*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PC/104 | |
| *Microcontrolador* | Infineon SAB C165 (16 bits) | |
| *Controlador CAN* | 1 x Philips SJA1000 | |
| *Descripción* | Posee un solo canal, adecuado para operar en condiciones de ambiente ruidosos. | |
| *Precio* | $ 559 Dólares | |

**Tabla 4.4:** Características generales CAN-AC1-104[126]

[124 - 125 - 126] Hitex (UK) Webshop, 2008; “CAN Tools”

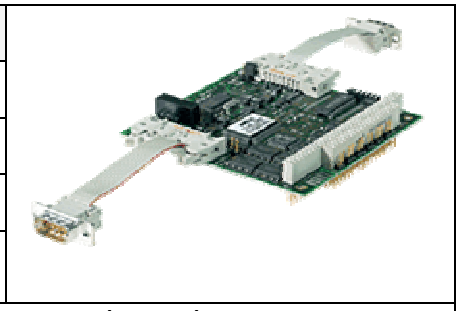
| ***CAN-AC2-104*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PC/104 | |
| *Microcontrolador* | Infineon SAB C165 (16 bits) | |
| *Controlador CAN* | 2 x Philips SJA1000 | |
| *Descripción* | Posee dos canales independientes, adecuado para operar en condiciones de ambiente ruidosos. | |
| *Precio* | $ 937 Dólares | |

**Tabla 4.5:** Características generales CAN-AC2-104[127]

| ***CANCard2*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PCMCIA | |
| *Microcontrolador* | Infineon SAB C165 (16 bits) | |
| *Controlador CAN* | 2 x Philips SJA1000 | |
| *Descripción* | Posee dos canales, adecuado para sistemas CAN móviles, transceiver integrado para soportar diferentes esquemas de cableado. | |
| *Precio* | $ 1586 Dólares | |

**Tabla 4.6** Características generales CAN-AC1-PCI[128]

| ***CANusb*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | USB | |
| *Microcontrolador* | Infineon SAB C165 (16 bits) | |
| *Controlador CAN* | 1 x Philips SJA1000 | |
| *Descripción* | Interface plug&play, posee un solo canal, adecuado para conectar un computador personal a una red CAN. | |
| *Precio* | $ 1042 Dólares | |

**Tabla 4.7** Características generales CAN-AC1-PCI[129]

[127 - 128 - 129] Hitex (UK) Webshop, 2008; “CAN Tools”

#### 4.2.2.2 DeviceNet

DeviceNet ofrece varios tipos de interfaces que permiten operar a una PC como un dispositivo CAN maestro y/o esclavo, lo que resulta adecuado para arquitecturas multi-maestro.

El protocolo CAN es implementado directamente en las interfaces como firmware, ésta característica hace que las tarjetas DeviceNet sean particularmente veloces. DeviceNet ofrece interfaces del tipo activo, de uno o dos canales. Similar a las interfaces de CAN bus y CANopen, estas tarjetas son configurables y permiten controlar y visualizar las tareas dentro de la aplicación.

Las principales características que poseen estas interfaces son:

- Tarjetas activas, maestro/esclavo con su propio microcontrolador.
- DeviceNet API.
- Protocolo CAN procesado directamente en la tarjeta.
- Tiempo de respuesta exacto con resolución de milisegundos.
- Funciones de escaneo a los esclavos lo que permite asignaciones dinámicas.
- Procesamiento de datos de tiempo crítico son almacenados en la DPRAM.
- Transmisión de cualquier cantidad de datos vía “mensajería explícita”.
- Soporte de aplicaciones asincrónicas con tareas y colas de eventos.
- Compatibilidad con Windows XP/2000/NT/ME/98.

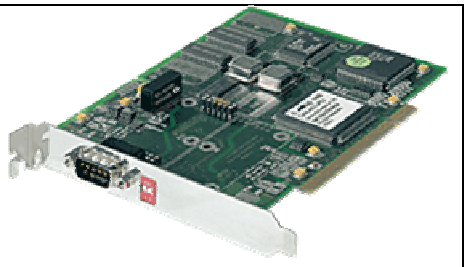
| ***CAN-AC1-PCI/DN*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PCI | |
| *Microcontrolador* | Motorola 68331 (16 bits) | |
| *Controlador CAN* | 1 x Philips SJA1000 | |
| *Descripción* | Posee un canal, permite filtrado, preprocesamiento, trasmisión y almacenamiento de mensajes CAN. | |
| *Precio* | $ 597 Dólares | |

**Tabla 4.8:** Características generales CAN-AC1-PCI/DN[130]

[130] Hitex (UK) Webshop, 2008; “CAN Tools”

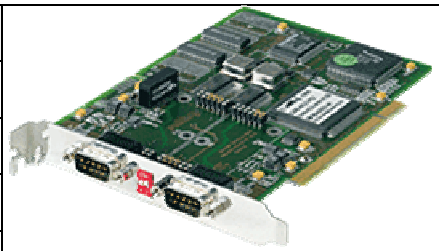
| ***CAN-AC2-PCI/DN*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PCI | |
| *Microcontrolador* | Motorola 68331 (16 bits) | |
| *Controlador CAN* | 2 x Philips SJA1000 | |
| *Descripción* | Posee dos canales, permite filtrado, preprocesamiento, trasmisión y almacenamiento de mensajes CAN. | |
| *Precio* | $ 1012 Dólares | |

**Tabla 4.9:** Características generales CAN-AC2-PCI/DN[131]

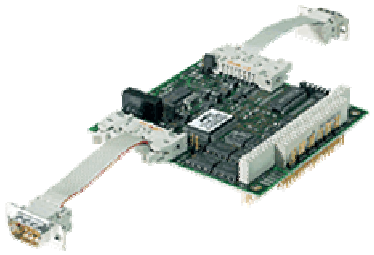
| ***CAN-AC1-104/DN*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PC/104 | |
| *Microcontrolador* | Motorola 68331 (16 bits) | |
| *Controlador CAN* | 1 x Philips SJA1000 | |
| *Descripción* | Posee un canal, permite filtrado, preprocesamiento, trasmisión y almacenamiento de mensajes CAN. | |
| *Precio* | $ 693 Dólares | |

**Tabla 4.10:** Características generales CAN-AC1-PCI/DN[132]

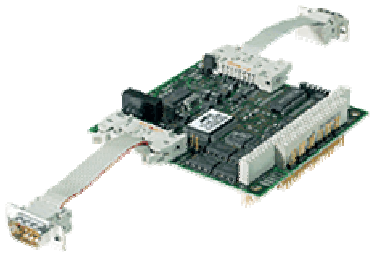
| ***CAN-AC2-104/DN*** | |  |
|---|---|---|
| *Tipo* | Activo | |
| *Técnica* | PC/104 | |
| *Microcontrolador* | Motorola 68331 (16 bits) | |
| *Controlador CAN* | 2 x Philips SJA1000 | |
| *Descripción* | Posee dos canales, permite filtrado, preprocesamiento, trasmisión y almacenamiento de mensajes CAN. | |
| *Precio* | $ 997 Dólares | |

**Tabla 4.11:** Características generales CAN-AC1-PCI/DN[133]

131 - 132 - 133 Hitex (UK) Webshop, 2008; “CAN Tools”

# REFERENCIAS BIBLIOGRÁFICAS CAPÍTULO 4

## Direcciones Electrónicas

[1] Softing, 2008; “CAN bus Interface Cards and Boards CAN PCI, CAN PC/104, CAN USB CAN PC Card, PCMCIA”, http://www.softing.com/home/en/industrial-automation/products/can-bus/interface cards/can/index.php?navanchor=3010132

[2] Hitex (UK) Webshop, 2008; “CAN Tools”, http://www.hitex.co.uk/shop/index1.html

# CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

## 5.1 CONCLUSIONES

- En base a la bibliografía consultada, se logró comprender los fundamentos del protocolo CAN y sus características, como resultado de esto se presentó un prototipo que muestra el funcionamiento del protocolo en base a intercambio de información entre diferentes microcontroladores, simulando una red de sensores.

- El protocolo CAN puede ser considerado como un protocolo estándar para la construcción de máquinas actuadoras independientes y sensores inteligentes. CAN es suficientemente rápido para aplicaciones en tiempo real donde las exigencias de rendimiento son indispensables, además por su excelente confiabilidad, es adecuado para ambientes de alto ruido tales como vehículos, industrias textiles, aeronáutica, sistemas marítimos etc.

- Las necesidades actuales que requieren ciertas aplicaciones domóticas, industriales y automotrices, han impulsado el avanzado desarrollo tecnológico de los buses de comunicaciones, razón por la cual el bus CAN consiguió un enorme progreso y especialmente gran acogida a niveles industriales y automotrices.

- Se conoció aspectos fundamentales sobre algunos buses de comunicaciones, características básicas como velocidad de transmisión, modo de operación, formato de trama, fueron analizadas y comparadas con el bus CAN.

- Se logró conocer el funcionamiento básico y algunas características de los sensores de humo, magnético, movimiento y temperatura, así como también ciertas aplicaciones en las que dichos sensores podrían resultar de utilidad.

- Se implementaron tres nodos interconectados por un bus de comunicaciones CAN. Los resultados obtenidos en la implementación del sistema, muestran que la solución es eficiente para el flujo de datos emitido a la velocidad de trabajo seleccionada, cumpliéndose así las expectativas del proyecto planteado.

- En la actualidad se dispone de microcontroladores CAN de diversos fabricantes que ofrecen soluciones que varían desde transceivers y controladores independientes hasta microcontroladores CAN.

- Aunque los componentes CAN son económicos, la solución implementada puede resultar poco atractiva en cuanto al precio debido al costo del diseño y a la licencia del software utilizado que, conjuntamente, representan el 90% del costo total del prototipo.

- De acuerdo a la especificación CAN, el tiempo de bit CAN está conformado por cuatro segmentos, segmento de sincronización, segmento de propagación, segmento de fase 1 y segmento de fase 2. Cada segmento consiste de un número de quantums programables, por tanto, la temporización del bit no es un proceso arbitrario, se debe tener en cuenta los componentes que afectan a la temporización y compensarlos para tratar de obtener un buen rendimiento del sistema a la velocidad configurada.

- La utilidad que ofrecen los compiladores para microcontroladores aumenta notablemente, el prototipo fue desarrollado utilizando MikroC, compilador para microcontroladores PICs y DSPICs. Una característica importante de este software es la inclusión de librerías y depuradores, esto facilita enormemente el desarrollo de sistemas microprocesados complejos, en un entorno de programación ciertamente intuitiva. Un proceso importante que debe destacarse dentro del programa fue el manejo de interrupciones ya que tres de los cuatro sensores fueron manejados utilizando interrupciones, la herramienta de desarrollo facilita el manejo de las mismas.

- Cada nodo permite la desactivación manual de la sirena, esto es útil ya que en caso de no deshabilitarla remotamente, el usuario puede hacerlo localmente.

- Debido a la naturaleza del prototipo, el software de monitoreo tiene opciones restringidas, únicamente permite enviar tramas específicas (mensajes de control) y además no permite configurar la velocidad del bus.

## 5.2 RECOMENDACIONES

- Se recomienda hacer un estudio del bus de comunicaciones LIN (Local Interconnect Network) como subsistema de una red CAN.

- Para la programación de aplicaciones sencillas es recomendable utilizar lenguajes de programación menos complejos como pueden ser PICSimulator IDE, mientras q para aplicaciones más complejas es conveniente utilizar compiladores más poderosos como lo son MikroC, MikroBasic, MPLAB, etc.

- Los nodos CAN con reloj propio necesariamente deben ser configurados para trabajar a la misma velocidad, es decir con los mismos parámetros de temporización de bit, de otro modo el sistema no funciona adecuadamente.

- Para el diseño de redes CAN más complejas, o de un número de nodos elevados, se recomienda considerar la existencia de las interfaces CAN mencionadas en el capítulo 4.

- El número de nodos CAN permitidos en el bus es teóricamente ilimitado, en la práctica el número total se ve restringido debido a los retardos de propagación y a retardos introducidos por el hardware, en general se recomienda remitirse al estándar ISO 11898 que especifica un valor máximo de 32 nodos para

trabajar a la velocidad máxima 1Mbps. En el caso de requerir un mayor número de nodos, se podría utilizador repetidores o transceivers.

- El prototipo fue diseñado para manejar los sensores utilizando las líneas de I/O que poseen interrupción por tanto el límite de sensores se ve restringido por el número de interrupciones permitidas por el microcontrolador PIC18F2480, que en este caso son tres. En el caso de requerir el control de un mayor número de sensores, se recomienda utilizar multiplexores u otro microcontrolador que posea un número mayor de interrupciones acoplado a un controlador CAN independiente.

- Se recomienda que en estudios futuros se implementen sistemas más complejos que permitan realizar la configuración de velocidad del bus al igual que enviar información personalizada.

- Para el adecuado funcionamiento del programa de monitoreo, se recomienda utilizar un computador que posea las siguientes características: Sistema Operativo Windows XP o superior, procesador de 700Mhz o superior, mínimo 256MB de memoria RAM, Microsoft Framework.Net versión 2.0 o superior y tener un puerto serial disponible.

- Para el correcto manejo del prototipo, se recomienda remitirse al Anexo I que corresponde al manual de usuario.

# GLOSARIO

**1-Wire.-** Bus serial asincrónico desarrollado por Dallas/Maxim con el propósito de simplificar los sistemas de redes de control utilizando un único cable de par trenzado en comunicaciones bidireccionales.

**ACK.-** Acknowledgement o acuse de recibo, es un mensaje que se envía para confirmar que una trama o un conjunto de tramas han llegado.

**BasicCAN.-** Tipo de controlador CAN que permite funciones de filtrado básico para mensajes CAN.

**Bits-stuffing.-** Inserción de bits o método de relleno de bits o violación de bits consiste en emitir un bit de polaridad opuesta cada 5 bits consecutivos de la misma polaridad.

**CAN.-** Controller Area Network, bus de comunicaciones seriales cuya velocidad máxima de trasmisión es 1Mbps. Bus comúnmente utilizado a nivel industrial y automotriz.

**CAN_H.-** CAN_High, línea alta del bus de comunicaciones CAN.

**CAN_L.-** CAN_Low, línea baja del bus de comunicaciones CAN.

**CiA.-** CAN in Automation, organización de fabricantes y usuarios de dispositivos CAN.

**CRC.-** Cyclic Redundancy Check, Chequeo de Redundancia Cíclica o códigos polinómicos, son utilizados como mecanismo de verificación de errores en una trama, su uso está muy extendido porque pueden implementarse en hardware con mucha facilidad y son muy potentes.

**CSMA/CD with AMP.-** Carrier Sense Multiple Access/ Collision Detection with Arbitration on Message Priority, método de acceso al medio basado en el método de detección de colisión pero con priorización de mensajes.

**DLC.-** Data Length Code, bits pertenecientes al campo de control de trama CAN, permiten identificar el número de octetos (0 a 8) que contendrá el campo de datos.

**EEPROM.-** Electrical Erasable Programmable Read Only Memory, tipo de memoria que permite grabado y borrado de manera eléctrica.

**EOF.-** End of Frame o final de trama, es un campo de la trama CAN que contiene una secuencia de siete bits en estado recesivo, esta bandera especial es utilizada para delimitar el fin de las tramas de datos y remotas.

**EPROM.-** Erasable Programmable Read Only Memory, tipo de memoria que se graba eléctricamente, para su borrado se utiliza luz ultravioleta.

**EUSART.-** Enhanced USART, Enhanced Universal Synchronous Asynchronous Receiver Transmitter, es una versión mejorada del bus de comunicaciones USART, introduce mejoras como calibración de velocidad, capacidad de auto-wake-up entre otras.

**Firmware.-** Es un híbrido entre el hardware y el software, es decir tiene parte física y una parte de programación consistente en programas internos implementados en memorias no volátiles.

**FLASH.-** Memoria no volátil de bajo consumo, similar a la EEPROM pero con mayor capacidad. Ideales para aplicaciones que serán modificadas continuamente.

**FullCAN.-** Tipo de controlador CAN que permite funciones de filtrado más complejas para mensajes CAN.

**GPS.-** Global Positioning System o Sistema de Posicionamiento Global, consiste en un sistema global de navegación por satélite que permite localizar usuarios en cualquier parte del mundo.

**$I^2C$.-** Inter-Integrated Circuits o I2C define un bus serial sincrónico de comunicaciones que usa dos hilos para transmitir la información, por un hilo viajan los datos y por el otro la señal de reloj.

**ICSP.-**In Circuit Serial Program, programación serial en circuito, es una técnica que permite la programación del microcontrolador utilizando dos pines I/O de modo que puede ser programado en el mismo circuito.

**IDE.-** Identifier Extension, pertenece al campo de control en una trama estándar y en una trama extendida, pertenece al campo de arbitraje, sirve para determinar la versión del protocolo.

**ISO.-** Organización Internacional de Estandarización, organismo de normalización, fundada en 1946, tiene como funciones estandarizar diversas áreas entre ellas las telecomunicaciones.

**LIN.-** Local Interconnect Network, subconjunto del bus de datos CAN, es un bus unidireccional que permite transferencias de hasta 20 Kbps. El bus LIN conecta actuadores o sensores con las correspondientes unidades de control dentro de un vehículo.

**LLC.-** Logical Link Control, subcapa de la capa enlace de datos del modelo de referencia ISO/OSI.

**MAC.-**Medium Access Control, subcapa de la capa enlace de datos del modelo de referencia ISO/OSI.

**Microwire.-** Microwire es un subconjunto del bus SPI, desarrollado por National Semiconductor Corporation como un bus serial sincrónico utilizado para los mismos propósitos que el bus SPI.

**MIPS.-** Millon Instructions Per Second, número de instrucciones en millones que el procesador puede ejecutar por segundo, es una forma de medir la potencia de los procesadores.

**Multicast.-** Modo de difusión de información que permite tener múltiples receptores en una red.

**MVB.-** Multiple Vehicle Bus, bus de comunicaciones que permite el intercambio de información entre los equipos instalados de manera local dentro de un vehículo.

**OTP.-** One Time Programmable, memorias que pueden ser programas por el usuario una sola vez. Por su bajo costo son ideales para aplicaciones industriales.

**Prescaler.-** Divisor de frecuencia que habitualmente puede configurarse o programarse en el programa utilizando registros específicos de un microcontrolador.

**REC.-** Reception Error Counter, es el contador de error de recepción que permite cambiar a un nodo de un estado de error pasivo a un estado de bus apagado.

**RJW.-** Resynchronization Jump Width, parámetro utilizado para resincronización dentro del bit CAN.

**ROM.-** Read Only Memory, memoria de tipo no volátil, comúnmente programada por el fabricante.

**Round-Robin.-** Sistema de contienda en donde todos los participantes se enfrentan entre ellos un número constante de oportunidades.

**RTR.-** Remote Transmission Request, bit del campo de arbitraje de la trama de datos CAN.

**SAE.-** Society of Automotive Engineers, es una organización de profesionales en ingeniería automotriz, aeroespacial y vehículos industriales. Esta organización es responsable del establecimiento de normas en la industria automotriz y de aviación.

**SCI.-** Serial Communication Interfaz, es un bus serial asincrónico, full-duplex que posee muchas similitudes con el bus UART.

**SMD.-** Surface Mount Device, Dispositivos de Montaje en Superficie son circuitos integrados de silicio montados en dispositivos de diminutas dimensiones.

**SOF.-** Start of Frame o delimitador de inicio de trama, es el campo de la trama CAN que determina el inicio de una trama remota o una trama de datos. Consiste de un bit dominante que sincroniza a todos los nodos activos en la red.

**SPI.-** Serial Peripheral Interface, fue desarrollado por Motorola como un estándar de comunicaciones utilizado para la interconexión serial sincrónica a tres hilos full-duplex.

**SRR.-** Substitutte Remote Request, es un bit del campo de arbitraje de la trama de datos CAN extendida.

**TEC.-** Transmission Error Counter, es el contador de error de trasmisión, que permite cambiar a un nodo de un estado de error activo a un estado de error pasivo.

**Tiempo Quantum.-** Es una unidad básica de tiempo de bit CAN que se deriva del periodo del oscilador.

**UART.-** Universal Asynchronous Receiver Transmitter, Transmisor-Receptor Asíncrono Universal es un bus de comunicaciones serial, asincrónico, full-duplex, comúnmente utilizado conjuntamente con otros estándares para comunicación seriales como el EIA RS-232

**UPB.-** Universal Powerline Bus, es un estándar de comunicación digital diseñado para control domótico, ciertamente similar a X10, éste bus trasmite información codificada sobre las líneas de corriente alterna utilizando pulsos eléctricos.

**USART.-** Universal Synchronous Asynchronous Receiver Transmitter, Transmisor-Receptor Sincrónico Asíncrono Universal es un bus de comunicaciones serial UART que permite trabajar tanto en modo asincrónico como en modo sincrónico.

**USB.-** Universal Serial Bus, desarrollado por IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC como un estándar de bus serial para la interconexión de dispositivos a un computador personal. USB es un bus que puede operar tanto en modo sincrónico como asincrónico.

**WTB.-** Wide Train Bus, bus serial que permite la comunicación entre diferentes vagones de un mismo tren.

**X10.-** Protocolo diseñado por Pico Electronics con el propósito de conseguir un control de los equipos domésticos utilizando las líneas de trasmisión eléctrica (220V o 110V).