UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA EN MECÁNICA AUTOMOTRIZ

ANÁLISIS DEL FUNCIONAMIENTO Y APLICACIÓN DE LAS REDES DE COMUNICACIÓN MULTIPLEXADAS EN VEHÍCULOS AUTOMOTRICES

Trabajo de grado previo a la obtención del título de ingeniero en mecánica automotriz

Autor:

Jimmy Gustavo Ruiz Campoverde

Director:

Pedro Jose Crespo Vintimilla

Cuenca – Ecuador

2013

DEDICATORIA

Para mis queridos padres, Ninfa y Enrique, por su apoyo incondicional, su comprensión y paciencia que me supieron dar tanto en los buenos momentos como en los malos.

AGRADECIMIENTOS

Agradezco a todos los docentes que me ayudaron en la realización de este trabajo, especialmente a mi director de tesis, Ing. Pedro Crespo, por su colaboración y paciencia.

INDICE DE CONTENIDOS

CAPÍTULO 2: TIPOS DE REDES MULTIPLEXADAS EN EL CAMPO AUTOMOTRIZ

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

TEMA:

ANÁLISIS DEL FUNCIONAMIENTO Y APLICACIÓN DE LAS REDES DE COMUNICACIÓN MULTIPLEXADAS EN VEHÍCULOS AUTOMOTRICES

RESUMEN:

En el presente trabajo de grado se realizó el compendio teórico del funcionamiento de las redes multiplexadas automotrices y además se estableció un procedimiento sistematizado para diagnosticar y reparar las mismas. De igual manera, se analizó y comparó los tipos de redes multiplexadas automotrices acorde a la clasificación SAE. Se simuló un sistema multiplexado para las puertas de un vehículo en base a 4 microcontroladores AVR; como resultado de ello, se obtuvo parámetros técnicos para seleccionar, comparar y reparar un bus de datos. Finalmente, se evidenció la simplicidad y funcionalidad del circuito simulado, concluyendo que las redes multiplexadas en un vehículo representan mayores ventajas que el cableado convencional.

PALABRAS CLAVE: redes, multiplexado, red CAN, sistemas embebidos, comunicación serial.

Ing. Pedro Crespo
Director de Tesis

Mg. Mauricio Barros
Director de Escuela

Jimmy Ruiz
Alumno

14/10/13

ABSTRACT

PERFORMANCE ANALYSIS AND APPLICATION OF MULTIPLEXED COMMUNICATION NETWORKS IN AUTOMOTIVE VEHICLES

In the present work we performed a theoretical compilation of the performance of automotive multiplexed network; in addition, we established a systematic procedure for their diagnostic and repair.

Similarly, we analyzed and compared the automotive multiplexed network types according to the SAE classification. We simulated a multiplexing system for vehicle doors based on 4 AVR microcontrollers. As a result, we obtained technical parameters to select, compare and repair a data bus.

Finally, we evidenced the simplicity and functionality of the simulated circuit, concluding that the multiplexed networks in a vehicle represent major advantages than the conventional wiring

Keywords: network, multiplexing, CAN network, embedded system, serial communication.

Ing. Pedro Crespo
THESIS DIRECTOR

Magister Mauricio Barros
SCHOOL DIRECTOR

Jimmy Ruiz
STUDENT

UNIVERSIDAD DEL AZUAY
DPTO. IDIOMAS

Translated by:
Lic. Lourdes Crespo

Jimmy Gustavo Ruiz Campoverde

Trabajo de Graduación

Ing. Pedro Jose Crespo Vintimilla

Junio, 2013

ANÁLISIS DEL FUNCIONAMIENTO Y APLICACIÓN DE LAS REDES DE COMUNICACIÓN MULTIPLEXADAS EN VEHÍCULOS AUTOMOTRICES

INTRODUCCIÓN

La eminente evolución de la tecnología automotriz ha dado como resultado la implementación de sistemas cada vez más complejos que usan control electrónico. Como resultado de ésta evolución, se han incrementado notablemente los módulos electrónicos en los diferentes sistemas del vehículo, hasta tal punto que se están remplazando gran parte de los sistemas que anteriormente eran controlados con mandos mecánicos y neumáticos por sistemas controlados electrónicamente. Debido a éste notable incremento en el número de módulos electrónicos, ha surgido la necesidad de interconectarlos para que todos los sistemas del vehículo puedan compartir datos. Las redes multiplexadas automotrices tienen las función de interconectar los módulos electrónicos de los diferentes sistemas del vehículo para que éstos trabajen eficientemente. El presente trabajo de investigación analiza el funcionamiento y aplicación de las redes de comunicación multiplexadas en vehículos automotrices.

En primer lugar se explica el funcionamiento de las redes multiplexadas automotrices mediante la definición de conceptos usados en redes, estableciendo como se organiza una red, analizando las diferentes tipologías de red de uso automotriz, evaluando los parámetros para la selección de un bus de datos, analizando el modelo de referencia OSI, explicando los mecanismos de control, y dando una guía de diagnostico y reparación de redes automotrices.

Se compara las características de los diferentes buses de datos basándose en la clasificación SAE. Por lo tanto características de funcionamiento, aplicaciones en el vehículo, ventajas y desventajas de cada una de las clases A, B, C, y D. Además se

va ha comparar dentro de cada clase las características de los protocolos mas comunes.

Finalmente mediante la simulación de una red multiplexada se controla el sistema eléctrico de las puertas del vehículo. Esta red multiplexada consta de cuatro módulos electrónico y cada uno usa un microcontrolador AVR Atmega8. El primer módulo se usa como master y controlará los otros tres módulos subordinados, por medio del puerto USART. Los tres módulos subordinados serán la unidad de seguros eléctricos, la unidad de vidrios eléctricos, y la unidad de retrovisores eléctricos.

.

CAPÍTULO I

DESCRIPCIÓN GENERAL DEL FUNCIONAMIENTO DE REDES MULTIPLEXADAS EN EL VEHÍCULO Y GUÍA PARA REPARARLAS

1.1. Introducción

Las redes de comunicación multiplexadas en el vehículo hacen referencia a la interconexión que existe entre computadoras (Ecus) o módulos electrónicos a través de un medio de transportación. Esta interconexión se hace con el fin de compartir información de una computadora o modulo electrónico hacia otro u otros que la requieran. Con la comunicación multiplexada obtenemos muchas ventajas con respecto a un cableado convencional, entre ellas:

- "*Reducimos el costo con menos peso y espacio de instalación debido a los pocos cables en el arnés de cableado.*
- *Mejor confiabilidad y funcionalidad debido a las pocas conexiones plug-in.*
- *Simplificación del ensamblaje del vehículo durante la producción.*
- *Múltiples usos de las señales de los sensores.*
- *Conexión sencilla de componentes de sistema a un bus.*
- *Un manejo mas fácil de equipos y variantes de equipos especiales en un vehículo*."[1]

[1] Bosch, *Automotive Networking*, 2007, pág. 17

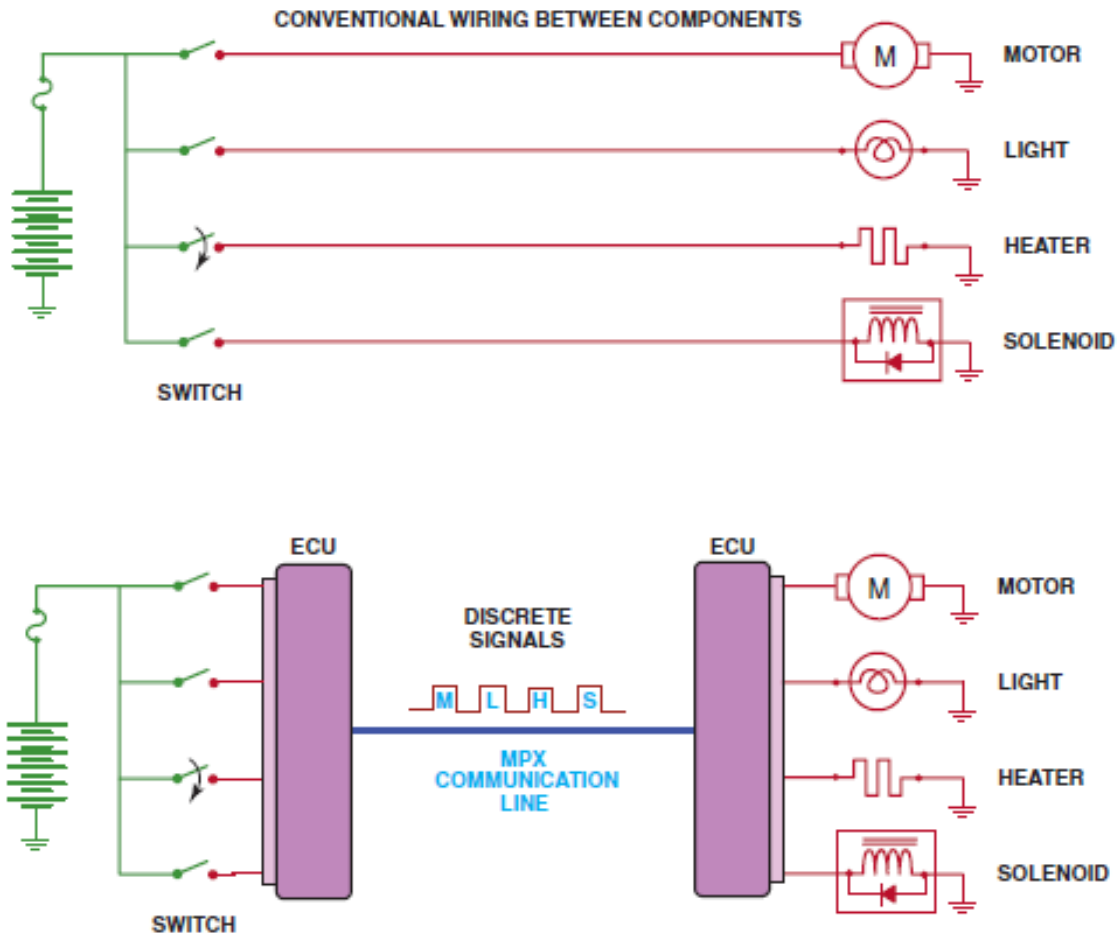

Fig.1.1. Relación entre un cableado convencional y una línea de comunicación multiplexada.
Fuente: Halderman, *Automotive Technology*, 2012, pág. 525

En la Figura 1.1 se puede observar gráficamente las diferencias y las ventajas que existen entre un cableado convencional y una línea de comunicación multiplexada. En el cableado convencional se usa una línea directa entre el pulsante y el actuador, utilizando una gran cantidad de cables dependiendo de la ubicación entre actuador y el tablero de mandos. Con el uso de una red multiplexada se usa un bus que lleva todos los datos en un solo cable, esto conlleva el uso de más módulos electrónicos. Sin embargo, representa una ventaja tanto en eficiencia como economía el uso de un mayor número de módulos que el uso excesivo de cables.

Con el fin de analizar el funcionamiento de las redes de comunicación multiplexadas en el automóvil, el presente capítulo consta de seis secciones que describen el funcionamiento de ésta (sección 1.2 a 1.7). Adicionalmente presentará una guía de diagnóstico y reparación de redes automotrices (sección 1.8):

- Definición de conceptos usados en redes. (sección 1.2)
- Organización de una red (sección 1.3)
- Análisis de las diferentes tipologías de red que usan en vehículos automotrices. (sección 1.4)
- Análisis de los parámetros para la selección de un bus de datos. (sección 1.5)
- Análisis del estándar OSI que da la base para comparar los protocolos de comunicación. (sección 1.6)
- Mecanismos de Control. (sección 1.7)
- Guía de Diagnóstico y Reparación de Redes Automotrices. (sección 1.8)

1.2. Conceptos Usado en Redes

1.2.1. Bit y Byte

Bit.- Es un digito en el sistema binario, y se representa por un periodo completo dentro de una secuencia de bits.

Byte.- El byte representa una unidad de información**,** y además es múltiplo del bit y su equivalencia generalmente es de 8 bits.

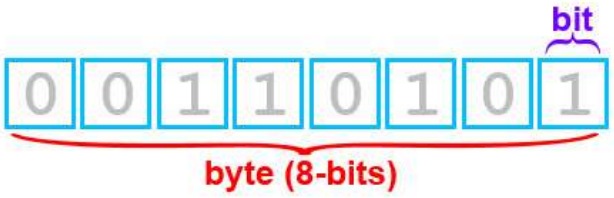

Fig.1.2. Representación esquemática de un Byte
Fuente: Imagen tomada de: USGS, http://isis.astrogeology.usgs.gov/IsisWorkshop/index.php/Understanding_Bit_Types , junio del 2013

1.2.2. Red

"*Una red se entiende como un sistema en cual un grupo de elementos puede intercambiar información a través de un medio de transportación*."[2]

1.2.3. Nodo

Es una computadora o módulo electrónico que esta conectado a una red. Los nodos de red también se los conoce como subscriptores o estaciones de red.

1.2.4. Niveles de Transmisión de Datos: Recesivo y Dominante

En la electrónica digital se usan dos niveles lógicos 0 y 1, que representan por ejemplo, que un *led* se encuentra apagado cuando su nivel lógico es 0, mientras que cuando su nivel lógico es uno, el *led* se enciende. Por lo tanto, el **nivel recesivo** está representado por 1, mientras que el **nivel dominante** se lo representa por 0.

1.2.5. Bus de Datos

"Es un término usado para describir una red de comunicación. Por lo tanto, hay conexiones hacia el BUS y comunicaciones de BUS, ambos se refieren a mensajes digitales siendo trasmitidos entre módulos electrónicos o computadoras."[3]

En otras palabras bus de datos es la línea o líneas (1 cable o 2 cables trenzados) por donde circulan los datos en una red.

1.2.6. ¿Por qué el uso de el par de cables trenzados?

"*Un par de cables trenzados se utiliza para prevenir radiaciones electromagnéticas que afecten las señales que están pasando a través de los cables. Trenzando dos cables alrededor de una vez por cada pulgada, la interferencia es cancelada por el cable adyacente*."[4]

---

[2] Bosch, *Automotive Networking*, 2007, pág. 4
[3] Halderman, *Automotive Technology*, 2012, pág. 526
[4] Halderman, *Automotive Technology*, 2012, pág. 529

1.2.7. Modos *Sleep* y *Wake-up*

Con el fin de reducir el consumo de energía en una red se implementan los modos *Sleep* y *wake-up.* Cuando un nodo esta en *Sleep mode* cesa su actividad y su controlador del bus se desconecta de la línea. El estado *Sleep* termina cuando el nodo se ve afectado por alguna actividad o condiciones internas del sistema local, y entonces el nodo pasa a tener un estado *wake-up*.

1.2.8. Protocolos de Comunicación

"*Un protocolo es un conjunto de reglas o un estándar usado entre computadoras o módulos de control electrónico. Los protocolos incluyen conectores eléctricos y niveles de voltaje y frecuencia de los mensajes transmitidos. Los protocolos, además, incluyen el hardware y el software que se necesita para la comunicación entre módulos*."[5]

[5] Halderman, *Automotive Technology*, 2012, pág. 527

1.3. Organización de una red

Una red de comunicación multiplexada debe estar correctamente organizada para evitar colisión de datos y por lo tanto errores en la comunicación. Hay dos aspectos que se pueden tratar dentro de la organización de la una red: el direccionamiento y el método de acceso al bus.

1.3.1. Direccionamiento

Hay dos métodos que se utilizan para que un mensaje de datos llegue eficazmente desde el nodo transmisor hacia un nodo específico receptor, por lo tanto tenemos: el método orientado por el suscriptor y método orientado por el mensaje.

Método Orientado por el Suscriptor

En este método de direccionamiento de datos, el transmisor envía mensajes que contienen dos partes: los datos de información y además la dirección del nodo destinatario.

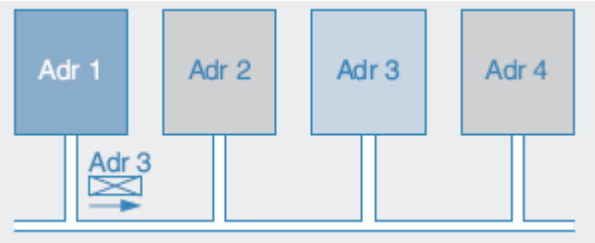

Fig.1.3. Direccionamiento de un mensaje hacia los nodos por el método orientado por el suscriptor.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 8

Método Orientado por el Mensaje

En este método el nodo transmisor no necesita poner ninguna dirección del nodo destinatario debido a que el nodo receptor evalúa si le sirve o no la información. El mensaje de datos puede ser evaluado por varios nodos.

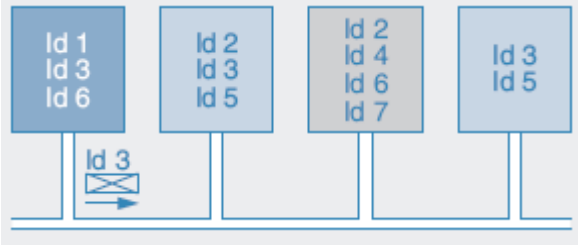

Fig.1.4. Direccionamiento de un mensaje hacia los nodos por el método orientado por el mensaje.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 8

En una trama de datos, el direccionamiento del mensaje lo lleva la sección de mensaje de cabecera (*header*), siendo así:

Mensaje de Cabecera (*header*).- esta compuesto de uno a tres bytes de información concerniente al tipo, longitud, prioridad, módulo objetivo y módulo que envía.
En la siguiente figura 1.5 de trama de datos, el mensaje de cabecera esta en la segunda sección después del SOF (*start of a message*), que es un bit de inicio del mensaje:

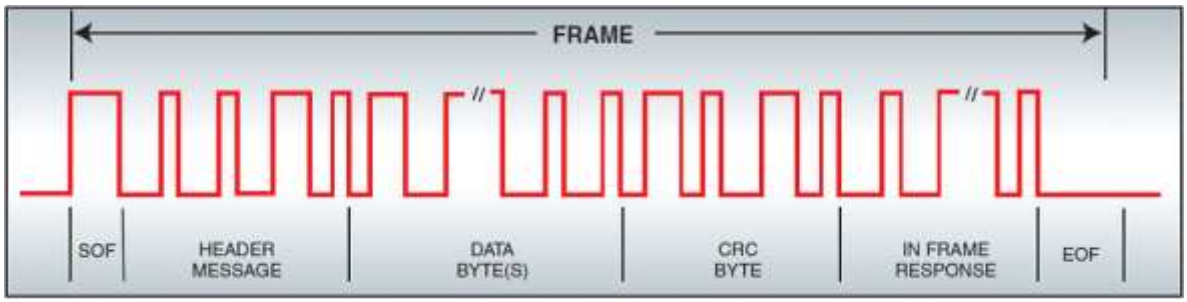

Fig.1.5. Partes de una trama de datos en un mensaje.
Fuente: Hollembeak, *Classroom Manual for Automotive Electricity And Electronics*, 2011, pág. 292

1.3.2. Método de Acceso al Bus

Para que un nodo pueda transmitir información, debe acceder al BUS. Hay dos métodos para acceder al BUS:

"***Método Predecible:*** *el acceso al bus es determinado por características de red tiempo-dependientes, en donde solamente un nodo puede transmitir a la vez.*

Método aleatorio: *cualquier nodo puede intentar trasmitir datos si el bus parece estar libre.*"[6]

Siendo así, tenemos tres tipos de mecanismos de control para acceder y trasmitir al bus:

- *Time Division Multiple Access* (TDMA)
- Maestro-Esclavo
- *Multimaster*

***Time Division Multiple Access* (TDMA)**

"*TDMA es un método de acceso determinístico (predictivo). En este caso a cada nodo es asignada una ventana de tiempo en cual este se le permite transmitir. Un horario establecido es además requerido para para la red. Los clocks internos de los diferentes nodos deben correr extremadamente sincrónicos con el TDMA*."[7]

En otras palabras el TDMA utiliza una partición de tiempo entre los diferentes nodos, por ejemplo, si el ciclo dura 4 segundos y hay 4 nodos, cada nodo disfrutará para sí solo de 1 segundo para transmitir cada 4 segundos.

Maestro-Esclavo

Bosch (2007) explica el sistema maestro-esclavo de la siguiente manera:

> En el sistema maestro-esclavo, un nodo en la red opera como el maestro. Éste nodo determina la frecuencia de comunicación por interrogación de sus nodos subordinados (esclavos). Un esclavo solamente responde si este es permitido por el master (figura 1.6). Sin embargo algunos protocolos maestro-esclavo permiten a un esclavo contactar un master con el fin de transmitir un mensaje (por ejemplo, transmitir información acerca de la posición de la unidad de vidrios eléctricos al módulo de la puerta) (pág. 9).

[6] Bosch, *Automotive Networking*, 2007, pág. 8
[7] Bosch, *Automotive Networking*, 2007, pág. 9

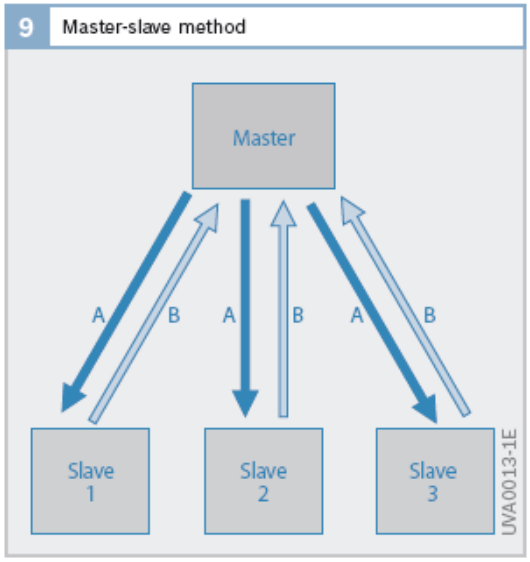

Fig.1.6. Configuración de nodos en método maestro-esclavo.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 9

Multimaster

"*En una red multimaster, cada nodo en la red puede transmitir si el bus parece estar libre. Ningún nodo depende de otro para poder trasmitir. Esta configuración es muy segura ya que ya que si un nodo deja de funcionar, el resto de nodos continúa funcionando correctamente ya que no depende de un master para transmitir*."[8]

[8] Bosch, *Automotive Networking*, 2007, pág. 17

1.4. Tipología de Red

Los módulos electrónicos en vehículos automotrices se los puede encontrar en configuraciones de tipo:

- Lineal
- Estrella
- Anillo
- Malla
- Híbrida

1.4.1. Tipo Lineal

Este tipo de configuración de bus lineal tiene un único cable principal, en donde los nodos se conectan por medio de cables pequeños. En este tipo de red, sin un nodo falla, el resto de nodos continua transmitiendo información. Sin embargo, la red falla cuando el cable principal esta defectuoso.

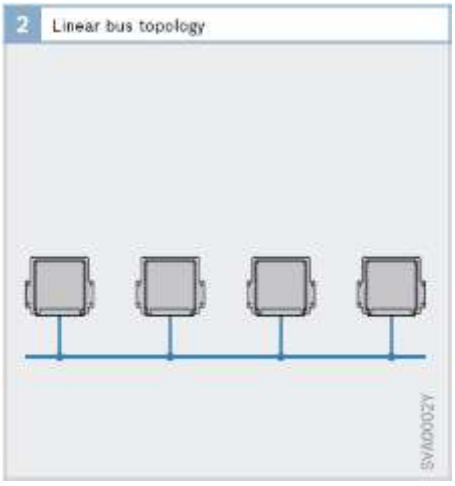

Fig.1.7. Configuración de red tipo lineal.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 5

1.4.2. Tipo estrella

En esta configuración, el nodo principal se conecta se conecta a los otros nodos por una única línea. Este tipo de red falla únicamente cuando el nodo principal falla. "*En automotriz, esta tipología de red, esta bajo discusión de seguridad, debido a que la red puede fallar si falla el nodo principal*."[9]

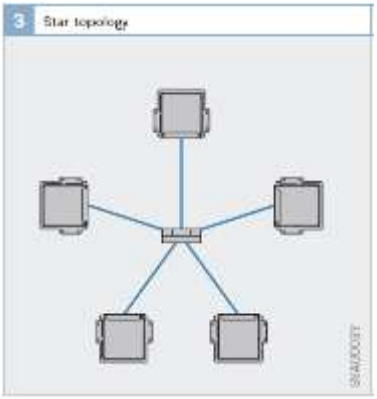

Fig.1.8. Configuración de red tipo estrella.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 5

1.4.3. Tipo anillo

En redes de tipo anillo cada nodo es conectado a sus dos vecinos. Los datos se transmiten unidireccionalmente de un nodo al siguiente. Si un nodo falla, todo la red se ve afectada.

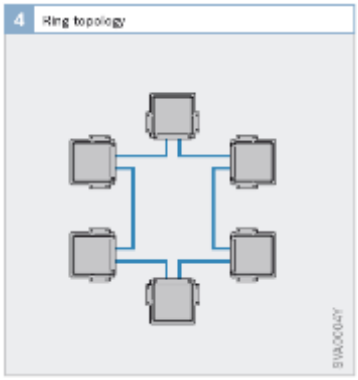

Fig.1.9. Configuración de red tipo anillo.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 6

[9] Bosch, *Automotive Networking*, 2007, pág. 6

1.4.4. Tipo malla

En esta configuración cada nodo esta interconectado con cada uno de los otros a través de una línea individual. Esta tipología de red es la mas estable y segura debido a que si una línea falla en trasmitir el mensaje, el mensaje se re-direcciona a través de otra. Sin embargo el costo es elevado debido al número de cables.

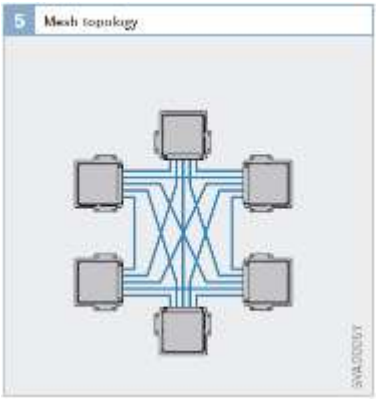

Fig.1.10. Configuración de red tipo malla.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 6

1.4.5. Tipo Híbrida

Este tipo de red es la combinación de dos tipologías de red, de esta manera podemos tener:

- Tipo Estrella-Bus (figura.11-6)
- Tipo Estrella- Anillo (figura.11-7)

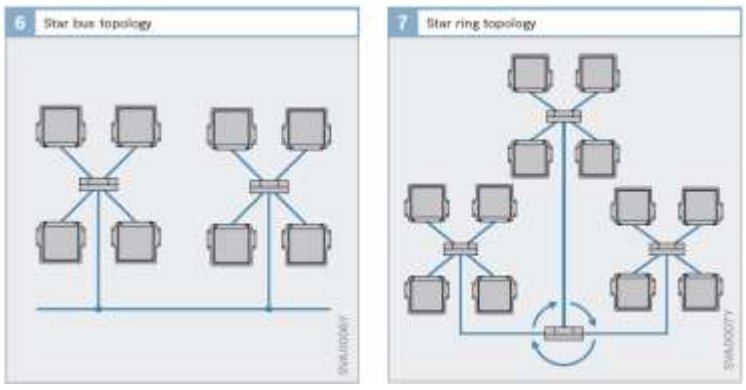

Fig.11. Configuraciones de red tipo estrella-bus y estrella-anillo.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 7

1.5. Requerimientos para los Sistemas de Bus

Bosch (2007) establece que para seleccionar un bus se deben tomar en cuenta tanto las restricciones económicas (costo del cable, costo de componentes, etc.) como técnicas. Los criterios técnicos de selección más importantes son:

- Velocidad de transferencia de datos
- Inmunidad de la Interferencia
- Capacidad de tiempo real
- Número de nodos de red

1.5.1. Velocidad de transferencia de datos

"*Esta variable especifica el volumen de datos que se transmiten durante una unidad de tiempo. La unidad mas pequeña de datos es el bit, y la velocidad de transferencia de datos es usualmente especificada en bits/segundo*."[10]

Para seleccionar la velocidad de transferencia de datos se debe tomar en cuenta la aplicación del bus. No es lo mismo transferir datos para activar el motor del *sunroof* que para recibir datos de navegación.

1.5.2. Inmunidad de la Interferencia

Lo ideal seria que los datos se transmitan sin interferencia, pero debido a que existen muchas interferencias electromagnéticas en el vehículo automotriz esto no puede ser posible. "*Los requerimientos de inmunidad de interferencias que son hechos depende de la relevancia de seguridad del sistema electrónico concernido. Por ejemplo, menos requerimientos se necesitan para confort y comodidad que para el sistema de frenos antibloqueo (ABS). Con el fin de reunir estos requerimientos, se incorporan mecanismos que detectan errores de transmisión en los protocolos de red*."[11]

Existen tres maneras para detectar errores en la transmisión de datos: el bit de paridad y la suma de verificación (*checksum)* y verificación de redundancia cíclica (CRC).

[10] Bosch, *Automotive Networking*, 2007, pág. 17
[11] Bosch, *Automotive Networking*, 2007, pág. 17,18

Bit de Paridad.- el bit de paridad es usado en la comunicación serial para detectar errores individuales (1byte). El bit de paridad es un bit (1 o 0) adicional añadido al bloque de bits que se desea proteger para detectar un posible error. El bit de paridad es calculado por el transmisor y la información es recibida por el receptor.

Suma de Verificación (*Checksum*).- Este es un algoritmo que también se utiliza en la comunicación serial para calcular errores en una corriente datos (varios bytes). El nodo trasmisor calcula el *checksum* mediante una formula y transmite el valor. El nodo receptor también calcula el *checksum* y compara con el *checksum* recibido.

Verificación de Redundancia Cíclica (CRC).- "*es una técnica poderosa para la detección de errores. Por lo tanto este es usado en todos los sistemas de comunicación de datos. En el transmisor, los bits adicionales son adjuntados al bloque, mientras que en lado receptor el CRC recibido es comparado con el CRC calculado. Si los dos son iguales, se considera que los bits de información fueron recibidos correctamente*."[12]

1.5.3. Capacidad de tiempo real

"*Los sistemas de tiempo real garantizan que sus resultados son calculados dentro de un intervalo de tiempo fijo.*"[13]

Requerimientos de tiempo-real flexibles.- "*el sistema generalmente se acopla a una respuesta de tiempo específico. Si los tiempos son ocasionalmente excedidos no produce un efecto serio en el sistema*."[14]

Por ejemplo en el sistema de aire acondicionado, no se precisaría de una capacidad de tiempo real ya que en este sistema si hay un retraso no representaría ningún peligro en el vehículo. El CAN es un ejemplo de un bus que reúne características de tiempo real flexibles.

Requerimientos de tiempo-real estrictos.- "*La especificación del tiempo debe ser estrictamente cumplido. Si el tiempo de respuesta del tiempo fue excedido, el*

[12] Prasad, *Principles Digital Communication System & Computer Networks*, 2004, pág. 61
[13] Bosch, *Automotive Networking*, 2007, pág. 18
[14] Bosch, *Automotive Networking*, 2007, pág. 18

resultado calculado no seria capaz de ser usado. Esto puede dirigirse a un serio problema en sistemas críticos de seguridad."[15]

Por ejemplo, el sistema de antibloqueo de frenos precisa de transmisión de datos en tiempo real con requerimientos estrictos, debido a que el incipiente bloqueo de ruedas debe ser detectado tan pronto como sea posible para que el cilindro master y la presión sea reducida para no bloquear las ruedas, y además sin dar lugar a retrasos ya que puede ser peligroso en la seguridad del vehículo. Por lo tanto, la característica de tiempo real estricto se utiliza en buses que son aplicados a sistemas que son activados por eventos latentes (por ejemplo, bus FLEXRAY).

1.5.4. Número de Nodos de Red

"*El máximo numero de nodos que puede ser integrado varía para la diferentes áreas de operación del vehículo. El numero de nodos para el sistema de confort y comodidad podría ser alto debido a los servomotores interconectados (por ejemplo en el ajuste de asientos) y sensores inteligentes (por ejemplo los sensores de lluvia). Varios buses idénticos pueden ser usados si es necesario*."[16]

[15] Bosch, *Automotive Networking*, 2007, pág. 18
[16] Bosch, *Automotive Networking*, 2007, pág. 18

1.6. Modelo de Referencia OSI

La OSI (Interconexión de Sistemas Abiertos) es un estándar regulado por la ISO (Organización de Estandarización Internacional) y la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) y da una base para describir y comparar los protocolos de comunicación. Los protocolos de red en el área automotriz según OSI, son a menudo divididos en tres capas:

- Capa física.
- Capa de comunicación.
- Capa de aplicación.

1.6.1. Capa Física

Son todos los parámetros eléctricos y procedimentales para conectar físicamente los nodos de red. Entre los parámetros están: el nivel de señal, el flujo de datos.

El nivel de señal.- El nivel de señal se refiere al nivel de voltaje que tienen los estados recesivo y dominante. Esto varía entre protocolos por ejemplo, "*la interfaz serial de una PC varía entre 12V y -12V. Mientras que el bus CAN-B usa voltajes entre 0V y 5V. Los voltajes de la interfaz serial no son adecuados para un bus, debido a que un corto circuito puede ocurrir si varios subscriptores desean transmitir estados binarios conflictivos simultáneamente*."[17]

El flujo de datos.- "*Con el fin de hacer posible la transmisión, la información es primero incorporada como una carga útil en la trama de un mensaje que contiene información para ser transmitida. Debido a que todos los protocolos han sido desarrollados en acuerdo con diferentes requerimientos, el formato de trama difiere de protocolo a protocolo*."[18]

Dos ejemplos de formato o codificación de trama son: la codificación NRZ y la codificación *Byphase*.

[17] Bosch, *Automotive Networking*, 2007, pág. 11
[18] Bosch, *Automotive Networking*, 2007, pág. 11

NRZ (*Non-Return-To-zero*).- Esta codificación de datos significa que para la duración total de un bit generado, su nivel permanece constante ya sea recesivo o dominante.

***BiPhase*.-** "*Biphase añade un nivel de complejidad al proceso de codificación, pero a cambio incluye una forma de transferir el clock de datos de trama de bit que se puede utilizar en la descodificación para aumentar la precisión. Un "1" lógico tendrá una transición adicional a mediados de los bits. Esto permite que el sistema de demodulación recuperar la velocidad de datos y también sincronizar a los períodos de bits de borde. Con esta información de reloj, el flujo de datos se puede reconstruir.*"[19]

La figura 1.12 muestra la diferencia entre una trama NRZ y una *Biphase*, con respecto a la frecuencia de un mismo *clock.*

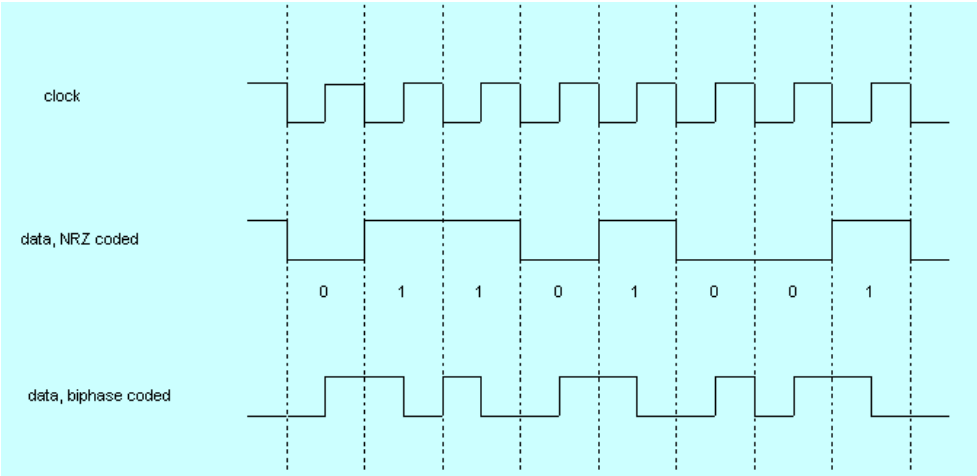

Fig.1.12. Comparación entre una trama NRZ y una *Biphase*, con respecto a la frecuencia de un mismo *clock*
Fuente: Imagen tomada de: Jean-François Fourcadier, http://jf.fourcadier.pagesperso-orange.fr/haut_debit/biphase/biphase_e.htm, junio del 2013

[19] Atmel, http://www.atmel.com/images/doc9164.pdf , junio del 2013

1.6.2. Capa de Comunicación

Las unidades de control o nodos se pueden interconectar e intercambiar datos solamente si hablan el mismo "lenguaje". Este lenguaje determina las reglas que son usadas para intercambiar datos entre los subscriptores de red individuales. La capa de comunicación acepta datos de la capa de aplicación y la prepara para transmitirla hacia la capa física.

"*Las características esenciales de esta capa de protocolo son:*

- *Formato de trama de mensaje*
- *Control de acceso al bus*
- *Direccionamiento de mensajes*
- *Detección y manejo de colisiones*
- *Sincronización del nodo de red*
- *Cálculo del checksum.*"[20]

1.6.3. Capa de Aplicación.

"*Esta capa consiste de las aplicaciones que procesan y proveen información. Es la capa de protocolo que puede ser afectada por el usuario o por un sensor de entrada*."[21]

1.7. Mecanismos de Control

Los mecanismos de control principales para poder transmitir y priorizar mensajes sobre un bus son:

- Controlado por Eventos (*Event Triggered*)
- Controlado por Temporizador (*Time Triggered*)

[20] Bosch, *Automotive Networking*, 2007, pág. 12
[21] Bosch, *Automotive Networking*, 2007, pág. 12

1.7.1. Controlado por Eventos (*Event Triggered*)

Este mecanismo de control de bus es aleatorio, y funciona de la siguiente manera: "*los mensajes son transmitidos tan pronto como un evento que active la transmisión del mensaje haya ocurrido. Ejemplos de tales eventos son:*

- *Presionar un botón en el panel de control del sistema de aire acondicionado.*
- *Accionar el interruptor luces de emergencia.*
- *Mensajes de entrada que requieren reacción (por ejemplo, información del sensor de rpm para la aguja del tacómetro.*"[22]

En el mecanismo *Event Tirggered*, los eventos se pueden transmitir sobre un bus, tomando en cuenta su prioridad. En la figura 1.13 se muestra un gráfico, con dos posibles situaciones que podrían ocurrir en un bus de datos en un mecanismo controlado por eventos. El grafico (a) muestra como 3 eventos se transmiten libremente, ya que el bus no esta saturado. Por otra parte, el gráfico (b) muestra una saturación de eventos en donde pueden transmitir primero, los de más alta prioridad.

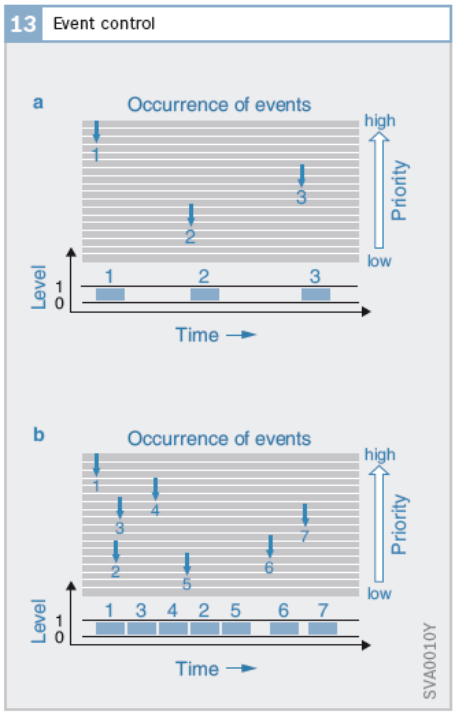

Fig.1.13. Ocurrencia de eventos controlados por mecanismo de eventos.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 12

[22] Bosch, *Automotive Networking*, 2007, pág. 12

1.7.2. Controlado por Temporizador (*Time Triggered*)

Este tipo de mecanismo de control se utiliza en la tecnología *drive-by-wire,* que se refiere a que los mecanismos mecánicos e hidráulicos son remplazados por sistemas electrónicos.

El mecanismo de acceso al bus TDMA es un claro ejemplo de mecanismo *time triggered*, y se utiliza en el reciente bus FLEXRAY que es destinado para tecnología *drive-by-wire.*

En el mecanismo *time triggered,* "*todas las transmisiones son procesadas secuencialmente acorde con un planeamiento de la red (sin colisiones). Además, el tiempo de retraso entre la ocurrencia del evento y la transmisión de los datos pueden ser tan pequeños que los sistemas cumplen con requerimientos de tiempo real estrictos.*"[23]

La confiabilidad, seguridad y los requisitos de tolerancia a fallos son extremadamente altos, esto significa que:

- "*Los mensajes deben ser recibidos a tiempo*
- *Los tiempos de latencia de mensajes críticos deben ser extremadamente pequeños.*
- *El sistema debe tener un diseño redundante.*
- *La falla de un nodo debe afectar el resto de los sistemas tan poco como sea posible.*
- *Debe ser posible alcanzar modo de operación segura de cualquier situación de falla*."[24]

Sin embargo en este tipo de mecanismos debido a su gran eficiencia, aumenta la complejidad en la arquitectura de la red, y en los mecanismos de detección de errores.

[23] Bosch, *Automotive Networking*, 2007, pág. 14
[24] Bosch, *Automotive Networking*, 2007, pág. 14

En la figura 1.14 se puede observar que los eventos solo pueden transmitirse en un tiempo determinado.

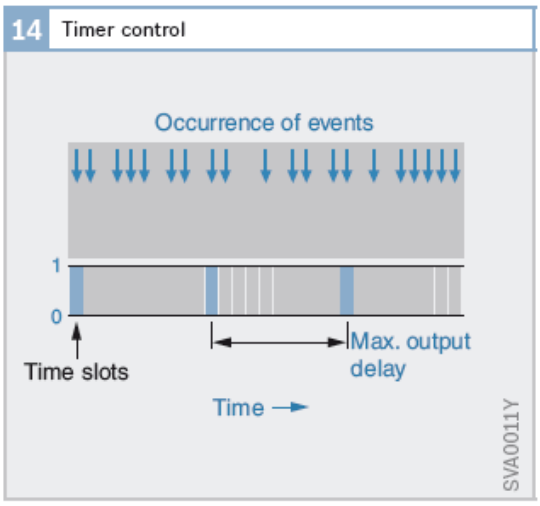

Fig.1.14. Ocurrencia de eventos controlados por mecanismo temporizador.
Fuente: Bosch, *Automotive Networking*, 2007, pág. 8

1.8. Guía de Diagnóstico y Reparación de Redes Automotrices

"*Los códigos de diagnóstico U fueron al principio "indefinidos", pero ahora son códigos relacionados a la red. Utilicé los códigos para ayudar a identificar el circuito o módulo que no está trabajando correctamente*."[25]

Halderman (2012) establece que cuando se sospeche de una falla de red, se lleve a cabo los siguientes pasos:

Paso 1: Revise cualquier cosa que funcione o no funcione.- A menudo los accesorios que no parecen estar conectados pueden ayudar a identificar cual modulo o circuito del BUS esta fallando.

Paso 2: Realice la prueba de estado del módulo.- Use un escáner de fábrica o un escáner del mercado equipado con software mejorado que permita las funciones *OE-like*. Revise si los componentes o sistemas pueden ser operados a través del escáner.

Paso 3: Revise los valores de resistencia de las resistencias terminales.- La mayoría de los sistemas bus de alta velocidad usan resistencias en cada terminal, llamadas resistencias terminales. Estos resistores son usados para ayudar a reducir la interferencia dentro de otros sistemas en el vehículo. Usualmente dos resistencias de 120 ohm son instaladas en cada terminal y además conectadas eléctricamente en paralelo. Las dos resistencias conectadas en paralelo medirían 120Ω si se las prueba con multímetro.

[25] Halderman, *Automotive Technology*, 2012, pág. 530

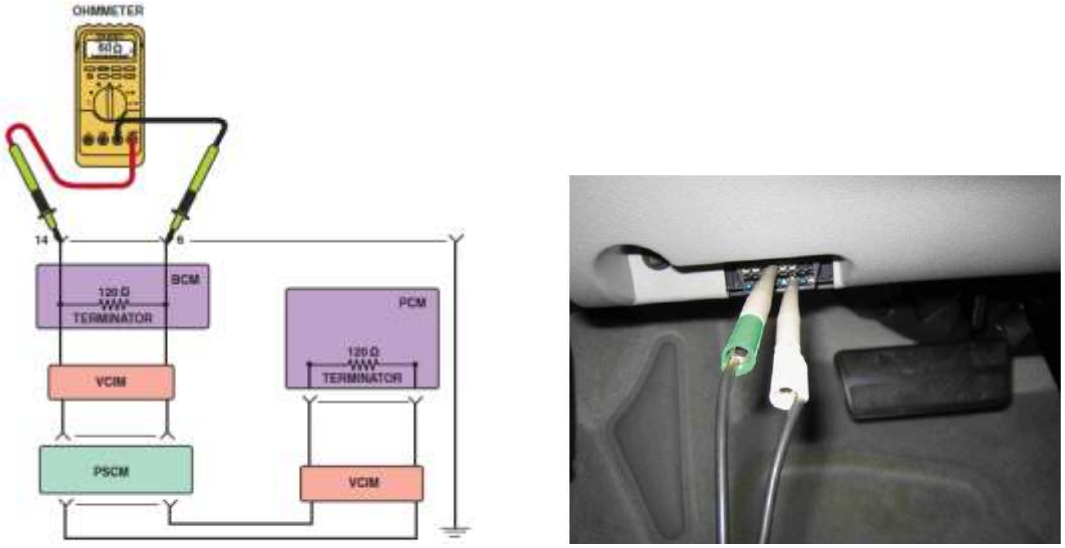

Fig.1.15. Medición de la resistencia en los terminales de una red.
Fuente: Halderman, *Automotive Technology*, 2012, pág. 535

Paso 4: Revise los voltajes del BUS de datos.- Use un multímetro digital y establezca en volteos DC, para monitorear la comunicación y revisar la operación correcta del BUS. Algunas condiciones del BUS y posibles causas incluyen:

- **La señal es cero voltios todo el tiempo.-** Revise un cortos a tierra desenchufando los módulos uno a la vez para revisar si un módulo está causando el problema.
- **La señal es alta o 12V todo el tiempo.-** el circuito BUS podría estar cortado a 12V. Revise con el cliente para ver si algún servicio o trabajo de reparación de la carrocería fue hecha recientemente. Intente desconectar cada módulo uno a la vez para precisar cual modulo está causando el problema de comunicación.
- **Un voltaje variable usualmente indica que los mensajes están siendo enviados y recibidos.-** CAN y clase 2 pueden ser identificados observando en el conector DLC (*Data Link Connector*) por el terminal en el pin numero 2. La clase 2 está activa todo el tiempo cuando el encendido está en ON, y además la variación del voltaje entre 0 y 7V pueden ser medidas usando una configuración DDM para leer voltios DC.

Paso 5: Utilice un osciloscopio de almacenamiento digital para monitorear las formas de onda del circuito BUS.- Usando un osciloscopio en los terminales de la línea de datos, se puede mostrar si la comunicación está siendo transmitida. Típicamente las fallas y sus causas incluyen:

- **Operación Normal.-** Una operación normal muestra señales de voltaje variable en la línea de datos. Es posible saber qué información está siendo transmitida, pero si hay actividad con secciones cortadas de inactividad.

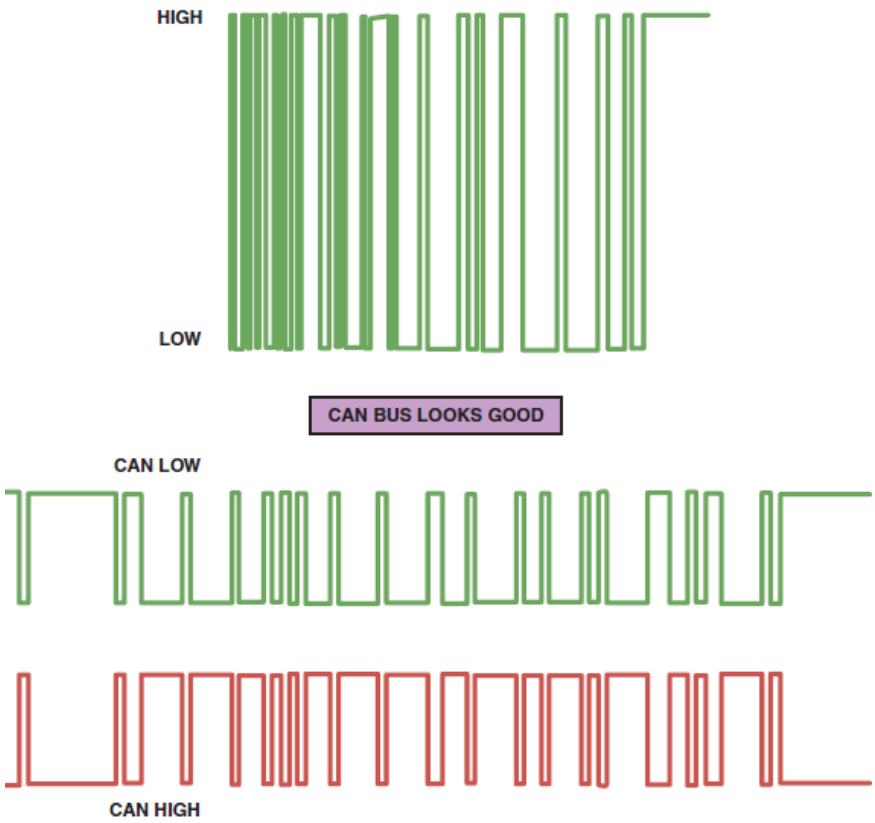

Fig.1.16. Trama normal de un bus CAN.
Fuente: Halderman, *Automotive Technology*, 2012, pág. 536

- **Alto Voltaje.-** Si hay una señal de alto voltaje constante sin ningún cambio, indica que la línea de datos está cortada a voltaje.
- **Cero o Bajo Voltaje.-** Si el voltaje de la línea de datos es cero o casi cero y no está mostrando ninguna señal de voltaje alto, se debe a que la línea de datos está cortada a tierra.

Paso 6: Siga las instrucciones del manual de servicio de fábrica para aislar la causa de la falla.- Este paso a menudo involucra desconectar un módulo a la vez, para ver si es la causa un corto a tierra o un circuito abierto en el circuito del BUS.

Reparación del Cable BUS

"*Si el bus necesita ser reparado debido a un circuito abierto, corto(a voltaje o masa), o alta resistencia, éste no debe ser reubicado o des-trenzado. El trenzado sirve para un propósito muy importante. Después que un cable bus ha sido reparado por soldadura, envuelva esa parte del cable con cinta de vinilo. Nunca haga funcionar el alambre de reparación si hay secciones no trenzadas. Cables de bus CAN son propensos a ser influenciados por el ruido si se omiten los cables trenzados*."[26]

[26] Erjavec, *AUTOMOTIVE TECHNOLOGY: A Systems Approach, 5e,* 2010, pág. 663

CAPÍTULO 2

TIPOS DE REDES MULTIPLEXADAS EN EL CAMPO AUTOMOTRIZ

2.1. Introducción

Debido a las diferentes necesidades es cada uno de los sistemas del vehículo, La Sociedad de Ingenieros Automotrices (SAE) clasificó a los buses acorde a su velocidad de transferencia de datos en clases A, B, C y D. En presente capitulo se va ha comparar todas estas clases tomando en cuenta características de funcionamiento, aplicaciones, ventajas y desventajas de cada una de las clases. Además se va ha comparar dentro de cada clase los protocolos mas comunes, con el fin de comparar las características de los mismos.

2.2. Redes Clase A

2.2.1. Características de Funcionamiento

Los buses de clase A son de baja velocidad, y su máxima velocidad de transferencia de datos es 10kbits/s. Hollembeak (2011) define que: "*los buses clase A prácticamente son protocolos genéricos del UART (Universal Asynchronous Receiver/Transmitter)*" (pág. 284).

El más común de esta clase de buses es el LIN que es usado por un gran número de marcas automotrices. Sin embargo también analizaremos dos buses más, el BEAN (de Toyota) y el UART (de General Motors), con el fin de comparar sus características.

Buses:

LIN.- El bus LIN (*local Interconnect Network*) fue desarrollado por un consorcio fundado por varias fábricas automotrices. El consorcio LIN busco desarrollar una estándar para un bus serial para la interconexión de sensores, actuadores en toda el área electrónica de la carrocería.

El bus LIN típicamente usa la configuración maestro-esclavo. El master generalmente es una la unidad de control electrónica conectado a un sistema de bus subordinado, y los esclavos. Los esclavos generalmente son actuadores inteligentes, sensores inteligentes o simplemente *switches* con hardware adicional para la interface del bus LIN.

El bus LIN trabaja en un rango de voltaje de 0 a 12V. Su gráfico es característico es como se muestra en la figura 2.1.

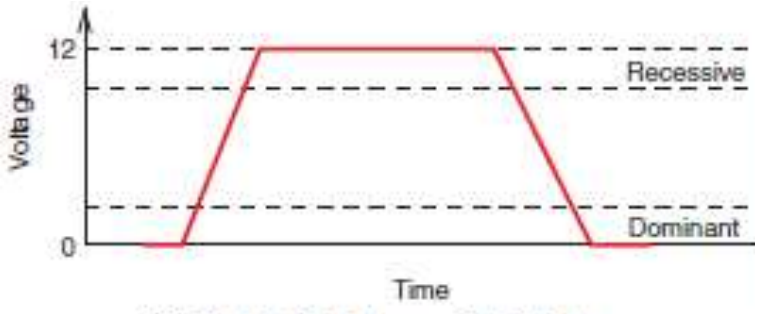

Fig.2.1. Niveles de voltaje del bus LIN.
Fuente: Hollembeak, *Classroom Manual for Automotive Electricity And Electronics*, 2011, pág. 298

El bus LIN además funciona de manera de tiempo sincrónico, donde el master define los cuadros de tiempo. Consecuentemente, allí surge una respuesta del bus LIN estrictamente determinística. En la figura 2.2 se puede observar las aplicaciones del bus LIN, que generalmente trabaja como un bus complementario al CAN.

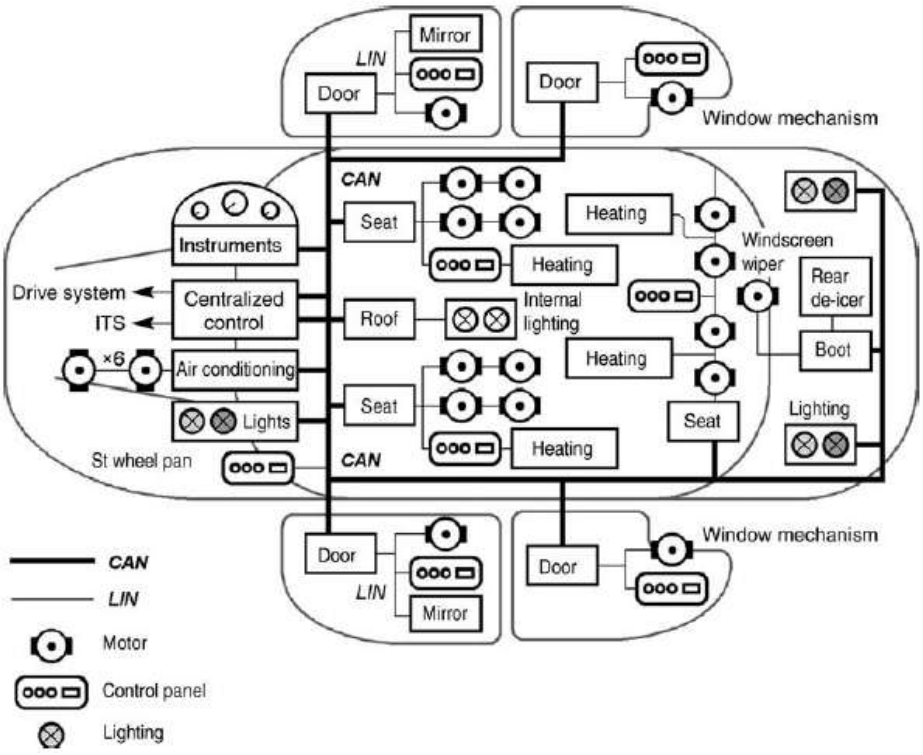

Fig.2.2. Diagrama que relaciona las líneas CAN y LIN.
Fuente: Paret, Multiplexed Networks for Embedded Systems, 2007 , pág. 286

BEAN.- El bus BEAN (*Body Electronics Area Network)* es un bus original de la marca Toyota. Éste bus se usa para el sistema eléctrico de la carrocería. La configuración típica del BEAN es de tipo anillo (*Daisy Chain* denominada por Toyota), en la cual todas las ECUs están interconectadas a la ECU de entrada (Gateway) en círculo, todas entre sí. Este tipo de red mantiene la comunicación aun sí el arnés de cables tiene un circuito abierto. El bus BEAN trabaja en un rango de voltaje de 0 a 10V.

En la figura 2.3 se muestra una configuración típica del bus BEAN, en donde se observa que la ECU de entrada (Gateway) esta interconectada al resto de ECUs en círculo. La ECU de entrada sirve como como interfaz que comunica el bus BEAN con el CAN y el AVC-LAN, con el fin que se pueda intercambiar información entre todos estos buses.

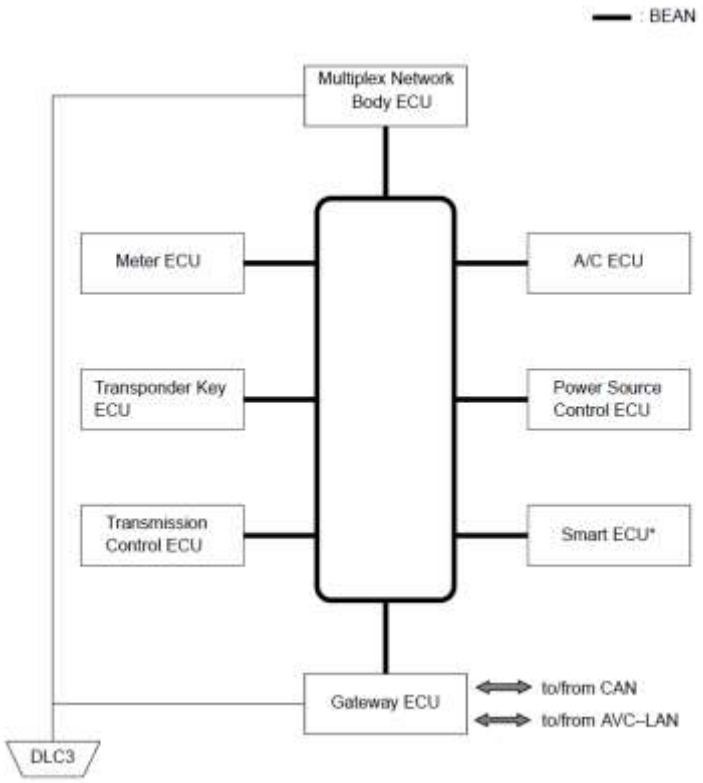

Fig.2.3. Configuración típica de un bus BEAN
Fuente: Manual de Servicio Toyota Prius, *MULTIPLEX COMMUNICATION SYSTEM,* 2004

UART. - El bus UART (*Universal Asynchronous Receiver Transmitter*) es utilizado por los vehículos de General Motors para algunos módulos o sistemas electrónicos UART usa una configuración de módulos maestro-esclavo. "*El módulo master es usado para controlar el tráfico de mensajes sobre la línea de datos por polarización de todos los otros módulos UART. Los módulos remotos envían un mensaje de respuesta de vuelta al módulo maestro*."[27]

El bus UART trabajo en un rango de voltaje de 0 a 5V. Además una característica de este bus es que trae su propio puerto de diagnostico directo al conector de vinculo de datos (DLC). En los vehículos General Motors que equipan UART, viene habilitado el pin 9, que es el pin de diagnostico del bus.

En la figura 2.4 se puede observar el gráfico característico del bus UART y el pin de diagnostico en el DLC.

[27] Halderman, *Automotive Technology*, 2012, pág. 527

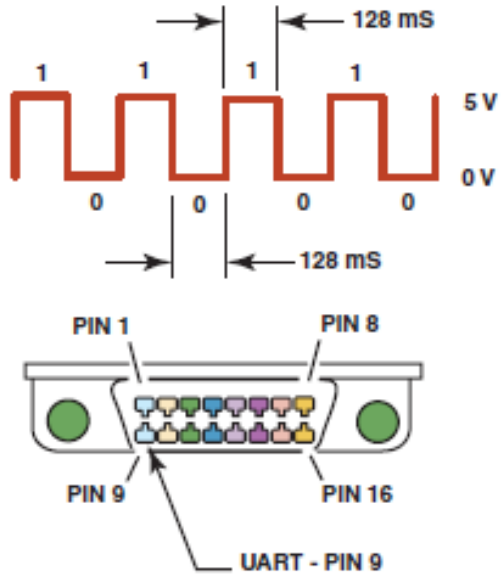

Fig.2.4. Niveles de voltage y pines de diagnostico del bus UART.
Fuente: Halderman, *Automotive Technology*, 2012, pág. 527

A continuación una tabla comparativa de los buses clase A tratados:

	NOMBRE DEL BUS		
CARACTERÍSTICAS	**LIN**	**BEAN**	**UART**
AFILIACIÓN	Motorola	Toyota	General Motors
APLICACIÓN	Sensores Inteligentes	Control de Chasis y Diagnóstico	General y Diagnóstico
MEDIO FÍSICO	1 cable	1 cable	1 cable
BIT DE CODIFICACIÓN	NRZ	NRZ	NRZ
ACCESO DE MEDIOS	Maestro/Esclavo	Contención	Maestro/Esclavo
DETECCIÓN DE ERRORES	*Checksum* de 8bits	CRC de 8bits	*Checksum* de 8bits
LONGITUD DE CABECERA	2 bits por byte	25 bits	16 bits
LONGITUD DE DATOS	8 bytes	1 a 11 bytes	0 a 85 bytes
MENSAJE DE CABECERA	2 BYTES	28%	Variable
VELOCIDAD	1 - 20 kbits/s	10 kbits/s	8192 bits/s
LONGITUD MÁXIMA DEL BUS	40 metros	No especifica	No especifica
NODOS MÁXIMOS	16	20	10
SLEEP/WAKEUP	Si	No	No
COSTO	Bajo	Bajo	Bajo

Tabla 2.1. Comparación de buses clase A[28]

2.2.2. Aplicación en el Vehículo

Bosch (2007) destaca las siguientes aplicaciones de los buses clase A:

- Módulo de puertas con seguro de puertas, unidad de elevalunas eléctrica, y ajuste de retrovisores.
- Unidad de control del *sunrooof.*
- Control del motor del limpia parabrisas.
- Sensor de lluvia y detección de luz.
- Sistema de aire acondicionado (transmisión de señales del elemento de control, activación del ventilador de aire fresco).
- electrónica de los faros.

[28] Tabla tomada de: Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 2

- Control de motores para ajuste de asientos.
- Sistema de Antirrobo.
- Abridor de puerta de garaje

2.2.3. Ventaja y Desventajas

Ventajas

- Bajo costo. "*Como referencia X por nodo. Incluye silicón (usado en modulo microprocesador o transductor), software, conector pin y servicio. El valor es muy crudo, se usa solo como referencia*."[29]
- "*La interfaz eléctrica puede ser creada fácilmente y de bajo costo en los nodos de la red.*"[30]
- Usa microcontroladores de baja capacidad sin hardware adicional para la interface de comunicación.

Desventajas:

- Baja velocidad
- No cumple con requisitos de tiempo real
- Se aplica solo como complementos de buses principales

[29] Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 2
[30] Bosch, *Automotive Networking*, 2007, pág. 44

2.3. Redes Clase B

2.3.1. Características de Funcionamiento

Los buses de clase B son de media velocidad, y su velocidad de transferencia de datos esta entre 10kbits/s y 125kbits/s. El estándar de esta clase es el bus CAN-B o CAN de baja velocidad, que es usado por la mayoría de los vehículos modernos de media y alta gama. Sin embargo, se va ha describir también el bus J1850 que es usado por Ford, General Motors y Daimler Chrysler.

Buses

CAN de baja velocidad (CAN-B).- El bus CAN es el sistema estándar en el sector automotriz. El bus CAN tiene varios dominios en el vehículo, el de baja velocidad es requerido en el área de confort y comodidad. Su estándar es el ISO 11898-3 y opera en un rango de velocidad de entre 5 a 125 kbits/s, esta velocidad es suficiente para reunir requerimientos de tiempo real flexible, demandadas en esta área.

El bus CAN-B trabaja dentro de un rango de voltaje de 0 a 5V y en la figura 2.5 se observa su gráfico característico:

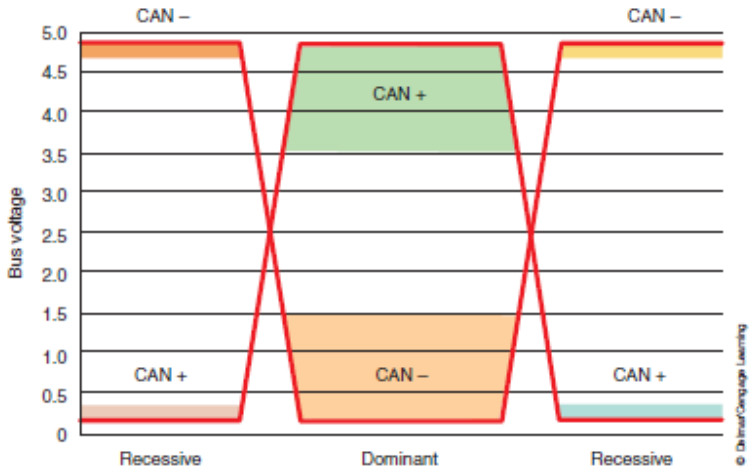

Fig.2.5. Niveles de voltaje en el bus CAN-B.
Fuente: Hollembeak, *Classroom Manual for Automotive Electricity And Electronics*, 2011, pág. 296

Bosch (2007) establece que los nodos en la red CAN están compuestos por los siguientes bloques:

- Microcontrolador con su software de aplicación.- Es él encargado de correr su software de aplicación (ejemplo: Bosch EDC), controla al controlador CAN, prepara los datos para ser enviados y evalúa los recibidos.
- El controlador CAN.- Es el encargado de los modos de transmisión y recepción. Genera la corriente de datos y los envía por la línea TxD. La línea RxD recibe los datos desde el transductor
- Y el transceptor CAN.- Este amplifica la señal a un nivel de voltaje requerido por la línea de bus (CAN_H y CAN_L).

En la figura 2.6 se observa los tres bloques que componen un nodo en bus CAN

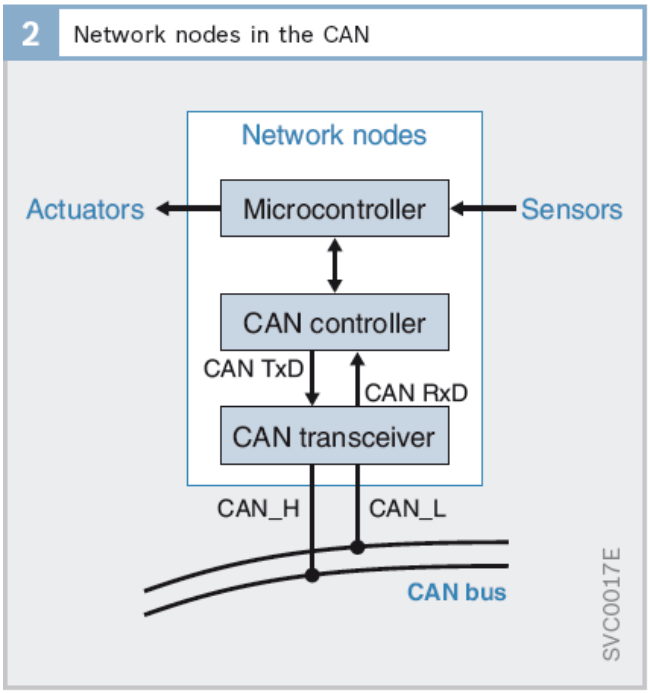

Fig.2.6. Elementos de un nodo CAN
Fuente: Bosch, *Automotive Networking*, 2007, pág. 31

J 1850.- Este bus es usado por las marcas General Motors, Ford y Chrysler. Este bus es para uso general en el sistema eléctrico del vehículo. El bus J1850 tiene dos protocolos el J1850 VPW y el J1850 PWM. Estos dos protocolos se diferencian principalmente por su capa física, el J1850 VPW utiliza como medio de transmisión un cable, mientras que el J1850 PWM utiliza como medio de transmisión dos cables.

El bus J1850 además se usa para el diagnostico, ya que tienen su pin de diagnostico en el DLC.

A continuación una tabla comparativa de los buses clase B tratados:

	NOMBRE DEL BUS			
CARACTERÍSTICAS	**CAN 2.0** **ISO 11898** **ISO 11519-2** **ISO 11992** **J2284**	**J 1850**		
AFILIACIÓN	Bosch/ISO/SAE	GM	Ford	Chrysler
APLICACIÓN	Control y Diagnóstico	General y Diagnóstico	General y Diagnóstico	General y Diagnóstico
MEDIO FÍSICO	2 cables trenzados	1 cable	2 cables trenzados	1 cable
BIT DE CODIFICACIÓN	NRZ-5	VPW	PWM	VPW
ACCESO DE MEDIOS	Contención	Contención	Contención	Contención
DETECCIÓN DE ERRORES	CRC	CRC	CRC	CRC
LONGITUD DE CABECERA	11 o 29 bits	32 bits	32 bits	8bits
LONGITUD DE DATOS	0-8 bytes	0-8 bytes	0-8 bytes	0-10 bytes
SOBRE CABEZA	9.9 % - 22 %	33.3%	33.3%	8.3%
VELOCIDAD	10 kbits/s a 1 Mbit	10.4 kbits/s	41.6 kbits/s	10.4 kbits/s
LONGITUD MÁXIMA DEL BUS	No especifica (típico 40m)	35m (5m para el puerto de diagnóstico)	35m (5m para el puerto de diagnóstico)	35m (5m para el puerto de diagnóstico)
NODOS MÁXIMOS	No especifica 32 (típico)	32	32	32
SLEEP/WAKEUP	No	Si	No	No
COSTO	Medio	Bajo	Bajo	Bajo

Tabla 2.2. Comparación de buses clase B[31]

[31] Tabla tomada de: Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 2

2.3.2. Aplicación en el Vehículo

Bosch (2007) destaca las siguientes aplicaciones de los buses clase B:

- Control del sistema de aire acondicionado
- Ajuste de asientos
- Unidad de vidrios eléctricos
- Control de deslizamiento de techo corredizo
- Ajuste de retrovisores
- Sistema de alumbrado

2.3.3. Ventajas y Desventajas

Ventaja:

- Velocidad Media
- Mejor Desempeño

Desventajas:

- Costo relativamente alto en relación a sus aplicaciones. "*Como referencia 2X por nodo.*"[32]
- Software complejo

[32] Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 3

2.4. Redes Clase C

2.4.1. Características de Funcionamiento

Los buses de clase C son de alta velocidad, y su velocidad de transferencia de datos esta entre 125kbits/s y 1Mbit/s. En esta clase el bus estándar es el CAN de alta velocidad, y no tiene rivales significativos en esta clase.

CAN de alta velocidad.- El estándar de este dominio de CAN es el ISO 11898-2, y opera en un rango de datos de entre 125kbits/ y un 1Mbit/s. La transferencia de datos es además capaz de reunir requerimientos de tiempo real flexible para tren de potencia.

El bus CAN C trabaja en un rango de voltaje de 1.5 a 2.5V y de 2.5 a 3.5V.

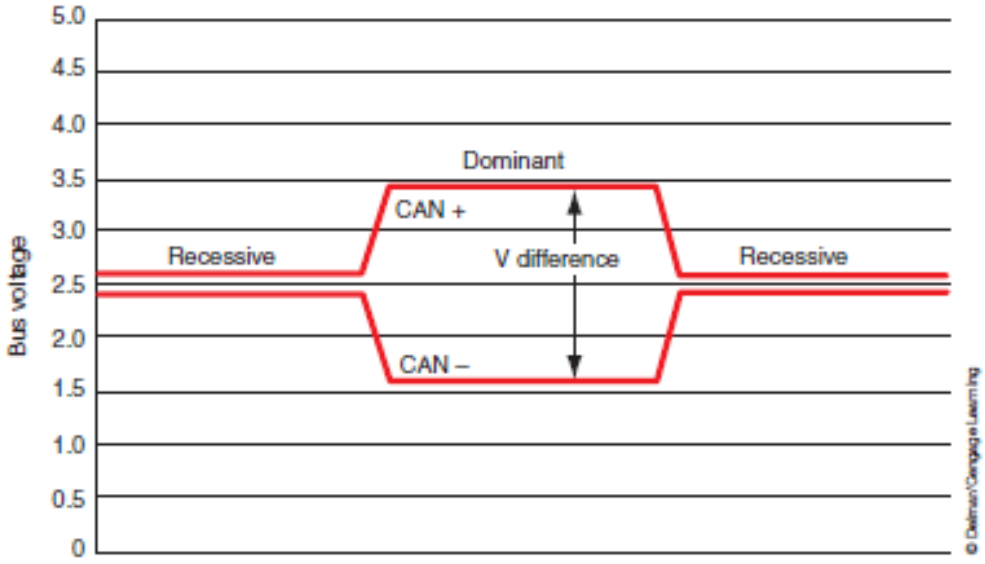

Fig.2.7. Niveles de voltaje en el bus CAN-C.
Fuente: Hollembeak, *Classroom Manual for Automotive Electricity And Electronics*, 2011, pág. 295

Paret (2007) destaca sus principales características que se describen a continuación:

- La longitud máxima del bus es 40m.
- El rango de nodos participantes para la red está entre 2 a 30 nodos.
- La capa física es un par de cables trenzados con retorno a tierra.
- una característica de impedancia de la línea de 120Ω.
- Una resistencia del cable de 70mΩ/m

- Un tiempo de propagación de señal nominal de 5ns/m sobre el bus.
- Una corriente de salida de más de 25 mA suministrada por los transmisores.
- Una línea protegida de cortos circuitos (de 3 a 16 o 32V)
- un rango de modo común de -2 a +7 V, fuente de alimentación simple bajo 5 V. (pág. 132)

2.4.2. Aplicación en el Vehículo

Bosch (2007) destaca las siguientes aplicaciones de los buses clase C:

- Sistemas de gestión del motor (por ejemplo, *motronic* para motores de gasolina o EDC para motores de diesel).
- Control de transmisión electrónica.
- Sistema de estabilización del vehículo.
- Panel de instrumentos.

2.4.3. Ventajas y Desventajas

Ventajas:

- Reúne la característica de tiempo real flexible.
- Permite una conexión de un numero de nodos relativamente alto

Desventajas:

- Costo relativamente alto. "*Varía de 3 a 4X por nodo*."[33]
- Software bastante complejo.

[33] Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 4

2.5. Redes Clase D

2.5.1. Características de Funcionamiento

Los buses de clase D son de muy alta velocidad, sus velocidades de transferencia de datos esta sobre los 10Mbits/s. Los Buses más comunes son el MOST y el IEEE 1394.

Buses

MOST.- El bus MOST (*Media Oriented System Transport*) fue específicamente desarrollado para la interconexión de sistemas *infotainment* (Sistemas de Entretenimiento de audio y video).

Este bus es usado por los vehículos de gama alta en las marcas "*BMW, General Motors, Ford, Volkswagen y Toyota*."[34]

El bus MOST fue originalmente definido como un agente de transmisión óptica que usa fibra óptica plástica (POF). La señal óptica es generada por un LED en el lado de transmisión designado con Tx-FOT (FOT= transceptor de fibra óptica). En el lado receptor, Rx-FOT, la señal óptica es convertida a señal eléctrica por un fotodiodo.
En la figura 2.8 se observa la configuración de un nodo MOST.

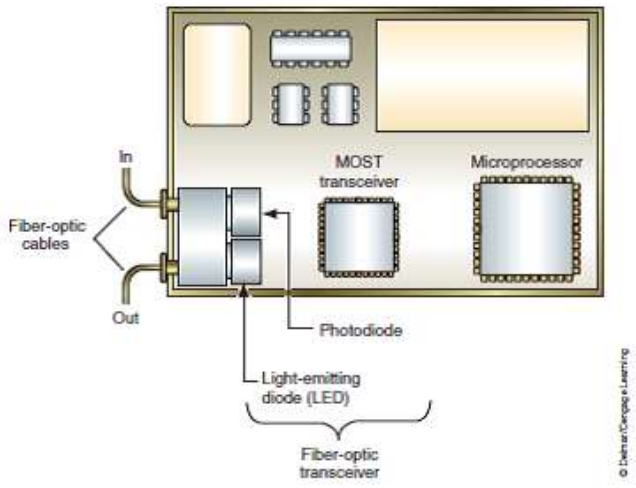

Fig.2.8. Elementos de un nodo en un bus MOST.
Fuente: Hollembeak, *Classroom Manual for Automotive Electricity And Electronics*, 2011, pág. 298

[34] Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 8

"*Los cables POF usados para aplicaciones automotrices consiste de un núcleo óptico grueso aislado de 980µm por un revestimiento óptico grueso de 20µm con un bajo índice de refracción, en total el conductor óptico tiene un diámetro de 1mm. La fibra óptica es aislada con un buffer negro, que es revestido por un protector de cable. Esto da al cable un diámetro total de 2.3mm*."[35]

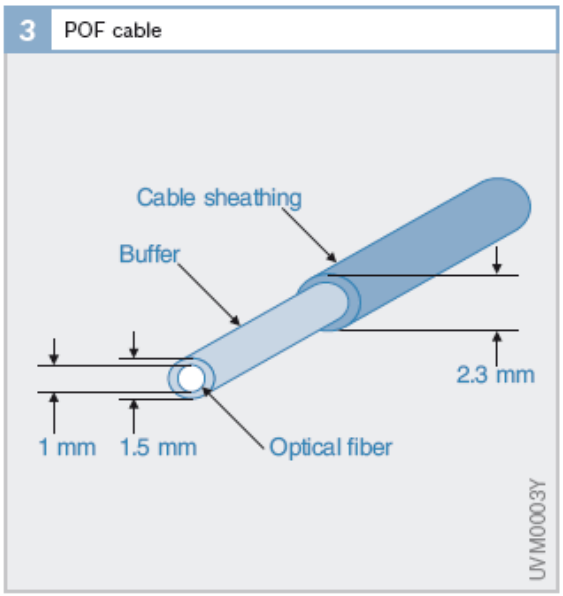

Fig.2.9. Constitución de un cable de fibra óptica POF
Fuente: Bosch, *Automotive Networking*, 2007, pág. 62

IEEE1394 o Firewire.- el bus IEEE 1394 tambien es un bus para sistemas de audio y video de abordo, ademas este bus es el competidor directo del bus MOST. La clave para el éxito del bus 1394 es el apoyo de la poderosa *Automotive Multimedia Interface Collaboration* (AMIC), un grupo de la industria que incluye 12 de los mayores fabricantes de automóviles del mundo, como son: BMW, DaimlerChrysler, Ford, Fiat, General Motors, Honda, Mitsubishi, Nissan, PSA Peugeot-Citroen, Renault, Toyota y Volkswagen.

"*La nueva especificación 1394 CU permite a los fabricantes de automóviles a utilizar los medios de cobre, solos o en combinación con la fibra óptica, en función de sus necesidades. Es un avance significativo en la tecnología del automóvil, permitiendo que la información, como la navegación de alta resolución, entretenimiento y características de seguridad que deben aplicarse en una sola red extensible con un excelente rendimiento. Otras tecnologías de red más lenta de*

[35] Bosch, *Automotive Networking*, 2007, pág. 62

automóviles trabajan en areas específicos. Sólo el estándar Automotriz 1394 ofrece una solución única, flexible y asequible para todas las necesidades y aplicaciones."[36]

A continuación una tabla comparativa de los buses clase D tratados:

	NOMBRE DEL BUS	
CARACTERÍSTICAS	**MOST**	**IEEE1394 o *Firewire***
AFILIACIÓN	OASIS	IEEE
APLICACIÓN	Flujo de Datos & Control	Dispositivos de PC
MEDIO FÍSICO	Fibra óptica	2 cables trenzados blindados
BIT DE CODIFICACIÓN	*BiPhase*	NRZ
ACCESO DE MEDIOS	Maestro/Esclavo	Contención
DETECCIÓN DE ERRORES	CRC	CRC
LONGITUD DE CABECERA		25 bits
LONGITUD DE DATOS	8 bytes	1 a 11 bytes
VELOCIDAD	POF= 25-150 Mbits/s	98-393 Mbits/s Fujitsu MB88395 = 800 Mbits/s[37]
LONGITUD MÁX. DEL BUS	TBD(para ser determinado)	72 metros
NODOS MÁXIMOS	24	16
SLEEP/WAKEUP	Si	No
COSTO	Alto	Medio

Tabla 2.3. Comparación de buses clase D[38]

[36] 1394 Trade Association, http://www.1394ta.org/press/TAPress/2008_0725.html, junio del 2013
[37] 1394 Trade Association, http://www.1394ta.org/press/TAPress/LookWhat1394AutomotiveCanDo.html, junio del 2013
[38] Tabla tomada de: Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 2

2.5.2. Aplicación en el Vehículo

Bosch (2007) destaca las siguientes aplicaciones de los buses clase D:

- Amplificador de Audio.
- Auxiliar de entrada. Interfaz para conectar mp3 *players.*
- Entrada de micrófono.
- Toca cintas.
- CD *player* o cargador de CD.
- DVD *player.* o cargador de DVD.
- Receptor radio Am/Fm.
- TMC *tuner* (receptor especial para señales de mensajes de trafico).
- Receptor de TV.
- DAB *tuner* (receptor de radios digitales).
- SDARS (receptor para radio satélite).
- Módulo de teléfono o una conexión a teléfono celular.
- *General Phone Book* (acceso a contacto de teléfono).
- Sistema de Navegación (Sólo las interfaces que estén aprobadas por la cooperación MOST).
- Pantalla de Gráficos.

2.5.3. Ventajas y Desventajas

Ventajas:

- Alta resistencia a la interferencia.
- Envía una considerable cantidad de datos a una muy alta velocidad.

Desventajas:

- Costo excesivamente alto. "*Como referencia esta entre 15X a 25X por nodo, debido a la fibra óptica*."[39]

[39] Lupini, *Multiplex Bus Progression 2003*, 2003, pág. 6

CAPÍTULO 3

SIMULACIÓN DE UN ESQUEMA ELECTRÓNICO DE RED

3.1. Introducción

En el presente capitulo se va simular una red multiplexada prototipo para aplicar al sistema eléctrico de las puertas del vehículo. La red constará de cuatro módulos electrónicos, el módulo master (módulo de la puerta) más tres módulos subordinados. Los módulos subordinados serán la unidad de vidrios eléctricos, la unidad de retrovisores eléctricos, y la unidad de seguros eléctricos.

En la figura 3.1 se ilustra esquemáticamente la configuración de los módulos del sistema multiplexado de las puertas.

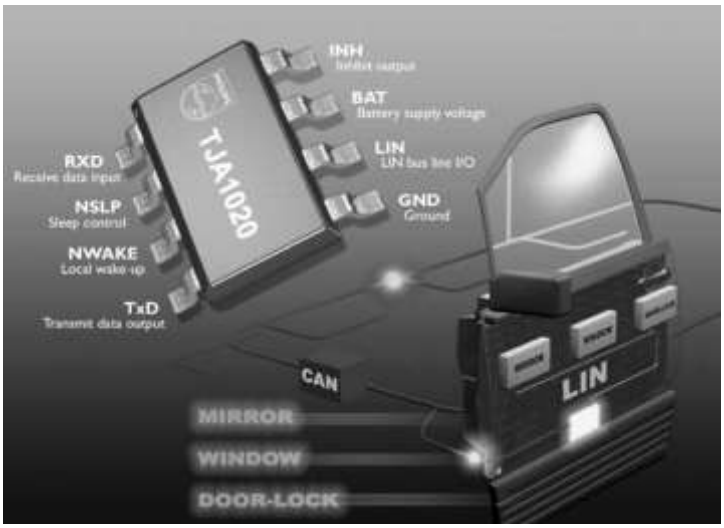

Fig.3.1. Configuración de módulos de un sistema multiplexado en las puertas de un vehículo
Fuente: Paret, Multiplexed Networks for Embedded Systems, 2007 , pág. 287

Para crear la red prototipo, en cada modulo electrónico, se utilizará un microcontrolador AVR ATmega8. El primer ATmega8 se usará como master y controlará los otros tres ATmega8 subordinados, por medio del puerto USART. Además se utilizará tres motores DC de 12V, conectados a cada uno de los nodos subordinados. El primer motor generará el movimiento de apertura y cierre del seguro eléctrico de las puertas, el segundo motor generará el movimiento de subida y

bajada de la ventana eléctrica, y el tercer motor generará el movimiento del retrovisor eléctrico. Además, cada motor constará de un circuito que invierta el giro del mismo (horario-anti horario), para simular la apertura y cierre de la ventana, del seguro, o el movimiento del retrovisor según sea el caso.

Adicionalmente en el módulo de la ventana eléctrica usará dos *switches* (SW30 y SW40), que simularán los fines carreras de la ventana (superior e inferior). De la misma manera, el módulo del retrovisor eléctrico también usará dos *switches* (SW50 y SW60), que simularán los fines carreras del retrovisor (izquierda o derecha). Tanto el módulo de la ventana como el módulo del retrovisor informarán al modulo master que la ventana o retrovisor a llegado a su tope (superior, inferior, izquierda o derecha). El módulo master estará conectado a un grupo de *switches* (PF1, PR1, PF2, PR2, PF3 y PR3), que controlarán el giro de los motores (horario y anti horario) de cada uno de los módulos subordinados.

Además el modulo master estará conectado a un LCD, cuya función será presentar los datos que envíen los módulos subordinados, informado que la ventana ha llegado a su tope o que los seguros de las puertas ya están accionados, mediante los mensajes: “ventana cerrada”, “ventana abierta”, “seguro cerrado” y “seguro abierto”.

Finalmente se utilizará dos fuentes de alimentación, una de 12V para alimentar los motores DC y otra de 5V para alimentar los microcontroladores.

En la figura 3.2 se ilustra todo el conjunto de elementos electrónicos descritos anteriormente.

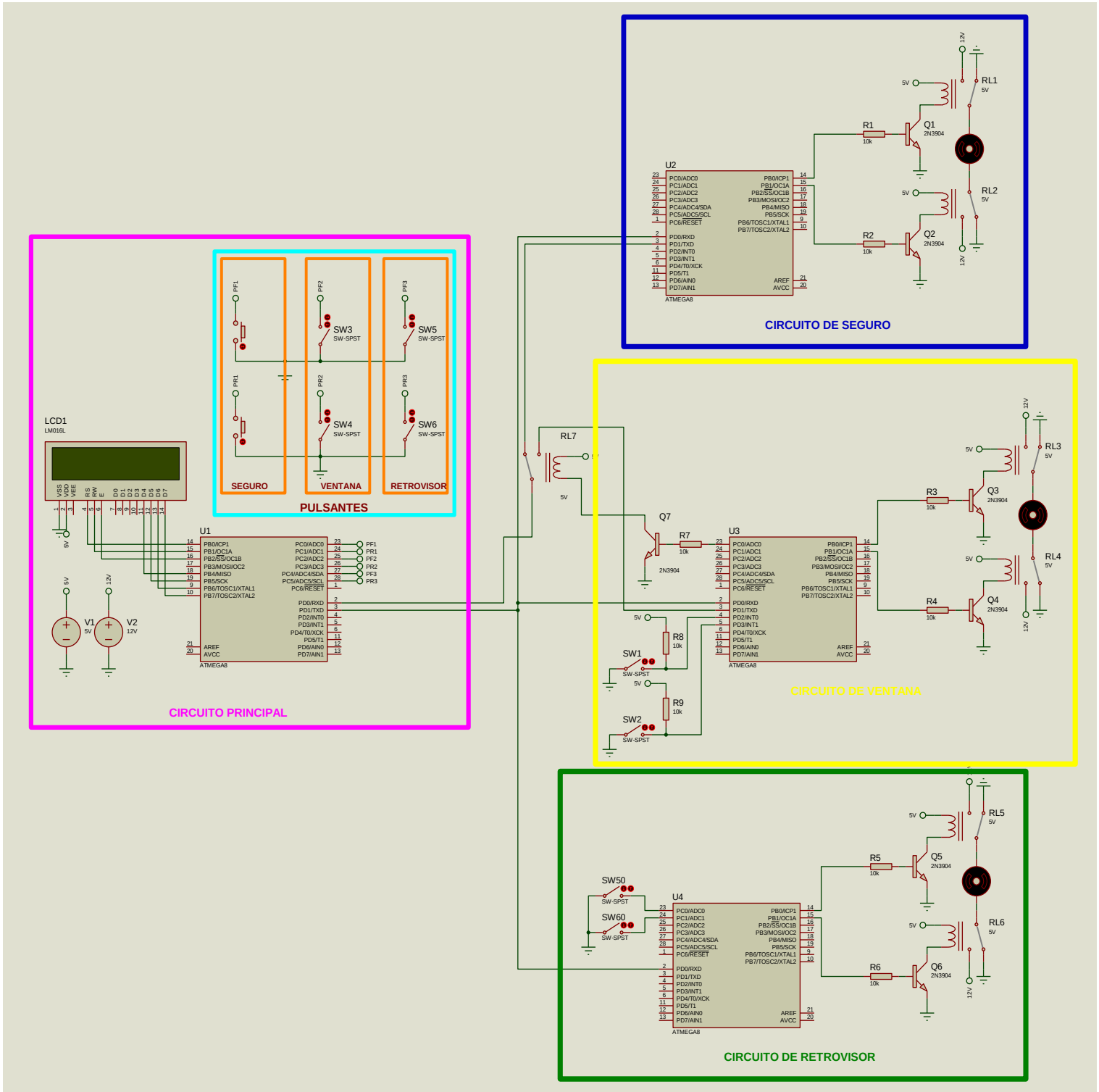

Fig.3.2. Esquema electrónico de un sistema multiplexado de las puertas del vehículo armado en el software ISIS Proteus 7.9. Ver esquema ampliado en ANEXO 5

3.2. Programación de los Microcontroladores

La codificación de los microcontroladores AVR ATmega8 se realizará en el *CodeVisionAVR V2.05.0* debido que es un *software* muy práctico y además presta la ayuda del *CodeWizardAVR* que genera las librerías de muchas funciones de manera automática.

3.2.1. Programación del Nodo Maestro

Para la codificación del nodo Maestro, se sigue los siguientes pasos:

- Primero, se corre el software *CodeVisionAVR V2.05.0.*
- Luego, se crea un nuevo proyecto con el *CodeWizardAVR.*
- Se Configura las pestañas del *CodeWizardAVR* acorde a la figura 3.3.

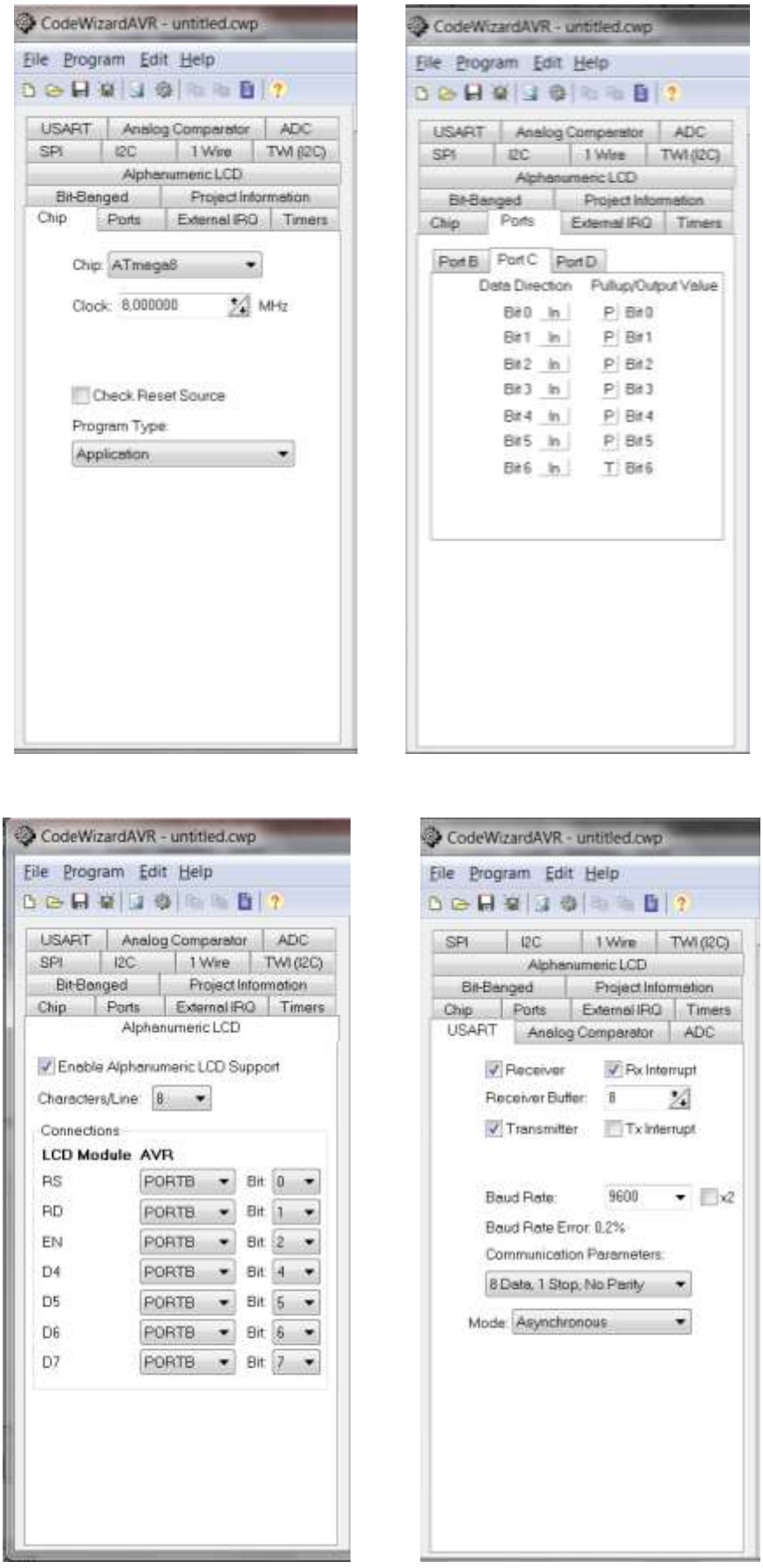

Fig.3.3. Configuración de las pestañas del *CodeWizardAVR*

- Se programa mediante lenguaje C (modificado para microcontroladores AVR), la siguiente codificación:

  Véase **ANEXO 1**

- Finalmente se presiona las teclas Ctrl+F9 para compilar la codificación y generar el archivo *.hex* que mas adelante se cargará en el microcontrolador master.

3.2.2. Programación de los Nodos Esclavos

3.2.2.1. Programación del Nodo de Seguro Eléctrico

Para la programación del nodo esclavo correspondiente al seguro eléctrico, se configura el *CodeWizardAVR* acorde a la figura 3.4.

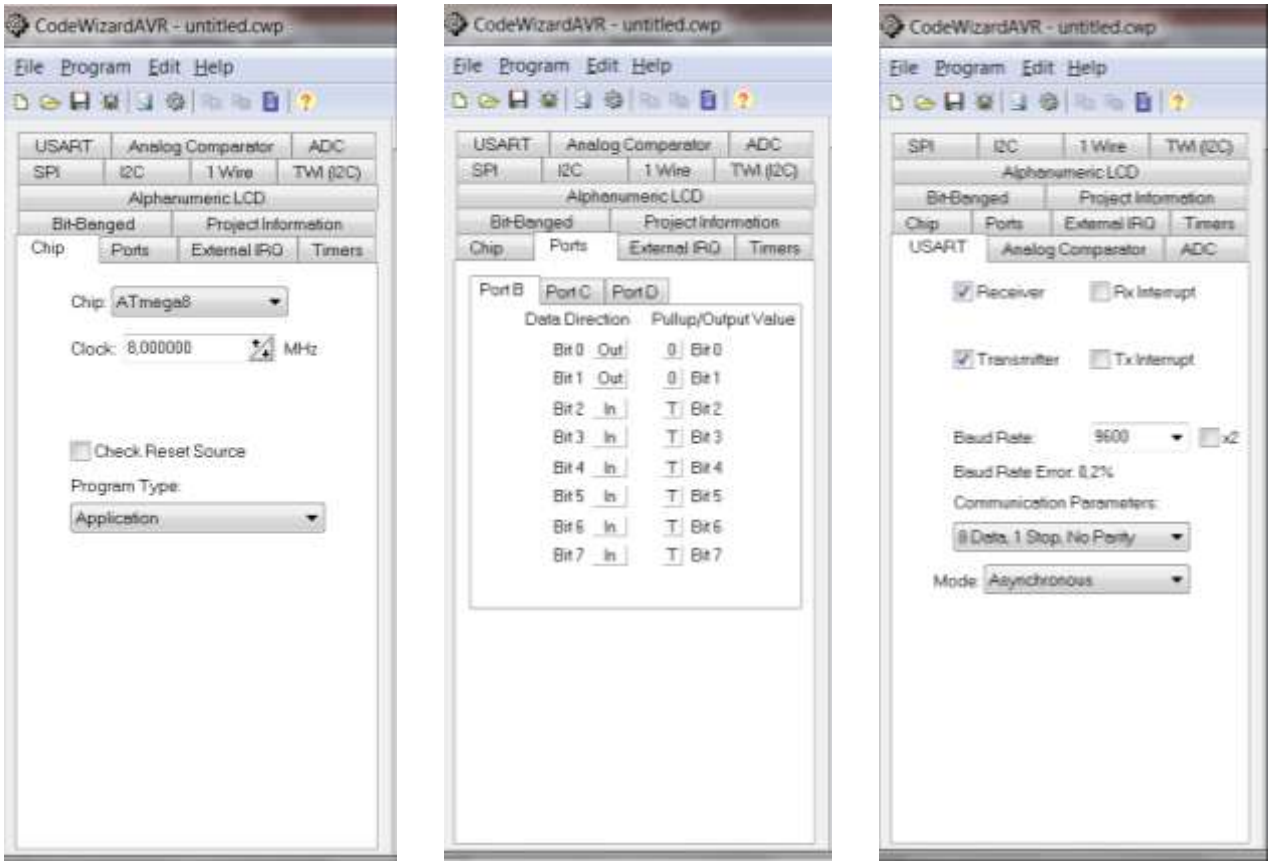

Fig.3.4. Configuración de las pestañas del *CodeWizardAVR*

Se programa mediante lenguaje C (modificado para microcontroladores AVR) la siguiente codificación:

Véase **ANEXO 2**

Finalmente se presiona las teclas Ctrl+F9 para compilar la codificación y generar el archivo *.hex* que mas adelante se cargará en el microcontrolador esclavo correspondiente al seguro eléctrico.

3.2.2.2. Programación del Nodo de Ventana Eléctrica

Para la programación del nodo esclavo correspondiente a la ventana eléctrica, se configura el *CodeWizardAVR* acorde a la figura 3.5.

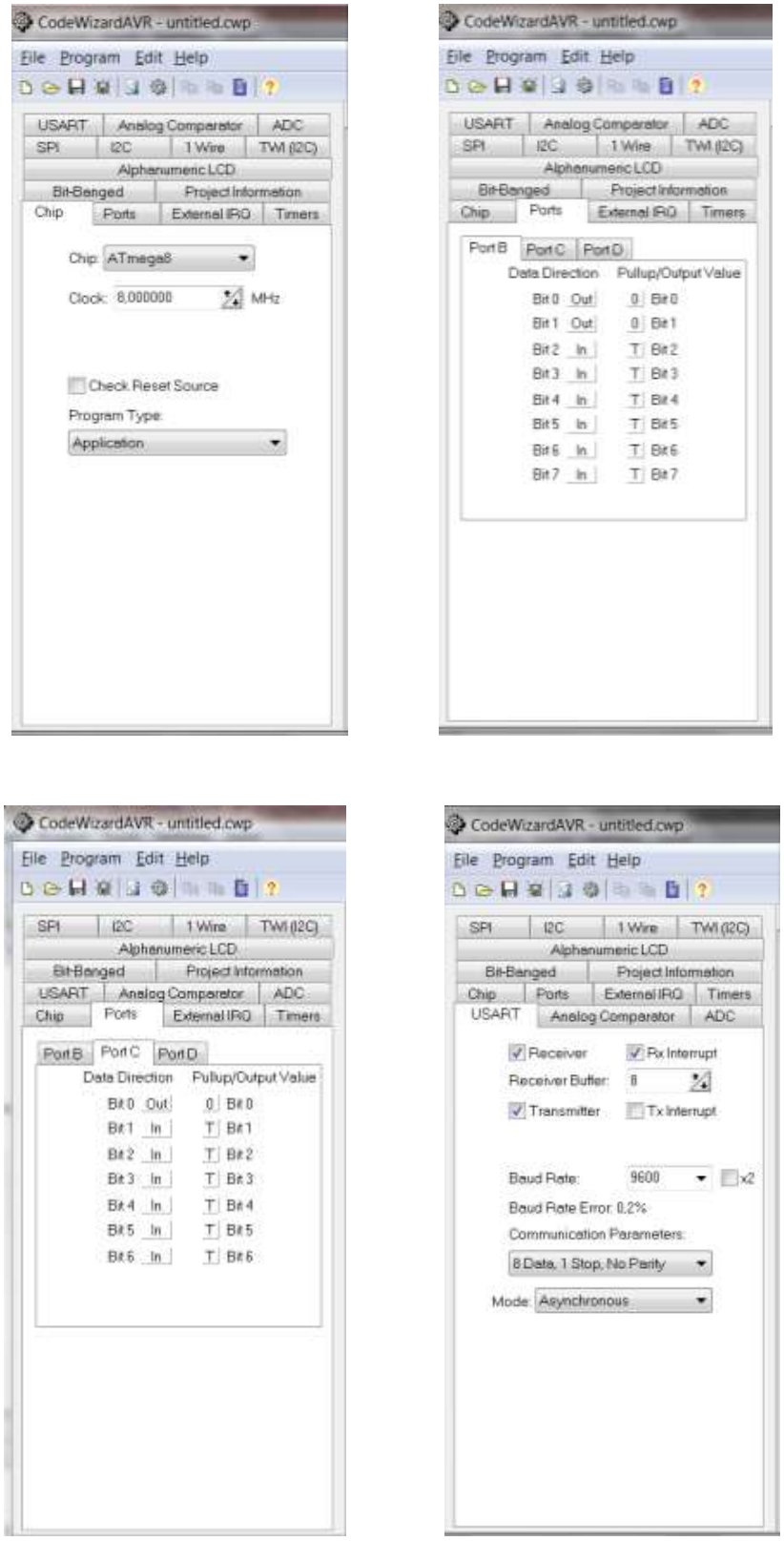

Fig.3.5. Configuración de las pestañas del *CodeWizardAVR*

Se programa mediante lenguaje C (modificado para microcontroladores AVR) la siguiente codificación:

Véase **ANEXO 3**

Finalmente se presiona las teclas Ctrl+F9 para compilar la codificación y generar el archivo *.hex* que mas adelante se cargará en el microcontrolador esclavo correspondiente a la ventana eléctrica.

3.2.2.3. Programación del Nodo de Retrovisor Eléctrico

Para la programación del nodo esclavo correspondiente al retrovisor eléctrico, se configura el *CodeWizardAVR* acorde a la figura 3.6.

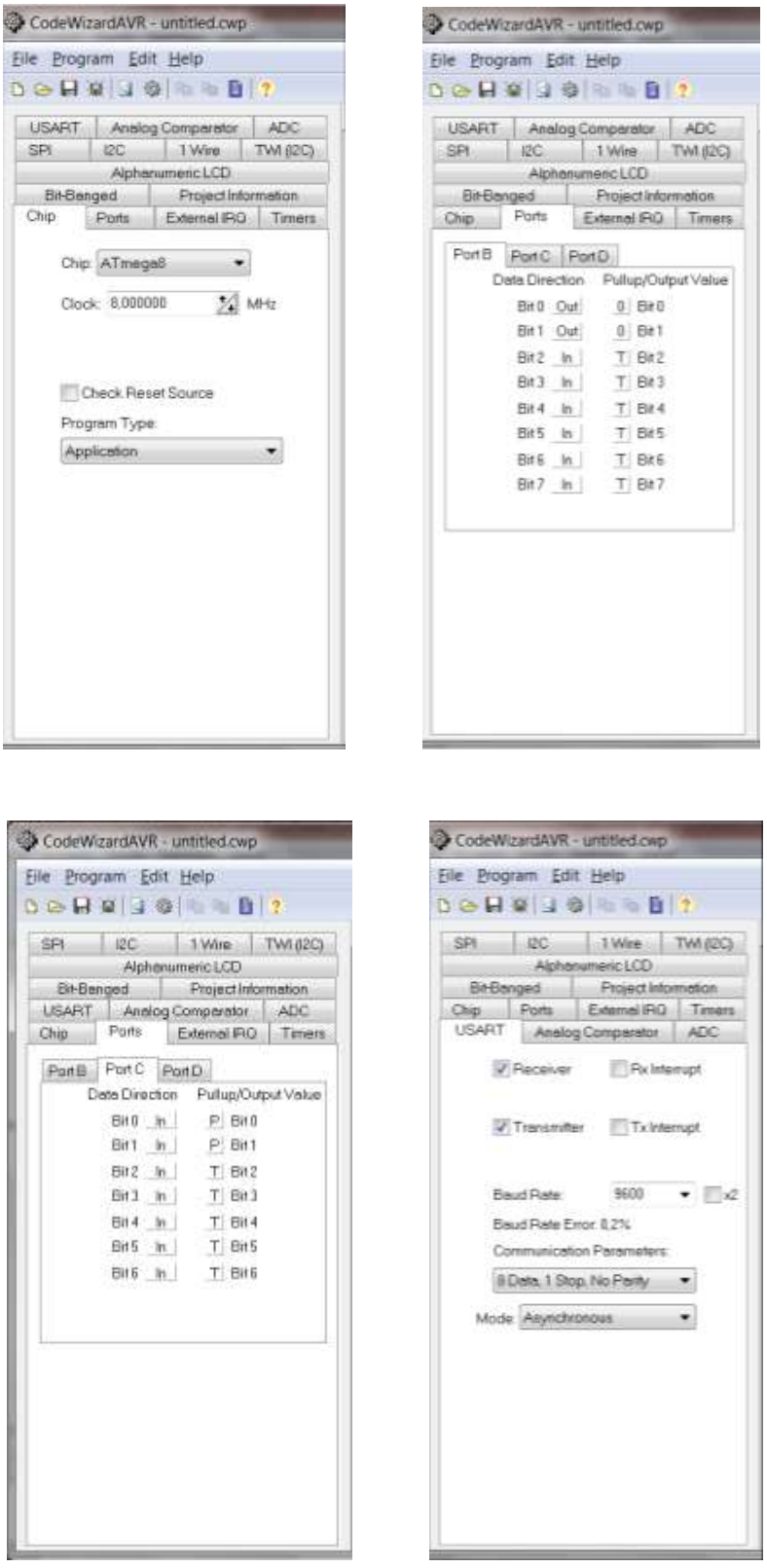

Fig.3.6. Configuración de las pestañas del *CodeWizardAVR*

Se programa mediante lenguaje C (modificado para microcontroladores AVR) la siguiente codificación:

Véase **ANEXO 4**

Finalmente se presiona las teclas Ctrl+F9 para compilar la codificación y generar el archivo *.hex* que mas adelante se cargará en el microcontrolador esclavo correspondiente al retrovisor eléctrico.

3.3. Simulación en software Proteus

Para terminar la simulación se corre el software ISIS de PROTEUS 7.9, y se precede a agregar los componentes descritos en la tabla 3.1.

ELEMENTO	CANTIDAD
ATMEGA 8	4
TRANSISTORES 2N3904	6
BOTONES	2
SWITCHES	8
RESISTENCIAS DE 10K	6
RELÉS	6
MOTORES DC	3
LCD LM016L	1
FUENTE DE CORRIENTE CONTINUA 12V	1
FUENTE DE CORRIENTE CONTINUA 5V	1

Tabla 3.1. Componentes del proyecto

Luego se arma el esquema de red acorde a la figura 3.7.

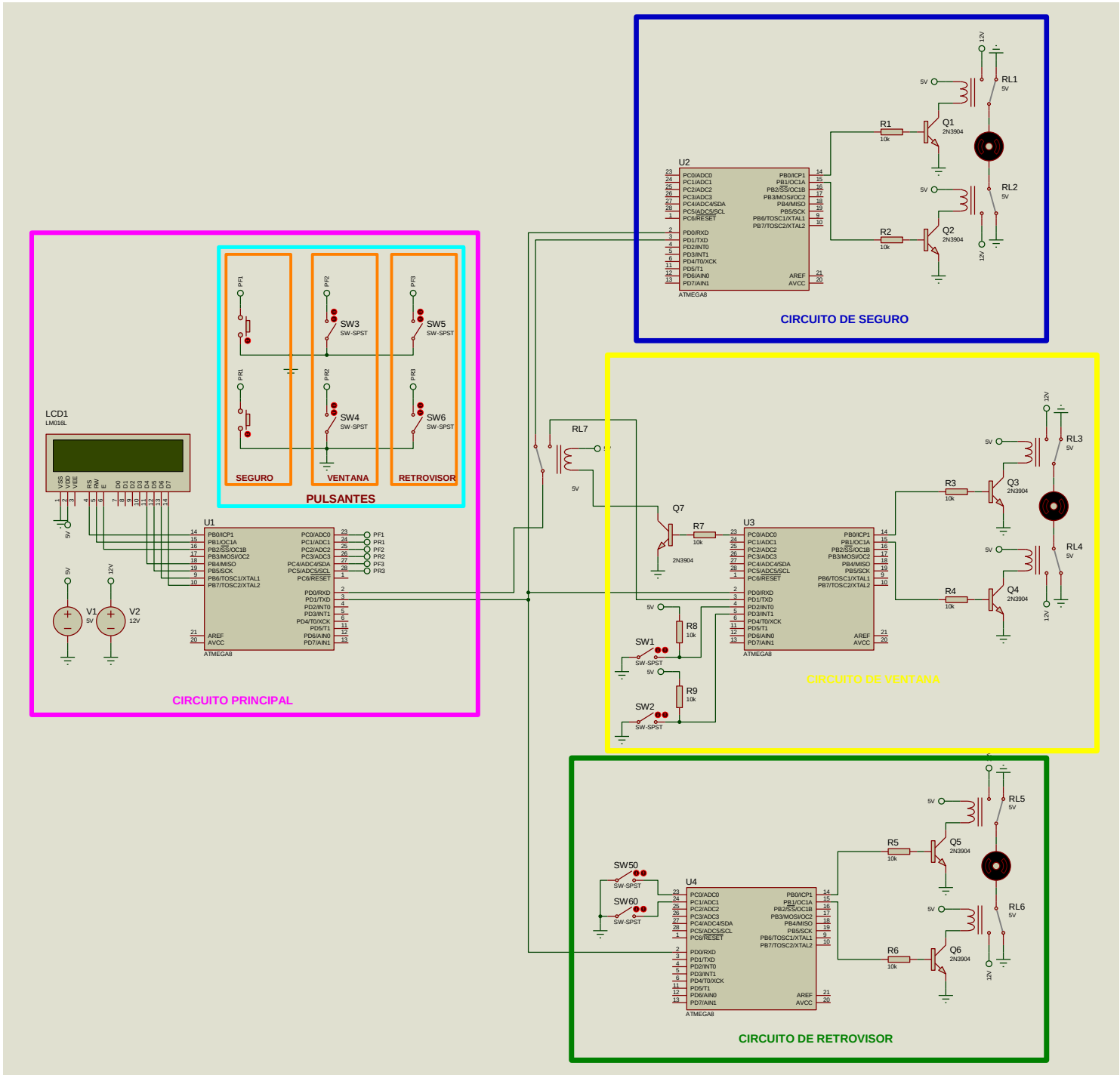

Fig.3.7. Esquema electrónico de un sistema multiplexado en las puertas del vehículo armado en el software ISIS Proteus 7.9. Ver esquema ampliado en ANEXO 5

Se da clic en cada microcontrolador, y se carga el archivo *.hex* correspondiente a cada AVR, como se muestra en la figura 3.8.

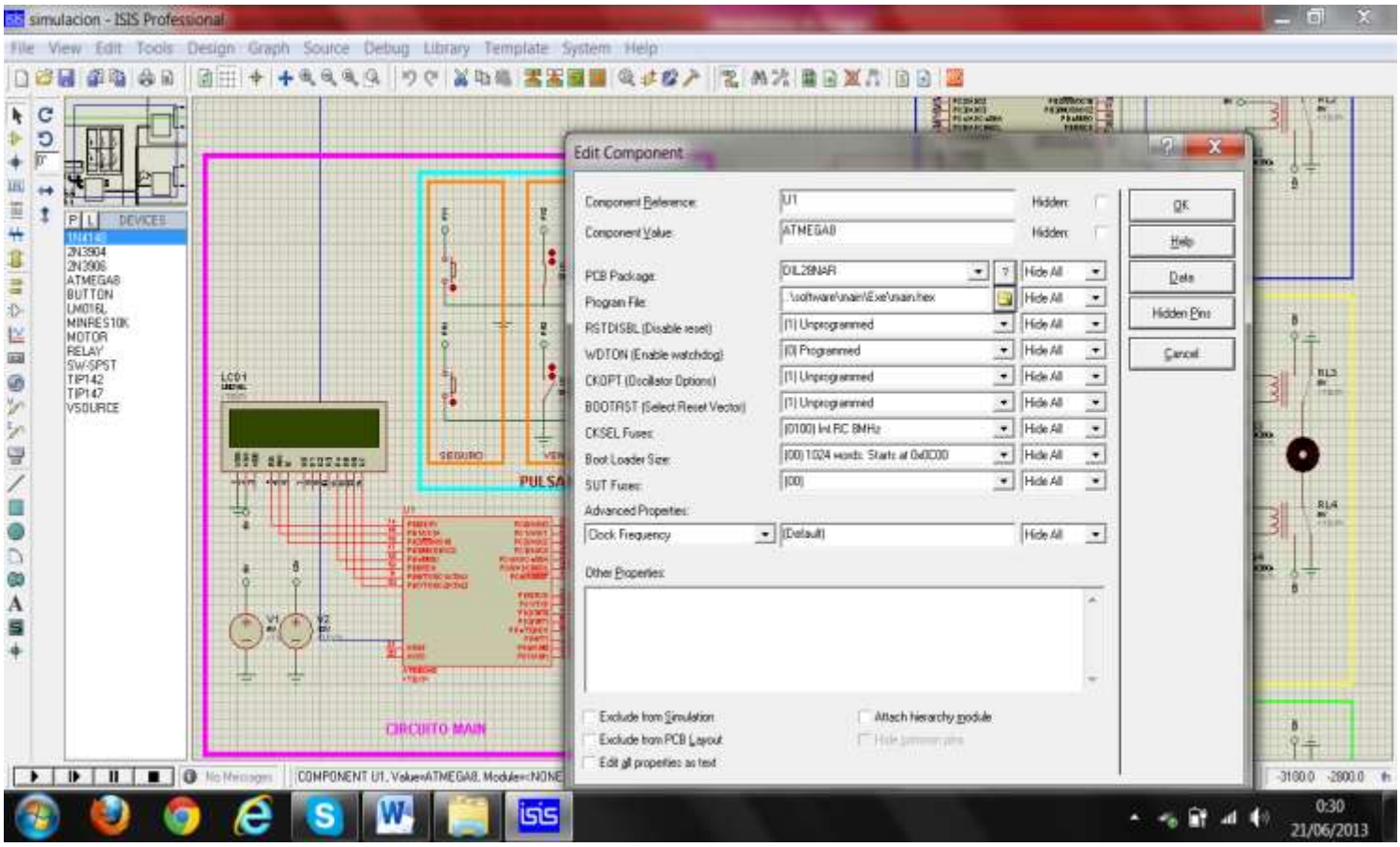

Fig.3.8. Carga del archivo *.hex* correspondiente a cada AVR

Finalmente se corre el simulador, y se procede a realizar las siguientes comprobaciones en la red:

- Si se presiona el botón SW1, el nodo master transmite al bus de datos el número 1 por medio del pin TX. Únicamente el nodo esclavo correspondiente a seguros eléctricos toma el dato 1, y lo gestiona para polarizar positivamente el motor DC del seguro eléctrico, y consecuentemente simulará la activación del seguro de la puerta del vehículo.

- Si se presiona el botón SW2, el nodo master transmite al bus de datos el número 2 por medio del pin TX. Únicamente el nodo esclavo correspondiente a seguros eléctricos toma el dato 2, y lo gestiona para polarizar negativamente el motor DC del seguro eléctrico, y consecuentemente simulará la desactivación del seguro de la puerta del vehículo.

- Si se presiona el *switch* SW3, el nodo master transmite al bus de datos el número 3 por medio del pin TX. Únicamente el nodo esclavo correspondiente a la ventana eléctrica toma el dato 3, y lo gestiona para polarizar positivamente el motor DC de la ventana eléctrica, y consecuentemente simulará la subida de la ventana de la puerta del vehículo.

  Suponiendo que la ventana llegue a tope (superior), se cierra el *switch* SW30, que simula el fin carrera de la ventana en la parte superior, consecuentemente se corta la alimentación del motor DC. Cuando se cierra *switch* SW30 se envía un dato 30 por medio del pin TX hacia el módulo master. El módulo master recibe el dato 30, y lo gestiona para mostrar el mensaje "ventana cerrada" en el *display*.

- Si se presiona el *switch* SW4, el nodo master transmite al bus de datos el número 4 por medio del pin TX. Únicamente el nodo esclavo correspondiente a la ventana eléctrica toma el dato 4, y lo gestiona para polarizar negativamente el motor DC de la ventana eléctrica, y consecuentemente simulará la bajada de la ventana de la puerta del vehículo.

  De la misma manera, suponiendo que la ventana llegue a tope (inferior), se cierra el *switch* SW40, que simula el fin carrera de la ventana en la parte inferior, consecuentemente se corta la alimentación del motor DC. Cuando se cierra *switch* SW40 se envía un dato 40 por medio del pin TX hacia el módulo master. El módulo master recibe el dato 40, y lo gestiona para mostrar el mensaje "ventana abierta" en el *display*.

- Si se presiona el *switch* SW5, el nodo master transmite al bus de datos el número 5 por medio del pin TX. Únicamente el nodo esclavo correspondiente al retrovisor eléctrico toma el dato 5, y lo gestiona para polarizar positivamente el motor DC del retrovisor eléctrico, y consecuentemente simulará el giro hacia la izquierda del retrovisor de la puerta del vehículo.
  Suponiendo que el retrovisor llegue a tope (izquierdo), se cierra el *switch* SW50, que simula el fin carrera del retrovisor en la parte izquierda, consecuentemente se corta la alimentación del motor DC.

- Si se presiona el *switch* SW6, el nodo master transmite al bus de datos el número 6 por medio del pin TX. Únicamente el nodo esclavo correspondiente al retrovisor eléctrico toma el dato 6, y lo gestiona para polarizar negativamente el motor DC del retrovisor eléctrico, y consecuentemente simulará el giro hacia la derecha del retrovisor de la puerta del vehículo.

  De la misma manera, suponiendo que la ventana llegue a tope (derecho), se cierra el *switch* SW60, que simula el fin carrera de la ventana en la parte derecha, consecuentemente se corta la alimentación del motor DC.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- El uso de redes multiplexada en el vehículo representa mayores ventajas que el cableado convencional.
- Para seleccionar un bus se deben tomar en cuenta tanto las restricciones técnicas como económicas.
- Los criterios técnicos para seleccionar un bus son: velocidad de transferencia de datos, inmunidad de la Interferencia, capacidad de tiempo real, número de nodos de red.
- El Modelo de referencia OSI da parámetros para comparar los diferentes protocolos de comunicación en el área automotriz.
- Hay dos mecanismos para transmitir y priorizar mensajes en un bus de datos automotriz, que son: el controlado por eventos (*event triggered*) y el controlado por temporizador (*time triggered*).
- Los códigos de falla de la red se designan con la letra U.
- Para aislar una falla de red se recomienda seguir los pasos que se obtiene mediante el manual del fabricante.
- Los buses clase A y B se usan generalmente para el sistema eléctrico de la carrocería, siendo más costosos los de clase B.
- Los buses clase C generalmente se usan para tren de potencia por que reúnen requerimientos de tiempo real.
- Actualmente, el bus CAN es el bus mas usado para tren de potencia.
- El Bus MOST generalmente usa como medio de trasmisión la fibra óptica y se usa en sistemas de multimedia (ejemplo: audio, video, navegación).

RECOMENDACIONES

- Para la simulación de una red multiplexada, se precisa el uso de microcontroladores que posean puerto USART.
- Para la programación de los microcontroladores del módulo principal y del módulo de la ventana, se recomienda usar recepción por interrupción, debido a que existen eventos externos que interrumpen el ciclo del programa

principal del microcontrolador (los mensajes entrantes que se presentan en el LCD para el módulo master y el accionamiento de los fin carreras en el módulo de la ventana).

BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS

BARNETT, R., Cox, S., & O'Cull, L. (2007). *Embedded C Programming and the Atmel AVR.* Clifton Park, New York: Thomson Delmar Learning.

BOSCH, R. (2007). *Automotive Networking.* Plochingen: Dipl.-Ing. Karl-Heinz Dietsche.

EADY, F. (2004). *Networking and Internetworking with Microcontrollers.* Burlington, USA: Newnes.

ERJAVEC, J. (2010). *AUTOMOTIVE TECHNOLOGY: A Systems Approach, 5e.* Clifton Park, NY, USA: Delmar, Cengage Learning.

HALDERMAN, J. (2012). *Automotive Technology.* New Jersey: Prentice Hall.

HOLLEMBEAK, B. (2011). *Classroom Manual for Automotive Electricity And Electronics.* Clifton Park, NY, USA: Delmar, Cengage Learning.

LUPINI, C. (2003). *Multiplex Bus Progression 2003.* Warrendale, USA: SAE international.

MAZIDI, M., Naimi, S., & Naimi, S. (2011). *The AVR Microcontroller and Embedded System: Using Assembly and C.* New Jersey: Prentice Hall.

NAVET, N., & Simonot-Lion, F. (2009). *Automotive Embedded Systems Handbook.* Boca Raton, Florida, USA.: CRC Press.

PARAB, J., Shelake, V., Kamat, R., & Naik, G. (2007). *EXPLORING C FOR MICROCONTROLLERS.* Dordrecht, The Netherlands: Springer.

PARET, D. (2007). *Multiplexed Networks for Embedded Systems.* Wiltshire, Gran Bretaña: Wiley.

PARET, D. (2012). *FLEXRAY AND ITS APPLICATIONS REAL TIME MULTIPLEXED NETWORK.* Paris: WILEY.

PETERSON, L., & Davie, B. (2003). *Computer Networks: A Systems Approach.* San Francisco, CA: Morgan Kaufmann Publishers.

PRASAD, K. V. (2004). *Principles Digital Communication System & Computer Networks.* Hingham, Massachusetts: Charles River Media, INC. .

TREVENNOR, A. (2012). *Practical AVR Microcontrollers.* New York: Apress.

REFERENCIAS DIGITALES

USGS; Representación esquemática de un Byte; http://isis.astrogeology.usgs.gov/IsisWorkshop/index.php/Understanding_Bit_Types ; junio del 2013

Atmel; http://www.atmel.com/images/doc9164.pdf ; junio del 2013

Jean-François Fourcadier; Comparación entre una trama NRZ y una *Biphase*, con respecto a la frecuencia de un mismo *clock*; http://jf.fourcadier.pagesperso-orange.fr/haut_debit/biphase/biphase_e.htm ; junio del 2013

1394 TRADE ASSOCIATION PRESS RELEASE; http://www.1394ta.org/press/TAPress/2008_0725.html ; junio del 2013

1394 TRADE ASSOCIATION PRESS RELEASE; http://www.1394ta.org/press/TAPress/LookWhat1394AutomotiveCanDo.html; junio del 2013

ANEXOS

ANEXO 1: Programación del Nodo Maestro

```
/*-------------------------------------------------------
        LIBRERIAS
-------------------------------------------------------*/
#include <mega8.h>
#include <stdlib.h>
#include <alcd.h>
/*-------------------------------------------------------
        VARIABLES
-------------------------------------------------------*/
unsigned char i=0;
unsigned char x=0;


/*-------------------------------------------------------
CONFIGURACION USART (RECEPCION DE DATOS)
-------------------------------------------------------*/

#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif
```

```
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

/*-----------------------------------------------------------------------------------------
        INTERRUPCION POR RECEPCION DE DATOS
-----------------------------------------------------------------------------------------*/
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if (data==10)                       //PREGUNTA SI DATO ES 10
      {
      lcd_gotoxy(0,0);              //SI ES QUE SI UBICA AL CURSOR DEL LCD EN LA UBICACION 0
      lcd_putsf("seguro cerrado");   //PONEMOS EN EL LCD SEGURO CERRADO
      };
     if (data==20)                  //PREGUNTA SI DATO ES 20
      {
      lcd_gotoxy(0,0);              //SI ES QUE SI UBICA AL CURSOR DEL LCD EN LA UBICACION 0
      lcd_putsf("seguro abierto");   //PONEMOS EN EL LCD SEGURO ABIERTO
      };
     if (data==30)                  //PREGUNTA SI DATO ES 30
      {
      x=1;                          //PONEMOS 1 A LA VARIABLE X
      i=0;                          //PONEMOS 0 A LA VARIABLE I
      lcd_gotoxy(0,1);              //SI ES QUE SI UBICA AL CURSOR DEL LCD EN LA UBICACION 0 EN X
Y 1 EN Y
      lcd_putsf("ventana cerrada");   //PONEMOS EN EL LCD SEGURO CERRADO
      };
     if (data==40)                  //PREGUNTA SI DATO ES 40
      {
      x=0;
      i=1;
      lcd_gotoxy(0,1);              //SI ES QUE SI UBICA AL CURSOR DEL LCD EN LA UBICACION 0 EN X
Y 1 EN Y
```

```
       lcd_putsf("ventana abierta");   //PONEMOS EN EL LCD SEGURO ABIERTO
       };
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index++]=data;
#if RX_BUFFER_SIZE == 256
  // special case for receiver buffer size=256
  if (++rx_counter == 0)
    {
#else
  if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
#endif
    rx_buffer_overflow=1;
    }
  }

}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index++];
#if RX_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here


/*-----------------------------------------------------
          CONFIGURACION GENERAL
-----------------------------------------------------*/
void configuracion(void)
```

```
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=P State4=P State3=P State2=P State1=P State0=P
PORTC=0x3F;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;
// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTB Bit 0
```

```
// RD - PORTB Bit 1
// EN - PORTB Bit 2
// D4 - PORTB Bit 4
// D5 - PORTB Bit 5
// D6 - PORTB Bit 6
// D7 - PORTB Bit 7
// Characters/line: 16
lcd_init(16);

// Global enable interrupts
#asm("sei")
}

/*-------------------------------------------------------
        PROGRAMA PRINCIPAL
-------------------------------------------------------*/
void main(void)
{
configuracion();
while (1)
    {
    putchar(0);                  //ENVIAMOS AL PUERTO SERIAL EL VALOR 0
    while (PINC.0==0)              //SI PRESIONO EL PULSANTE DE SEGURO CERRADO
     {
     putchar(1);               //ENVIO ATRAVES DEL PUERTO SERIAL EL NUMERO 1
     };
    while (PINC.1==0)             //SI PRESIONO EL PULSANTE DE SEGURO ABIERTO
     {
     putchar(2);               //ENVIO ATRAVES DEL PUERTO SERIAL EL NUMERO 2
     };
    while (PINC.2==0)             //SI PRESIONO EL PULSANTE DE VENTANA CERRADA
     {
     if (x==0)                 //PREGUNTO SI LA VARIABLE X ESTA EN 0
        {
        putchar(3);               //ENVIO ATRAVES DEL PUERTO SERIAL EL NUMERO 3
        i=0;                   //PONGO 0 EN VARIABLE i
        }
     else
        {
        putchar(0);               //ENVIO AL PUERTO SERIAL 0
        }
     };
    while (PINC.3==0)             //SI PRESIONO EL PULSANTE DE VENTANA ABIERTA
     {
     if (i==0)                 //PREGUNTO SI LA VARIABLE I ES 0
        {
        putchar(4);             //ENVIO ATRAVES DEL PUERTO SERIAL EL NUMERO 4
        x=0;                 //PONGO 0 A LA VARIABLE X
        }
     else
        {
```

```
        putchar(0);          //ENVIO AL PUERTO SERIAL 0
        }
      };
    while (PINC.4==0)          //SI PRESIONO EL PULSANTE DE RETROVISOR A LA DERECHA
      {
      putchar(5);            //ENVIO ATRAVES DEL PUERTO SERIAL EL NUMERO 5
      };
    while (PINC.5==0)          //SI PRESIONO EL PULSANTE DE RETROVISOR A LA IZQUIERDA
      {
      putchar(6);            //ENVIO ATRAVES DEL PUERTO SERIAL EL NUMERO 6
      };

    }
}
```

ANEXO 2: Programación del Nodo de Seguro Eléctrico

```
/*-----------------------------------------------
            LIBRERIAS
-----------------------------------------------*/
#include <mega8.h>
#include <stdio.h>
#include <delay.h>


/*-----------------------------------------------
            VARIABLES
-----------------------------------------------*/
unsigned char dato;

/*-----------------------------------------------
        SUBRUTINA DE CONFIGURACION
-----------------------------------------------*/
void configuracion(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=0
PORTB=0x00;
DDRB=0x03;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
```

```
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

// Analog Comparator initialization
```

```
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;
}
/*---------------------------------------------------------------------
            PROGRAMA PRINCIPAL
---------------------------------------------------------------------*/
void main(void)
{
configuracion();           //llamamos a subrutina de configuracion inicial
while (1)
    {
    dato=getchar();      //RECIBE EL VALOR TRANSMITIDO POR EL MAIN
    if (dato == 1)       //PREGUNTA SI LA VARIABLE DATO ES 3
     {
     PORTB.0=1;          //si la respuesta es si activamos el puerto b.0
     delay_ms(200);     //esperamos 2 segundos
     PORTB.0=0;          //desactivamos el puerto b.0
     putchar(10);        //enviamos al puerto serial 10 seguro cerrado
     }
    if (dato == 2)       //PREGUNTA SI LA VARIABLE DATO ES 4
     {
     PORTB.1=1;          //si la respuesta es si activamos el puerto b.1
     delay_ms(200);     //esperamos 2 segundos
     PORTB.1=0;          //desactivamos el puerto b.1
     putchar(20);        //enviamos al puerto serial 20 seguro abierto
     }
    }
}
```

ANEXO 3: Programación del Nodo de Ventana Eléctrica

```
/*---------------------------------------------------------------------
                    LIBRERIAS
---------------------------------------------------------------------*/
#include <mega8.h>
#include <delay.h>
#include <stdio.h>
/*---------------------------------------------------------------------
                    VARIABLES
---------------------------------------------------------------------*/
unsigned char dato;

/*---------------------------------------------------------------------
                  INTERRUPCION
---------------------------------------------------------------------*/

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
PORTB.0=0;            //APAGAMOS EL MOTOR PORTB.0=0
PORTC.0=1;            //ACTIVAMOS AL RELE
putchar(30);          //MANDAMOS AL PUERTO SERIAL 30 PARA DECIR VENTANA CERRADA
delay_ms(500);        //ESPERAMOS MEDIO SEGUNDO
PORTC.0=0;            //DESACTIVAMOS EL RELE

}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
PORTB.1=0;            //APAGAMOS EL MOTOR
PORTC.0=1;            //ACTIVAMOS AL RELE
putchar(40);          //MANDAMOS AL PUERTO SERIAL 40 PARA DECIR VENTANA ABIERTA
delay_ms(500);        //ESPERAMOS MEDIO SEGUNDO
PORTC.0=0;            //DESACTIVAMOS EL RELE
}

#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif
```

```
#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
dato=data;        //DATO ES IGUAL A DATA
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index++]=data;
#if RX_BUFFER_SIZE == 256
  // special case for receiver buffer size=256
  if (++rx_counter == 0)
    {
#else
  if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
```

```
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
#endif
    rx_buffer_overflow=1;
    }
  }
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index++];
#if RX_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

/*-----------------------------------------------------------------
                    PROGRAMA PRINCIPAL
-----------------------------------------------------------------*/
void configuracion(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=0
PORTB=0x00;
DDRB=0x03;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=Out
// State6=T State5=T State4=T State3=T State2=T State1=T State0=0
PORTC=0x00;
DDRC=0x01;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
```

```
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: On
// INT1 Mode: Falling Edge
GICR|=0xC0;
MCUCR=0x0A;
GIFR=0xC0;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Global enable interrupts
#asm("sei")
}

/*---------------------------------------------------------------
                    PROGRAMA PRINCIPAL
---------------------------------------------------------------*/

void main(void)
{
configuracion();              //LLAMA A SUBRUTINA DE CONFIGURACION
while (1)
    {
    if (dato == 3)          //PREGUNTA SI LA VARIABLE DATO ES 3
      {
      PORTB.0=1;            //MOTOR GIRA PARA ARRIBA
      }
```

```
    if (dato == 4)          //PREGUNTA SI LA VARIABLE DATO ES 4
      {
      PORTB.1=1;            //MOTOR GIRA PARA ABAJO
      }
    if (dato == 0)          //PREGUNTA SI LA VARIABLE DATO ES 0
      {
      PORTB.1=0;            //APAGO PUERTO B.1
      PORTB.0=0;            //APAGO PUERTO B.0
      }
    };
}
```

ANEXO 4: Programación del Nodo de Retrovisor Eléctrico

```
/*---------------------------------------------------------------
                    LIBRERIAS
---------------------------------------------------------------*/
#include <mega8.h>
#include <stdio.h>

/*---------------------------------------------------------------
                    VARIABLES
---------------------------------------------------------------*/
unsigned char dato;

/*---------------------------------------------------------------
              SUBRUTINA CONFIGURACION
---------------------------------------------------------------*/
void configuracion(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=0
PORTB=0x00;
DDRB=0x03;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=P State0=P
PORTC=0x03;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
```

```
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
```

```
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;
}

/*---------------------------------------------------------------
                PROGRAMA PRINCIPAL
---------------------------------------------------------------*/
void main(void)
{
configuracion();              //LLAMA A SUBRUTINA DE CONFIGURACION
while (1)
    {
    dato=getchar();          //RECIBE EL VALOR TRANSMITIDO POR EL MAIN
    if (dato == 5)           //PREGUNTA SI LA VARIABLE DATO ES 3
      {
      PORTB.0=1;             //MOTOR GIRA PARA ARRIBA
      }
    if (dato == 6)           //PREGUNTA SI LA VARIABLE DATO ES 4
      {
      PORTB.1=1;             //MOTOR GIRA PARA ABAJO
      }
    if (dato == 0)           //PREGUNTA SI LA VARIABLE DATO ES 0
      {
      PORTB.1=0;             //APAGO PUERTO B.1
      PORTB.0=0;             //APAGO PUERTO B.0
      }
    while (PINC.0==0)        //SI LA VENTANA LLEGA ARRIBA EL PORTC.0 ES 0
      {
      PORTB.0=0;             //APAGAMOS EL MOTOR PORTB.0=0
      dato=getchar();        //RECIBE EL VALOR TRANSMITIDO POR EL MAIN
      if (dato == 6)         //PREGUNTA SI LA VARIABLE DATO ES 4
        {
        PORTB.1=1;           //HABILITAMOS AL PORTB.1 PARA QUE MOTOR BAJE
        }
      if (dato == 0)         //PREGUNTA SI LA VARIABLE DATO ES 0
        {
        PORTB.1=0;           //APAGA MOTOR
        PORTB.0=0;           //APAGA MOTOR
        }
```

```
    };
  while (PINC.1==0)          //SI LA VENTANA LLEGA ABAJO EL PORTC.1 ES 0
    {
    PORTB.1=0;               //APAGAMOS EL MOTOR
    dato=getchar();          //RECIBE EL VALOR TRANSMITIDO POR EL MAIN
    if (dato == 5)           //PREGUNTA SI LA VARIABLE DATO ES 4
      {
      PORTB.0=1;             //HABILITAMOS EL MOTOR PARA QUE SUBA LA VENTANA
      }
    if (dato == 0)           //PREGUNTA SI LA VARIABLE DATO ES 0
      {
      PORTB.1=0;             //APAGA EL MOTOR
      PORTB.0=0;             //APAGA EL MOTOR
      }
    };
  }
}
```