BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

IDENTIFICACIÓN Y CONTROL DE PARÁMETROS DE CLÚSTER DE INSTRUMENTOS AUTOMOTRIZ MEDIANTE RED CAN

TESIS

QUE PARA OBTENER EL GRADO DE:

LICENCIATURA EN INGENIERÍA EN SISTEMAS AUTOMOTRICES

PRESENTA:

LADY GUADALUPE FELICIANO FUENTES

ASESOR:

DR. ROBERTO CARLOS AMBROSIO LÁZARO

PUEBLA, PUE. NOVIEMBRE 2019

Dedicatoria

A mis padres por su apoyo, consejos, confianza, amor y motivación. Gracias por estar a mi lado en cada paso de mi vida. Que Dios los proteja y les dé una larga vida para seguir compartiendo momentos de felicidad y éxito juntos.

Agradecimientos

Agradezco a mis padres, Lorena Fuentes y Martin Feliciano por su apoyo incondicional durante mi formación. Gracias por darme el ejemplo de perseverancia y preocuparse por mi bienestar y educación.

Agradezco a mi novio, Edgar Hernández por su apoyo, amor, comprensión y por compartir conmigo sus conocimientos durante toda la carrera. Gracias por estar a mi lado inclusive en los momentos más difíciles, tu ayuda y motivación han sido fundamentales.

Mi más sincero agradecimiento a mi asesor de Tesis, Dr. Roberto Carlos Ambrosio Lázaro por su apoyo, por compartir sus conocimientos, darme asesorías y por brindarme el material necesario durante la realización de esta Tesis. Gracias por darme la oportunidad y confianza de desarrollar proyectos de investigación bajo su tutela.

Agradezco a mis sinodales, M.C Julia Chávez Chávez, M.C. Nicolás Quiroz Hernández y Dr. Luis Abraham Sánchez Gaspariano por su tiempo, observaciones y correcciones que me ayudaron a mejorar y concluir esta Tesis.

Agradezco a la Benemérita Universidad Autónoma de Puebla, la Facultad de Ciencias de la Electrónica y a mis profesores por brindarme una Excelente preparación profesional.

También agradezco a la Vicerrectoría de Investigación y Estudios de Posgrado (VIEP) por la oportunidad de trabajar en el proyecto de investigación VIEP-2019 “Sistema de control para panel de instrumentos automotrices basados en Hardware in the Loop (HIL) y comunicación CAN”. Gracias por fomentar en los estudiantes la investigación.

Índice General

Índice de Figuras

Índice de Figuras

Índice de Tablas

Glosario de Términos

CAN	Red de área de controlador	Controller Area Network
CAN_H	CAN alto	CAN High
CAN_L	CAN bajo	CAN Low
LIN	Red de interconexión local	Local InterConnect Network
ISO	Organización internacional para la estandarización	International Organization for Standardization
SAE	Sociedad de ingenieros automotrices	Society of Automotive Engineers
TX	Transmisión	Transmission
RX	Receptor	Receiver
ID	Identificador	Identifier
SPI	Interfaz periférica serial	Seria Peripherical Interface
UTP	Par trenzado sin blindaje	Unshielded twisted pair
STP	Par trenzado blindado	Shielded Twisted Pair
ABS	Sistema antibloqueo de frenos	Antilock Brake System
MOST	Transporte de sistemas orientados a medios	Media Oriented Systems Transport
BIT	Dígito binario	Binary digit
kbps	kilobits por segundo	kilobits per second
RPM	Revoluciones Por Minuto	Revolutions per minute
EPC	Control Electrónico de Potencia	Electronic Power Control
MOSI	Salida de maestro / Entrada de esclavo	Master Out Slave In
MISO	Entrada a maestro/ Salida de esclavo	Master In Slave Out
CLK	Reloj	Clock
SS	Esclavo seleccionado	Slave Select
PXI	Extensiones PCI para instrumentación	PCI eXtensions for Instrumentation
ICL	Clúster de instrumentos	Instrument clúster
IDs	Identificadores	Identifiers

Resumen

En esta tesis se presenta el control de tres de los parámetros de un clúster de instrumentos Automotriz mediante una Red CAN Bus. Los parámetros que se controlan son: Revoluciones por minuto, Velocidad y Luces. Para el control de los parámetros se envían mensajes al clúster utilizando sus identificadores de mensaje los cuales son: 280 para RPM, 5A0 para velocidad y 470 para el nodo de luces. Este trabajo se realiza con la intención de ser una herramienta de aprendizaje interactivo del protocolo de comunicación CAN.

Los nodos de la Red están formados por la unión de dos placas las cuales son: Placa CAN Bus Shield y Arduino UNO. La recepción y envío de mensajes al Bus se logra gracias al controlador MCP2515 y el transceptor MCP2551 integrados en la placa CAN Bus Shield. Para el envío y recepción de mensajes se realiza la programación de los nodos en la IDE de Arduino a una velocidad de 500 Kbps debido a que esta es la velocidad a la que trabaja el clúster de instrumentos. Los mensajes enviados son formato estándar, con un código de longitud de datos (también llamado payload) de 8 bytes.

Para la comunicación de la Red con el panel de instrumentos se necesitó de la obtención de los identificadores (IDs) de mensaje del clúster de instrumentos debido a que los fabricantes protegen el acceso a sus identificadores y la información que se envía en ellos, esto como medio de seguridad. Debido a esto se estableció una metodología con la cual se consiguió identificar los IDs de mensaje y los bytes en los cuales se envía la información al Bus.

La comunicación de la Red CAN y el panel de instrumentos se realizó de acuerdo a la prioridad de los mensajes. Se observó la intervención de algunos identificadores de mensaje con mayor frecuencia debido a la importancia de estos en el Bus. Las pruebas de comunicación se realizaron empleando tres diferentes métodos con el fin de verificar la comunicación y el comportamiento de la Red.

CAPÌTULO 1

Introducción y Objetivos

En la actualidad los automóviles cuentan con una gran cantidad de nodos encargados del intercambio de información de diversos subsistemas de automóvil como el motor, sistema de entretenimiento, bolsa de aire, entre otros, los que resultan importantes para el diagnóstico y control de distintos elementos del vehículo. Para comunicar los diversos nodos en el automóvil la empresa BOSCH en 1985 desarrolló el protocolo de comunicación CAN Bus[1]. El intercambio de datos entre los nodos es realizada en forma de señales eléctricas como se muestra en la figura 1.1 y se realiza mediante un Bus de dos cables CAN_H y CAN_L, en estado recesivo los dos canales tienen el mismo voltaje (2.5 v) y en estado dominante hay una diferencia de voltaje de 1 volt, esta señal al llegar a otros nodos es convertidas en código binario para ser interpretado por estas, por medio de este Bus de comunicación se transmite una la gran cantidad de información a alta velocidad.

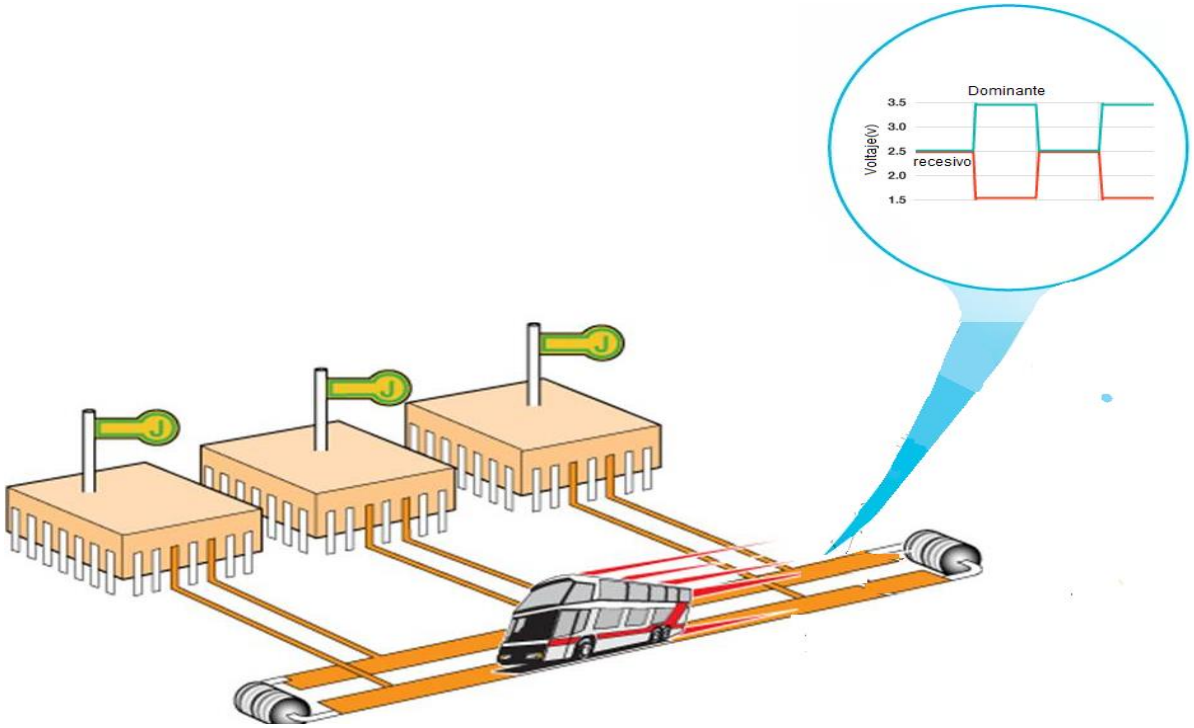


Figura 1.1 Comunicación CAN Bus [2]

> *Para que un mensaje sea transmitido, es necesario un emisor, un receptor, un canal de comunicación entre ambos y un lenguaje común. Los nodos están compuestos de un transceptor y un controlador, encargados de convertir la señal eléctrica del Bus en una señal digital y viceversa, y gestionar la información*[2].

El protocolo CAN admite la comunicación entre nodos de red sin la necesidad de una unidad de control central. Los nodos de la red se comunican mediante mensajes, el mensaje es una sucesión de bits y tiene una serie de campos de que permiten llevar a cabo el proceso de comunicación entre los nodos. Estos permiten identificar al nodo, indicar el principio y el final del mensaje y mostrar los datos. El protocolo CAN está orientado al mensaje y no al destinatario, los mensajes son recibidos por todos los nodos, pero son utilizados solo por quienes los necesitan. En caso de una trasmisión de mensajes simultánea, la prioridad se decide de acuerdo al identificador(campo de arbitraje), entre más bajo es el identificador el mensaje tiene más prioridad[3].

Este protocolo tiene como objetivo disminuir la cantidad de cables de comunicación en el automóvil, como se observa en la figura 1.2 este protocolo permite a los nodos tener una sola interfaz CAN, eliminando de esta manera la necesidad de tener muchas entradas analógicas o digitales para cada uno de los Nodos en el sistema, esto significa la reducción tanto del costo como del peso en el vehículo. Algunas otras ventajas de este protocolo de comunicación son: El aumento de velocidad de transmisión de la información entre los nodos, Aumento de espacio en el habitáculo debido a que los nodos son más compactos, la efectividad del envió de datos, disminución de sensores y aviso de fallas.

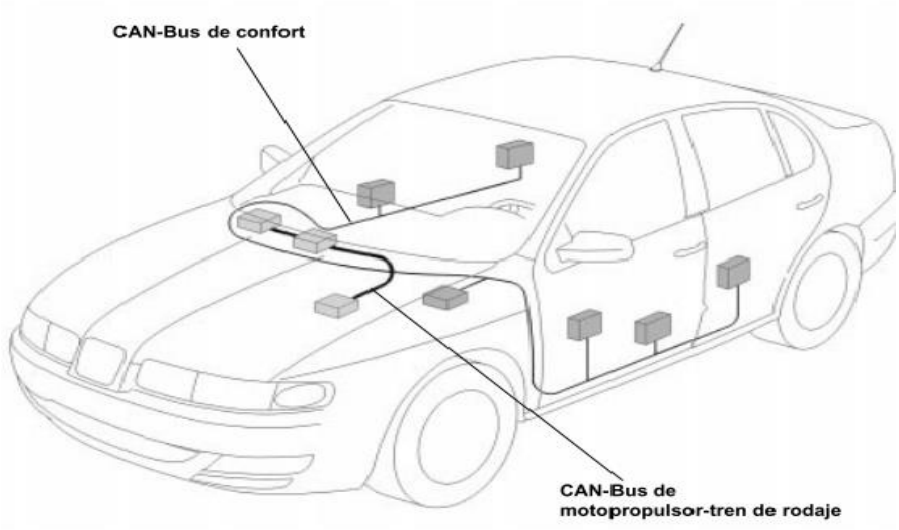


Figura 1.2 Cableado en el vehículo con CAN Bus[4]

Cada fabricante decide qué Bus y qué protocolos tienen más sentido para su vehículo. El protocolo CAN Bus, existe en una ubicación estándar en todos los vehículos: en el conector OBD-II. Dicho esto, los paquetes que viajan sobre el Bus CAN de un vehículo no están estandarizados. La comunicación crítica del vehículo, como la gestión de RPM y el frenado, ocurre en las líneas de Bus de alta velocidad, mientras que la comunicación no crítica, como el bloqueo de la puerta y el control de A/C ocurre en las líneas de bus de velocidad media a baja[5].

En 1991, el CAN Bus (Controller Area Network) fue el primer sistema de Bus que se introdujo en un vehículo motorizado en producción en masa. Desde entonces, se ha establecido como el sistema estándar en el sector automotriz, pero el Bus CAN también se usa comúnmente como Bus de campo en la ingeniería de automatización en general[3].

Los automóviles modernos contienen más de 50 nodos conectados en la red. La seguridad general del vehículo se basa en la comunicación en tiempo real entre estos diferentes nodos. Mientras que se comunican entre sí, los nodos son responsables de la predicción de accidentes, la detección de derrapes, la realización de antibloqueo de frenos, etc. Para realizar estas acciones los nodos cuentan con sensores y actuadores, los sensores proporcionan información al nodo para que puedan tomar decisiones sobre las acciones a tomar. Los actuadores permiten al nodo realizar acciones. Los nodos son dispositivos integrados especiales con fines específicos para percibir el entorno que les rodea y tomar medidas para ayudar al automóvil[6].

EL grupo de instrumentos representa el nodo más importante de un automóvil moderno, en relación con la interfaz con el conductor. A medida que más y más controles se realizan de forma automática, las funcionalidades proporcionadas por el clúster de instrumentos se vuelven cada vez más complejas. La funcionalidad se distribuye entre muchos nodos que proporcionan a los clientes información como condiciones de manejo, diagnósticos de fallas, señales de advertencia e información y entretenimiento [7]

La primera aplicación de un clúster de instrumentos electrónico en la industria automotriz fue en el año 1976 y fue realizada por Aston Martin Lagonda. Mientras que los primeros cuadros de instrumentos solo mostraban información analógica utilizando paneles de instrumentos electromecánicos, los clúster que se desarrollan hoy muestran todo de manera electrónica, utilizando interfaces gráficas con mayor cantidad de funciones[8].

En la industria automotriz actual, existen tres tipos de clústers de instrumentos, pueden ser analógicos, digitales e híbridos (figura 1.3), los cuales pueden ser implementados en vehículos para fines comerciales, industriales o especiales, el panel de instrumentos está expuesto a altas demandas y, por tanto, debe ser extremadamente resistentes[9].

Figura 1.3 Tipos de Cuadros de instrumentos[10].

Los factores del desarrollo del panel de instrumentos son el desarrollo tecnológico, la creciente demanda de clúster de instrumentos electrónicos, la implementación de sistemas avanzados de asistencia al conductor (ADAS) principalmente en auto de gama alta y el incremento de petición de autos eléctricos. La solicitud de clústers de instrumentos electrónicos con sistemas de visualización innovadores y sistemas de gráficos 2D Y 3D dirige al crecimiento del mercado del panel de instrumentos al futuro, en donde el diseño, la tecnología y las características de los circuitos integrados pueden ser muy diferentes, según las especificaciones del fabricante del equipo original (OEM)[8].

El clúster de instrumentos es una interfaz importante para el intercambio de información entre el conductor y el vehículo. Con la comunicación hombre-máquina y más requisitos de seguridad incorporadas en el cuadro de instrumento, es más conveniente para los conductores conocer información relevante sobre el estado actual del automóvil[11]. El cuadro de instrumentos contiene medidores que le proporcionan al conductor información sobre la velocidad, temperatura o el nivel combustible, información del módulo GPS e incluso información y entretenimiento, ya que distintas tecnologías han evolucionado para proporcionar mejor información para el conductor (ver figura 1.4). El clúster también contiene testigos que indican al conductor advertencias o informan del estado de algún nodo del automóvil. De esta forma los conductores pueden comprender y dominar la condición de marcha del coche correctamente para poder manejar diversas situaciones, asegurando que su transporte sea seguro y cómodo, debido a que se muestra la información de forma clara en el momento determinado.

Figura 1.4 Medidores y testigos del cuadro de instrumentos [4].

Con el desarrollo tecnológico en el automóvil y la implementación del CAN Bus, se proporciona la comunicación de diversos nodos que dan aviso del estado de los diferentes dispositivos del auto, el nodo debe procesar de manera rápida y precisa todo tipo de información que puede mostrarse en el cuadro de instrumentos y proporcionar al conductor información claramente estructurada, fácil de leer y en el momento específico. Sin el clúster el automovilista no conocería la situación de su automóvil y no estaría informado en caso de existir un problema grave o peligroso.

La comunicación CAN Bus es una de las más utilizadas en la industria automotriz, para realizar la comunicación entre los nodos y el clúster de instrumentos se necesita conocer los identificadores de mensaje, ya que sin estos en cuadro de instrumentos no reconocerá ningún dato recibido, este se vuelve un obstáculo, debido a que cada compañía tiene sus propios IDs, cambian de acuerdo al modelo del auto y no es información pública. En este trabajo se pretende controlar un cuadro de instrumentos, mediante la obtención de los identificadores de mensaje (IDs) y la realización de nodos encargadas de enviar y recibir mensajes, también se busca comprobar el correcto envió de datos de acuerdo a la prioridad del identificador del mensaje.

1.1 Trabajos previos

En [12] se realiza un trabajo sobre la creación de una red CAN Bus. Implementando un nodo basado en una placa Arduino UNO y una placa CAN Bus Shield, también se realiza la programación para la recepción y envío de mensajes de cuatro nodos en el Bus de comunicación de dicha red que se emplean para la monitorización de datos de la red CAN Bus de un vehículo y esta información se muestra en un display LCD. Por lo cual se utiliza dicho documento para iniciar con la realización de los nodos la Red CAN Bus, utilizando ambas placas para realizar la programación de cada uno de los nodos.

En algunos documentos [13]–[15] se tratan temas sobre los protocolos de comunicación en el vehículo y los identificadores de mensaje de cuadros de instrumentos de la familia Volkswagen. En [13] se presenta la manera de transmisión de mensajes por protocolo CAN Bus y se muestran algunos identificadores de mensaje sin mencionar el modelo al cual pertenecen. En [14] identifican algunos IDs de mensaje para un clúster de un automóvil VW Passat B6. En [16] se muestra una tabla de identificadores de mensaje de un VW Polo de los cuales el referente a las revoluciones por minuto concuerda. Mientras que en [15] se realiza el envío de mensajes a un clúster de instrumentos (VW Polo 6R). En [14], [15] se utilizan nodos formados por las tarjetas Arduino UNO y Shield y se emplea un panel de instrumentos similar al que se utilizará para la realización de la herramienta de aprendizaje interactivo. Existen muy pocos trabajos enfocados en controlar los parámetros de un cuadro de instrumentos debido a la protección de la información de Identificadores de mensajes y especificaciones técnicas a las que solo tienen acceso los fabricantes, esta limitación tiene gran parte de su justificación en mantener la seguridad del conductor, pues al ser información pública podría dar facilidad de intentos de falsificar identificadores de componentes críticos que perjudicarán al automovilista.

1.2 Objetivo General

Incorporar una Red CAN Bus a un clúster de instrumentos automotriz para el control de sus parámetros mediante sus identificadores de mensaje a fin de ser una herramienta de aprendizaje interactivo.

1.3 Objetivos específicos

- Estudiar y analizar el protocolo de comunicación CAN Bus en el vehículo.
- Establecer la metodología para la identificación de algunos testigos del clúster de instrumentos.
- Implementar un nodo y su código para el envío y recepción de datos
- Desarrollar una Red CAN Bus de 3 nodos e integración de un sensor a cada uno.
- Implementar la Red CAN Bus y el clúster de instrumentos.
- Evaluar él envió de mensajes en la Red de acuerdo a su prioridad.

1.4 Justificación

El CAN Bus es uno de los protocolos más utilizados para transmitir información en el vehículo, además de que es considerado un sistema de mayor seguridad debido a que utiliza cables para enviar la información [2], por estas razones es importante conocer la comunicación entre los nodos, es decir, como envían y reciben la información del automóvil. El cuadro de instrumentos es el encargado de mostrar al conductor la información del estado del vehículo, en este podemos encontrar el indicador de velocidad, el tacómetro y testigos luminosos de símbolos estandarizados los cuales tienen distinto significado, tal como la velocidad, las RPM, ABS, entre otros. El cuadro de instrumentos está conectado a los canales CA_H y CAN_L, por lo tanto, también puede comunicarse con los nodos del automóvil, este al igual que los nodos tiene un Identificador y puede recibir mensajes de los nodos encargados los sistemas del automóvil y utilizar esta información para encender uno de sus testigos o mover las agujas del tacómetro o velocímetro, para indicarle al conductor de alguna falla o simplemente encender un testigo informativo.

Otro motivo es poder controlar un clúster de instrumentos (Modelo 6J0920801X) utilizado en la industria automotriz a través de una Red CAN Bus a base de Arduino y Shield. En este trabajo se pretende controlar un cuadro de instrumentos mediante el protocolo de comunicación CAN Bus, pero debido a que en este protocolo los nodos necesitan un identificador primero necesitamos encontrar estos IDs de cada nodo de la Red, cada marca de vehículos tiene su propia base de datos de los identificadores de cada nodo, sin embargo es muy difícil acceder a esta base de datos, la única manera de conocerlos es en caso de ser un proveedor o si se tiene un convenio con la empresa automovilística, en nuestro caso no se cumple con ninguna de las dos condiciones por lo tanto al no poder acceder a una base de datos de los identificadores es necesario hace uso de la ingeniería inversa para poder obtenerlos. Con este trabajo no solo se espera poder controlar el cuadro de instrumentos sino también dejar una base de datos de los identificadores y la información que se transmite en cada uno de los bytes del mensaje.

El laboratorio de sistemas automotrices no cuenta con un módulo de aprendizaje de comunicación CAN, por lo cual se desea realizar una herramienta de aprendizaje interactivo que ayude a comprender mejor los temas de comunicación en el vehículo, los componentes que intervienen y las distintas condiciones que se deben de cumplir para tener una comunicación de los nodos en el vehículo, de esta manera se facilita el aprendizaje de estos temas. Con esta herramienta los alumnos de la carrera de ingeniería en sistemas automotrices podrán realizar mediciones e implementar más nodos, sensores y actuadores a la red.

El último propósito que motiva la realización de este trabajo es aplicar los conocimientos que se aprendieron durante la formación para realizar una herramienta de aprendizaje que ayude al aprendizaje de los alumnos de ingeniería en sistemas automotrices.

1.5 Metodología

Este trabajo presenta el control de un clúster de instrumentos modelo 6J0920 801X, el cual realiza su comunicación mediante CAN Bus, este tema se desarrolla para poder usarlo en la enseñanza de temas de comunicación en el vehículo para que los alumnos puedan comprender mejor el protocolo de comunicación CAN Bus, en la figura 1.5 se muestra un diagrama del procedimiento para realizar este trabajo, primero es necesario el estudio y comprensión del protocolo CAN Bus, los elementos que intervienen y la características que se deben de cumplir para este tipo de comunicación. También el estudio del panel de instrumentos y su comunicación con la Red CAN. Posteriormente para poder comunicar el clúster de instrumentos es necesario conocer los identificadores de mensaje de cada uno de los nodos y al no existir un estándar de identificadores debido a que cada marca tiene los propios, se emplea una metodología para la localización de los identificadores de mensaje. En la figura 1.6 se muestra esta metodología la cual inicia con la investigación y recaudación de identificadores de mensaje de diferentes vehículos y modelos, durante este primer paso también se analizan los datos comparándolos con los identificadores de [13]–[15]. Los cuales son tomados como referencias principales, después se selecciona el medio para la obtención de los identificadores de mensaje, para lo cual se emplea el NI-XNET Bus monitor de National Instruments con el cual se puede realizar la lectura de los mensajes en el Bus y a través de este también es posible enviar mensajes a la red, para entablar la comunicación se realiza la conexión del clúster al NI-XNET Bus monitor y se inicia el envío de mensajes al clúster utilizando los identificadores mencionados anteriormente hasta localizar los IDs con los que trabaja el panel de instrumentos, estas acciones se realizan varias veces debido a que la única manera que se tiene para saber si el id corresponde a un nodo que trabaja con el clúster de instrumentos es mediante el encendido de algún testigo o movimiento de alguna aguja del clúster, además de localizar los identificadores también se identifica el byte del mensaje en el que se envía la información, todo esto para realizar una base de datos de los identificadores del clúster utilizado.

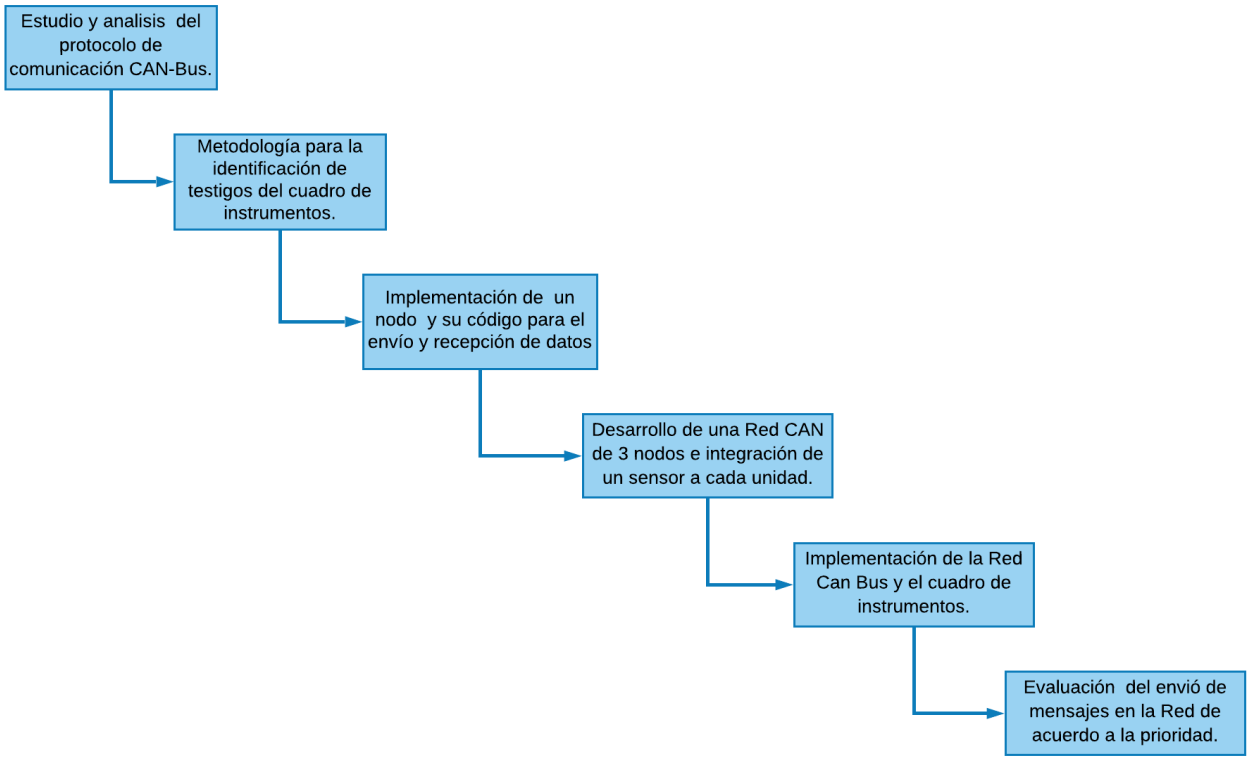


Figura 1.5 Diagrama de desarrollo del proyecto

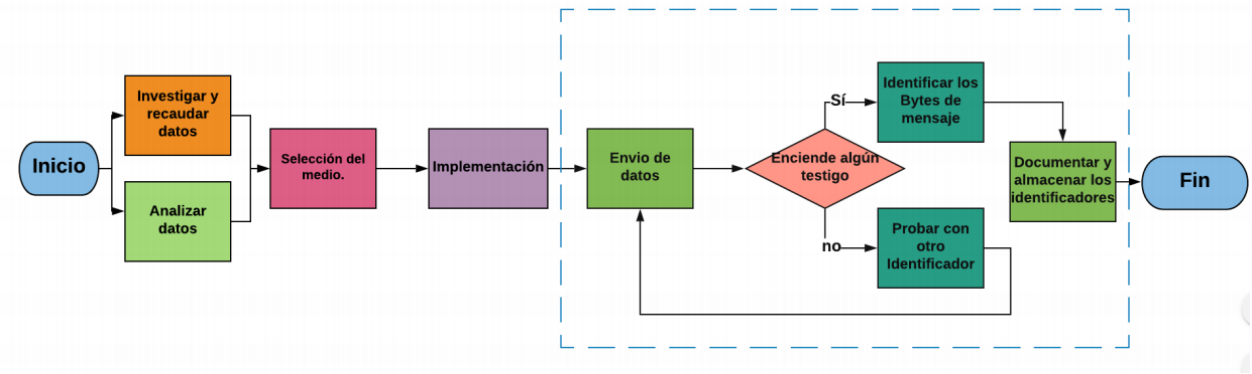


Figura 1.6 Metodología para la obtención de IDs de mensaje

Para implementar un nodo para el envío y recepción de mensajes se utiliza una placa CAN Bus Shield y una Arduino UNO. La placa CAN Bus Shield contiene un controlador CAN (Microchip MCP2515) y un transceptor CAN (MCP2551). Esta Shield proporciona conectividad CAN Bus a un Arduino mediante un controlador CAN Bus MCP2515 por SPI[17]-[18]. Mientras que el MCP2551 es un dispositivo que sirve como interfaz entre un

controlador de protocolo CAN y el Bus físico, además de que el MCP2551 es compatible con la norma ISO-11898[19]. La programación para el envío y recepción de mensajes del Bus se realiza en la IDE de Arduino.

Posteriormente se implementan 2 nodos más para formar la Red, a cada uno de los nodos se les asociará un sensor, debido a que los identificadores que se utilizan son los referentes a velocidad, Luces y RPM, los sensores que se emplean son una palanca de luces direccionales para las luces y un potenciómetro para velocidad y RPM. Seguidamente se une el clúster de instrumentos a la red, esto se hace mediante los canales CAN_H y CAN_L. Para realizar las pruebas de comunicación entre los nodos y la verificación del envió conforme a la prioridad se realizan pruebas utilizando el osciloscopio con analizador de protocolos para identificar las tramas de datos y la secuencia de las mismas, también se utilizara el NI-XNET Bus Monitor para leer todas las tramas de datos en el Bus, pero debido a que este no muestra la secuencia de envió se realizará un diagrama en LabVIEW para la lectura de los mensajes y comprobar que el orden de prelación se cumpla ya que como se muestra en la figura 1.7 se enviaran 3 mensajes, RPM, Velocidad y luces.

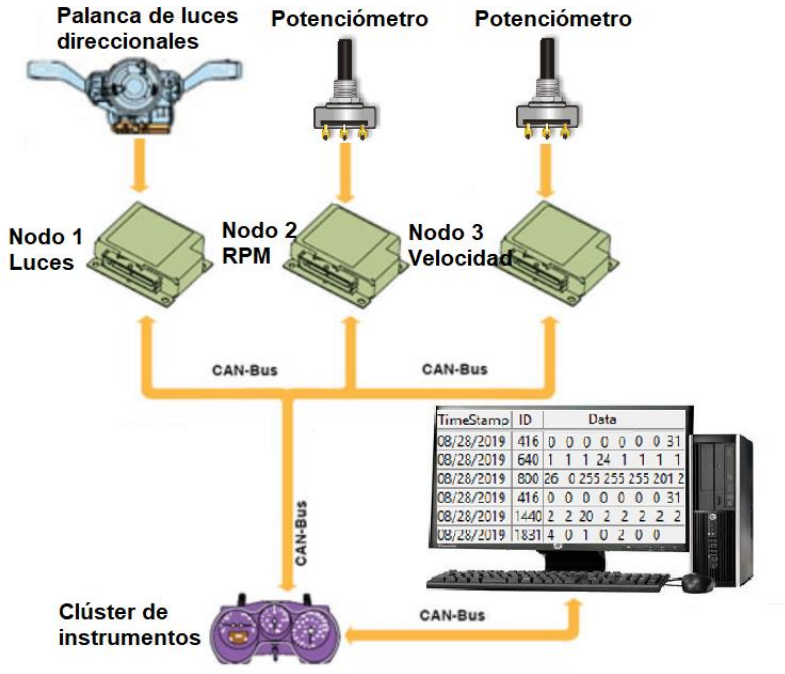


Figura 1.7 Conexión Red CAN Bus y clúster de instrumentos

1.6 Organización de la tesis

Capítulo 1

En este capítulo se describe el objetivo general y los objetivos específicos del trabajo, la justificación y la metodología con un diagrama de flujo de las diferentes actividades a realizar.

Capítulo 2

Se aborda el tema de los diferentes protocolos de comunicación en el vehículo, se describe el sistema OSI y el protocolo de comunicación CAN Bus, el cual se desprende del anterior mencionado. Se abordan las normas ISO que intervienen en el protocolo CAN.

Capítulo 3

En este capítulo se define la estructura interna del cuadro de instrumentos y los tipos de clúster existentes en el mercado. Se desarrolla el tema del sistema PXI y sus componentes

Capítulo 4

Se describen los elementos para la implementación de un Nodo CAN, las placas que se utilizan para realizar un nodo, así como la comunicación SPI entre estas placas.

Capítulo 5

En este capítulo se localizan los identificadores de mensaje del panel de instrumentos. También se realiza el código de cada nodo para el envío y recepción de mensajes, además se implementan diferentes sensores a cada uno de los nodos para simular distintas situaciones del automóvil y dependiendo de cada una encender un testigo del clúster.

Capítulo 6

Se realiza la implementación de la Red CAN Bus que comprende 3 nodos y el clúster de instrumentos, también se verifica la comunicación entre los diferentes nodos, la visualización de los testigos en el clúster al recibir los mensajes de los nodos, el envío de mensajes de acuerdo a la prioridad. Se realiza la lectura de la red usando tres métodos: osciloscopio con analizador de protocolos, NI-XNET Bus Monitor y LabVIEW. Se muestran los resultados y se dan las conclusiones.

CAPÍTULO 2

Protocolos de comunicación en el vehículo

El incremento de sistemas de confort en el auto, la protección contra robos, la mejora en prevención, detección o la forma de actuar ante un accidente, así como la implementación de seguridad activa y pasiva en el vehículo, ha generado el incremento tanto de sensores como nodos. Para comunicar a estos nodos existen diferentes protocolos de comunicación y dependiendo de las necesidades y costos que se deseen incrementar al vehículo se elige o eligen los sistemas de comunicación. El constante envío de información por parte de cada nodo es una necesidad del automóvil, ya que existen algunos sistemas que se encuentran interconectados y necesitan mutua información para actuar, por esta razón es importante que la velocidad de transmisión del mensaje sea elevada para asegurar que se reciba la información en tiempo real y actuar en caso de una situación inesperada.

Los sistemas electrónicos y componentes del automóvil se encuentran conectados mediante una Red con el fin de que se realice la comunicación, operaciones, toma de datos del vehículo y diagnóstico de una manera segura. El tipo de red que facilita la comunicación tanto dentro como fuera del auto es la del Bus serial, la mayoría de los automóviles cuentan con este tipo de sistema de comunicación, sin embargo, existen más protocolos los cuales tiene la misma finalidad, pero diferentes características como la velocidad de transmisión de datos, el costo, la seguridad etc. A continuación, se describen los distintos protocolos de comunicación en el automóvil.

2.1 Protocolo SAE J1850

El estándar SAE para las clases A y B (velocidad de transmisión baja y media), es una combinación del SCP de Ford y del protocolo de Clase 2 de General Motors. El protocolo SAE J1850 se adoptó en 1994 y aún se puede encontrar en algunos de los vehículos actuales, por ejemplo, algunos vehículos de General Motors y Chrysler. Estos sistemas de Bus son más antiguos y más lentos que CAN, pero más baratos de implementar. Los dos tipos de protocolos J1850 son: modulación de ancho de pulso (PWM) y ancho de pulso variable (VPW)[5], [20]. La figura 2.1 muestra dónde encontrar los pines PWM en el conector OBD-II. VPW utiliza solo el pin 2.

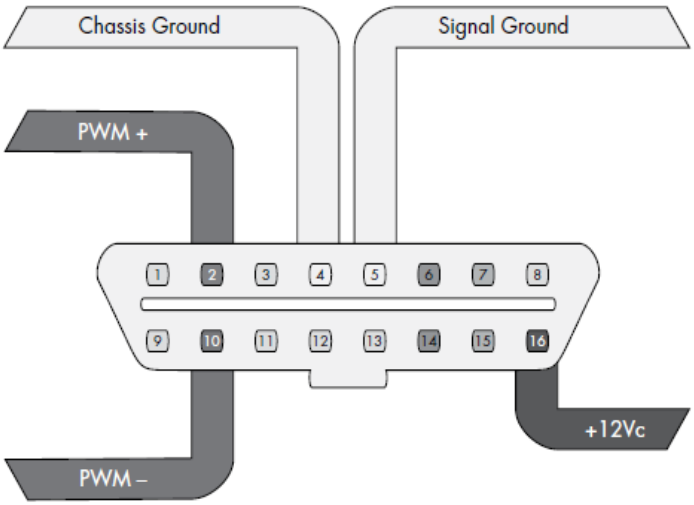


Figura 2.1 Pines en el conector OBDII del protocolo PWM [4]

2.1.1 SAE J1850 PWM

PWM usa señalización diferencial en los pines 2 y 10 y es usado principalmente por Ford. Funciona con un alto voltaje de 5V y a 41,6 kbps, y utiliza señalización diferencial de dos cables. PMW tiene una señal de bit fijo, por lo que un 1 es siempre una señal alta y un 0 siempre es una señal baja[5].

2.1.2 SAE J1850 VPW

VPW, un sistema de Bus de un solo cable, usa solo el pin 2 y es generalmente usado por General Motors y Chrysler. VPW tiene un alto voltaje de 7 V y una velocidad de 10.4 kbps. Dado que VPW usa señalización dependiente del tiempo, la recepción de 1 bit no se determina sólo por un alto potencial en el Bus. El bit debe permanecer alto o bajo durante un período de tiempo establecido para que se considere un solo bit 1 o un bit 0. Llevar el autobús a una posición alta lo pondrá a aproximadamente 7 V, mientras que el envío de una señal baja lo pondrá a nivel del suelo o cerca del suelo. Este Bus también se encuentra en una etapa de reposo o sin transmisión, a un nivel cercano al suelo (hasta 3 V)[5].

2.2 Protocolo ISO 9141-2

Este protocolo tiene una tasa de datos serie asíncrona de 10,4 kbps. Las comunicaciones en serie usan niveles de tensión altos y bajos para transmitir bits, un cero binario está representado por un nivel de tensión de cero voltios y un uno binario está representado por un nivel de tensión 12 voltios. ISO 9141-2, o línea k usa el pin 7 y, opcionalmente, el pin 15, en la figura 2.2 se muestran los pines del OBDII que utiliza este protocolo. Los paquetes K-Line tienen una fuente (transmisor) y una dirección de destino (receptor). K-Line puede usar la misma estructura de solicitud de ID de parámetro (ID) o una similar a la de CAN[5], [20]

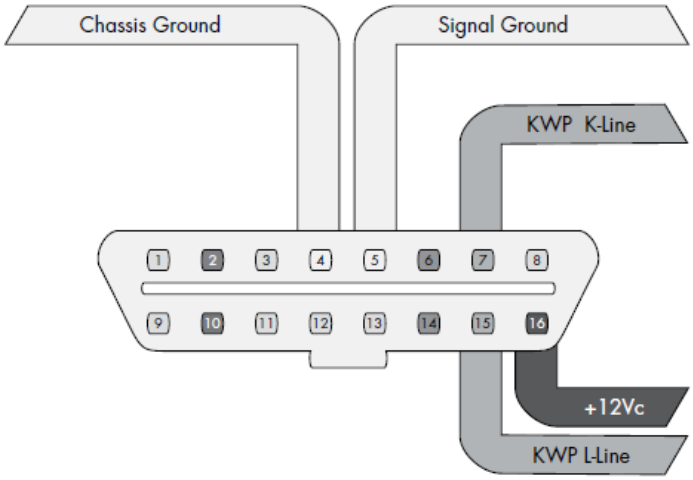


Figura 2.2 Protocolo ISO 9141-2 pin en el OBDII [4]

2.3 Protocolo LIN

El nombre, LIN (Red de Interconexión Local), se deriva del hecho de que todos los nodos están ubicados dentro de un espacio de instalación demarcado (por ejemplo, en la puerta). El LIN, por lo tanto, es un subsistema local para soportar la red del vehículo por medio de redes CAN de alto nivel. La red de interconexión local es el más barato de los protocolos del vehículo. Fue diseñado para complementar a CAN. No tiene código de arbitraje o de prioridad; en cambio, un solo nodo maestro hace toda la transmisión. LIN puede admitir hasta 16 nodos esclavos que principalmente solo escuchan el nodo maestro. Necesitan responder de vez en cuando, pero esa no es su función principal. La velocidad máxima de LIN es de 20 kbps. LIN es un Bus de un solo cable que funciona a 12V, esta tensión es bastante alta respecto a otros tipos de redes, pero se utiliza debido a que es más fácil para la electrónica del sistema eliminar la mayoría de interferencias que provienen del exterior. Los suscriptores de Bus generalmente están dispuestos en una topología de Bus lineal y

conectados entre sí por una línea de un solo cable[3], [5], [20]. En la figura 2.3 se muestra un ejemplo del protocolo de comunicación LIN y su conexión al protocolo CAN, también se pueden observar los esclavos y las funciones que realizan.

Figura 2.3 Protocolo LIN-Bus y su integración al CAN Bus [20].

2.4 Protocolo MOST

El Bus MOST se diferencia de los demás debido a que utiliza fibra óptica para el intercambio de información, la razón por la cual se optó por el uso de este material fue que al sustituir la corriente eléctrica por haces de luz y los cables por fibra óptica las dificultades que se presentaban debido a las radiaciones electromagnéticas ya no se presentan. Este protocolo se hizo específicamente para la red de sistemas de información y entretenimiento en vehículos motorizados como se visualiza en la figura 2.4. Además de las funciones de entretenimiento tradicionales, ofrecen funciones de video (DVD y TV), capacidades de guía de ruta y acceso a las comunicaciones e información móviles[3], [5], [21].

La transmisión de datos multimedia, tanto de audio como de vídeo, requiere una alta velocidad de datos, por lo que este Bus ofrece una velocidad de datos de 24.8 Mbps. Por lo general, MOST se presenta en una topología en anillo, o estrella virtual, que admite un máximo de 64 dispositivos [3], [5].

Figura 2.4 Red MOST de imagen y sonido en Audi [20]

2.5 Protocolo FlexRay

El objetivo de FlexRay es proporcionar un sistema con altas tasas de transferencia que funcionará de manera determinista y tolerante a fallas, a la vez que sea lo más flexible posible de usar y expandir. FlexRay es un Bus de alta velocidad que puede comunicarse a velocidades de hasta 10 Mbps. Está orientado a la comunicación sensible al tiempo por esta razón se utiliza en los sistemas de transmisión como se muestra en la figura 2.5, este protocolo usa cableado de par trenzado pero es más costoso de implementar que CAN, por lo que se usa para sistemas de gama alta[3].FlexRay admite una topología de Bus estándar, como CAN Bus, también admite topología en estrella, como Ethernet, que puede ejecutar segmentos más largos[3], [5].

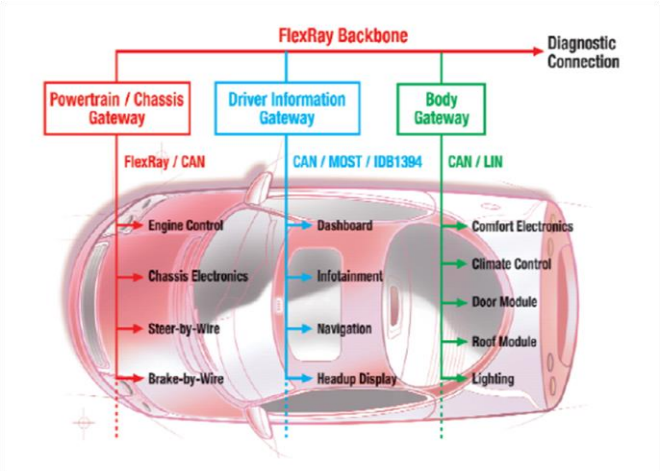


Figura 2.5 Comunicación FlexRay [20]

2.6 Protocolo CAN Bus

A comienzos de 1980 muchos sistemas electrónicos comenzaron a aparecer en la industria automotriz. Se añadieron un gran número de sensores al automóvil y se mejoraron las unidades de control de motor, tal fue el avance que a día de hoy podemos encontrar más de 200 sensores en un automóvil de gama media. Las compañías manufactureras de vehículos se vieron interesados en comunicar estos sistemas electrónicos entre sí, debido la ausencia de un Bus que fuera capaz de proveer una comunicación veloz multimaestro que pudiera operar correctamente en distancias y con un bajo costo, en 1983 Robert Bosch GmbH tomo la decisión de desarrollar un protocolo de comunicación orientado a sistemas distribuidos que operará en tiempo real, el desarrollo de CAN había comenzado (tabla 1). El enfoque principal de ese desarrollo fue un sistema de comunicación entre un número de nodos dentro de un vehículo Mercedes-Benz. El involucramiento de esta manufacturera automotriz, así como la manufacturera de semiconductores Inter y algunas universidades ayudaría al desarrollo exitoso de CAN. El estándar de CAN fue introducido en 1986 durante el congreso de la SAE en Detroit, Michigan. Los primeros chips controladores de CAN fue el Intel 82526 y el Phillips 82C200, introducidos en 1987[22].

Tabla 1 Historia del protocolo CAN Bus [21]

1983	*Comienza el desarrollo de CAN en Robert Bosch GmbH.*
1985	*Especificación de CAN V1.0.*
1986	*Comienza la estandarización ISO.*
1987	*Introducción del primer prototipo del circuito integrado de CAN.*
1989	*Comienzo de la primera aplicación industrial.*
1991	*Especificación de CAN extendido 2.0.* *Lanzamiento del primer vehículo - Mercedes class S, 5 unidades comunicadas*
1992	*Creación de CiA (Can en Automatización).*
1993	*Creación de grupo OSEK (Sistemas abiertos y sus interfaces para electrónicos en vehículos de motor).*
1994	*Primera estandarización ISO.*
1995	*Grupo de trabajo en los Estados Unidos con la SAE.*
1996	*CAN es aplicado en la mayoría de sistemas de control del motor.*
1997	*Todos los mayores productores de chips venden componentes de CAN.*
1998	*Nuevos set de estándares (Diagnósticos, conformidad, etc.).*
1999	*Fase de desarrollo de CAN activado por tiempo (TTCAN).*
2000	*Explosión de CAN en todos los equipamientos vehiculares.*
2001	*Introducción de CAN activado por tiempo en tiempo real (TTCAN).*

CAN es un protocolo simple utilizado en la fabricación y en la industria del automóvil. CAN ha sido un estándar en automóviles y camiones ligeros de EE. UU. En 1991, el Bus CAN (Controller Area Network) fue el primer sistema de Bus que se introdujo en un vehículo motorizado en producción en masa (tabla 1). Desde entonces, se ha establecido como el sistema estándar en el sector automotriz, pero el Bus CAN también se usa comúnmente como Bus de campo en la ingeniería de automatización en general.

2.6.1 Elementos del Protocolo CAN

Cables: La información circula por dos cables trenzados que unen todos los nodos que forman el sistema. Si las líneas se trenzan, los impulsos de perturbación tienen el mismo efecto en ambas líneas. Por lo tanto, la transferencia de datos diferenciales permite filtrar las interferencias en la línea [3], [13]. La información que se envía a través de los cables es transmitida por diferencia de tensión entre los dos cables, si uno de las dos fallas, el que sigue en funcionamiento se compara con tierra y el sistema se mantiene operando con uno de los cables.

Resistencias de terminación: Son resistencias conectadas a los extremos de los cables del Bus, que permiten adecuar el funcionamiento del sistema a diferentes longitudes de cables y número de nodos, ya que impiden fenómenos que pueden afectar el mensaje.

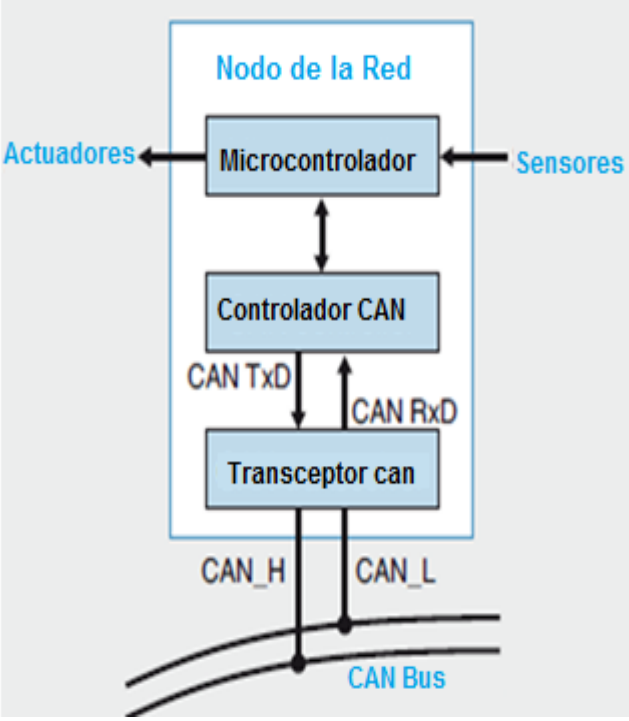


Figura 2.6 Componentes de un Nodo [4]

Nodos: Una Red CAN está compuesta de una serie de dispositivos conectados mediante un Bus serie, denominados nodos[23]. La forma de comunicación de dichos nodos es broadcast. Un nodo de red (figura 2.6) comprende el microcontrolador para el software de aplicación, el controlador CAN y el transceptor CAN (controlador de Bus).

Controlador: El controlador CAN es responsable de la comunicación entre el microprocesador y el transceptor, se encarga de los modos de transmisión y recepción. Genera el flujo de bits para la comunicación de datos desde los datos binarios que se van a transmitir y los envía al transceptor en la línea TxD. Este

amplifica las señales, genera el nivel de voltaje requerido para la transferencia de datos diferenciales y transmite el flujo de bits procesado en serie en la línea de Bus (CAN_H y CAN_L). Determina la velocidad de trasmisión de los mensajes, que será más o menos elevada, según el compromiso del sistema. Así, en la línea de CAN Bus del motor, frenos, o cambio automático es de 500 kbps; y en el sistema de confort a una velocidad mucho menor.

Transceptor: El transceptor CAN es un amplificador de transmisor y receptor, convierte los estados lógicos 0 y 1 recibidos por el controlador CAN en niveles de voltaje que se alimentan a las líneas de Bus CAN_H y CAN_L y viceversa. La comunicación que provee un transceptor solo puede ser Semiduplex, lo que significa que pueden enviarse señales entre dos terminales en ambos sentidos, pero no simultáneamente, es decir un nodo no podría transmitir si los otros nodos están también transmitiendo porque su transceptor estaría recibiendo en ese momento el mensaje. Algunos transceptores también evalúan el nivel de voltaje en las líneas CAN_H y CAN_L por separado. Entonces, la operación podría continuar en modo de una sola línea si una de las dos líneas de Bus fallara como consecuencia de un cortocircuito o una rotura del cable[3], [24]. Los mensajes entrantes son procesados por el transceptor y enviados al controlador CAN en la línea RxD. El microcontrolador, que ejecuta el programa de aplicación, controla el controlador CAN, prepara los datos que se enviarán y evalúa los datos recibidos. Los estados del Bus lógico y la codificación CAN utilizan dos estados para la comunicación, el dominante y el recesivo, con los que se transmiten los bits de información. El estado dominante representa un "0" binario, el "1" binario recesivo.

Sensores: Los nodos para realizar sus tareas están equipados con sensores y actuadores (figura 2.6). Los sensores se encargan de medir temperaturas, presiones, rotaciones y una gran cantidad de parámetros relacionados con el funcionamiento del vehículo. La información captada por estos sensores es enviada y almacenada en los nodos, una vez allí esta información es comparada con los valores óptimos que están almacenados en las memorias. Cuando se encuentra un valor incorrecto, el nodo notifica un fallo avisando al conductor de alguna forma (indicadores luminosos, sonidos, mensajes de alerta, etc.), los fallos quedan almacenados para su posterior verificación por personal calificado.

2.6.2 Funcionamiento del protocolo CAN

El protocolo CAN funciona con dos cables: CAN alto (CAN_H) y CAN bajo (CAN_L). CAN utiliza señalización diferencial lo que significa que cuando llega una señal, CAN aumenta el voltaje en una línea y deja caer la otra línea en una cantidad igual. La señalización diferencial se utiliza en entornos que deben ser tolerantes a las fallas del ruido, como en los sistemas automotrices y la fabricación. Los nodos tienen un transceptor que

verifica que ambas señales se activen; Si no lo hacen, el transceptor rechaza el paquete como ruido[3].

Los Buses CAN se pueden clasificar en dos tipos: CAN de alta velocidad (hasta 1 Mbps) y CAN de baja velocidad tolerante a fallos (hasta 125 kbps). El CAN de alta velocidad tiene un voltaje de reposo de 2.5V, cuando entra una señal, sumará o restará 1 V (3.5V o 1.5V). Por otro lado, el CAN de baja velocidad tiene un voltaje en reposo con CAN_H: 0V Y CAN_L=5V y una tensión con CAN_H: 3.5V, CAN_L: 1.5V. En la figura 2.7 se muestran dos niveles lógicos (recesivo y dominante) interpretados por los nodos y los niveles de voltaje del CAN dependiendo si es de alta o baja velocidad.

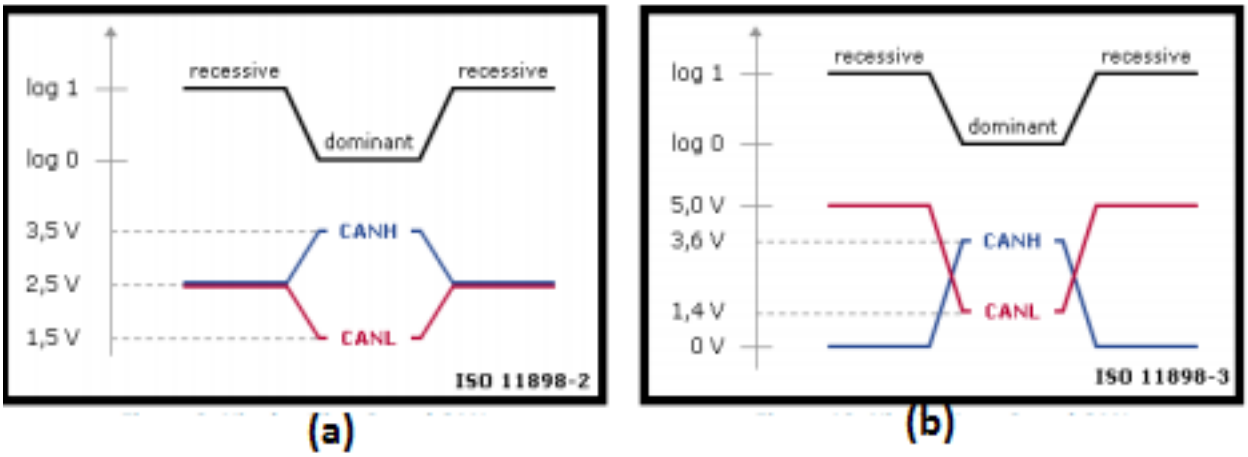


Figura 2.7 (a) Nivel de voltaje del CAN Bus de alta velocidad. (b) Nivel de voltaje CAN Bus de baja velocidad [16]

El tipo de cable utilizado en esta red para realizar la transmisión de datos es UTP esta denominación hace referencia a las siglas de las palabras inglesas Unshielded Twisted pair que significa “par trenzado sin protección”, la transmisión UTP formada por dos cables de 0,6 mm^2 que se encuentran trenzados entre sí, al ser una topología de línea los extremos tienen conectadas resistencias llamadas “terminadoras” y tienen el objetivo de atenuar las variaciones de tensión para evitar tensiones parásitas.

El valor de las resistencias depende del número de nodos que se conecten y la velocidad de transmisión. Para velocidades de transmisión alta tiene un valor de 120 ohmios (figura 2.8.a), a diferencia del Bus de alta velocidad, el Bus de baja velocidad requiere dos resistencias en cada transceptor: RTH para la señal CAN_H y RTL para la señal CAN_L (figura 2.8.b). Esta configuración permite al transceptor de Bus de baja velocidad detectar fallas en la red. La suma de todas las resistencias en paralelo, debe estar en el rango de 100-500 ohmios. El par de cables trenzados (CAN_H y CAN_L) constituyen una transmisión de línea. Si dicha transmisión de línea no está configurada con los valores correctos, cada

trama transferida causa una reflexión que puede originar fallos de comunicación. Como la comunicación en el Bus CAN fluye en ambos sentidos, ambos extremos de red deben de estar cerrados mediante una resistencia[3], [17].

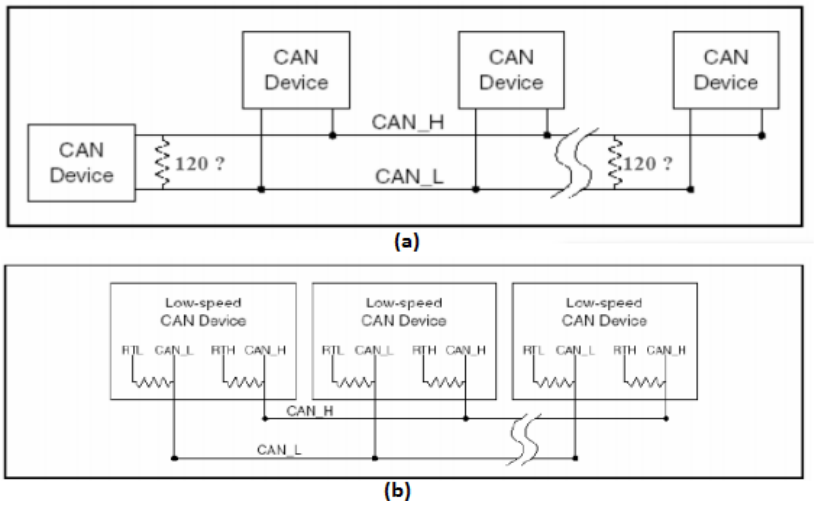


Figura 2.8 Resistencias terminadoras. (a) Bus de alta velocidad. (b) Bus de baja velocidad [16].

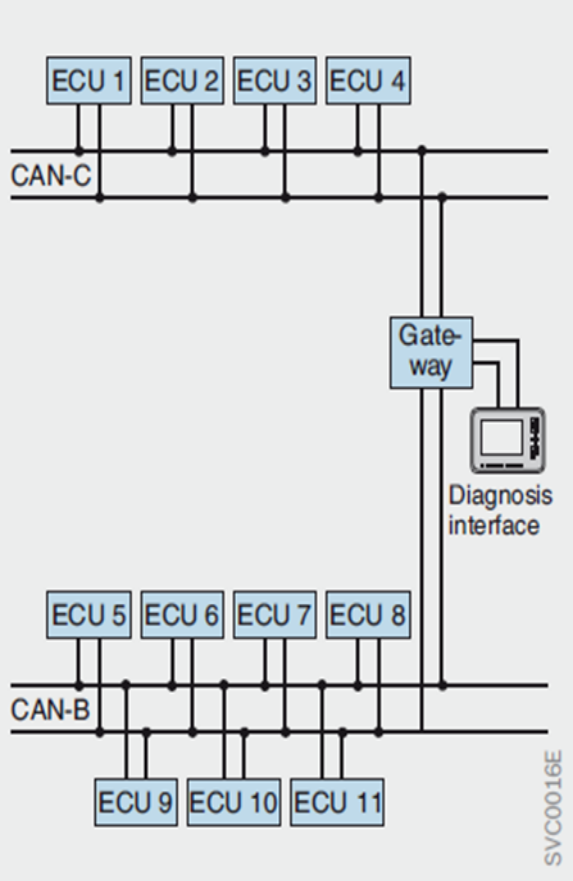
Los cables CAN pasan por el vehículo y se conectan entre los nodos y otros sensores, y siempre están en pares de dos cables. El protocolo CAN admite la comunicación entre nodos de red sin la necesidad de una unidad de control central. Cada nodo puede intentar enviar mensajes en cualquier momento. Si este intento tiene éxito o no depende esencialmente de dos factores: que el Bus esté libre antes del inicio de la transmisión y pasar con éxito la fase de arbitraje. CAN no se dirige a los nodos de red individuales sino a los mensajes que se han enviado. Cada mensaje tiene un marcador único o identificador[17].

El identificador clasifica el contenido del mensaje, por lo tanto, una estación puede transmitir un mensaje a todas las demás estaciones. Estas estaciones solo leen aquellos mensajes cuyos identificadores están almacenados en su lista de aceptación. De esta manera, cada estación decide por sí misma si necesita o no un mensaje enviado en el Bus. El identificador tiene 11 bits (formato estándar, CAN 2.0 A) o 29 bits (formato extendido, CAN 2.0 B). Con 11 bits en el formato estándar, es posible distinguir entre 2,048 mensajes CAN diferentes; En el formato extendido, este número se eleva a más de 536 millones. La ventaja de este método de direccionamiento es que los nodos de la red no requieren ninguna información sobre la configuración del sistema y, por lo tanto, son libres de operar de forma totalmente independiente entre sí. Esto da como resultado un sistema completo altamente flexible, que facilita la administración de las variantes de los equipos. Si uno de los nodos requiere información nueva que ya está en el Bus, todo lo que debe hacer es llamarla desde el Bus. Es posible integrar estaciones adicionales en el sistema sin tener que modificar las

estaciones existentes. Las ventajas del protocolo CAN Bus son: económico y sencillo, estandarizado, con un medio de transporte adaptable, una estructura definida, tiene una relación de velocidad distancia, está orientado al mensaje, multicast (todos los nodos acceden al Bus simultánea, broadcasting (la información se envía a todos los nodos de forma simultánea), detección y señalización de errores, jerarquía multimaestro[5], [20].

2.6.3 Aplicaciones

El Bus CAN se utiliza en varios dominios en el vehículo motorizado. Estos dominios difieren en los requisitos que exigen de la red. Debido a los procesos rápidos involucrados en el área de administración del motor, la información se requiere mucho más rápido aquí que en el área de comodidad / conveniencia donde los sistemas controlados están ubicados más alejados y, como tales, las líneas son más propensas a dañarse. Como resultado de estos diferentes requisitos, se utilizan Buses con diferentes tasas de datos que ofrecen una relación óptima de costo-beneficio para el campo de aplicación en cuestión[3]. En la figura 2.9 se aprecia un esquema de interconexión de nodos, la unidad de interconexión al ser un elemento tan importante dentro de la instalación eléctrica, cada fabricante le ha puesto un nombre propio, en este caso es nombrada GATEWAY por el grupo Volkswagen, BMW y Porsche. La unidad central juega un papel fundamental al permitir el paso de información desde la red de un subsistema a cualquier otra[21].



Bus CAN, se distingue entre Buses CAN de alta velocidad y de baja velocidad como se muestra en la figura 2.9.

CAN de alta velocidad (CAN-C) funciona a velocidades de bits de 40 kbps a 1 Mbps. Por lo tanto, la transferencia de datos puede cumplir los requisitos en tiempo real del tren motriz. Los Buses CAN-C se utilizan para conectar en red los siguientes sistemas: Sistema de gestión del motor, Control de transmisión electrónica, Sistemas de estabilización de vehículos (por ejemplo, ESP), el grupo de instrumentos.

Figura 2.9 CAN de alta velocidad (CAN-C) Y CAN de baja velocidad (CAN-B) [4].

CAN de baja velocidad (CAN-B) funciona a una velocidad de bits de 40 a 125 kbps. Se utiliza para muchas aplicaciones en la comodidad esta velocidad es suficiente para cumplir con los requisitos de tiempo real exigidos en esta área. Ejemplos de tales aplicaciones son: Control del sistema de climatización, Ajuste del asiento, unidad de ventana de energía, Control de techo corredizo, Ajustador de espejo, sistema de iluminación, Control del sistema de navegación[5].

2.6.4 El Modelo OSI y el Protocolo CAN Bus

El modelo OSI fue desarrollado en 1984 por la organización ISO (International Organization for Standardization). Este estándar tiene el objetivo de conseguir interconectar sistemas de distinta procedencia para que puedan intercambiar información sin ningún tipo de impedimentos debido a los protocolos con los que estos operan. El modelo OSI está conformado por 7 capas, cada una de ellas tiene una función para que en conjunto sea posible alcanzar el objetivo de interconexión entre sistemas[17].

El protocolo CAN está definido por las dos primeras capas del Modelo OSI: capa de enlace de datos y capa física. Las capas física y enlace de datos están completamente definidas en la especificación de CAN[25]. En la figura 2.10 se visualizan las capas del modelo OSI que utiliza el protocolo de comunicación CAN Bus.

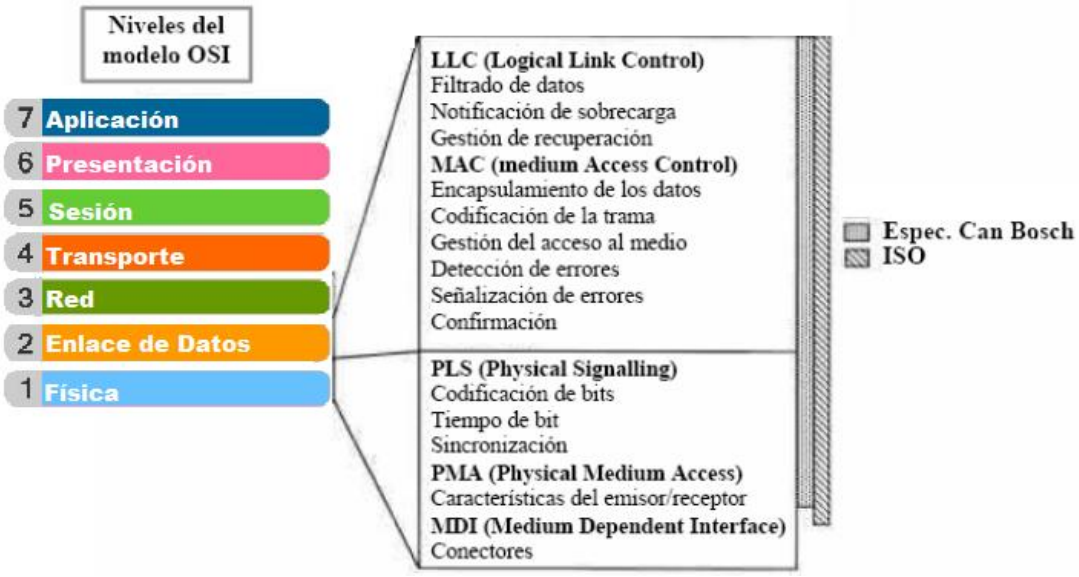


Figura 2.10 Sistemas de Bus automotriz en el modelo OSI: CAN Bus.

Capa física

La capa física de CAN, es responsable de la transferencia de bits entre los distintos nodos que componen la red. Define aspectos como niveles de señal, codificación, sincronización y tiempos en que los bits se transfieren al Bus. En la especificación original de CAN, la capa física no fue definida, permitiendo a los diseñadores diferentes opciones para la elección del medio y niveles eléctricos de transmisión. Las características de las señales eléctricas en el Bus, fueron establecidas más tarde por el ISO 11898. Para el Bus de alta velocidad se tienen velocidades de transmisión que van desde 40 kbps hasta 1 Mbps, dependiendo de la longitud del cable. Para el CAN de baja velocidad/ tolerante a fallos se tienen velocidades de transmisión que van desde 40 kbps hasta 125 kbps [3], [24], [25].

La capa física CAN se divide en tres subcapas:

- Señalización física (PLS, por sus siglas en inglés): se encarga de la codificación, sincronización y el tiempo de un bit.
- Acceso al medio físico (PMA, por sus siglas en inglés): Esta subcapa se encarga de describir las características del receptor.
- interfaz dependiente del medio (MDI, por sus siglas en inglés): Esta subcapa especifica las características de los cables y conectores utilizados en este protocolo de comunicación[26].

Capa de enlace

La capa de enlace describe como la información será transmitida entre los diferentes nodos de la red, es responsable del acceso al medio y el control lógico[17]. La capa de enlace estada integrada por dos subcapas:

- Control de enlace lógico (LLC, por sus siglas en inglés): esta subcapa es encargada de los filtros de los mensajes, proporciona medios para el restablecimiento y para notificar la sobrecarga del Bus.
- Control de acceso al medio (MAC, por sus siglas en inglés): esta subcapa es responsable de la trama de mensajes, el arbitraje, el reconocimiento, la detección de error y señalización. También se encarga de indicar si el Bus se encuentra libre para iniciar la transmisión de un mensaje o si la recepción ha iniciado[26].

2.6.5 Codificación de bits NRZ

La codificación de bits de no retorno a cero (NRZ, por sus siglas en inglés), se utiliza como método de codificación para la transmisión de datos. Con este método, no hay retorno

obligatorio a cero entre dos estados de transmisión del mismo valor. Lo cual significa que las señales binarias a transmitir se asignan directamente: una lógica "1" a un nivel bajo, una lógica "0" a un nivel alto. La característica de la codificación NRZ es que los bits consecutivos de la misma polaridad no presentan cambios de nivel. Es por esta razón que la codificación NRZ permite velocidades de datos muy altas y mantiene las emisiones dentro de los límites.

Sin embargo, la codificación NRZ no tiene propiedades de sincronización. Si no se produce un cambio de nivel durante un período de tiempo más largo, el receptor pierde la sincronización. Es por eso que el uso de la codificación NRZ requiere un mecanismo de sincronización explícito, el cual consiste en insertar un bit complementario (bit stuffing) en el flujo de bits después de cinco bits idénticos como se muestra en la figura 2.11. Este bit no cambia el mensaje ya que se borra en el nodo receptor.

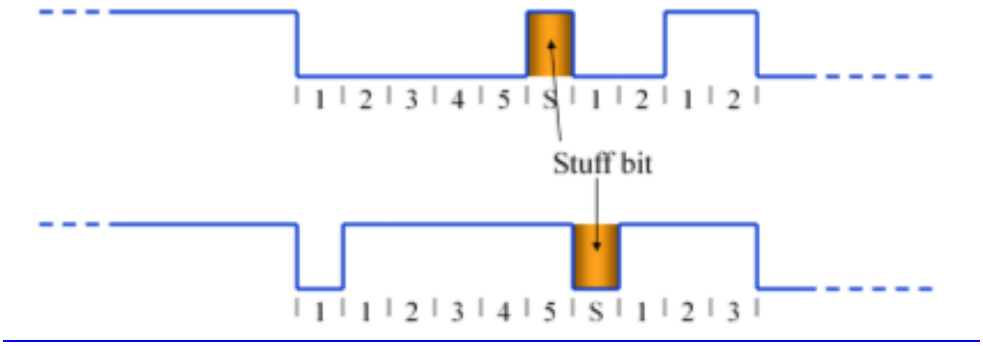


Figura 2.11 Bit stuffing

2.6.6 Controlando el acceso al Bus

Si el Bus está desocupado (estado recesivo) y los mensajes están disponibles para su envío, cada estación puede iniciar el envío de su mensaje. Pero si dos o más nodos comienzan a enviar mensajes al mismo tiempo el conflicto de acceso al Bus se soluciona mediante un arbitraje utilizando el identificador de cada uno de los nodos (figura 2.12). El método del control de acceso al medio se denomina “Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority” (CSMA/CD + AMP). Se utiliza el identificador para definir la prioridad del mensaje, garantizando que el mensaje de mayor prioridad sea enviado primero. Al mensaje con la prioridad más alta (el valor binario más bajo del identificador) se le asigna el primer acceso, sin ninguna pérdida o demora de datos (protocolo no destructivo).

El mensaje comienza con un bit dominante (bit de inicio de trama), seguido del identificador. Cuando varias estaciones comienzan a transmitir simultáneamente, el sistema responde empleando el arbitraje. Durante el arbitraje, cada uno de los nodos que quieren

enviar su mensaje al Bus compara el nivel de bit a ser transmitido con el nivel que es monitoreado en el Bus. Si los niveles son iguales, entonces el nodo puede continuar transmitiendo el próximo bit. Cuando un nodo transmite un bit con nivel recesivo y es monitoreado un nivel dominante, el nodo pierde el arbitraje y debe pasar a un estado de "oyente" del Bus. Este método es conocido como "Wired-AND" en el que, los "bits dominantes" (nivel lógico "cero") sobrescriben a los "bits recesivos" (nivel lógico "uno")[29].

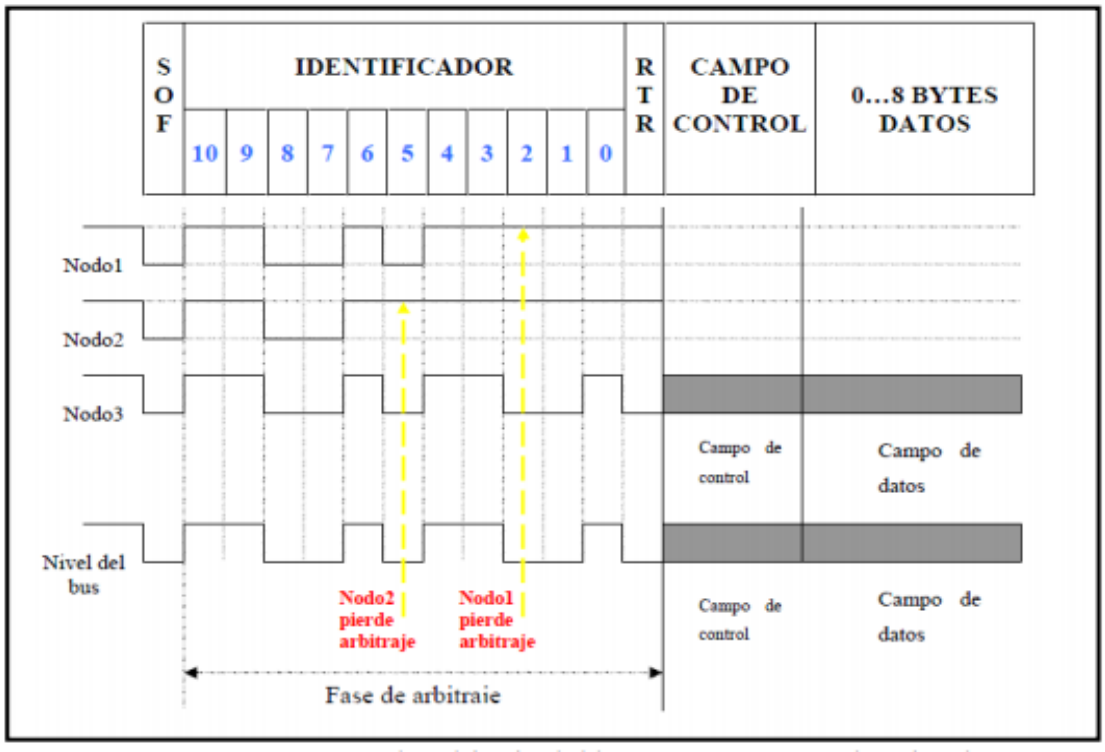


Figura 2.12 Arbitraje en una Red CAN Bus de 3 nodos [23]

En la figura 2.12 se puede ver que cada nodo que intenta enviar un bit recesivo, pero encuentra un bit dominante pierde el proceso de arbitraje como es el caso del nodo 2 en el quinto byte y el nodo 1 en el segundo byte. La estación con el identificador más bajo, es decir, la prioridad más alta (nodo 3), se abre paso en el Bus sin tener que repetir el mensaje (control de acceso no destructivo). Los transmisores de mensajes de prioridad más baja se convierten automáticamente en destinatarios del mensaje que acaba de enviar otra estación. Repiten su intento de enviar tan pronto como el autobús vuelva a estar libre. Sin este control de acceso, las colisiones de Bus darían como resultado fallas. Para garantizar un arbitraje de Bus inequívoco, por lo tanto, no está permitido que más de un nodo envíe un mensaje con el mismo identificador. Con este método de acceso, los mensajes de máxima prioridad solo tienen que esperar la transferencia del mensaje que se está enviando actualmente y, con tiempos de bit de 130 (CAN 2.0 A) o 150 (CAN 2.0 B), tienen la latencia más baja.

Para que todos los mensajes tengan la oportunidad de acceder al Bus, la velocidad de transferencia de datos debe coincidir con el número de suscriptores de Bus[3].

2.6.7 Manejo y Detección de errores

Si un controlador CAN detecta una falla o error de formato, interrumpe la transmisión actual al enviar una trama de error que comprende seis bits dominantes sucesivos. Esto rompe la regla de relleno que prohíbe este tipo de secuencia de bits. Si el remitente detecta que su mensaje ha sido interrumpido por un cuadro de error, deja de transmitir y realiza otro intento más adelante. Este efecto evita que otras estaciones acepten el mensaje erróneo y, por lo tanto, garantiza la consistencia de los datos en todo el sistema. Las estaciones defectuosas podrían tener un gran peso en el tráfico del Bus si enviaran con frecuencia mensajes erróneos o interrumpieran la transmisión de los mensajes correctos enviando repetidamente un cuadro de error. El protocolo CAN localiza las fallas de la estación mediante análisis de errores estadísticos. Una estación reconoce la probabilidad de su propio mal funcionamiento por la frecuencia con la que aborta los mensajes antes de que otras estaciones envíen un cuadro de error. La primera medida del protocolo en este caso es evitar que una estación como ésta continúe abortando las transmisiones. En caso de emergencia, la estación se apaga automáticamente[3].

Para detectar errores CAN emplea dos métodos, uno a nivel de mensaje y otro a nivel bit.

A nivel de mensajes el protocolo de comunicación CAN implementa tres mecanismos: chequeo de redundancia cíclica (CRC), errores de verificación de trama y acuse de recibo (ACK).

También implementa mecanismos para detectar errores a nivel de bit, estos son: Monitorización y Relleno de bits.

Monitorización: Cada nodo monitorea el nivel del Bus con el fin de detectar errores, si hay una diferencia entre el bit transmitido y recibido, se detecta un error en los bits.

Relleno de bits: Esta técnica consiste en insertar un bit complementario después de una sucesiva transferencia de cinco bits idénticos. Esta técnica se emplea en casi todos los campos de la estructura de datos, desde la transmisión del SOF hasta la transmisión del último bit de la secuencia CRC. Este bit insertado, se borra en el receptor[26], [29].

2.6.8 Estados de Error

Cuando un nodo funciona en forma defectuosa debe evitarse que este envíe mensajes, ya que pueden estar corrompidos. CAN provee un mecanismo para prevenir esto. Cada nodo posee dos contadores de error: REC (Receive Error Counter) que cuenta el número de

errores de transmisión en las tramas recibidas por el nodo y TEC (Transmit Error Counter) que cuenta el número de errores de transmisión en las tramas enviadas por el nodo. Cada vez que una trama es transmitida o recibida correctamente por un nodo, el contador correspondiente implementa una cuenta regresiva. De igual forma, cuando un error de transmisión es detectado produce un incremento en el contador correspondiente. El nodo puede encontrarse en uno de tres estados, de acuerdo al valor de ambos contadores:

- Error Activo: el nodo puede recibir y enviar tramas normalmente. Es el estado en que se encuentra una estación al ser inicializada.
- Error Pasivo: los mensajes pueden ser transmitido y recibidos, pero después de la transmisión de un mensaje el nodo debe suspender la transmisión y esperar un tiempo definido antes de volver a transmitir. Este sería el caso de un nodo con disturbios cortos o fallas temporales. Si los contadores de error del nodo llegan a determinado valor, configurable, el nodo puede volver al estado activo.
- Bus Apagado: si el nodo posee una falla permanente, este es automáticamente desconectado del Bus y ya no puede participar de la comunicación. Solo puede volver al estado activo si es reinicializado

2.6.9 Normas para el protocolo CAN Bus

La Organización Internacional de Normalización (ISO) y SAE (Sociedad de Ingenieros Automotrices) han emitido las normas CAN para el intercambio de datos en aplicaciones automotrices. En 1992 grupos de fabricantes fundaron la organización "CAN en la automoción" (CiA) promovida por Holger Zeltwanger, sin fines de lucro, con el fin de proporcionar información técnica, de productos y comercialización para el uso del Bus CAN; poco tiempo después la CiA publica un artículo técnico el cual describía la capa física del protocolo y recomendaba el uso de transceptores CAN que cumplieran la norma ISO 11898.

En noviembre de 1993, el protocolo CAN es estandarizado bajo la norma ISO 11898. ISO 11898 es el estándar internacional para comunicaciones CAN en vehículos de carretera. ISO 11898 consta de las siguientes partes, bajo el título general Vehículos de carretera - Red de área del controlador

Parte 1: Capa de enlace de datos y señalización física.

Parte 2: Unidad de acceso medio de alta velocidad

Parte 3: Interfaz de baja velocidad, tolerante a fallas, dependiente del medio

Parte 4: Comunicación activada por tiempo

Parte 5: Unidad de acceso medio de alta velocidad con modo de bajo consumo

Parte 6: Unidad de acceso medio de alta velocidad con funcionalidad de activación selectiva

- **ISO 11898-1** especifica la DLL (Capa de enlace de datos), incluidas las subcapas LLC (Control de enlace lógico) y MAC (control de acceso medio), así como la subcapa PLS. La capa física CAN se puede dividir en tres subcapas. La capa PLS se implementa en los chips del controlador CAN[30].

- **ISO-11898-2** especifica las subcapas PMA y MDI de la capa física. Las capas PMA y MDI están sujetas a diferentes estándares internacionales, nacionales e industriales, así como a especificaciones de propiedad. La capa PMA describe las características del transceptor. La capa MDI especifica las características del cable y del conector. Esta norma específica al CAN de alta velocidad y sus velocidades de transmisión (125 kbps a 1 Mbps.). ISO 11898-2 especifica que un transceptor debe poder conducir un Bus de 40 m a 1 Mbps. Se puede lograr una mayor longitud de Bus al disminuir la velocidad de datos. La mayor limitación de la longitud del Bus es el retardo de propagación del transceptor. En la tabla 2 se muestran las longitudes del cable y la velocidad de transmisión de información[31].

Tabla 2 Relación de velocidad de transmisión y longitud de cable en la Red CAN [4]

Tasa de transferencia (kbps)	Longitud del Bus (m)
1000	40
500	100
250	250
125	500
40	1000

La especificación ISO 11898-2 requiere que un transceptor compatible que debe cumplir una serie de especificaciones eléctricas. Algunas de estas especificaciones están destinadas a garantizar que el transceptor pueda sobrevivir a condiciones eléctricas severas, protegiendo así las comunicaciones del nodo CAN.

ISO 11898-2, especifica que los cables a elegir para líneas de Bus CAN deben tener una impedancia nominal de 120 ohmios ubicadas en ambos extremos de la línea y un retardo de línea específico de 5 ns / m nominales. De acuerdo a los estándares ISO 11898-2 se emplea cable UTP o STP.

- **ISO 11898-3**: 2006 especifica las características de configurar un intercambio de información digital entre los nodos de los vehículos de carretera equipados con la

red de área del controlador (CAN) a velocidades de transmisión superiores a 40 kbps hasta 125 kbps. Define el CAN de baja velocidad (CAN-B), puede utilizar un Bus lineal, un Bus en estrella o múltiples Buses en estrella conectados por un Bus lineal. El Bus está terminado en cada nodo por una fracción de la resistencia de terminación total. La resistencia de terminación total debería ser un valor próximo a 100 Ω, pero no inferior a 100 Ω. Este estándar permite velocidades de hasta 125 kbps[32].

- **ISO 11898-4:** 2004 específica la comunicación activada por tiempo en la red de área del controlador (CAN): un protocolo de comunicación en serie que admite el control distribuido en tiempo real y la multiplexación para uso dentro de vehículos de carretera. Es aplicable a la configuración de un intercambio desencadenado por tiempo de información digital entre los nodos de los vehículos de carretera equipados con CAN, y específica la entidad de sincronización de trama que coordina la operación de los enlaces lógicos y los controles de acceso a los medios de acuerdo con ISO 11898-1, para proporcionar el horario de comunicación activado por tiempo[33].

- **ISO 11898-5** especifica la capa física con tasas de transmisión de hasta 1 Mbps para sistemas que requieren bajo consumo de energía cuando no hay comunicaciones activas en el Bus de datos. representa una extensión de ISO 11898-2 y aquellas implementaciones que cumplan cualquiera de estas dos normas, es decir, los nodos CAN de alta velocidad con y sin bajo consumo de energía, son interoperables entre sí y pueden coexistir en la misma red[34].

- **ISO 11898-6** es una extensión de ISO 11898-2 y de ISO 11898-5. Esta extensión especifica la capa física de un Bus CAN de hasta 1 Mbps, proporcionando un método selectivo de activación de nodos (*wake-up*) usando tramas CAN configurables. Las implementaciones de ISO 11898-6, ISO 11898-2 e ISO 11898-5 son interoperables y se pueden usar en una misma red simultáneamente.

CAPÍTULO 3

Cuadro de instrumentos y Sistema PXI

Con la mayor complejidad de la electrónica del vehículo, una mayor funcionalidad requiere que la información de estado se muestre al conductor. El grupo de instrumentos (ICL) es la fuente de datos principal para el conductor, y proporciona información sobre el estado de varias funciones del vehículo. El ICL tiene tres formas de mostrar información: La primera es mediante indicaciones luminosas que se presentan por lámparas indicadoras, medidores y símbolos, mostrando mensajes utilizando símbolos y texto en la pantalla o por medios auditivos usando un altavoz[35], [36].

El ICL puede mostrar información sobre la velocidad actual del vehículo, la cantidad de combustible restante y la temperatura exterior. También se puede mostrar al conductor información más detallada, como advertencias basadas en símbolos y texto, menús de información y alarmas sobre partes del vehículo que no funcionan correctamente. Los medidores proporcionan la información de velocidad, distancia, calor y combustible. Las luces indicadoras proporcionan advertencias y actualizaciones como la luz del motor del cheque y la luz de bajo nivel de combustible. Diferentes vehículos tienen diferentes advertencias disponibles. El diseño en profundidad del ICL se basa en varias decisiones de diseño generales, dos de las cuales son que el ICL no debe sobrecargar al conductor con información y el conductor no debe distraerse mientras conduce. Esto hace que la ICL actúe como una Interfaz de Usuario Atento (AUI). Un AUI es una interfaz que prioriza y optimiza la comunicación con el usuario mediante la asignación dinámica de los recursos de procesamiento de información del usuario. La ICL, por ejemplo, da prioridad a mostrar alarmas al conductor antes de mostrar advertencias o información general[35], [37].

3.1 Historia del cuadro de instrumentos

Al comienzo de la instrumentación automotriz, solo había un velocímetro y algunas luces de advertencia para examinar las funciones más importantes. Henry Ford fue muy innovador y equipó todos sus autos con este instrumento a partir de 1910 como se observa en la figura 3.1. Fue inventado en 1902 por un ingeniero alemán y se convirtió en obligatorio a partir de 1935.

Figura 3.1 Velocímetro mecánico y amperímetro en un Ford Modelo T 1914[32].

Más adelante, se agregaron a la instrumentación indicadores individuales como la temperatura del radiador, el indicador de revolución y el indicador de combustible. En la figura 3.2 se observan los pasos de desarrollo desde el único velocímetro mecánico hasta el instrumento moderno de principios del siglo XXI. Con el tiempo, los instrumentos individuales fueron reemplazados por grupos de instrumentos menos costoso, más confiable y más fácil de fabricar e instalar en el vehículo.

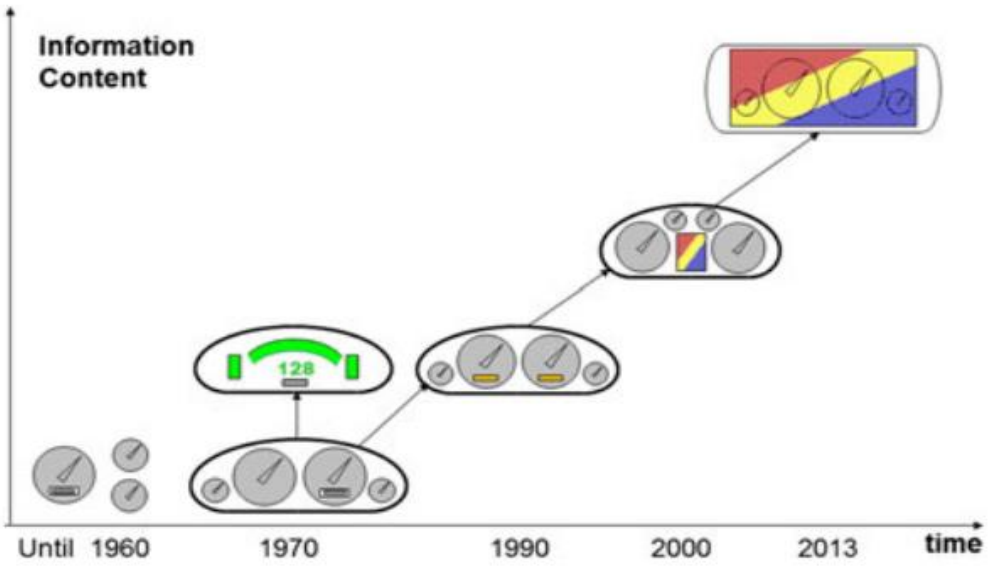


Figura 3.2 Evolución de los Cuadros de instrumentos[37]

Debido al desarrollo de nuevos equipos electrónicos y a la necesidad de monitorear, cada vez más nodos de información se han colocado dentro o en el área de cabina disponible, lo que a veces lleva a una apariencia confusa de la información. Hoy en día, los fabricantes ofrecen unas 10 variaciones diferentes de tablero de instrumentos por modelo que requieren enormes esfuerzos logísticos por parte del proveedor del instrumento. Desde principios de la década de 2000, las pantallas de visualización más grandes y con capacidad gráfica en la tecnología de matriz LCD activa con rango de temperatura extendido se utilizan en grupos de instrumentos para un reemplazo parcial de los medidores mecánicos y para aumentar el contenido de información del instrumento. En 2005, se introdujo un instrumento LCD parcialmente reconfigurable en el Mercedes S-class, seguido de algunos otros vehículos. Se utiliza para mostrar un gran instrumento de puntero analógico y alternativamente una imagen de visión nocturna desde una cámara de video[36].

3.2 Tipos de cuadros de instrumentos

Hay tres tipos de grupo de instrumentos, que incluyen el grupo de instrumentos estándar, el instrumento Híbrido y el grupo de instrumentos electrónicos (figura 3.3).

Figura 3.3 Tipos de Cuadros de instrumentos [10].

Las tres plataformas básicas correspondientes a los dispositivos de clúster son:

- **Plataforma Gráfica Clúster**

Este dispositivo consta principalmente de un procesador de gráficos, una memoria RAM de gráficos en el chip para el acceso de datos de gráficos, una interfaz de memoria externa de alta velocidad, controlador de pantalla para mostrar información de gráficos a través de

TFT, Pantalla LCD. Además, el conjunto de periféricos integrados como CAN, MOST, Ethernet y USB permiten una conexión directa con el resto de la red del automóvil[37].

- **Plataforma del controlador de clúster**

Este dispositivo contiene un controlador y controlador de motor paso a paso para la conducción de indicadores, módulo de generación de sonido, canales PWM para control de sonido y LED de grupo, varios sensores analógicos y periféricos de comunicación, HMI para comunicarse con el controlador, otros dispositivos del cuerpo y dispositivos de pasarela u otro subsistema[37].

- **Plataforma de clúster combinada**

El controlador de gráficos y el controlador de aplicaciones en tiempo real (Clúster MCU) se combinan en esta plataforma para crear una solución de chip único tanto para la aplicación de Gráficos como para la de Autosar. Este dispositivo mantiene un equilibrio entre el rendimiento de los gráficos y el rendimiento en tiempo real, por lo que es utilizado por los OEM (fabricante de equipos originales) cuando se requieren soluciones óptimas [32].

3.3 Estructura interna del cuadro de instrumentos

El uso del panel de instrumentos de los automóviles es esencial para una mayor sensación de seguridad. Todas las decisiones subjetivas con respecto a acciones adicionales realizadas durante el viaje son tomadas por el conductor sobre la base de lecturas objetivas obtenidas de los indicadores instalados y de las luces de advertencia individuales. Cuantos más datos recibe el conductor, más completo es su conocimiento de la condición del vehículo. El tiempo de recepción de la información sobre la condición del vehículo por parte del conductor también es importante. Teniendo en cuenta estos supuestos desde el punto de vista técnico, la alta confiabilidad, la legibilidad de la información mostrada y la certeza de que el conductor recibirá dicha información son los requisitos básicos que debe cumplir el panel de instrumentos[38].

El grupo de instrumentos de un automóvil alberga las distintas pantallas e indicadores que permiten al conductor operar el vehículo. Entre estos se encuentran varios medidores, así como varios indicadores de fallas y advertencias del sistema. Los grupos de instrumentos proporcionan a los conductores una ubicación centralizada y fácilmente visible para mostrar toda la información crítica del sistema[39]. En la figura 3.4 se muestra un diagrama a bloques de los subsistemas del clúster de instrumentos

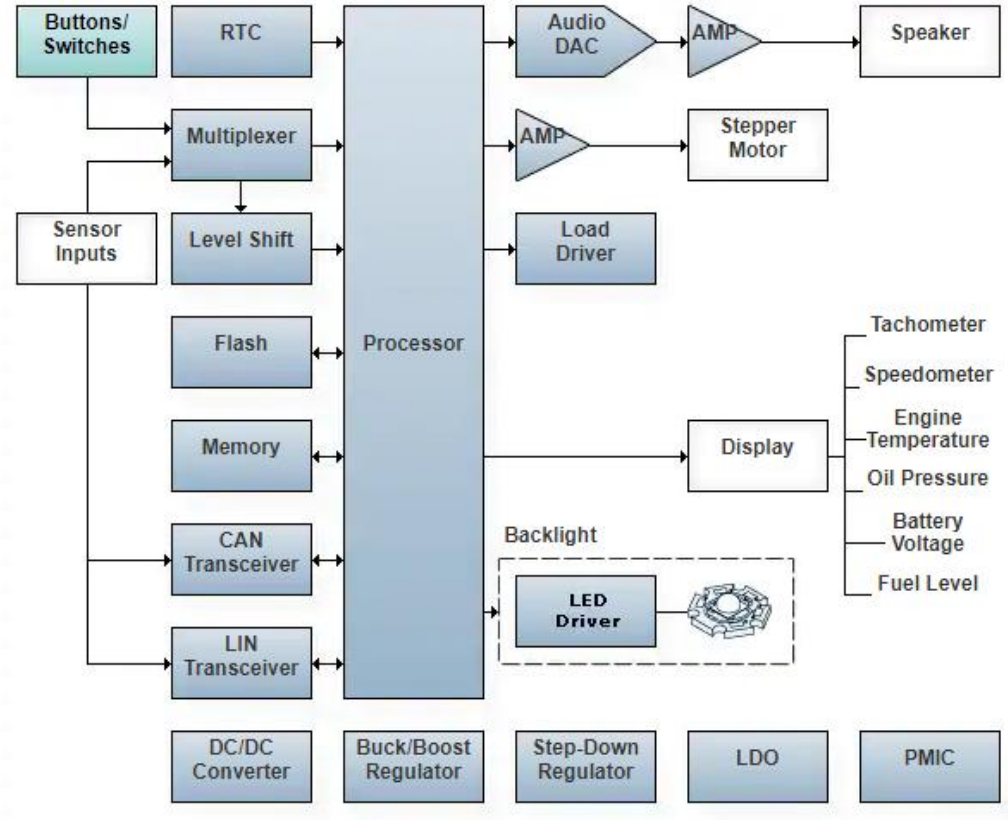


Figura 3.4 Subsistemas del Clúster de instrumentos [35].

Los principales subsistemas del clúster de instrumentos son:

- Conectividad: Siendo el estándar de facto para las comunicaciones en serie de alta integridad, el Bus CAN (red de área del controlador) de un automóvil constituye la "columna vertebral" de la red del vehículo. CAN está diseñado para aplicaciones que tienen que comunicar numerosos pero pequeños fragmentos de datos de forma consistente entre los nodos, así como autodiagnóstico y reparación de errores de datos. Del mismo modo, los LIN (red de interconexión local) manejan la comunicación de red dentro de un nodo. De bajo costo y relativamente sencillo de implementar, una red LIN usa una topología de transmisión con un solo maestro, generalmente una MCU, y hasta 12 dispositivos esclavos[39].

- Procesador: las aplicaciones automotrices son intrínsecamente críticas para la seguridad, por lo que el procesador o el microcontrolador deben ofrecer un nivel de rendimiento lo suficientemente alto como para garantizar un control confiable en

tiempo real. Además de su capacidad para ejecutar código rápidamente, el microcontrolador también se elige para sus periféricos integrados específicos de la aplicación. Las MCU del grupo de instrumentos suelen incluir un transceptor CAN y / o LIN integrado para comunicarse con varios sensores ubicados en todo el vehículo. El microcontrolador también puede contar con controladores de motor paso a paso para accionar varios medidores [34].

- Administración de energía: Un grupo de instrumentos puede incluir memoria externa, motores paso a paso, una o más MCU, interfaz CAN, interfaz LIN y retroiluminación LED, todos los cuales pueden operar a diferentes niveles de voltaje. Con tantos rieles de alimentación diferentes, se requiere una cuidadosa consideración al diseñar para la eficiencia, la compacidad, el bajo costo y la baja EMI [34].

3.4 Sistemas PXI

National Instruments desarrolló y anunció la especificación PXI (PCI eXtensions for Instrumentation) en 1997 y la lanzó en 1998 como una especificación abierta de industria para cubrir la creciente demanda de sistemas complejos de instrumentación. PXI es una plataforma basada en PC que ofrece una solución de despliegue de alto rendimiento para sistemas de medida y automatización. Los sistemas PXI están compuestos de tres componentes básicos: chasis, controlador de sistema y módulos periféricos[40]. En la figura 3.5 se muestran los componentes del sistema PXI.

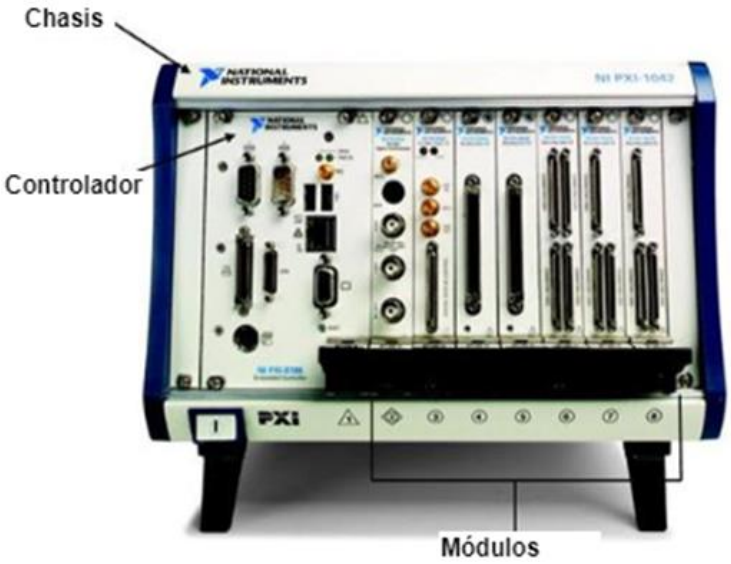


Figura 3.5 Sistema PXI [35]

Para este trabajo se emplea:

- Chasis NI PXI-1042 aloja módulos PXI y los conecta con un plano trasero de alto rendimiento que brinda habilidades de temporización y sincronización[41].

- PXI-8101 es un controlador embebido de dos ranuras para sistemas PXI y CompactPCI. Es usado para crear una plataforma compacta y/o portátil basada en PC para aplicaciones de control industrial, adquisición de datos y de pruebas y medidas[42].

- El módulo PXI-8513 es una interfaz de red de controladores de área (CAN). El módulo PXI-8513 funciona bien en aplicaciones que requieren manipulación de alta velocidad en tiempo real de cientos de marcos y señales CAN, como la simulación de hardware en el ciclo, la rápida generación de prototipos de control, el monitoreo de Bus, el control de automatización y más[43].

El PXI-1042 combinado con el controlador integrado NI PXI-8101 da como resultado una computadora totalmente compatible con PC en un paquete compacto y resistente, para adquisición de datos y aplicaciones de prueba y medición. El chasis es conectado a la PC a través de un cable Ethernet y luego se emplea el módulo de interfaz CAN que proporcionan conectividad directa con dispositivos externos, en este caso el panel de instrumentos. Para comunicarse con otros dispositivos los Módulos PXI de Interfaz CAN lo hacen usando transceptores internos.

Los Módulos de Interfaz CAN son compatibles con el controlador NI-XNET o el NI-985x, dependiendo del modelo. En este caso se usa el NI-XNET, con el cual se pueden crear aplicaciones que requieren manipulación de alta velocidad en tiempo real de cientos de marcos y señales CAN. El motor DMA (utilizado para transferir datos directamente desde o hacia un dispositivo y la memoria de la computadora) impulsado por el dispositivo NI-XNET permite al procesador integrado mover marcos y señales CAN entre la interfaz y el programa del usuario sin interrupciones de CPU, disminuyendo la latencia del mensaje y liberando tiempo de procesador principal. Los Módulos de Interfaz CAN funciona bien en aplicaciones como simulación de Hardware-in-the-Loop (HIL), rápida generación de prototipos de control, monitoreo de Bus y control de automatización[44]

CAPÍTULO 4

Elementos para la implementación del Nodo CAN

En este capítulo se enuncian los elementos que se utilizan para la construcción de un Nodo CAN y el tipo de comunicación que se emplea para realizar el intercambio de información entre las placas.

4.1 Placa Arduino

Arduino es una plataforma de prototipos electrónicos de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. La plataforma Arduino está basada en una sencilla placa con 14 pines digitales de entrada/salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16MHz, una conexión USB, un conector de alimentación y un botón de reinicio (figura 4.1). Arduino ofrece una serie de ventajas que la diferencia del resto: Barata, es multiplataforma (se ejecuta en diferentes sistemas operativos), Tiene un entorno de programación simple, es de código abierto y puede ser expandido mediante librerías. La plataforma Arduino puede captar señales del entorno, mediante la recepción de entradas desde una variedad de sensores, y puede utilizarse para diferentes elementos[45].

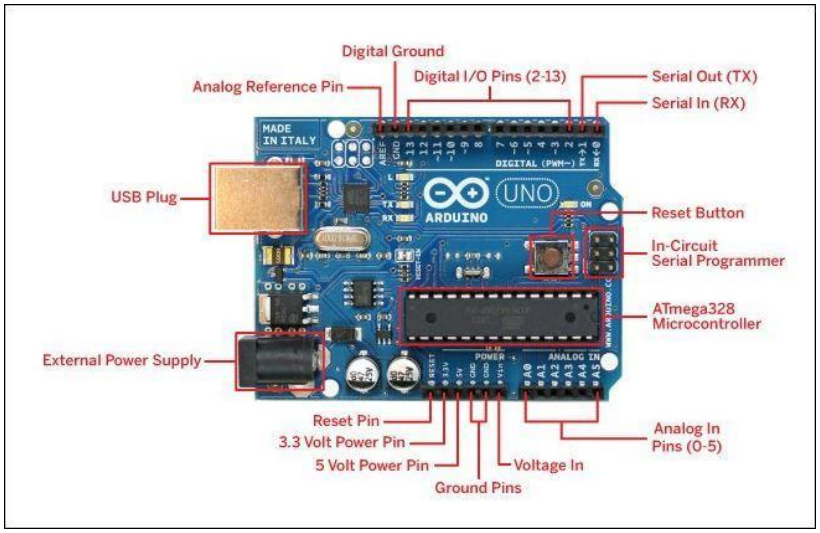


Figura 4.1 Pines de la placa Arduino UNO [40]

El elemento principal de dicha placa es el microcontrolador ATmega328P de la marca Atmel (figura 4.2), trabaja con una memoria flash de 32 kB (0.5 kB son usados para el cargador de arranque). Además, dispone de 2 kB de memoria SRAM (Static Random Access Memory) y 1 kB de memoria EEPROM (Electrically Erasable Programmable Read-Only Memory), es un chip sencillo y de bajo coste que permite el desarrollo de múltiples diseños. Cada uno de los 14 pines digitales de la placa Arduino UNO pueden ser usados como entrada o salida, usando las funciones pinMode (), digitalWrite () y digitalRead (). Estos pines operan a 5 voltios, cada pin puede proporcionar o recibir un máximo de 20 mA y tiene una resistencia interna pull-up (desconectada por defecto) de 20-50 KOhms.

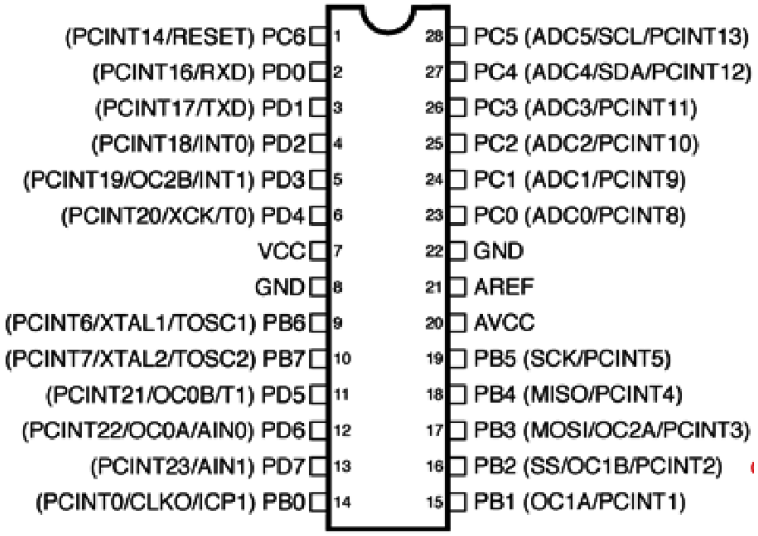


Figura 4.2 Pines del microprocesador ATmega328P [40]

Para este trabajo se necesita de la comunicación de la placa Arduino y Can Bus Shield, los pines para la comunicación SPI son: pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines soportan la comunicación SPI haciendo uso de la librería SPY. Arduino necesita de un programa externo ejecutado en otro ordenador para poder escribir códigos para la placa Arduino. Éste software es lo que llamamos Arduino IDE (Integrated Development Environment) o Entorno de Desarrollo Integrado en español[45].

4.2 CAN Bus Shield

El CAN Bus Shield le brinda a la placa Arduino capacidades CAN Bus. Este protector permite sondear al nodo para obtener información, también puede almacenar estos datos o leerlos.

Utiliza el controlador CAN Microchip MCP2515 y el transceptor CAN MCP2551. El protector también tiene un soporte de tarjeta MicroSD, un conector LCD serie y un conector

para un módulo GPS[46]. Estas características hacen que este escudo sea ideal para la aplicación de registro de datos, en la figura 4.3 se muestran las partes más importantes de la tarjeta y en la tabla 3 se describiré cada una de estas.

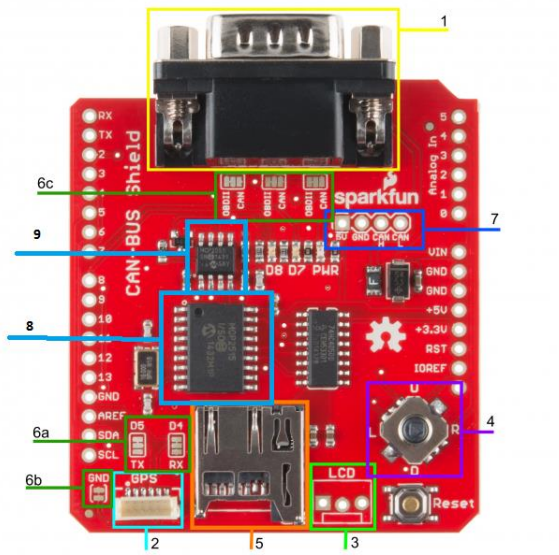


Figura 4.3 Placa CAN Bus Shield [42]

Tabla 3 Elementos de la Placa CAN Bus Shield

Parte	Descripción
1	Conector DB9
2	Conector GPS
3	Conector LCD
4	Joystick
5	Espacio MicroSD
6a	SJ1 y SJ2: permiten al usuario seleccionar entre UART y Software Serial para que la unidad GPS se comunique con el Arduino.
6b	SJ3: esto le permite al usuario separar el pin 5 en el conector GPS de la línea GND.
6c	SJ4, SJ5 y SJ6: permiten seleccionar la configuración del pin DB9 entre OBD-II y CAN.
7	CAN Pins: 5V, GND, CAN_H, CAN_L
8	MCP2515
9	MCP2551

Se utiliza esta tarjeta debido a que cuenta con el controlador MCP2515 y el transceptor MCP2551, los cuales cumples las características requeridas para la comunicación CAN Bus.

4.2.1 Controlador MCP2515

El MCP2515 de Microchip Technology es un controlador de red de área de controlador (CAN) independiente que implementa la especificación CAN, versión 2.0B a 1 Mbps. Es capaz de transmitir y recibir datos tanto estándar como extendidos y marcos remotos. El MCP2515 interactúa con microcontroladores (MCU) a través de una Interfaz Periférica Serial (SPI) estándar de la industria. El MCP2515 tiene dos máscaras de aceptación y seis filtros de aceptación que se utilizan para filtrar los mensajes no deseados, lo que reduce la sobrecarga de la MCU del host[47]. En la figura 4.4 se muestran los pines del controlador y en la tabla 4 se describen sus principales características.

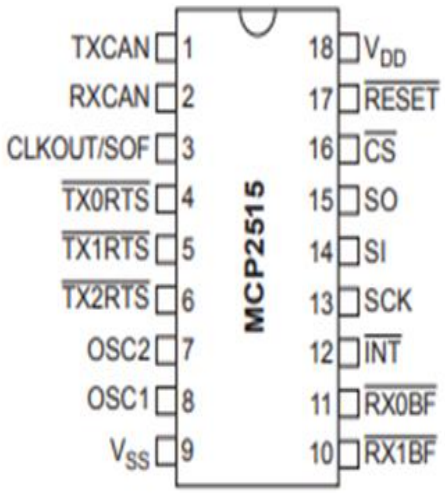


Figura 4.4 Controlador MCP2515 [42]

Tabla 4 Características del controlador MCP2515

Principales características del controlador MCP2515	
1	Implementa CAN V2.0B a 1 Mb / s
2	0 a 8 bytes de longitud en el campo de datos
3	Datos estándar y extendidos y marcos remotos
4	Búferes de recepción, máscaras y filtros
5	Interfaz SPI de alta velocidad (10 MHz)
6	Start-of-Frame La señal está disponible para monitorear la señal SOF
7	El modo One-Shot garantiza que la transmisión de mensajes se intente
8	Filtrado de bytes de datos en los primeros dos bytes de datos
9	Tres búferes de transmisión con funciones de priorización y cancelación

El MCP2515 es un controlador CAN desarrollado para simplificar las aplicaciones que requieren una interfaz con un Bus CAN. En la figura 4.5 se ven los tres bloques que lo integran: El módulo CAN, La lógica de control y los registros y el bloque de protocolo SPI.

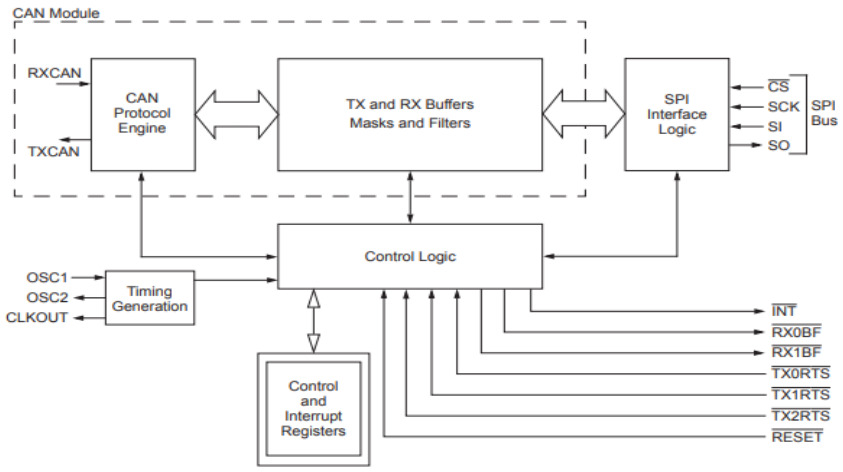


Figura 4.5 Diagrama de bloques del controlador MCP2515 [42]

- El módulo CAN, que incluye el motor de protocolo CAN, máscaras, filtros, transmisión y recepción de buffers. El módulo CAN maneja todas las funciones para recibir y transmitir mensajes en el Bus CAN. Los mensajes se transmiten cargando primero el búfer de mensajes apropiado y los registros de control. La transmisión se inicia utilizando bits de registro de control a través de la interfaz SPI o utilizando los pines de habilitación de transmisión.
- La lógica de control y los registros que se utilizan para configurar el dispositivo y su funcionamiento.
- El bloque de protocolo SPI. La MCU se conecta al dispositivo a través de la interfaz SPI. La escritura y lectura de todos los registros se realiza mediante los comandos estándar de lectura y escritura de SPI, además de los comandos especializados de SPI[47].

4.2.2 Transceptor MCP2551

El MCP2551 es un dispositivo tolerante a fallas CAN de alta velocidad que sirve como interfaz entre un controlador de protocolo CAN y el Bus físico. El MCP2551 proporciona una capacidad de transmisión y recepción diferencial para el controlador de protocolo CAN y es totalmente compatible con la norma ISO-11898, incluidos los requisitos de 12V a 24V. Funcionará a velocidades de hasta 1 Mbps. CAN_H y CAN_L están protegidos contra cortocircuitos de batería y transitorios eléctricos que pueden ocurrir en el Bus CAN, también proporciona protección contra apagones, así como una detección dominante permanente para garantizar que un nodo sin alimentación o defectuoso no perturbara el Bus. Normalmente, cada nodo en un sistema CAN debe tener un dispositivo para convertir las señales digitales generadas por un controlador CAN en señales adecuadas para la transmisión a través del cableado del Bus (salida diferencial). También proporciona un búfer entre el controlador CAN y los picos de alto voltaje que pueden generarse en el Bus CAN por fuentes externas. En la figura 4.6 se muestra un diagrama de bloques del transceptor y en la tabla 5 se describe la función de cada uno de sus pines[19], [48].

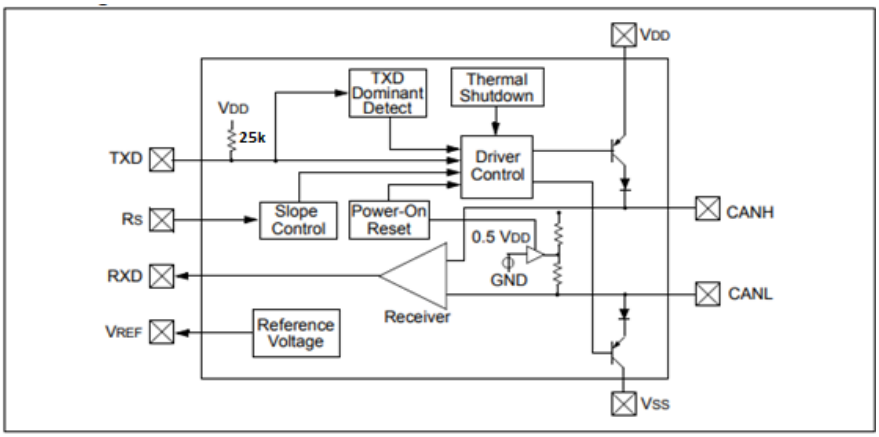


Figura 4.6 Diagrama de bloques del transceptor MCP2551 [18]

Tabla 5 Pines del transceptor MCP2551

Número de Pin	Nombre de Pin	Función
1	TXD	Entrada de datos del transmisor, los datos en este pin se expulsan en los pines de salida diferencial CAN_H y CAN_L
2	Vss	Pin de suministro de tierra
3	VDD	Pin de voltaje de suministro positivo.
4	RXD	Salida de datos del receptor (generalmente está conectada a la entrada de datos del receptor del dispositivo controlador CAN. RXD es alto cuando el Bus CAN es recesivo y bajo en el estado dominante)
5	VREF	Salida de tensión de referencia (definida como VDD / 2)
6	CAN_L	controla el lado bajo del Bus diferencial CAN, vinculado internamente al comparador de entrada de recepción
7	CAN_H	Controla el lado alto del Bus diferencial CAN, vinculado internamente al comparador de entrada de recepción.
8	Rs	Entrada del resistor de la pendiente

4.2.3 Operaciones del MCP2551

- TRANSMITIR

El controlador de protocolo CAN envía un flujo de datos en serie a la entrada lógica TXD del MCP2551. El estado correspondiente recesivo o dominante se emite en los pines CAN_H y CAN_L.

- RECIBIR

El MCP2551 recibe estados dominantes o recesivos en los mismos pines CAN_H y CAN_L cuando se produce la transmisión. Estos estados se emiten como niveles lógicos en el pin RXD para que el controlador de protocolo CAN reciba tramas CAN.

La especificación ISO 11898-2 requiere que un transceptor cumpla una serie de especificaciones eléctricas. Algunas de estas especificaciones están destinadas a garantizar que el transceptor pueda sobrevivir a condiciones eléctricas severas, protegiendo así las comunicaciones del nodo CAN. En la tabla 6 se comparan las características requeridas por ISO 11898-2 y las características de transceptor MCP2551 y se verifica que este cumple con las características y puede ser utilizado para realizar la comunicación CAN.

Tabla 6 Comparación características del transceptor MCP2551 y la norma ISO 11898-2[19]

Parameter	ISO-11898-4		MCP2551		Unit	Comments
	min	max	min	max		
DC Voltage on CANH and CANL	-3	+32	-40	+40	V	Exceeds ISO-11898
Transient voltage on CANH and CANL	-150	+100	-250	+250	V	Exceeds ISO-11898
Common Mode Bus Voltage	-2.0	+7.0	-12	+12	V	Exceeds ISO-11898
Recessive Output Bus Voltage	+2.0	+3.0	+2.0	+3.0	V	Meets ISO-11898
Recessive Differential Output Voltage	-500	+50	-500	+50	mV	Meets ISO-11898
Differential Internal Resistance	10	100	20	100	kΩ	Meets ISO-11898
Common Mode Input Resistance	5.0	50	5.0	50	kΩ	Meets ISO-11898
Differential Dominant Output Voltage	+1.5	+3.0	+1.5	+3.0	V	Meets ISO-11898
Dominant Output Voltage (CANH)	+2.75	+4.50	+2.75	+4.50	V	Meets ISO-11898
Dominant Output Voltage (CANL)	+0.50	+2.25	+0.50	+2.25	V	Meets ISO-11898
Permanent Dominant Detection (Driver)	Not Required		1.25	—	ms	
Power-On Reset and Brown-Out Detection	Not Required		Yes		--	

4.3 Interfaz de comunicación periférica serial

La Interfaz Periférica Serial (SPI) es un Bus de interfaz comúnmente utilizado para enviar datos entre microcontroladores y pequeños periféricos. Utiliza líneas separadas de reloj y datos, junto con una línea de selección para elegir el dispositivo con el que desea hablar. Los dispositivos que se comunican a través de SPI están en una relación maestro-esclavo. El maestro es el dispositivo de control, mientras que el esclavo toma las instrucciones del maestro.

Cuando los datos se envían del maestro a un esclavo (figura 4.7), se envían en una línea de datos llamada MOSI, para "Salida de maestro / Entrada de esclavo". Si el esclavo necesita enviar una respuesta al maestro, el maestro continuará generando un número preestablecido de ciclos de reloj, y el esclavo colocará los datos en una tercera línea de datos llamada MISO, para "Entrada / salida de esclavo". El maestro siempre genera la señal de reloj (primera línea de comunicación), debe saber en qué momento un esclavo debe devolver datos y cuántos datos se devolverán, SPI es "dúplex completo", por lo tanto, puede transmitir y recibir datos al mismo tiempo. Una última línea en este tipo de comunicación es SS para Slave Select. Esto le dice al esclavo que debe despertarse y recibir / enviar datos, y también se usa cuando hay varios esclavos presentes para seleccionar al que le gustaría hablar[18]

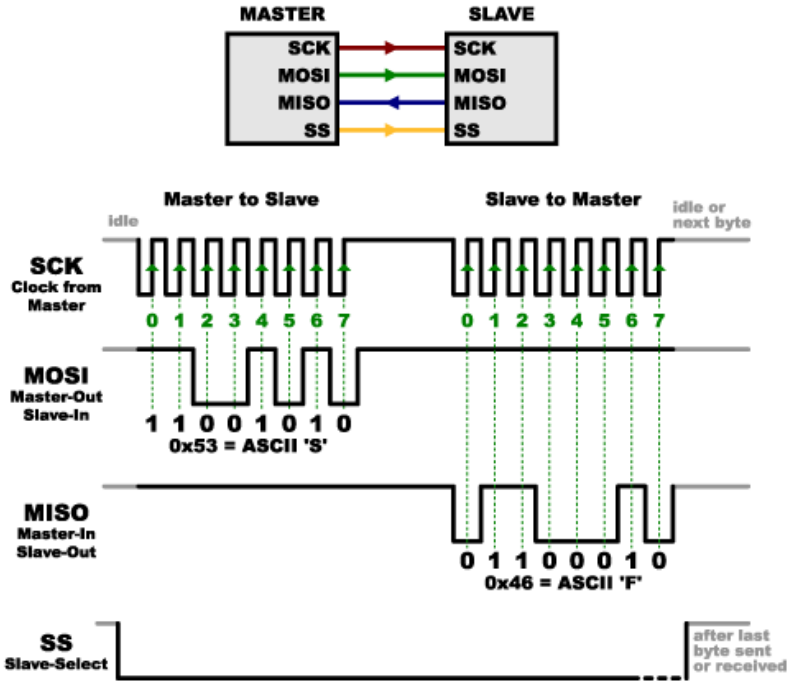


Figura 4.7 Comunicación SPI [36].

La línea SS normalmente se mantiene alta, lo que desconecta al esclavo del Bus SPI ("activo bajo" se usa para habilitar y establecer líneas). Justo antes de que se envíen los datos al esclavo, la línea baja, lo que activa al esclavo. Cuando hayas terminado de usar el esclavo, la línea vuelve a ser alta.

Muchos microcontroladores tienen periféricos SPI incorporados que manejan todos los detalles de envío y recepción de datos, y pueden hacerlo a velocidades muy altas. Para este trabajo se usa la biblioteca SPI, los pines de Arduino UNO para este tipo de comunicación son: MOSI=11, MISO=12, SCK=13, SS=10.

CAPÍTULO 5

Desarrollo del Proyecto

5.1 Obtención de los Identificadores de Mensajes del cuadro de instrumentos

El identificador no solo es parte de la trama de datos, sino que es utilizado para distinguir entre prioridades de mensaje, en el protocolo CAN el mensaje no va dirigido a ningún nodo en específico, ya que cada uno de ellos reconocerá mediante el identificador si el mensaje le interesa o no. Para localizar los identificadores de mensaje del clúster de instrumentos 6J0 920801X del vehículo Seat Ibiza, se utilizó el Monitor de Bus NI-XNET, este Monitor está incluido en el PXI-8513 el cual es un módulo de interfaz CAN. Para acceder a este Monitor se utiliza el software Measurement & Automation Explorer (NI MAX) que proporciona acceso a los dispositivos CAN. En la figura 5.1 se muestra la ruta de acceso al Monitor de Bus para registrar el tráfico de Bus CAN.

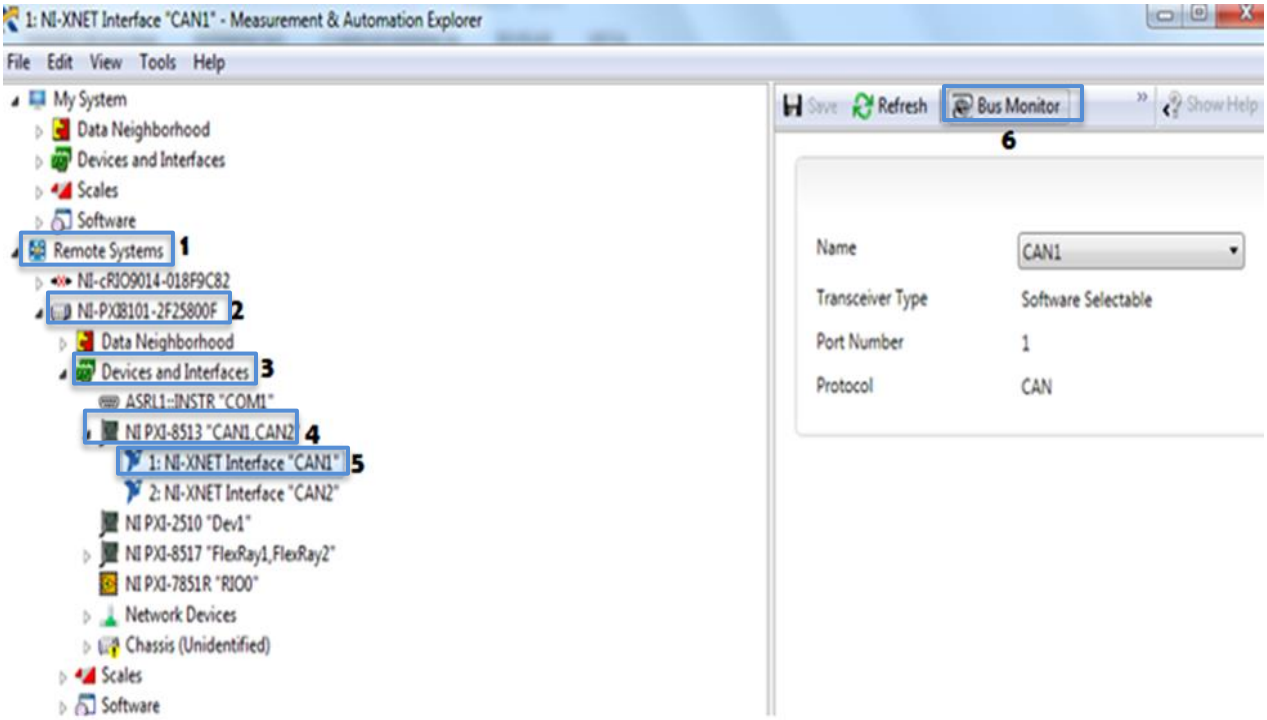


Figura 5.1 Bus Monitor NI-XNET

Además del uso del Monitor de Bus NI-XNET también se empleó el cuadro de instrumentos el cual tiene una interfaz CAN por lo que puede comunicarse mediante este protocolo con los nodos. Para la conexión del panel de instrumentos al Chasis se conectó el conector del panel de instrumentos empleando los pines que se muestran en la tabla 7.

Tabla 7 Pines de conexión del panel de instrumentos

Numero de Pin de panel de instrumentos	Descripción
Pin 16	GND
Pin 28	CAN_H
Pin 29	CAN_L
Pin 31	Terminal 15
Pin 32	Terminal 30

Para localizar los IDs de mensaje también se conectó el cuadro de instrumentos al conector DB9 del módulo de interfaz CAN (PXI 8513), en la figura 5.2 se muestra la conexión del clúster al módulo CAN. Entre la conexión del módulo y el clúster de instrumentos se encuentra una resistencia de 120 ohm para evitar interferencias.

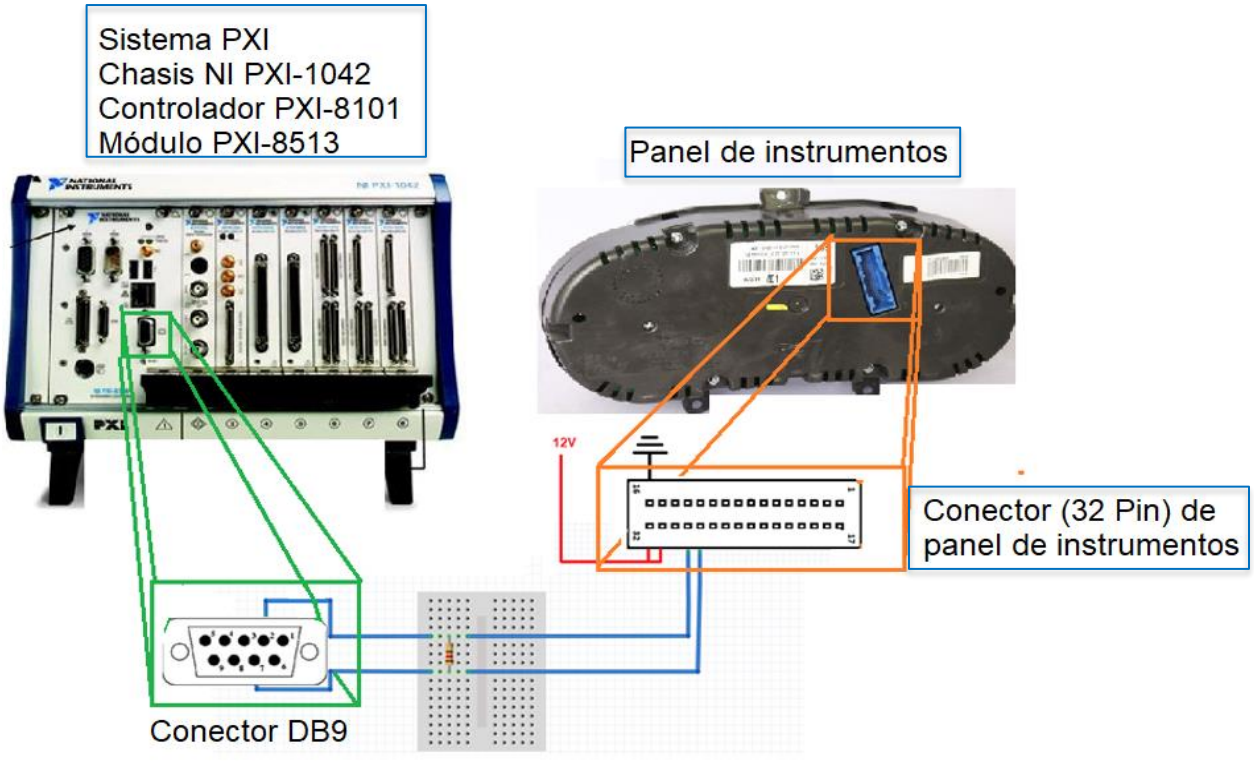
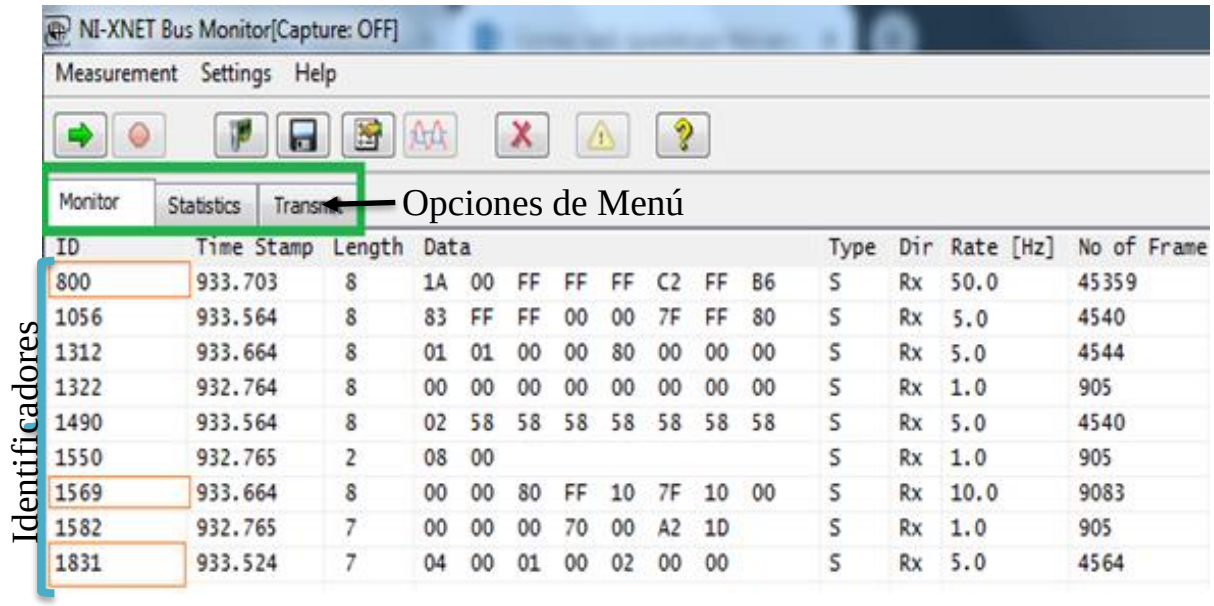


Figura 5.2 Conexión del clúster de instrumentos y el Chasis

El Monitor NI-XNET cuenta con un menú de tres opciones: Monitor, estadísticas y transmitir, en la figura 5.3 se observan resaltados en un recuadro verde.

La opción de Monitor se encarga de la lectura, es decir, de mostrar los mensajes que se envían y reciben, esta es la primera opción que se emplea. En cuanto inicia la lectura se observa que el clúster de instrumentos ya se encuentra enviando mensajes, esto es debido a que como se muestra en la figura 3.4 (capítulo 3) el cuadro de instrumentos tiene un microcontrolador, un controlador y un transceptor CAN, es por esta razón que el clúster puede enviar diferentes mensajes.

De acuerdo con [14] los mensajes con identificador 800, 1569 y 1831 corresponden a mensajes del clúster de instrumentos, estos mensajes se encuentran en sistema decimal por lo que en hexadecimal son: 320, 621 y 727 respectivamente. En la figura 5.3 se muestran los mensajes enviados por el Clúster de instrumentos y se resaltan en naranja los identificadores correspondientes al panel de instrumentos.



ID	Time Stamp	Length	Data	Type	Dir	Rate [Hz]	No of Frame
800	933.703	8	1A 00 FF FF FF C2 FF B6	S	Rx	50.0	45359
1056	933.564	8	83 FF FF 00 00 7F FF 80	S	Rx	5.0	4540
1312	933.664	8	01 01 00 00 80 00 00 00	S	Rx	5.0	4544
1322	932.764	8	00 00 00 00 00 00 00 00	S	Rx	1.0	905
1490	933.564	8	02 58 58 58 58 58 58 58	S	Rx	5.0	4540
1550	932.765	2	08 00	S	Rx	1.0	905
1569	933.664	8	00 00 80 FF 10 7F 10 00	S	Rx	10.0	9083
1582	932.765	7	00 00 00 70 00 A2 1D	S	Rx	1.0	905
1831	933.524	7	04 00 01 00 02 00 00	S	Rx	5.0	4564

Figura 5.3 Lectura de los mensajes del panel de instrumentos

En la figura 5.3 se observan los apartados de la opción monitor los cuales son: ID, Time Stamp (marca de tiempo), Lenght (Longitud), data, Type, Dir, Rate [Hz], Número de trama. El ID corresponde al identificador del mensaje, la marca de tiempo de LabVIEW representa el tiempo absoluto en que la interfaz XNET recibió el marco (final del marco), con precisión de microsegundos[49], la longitud se refiere al número de bytes del mensaje, data corresponde a la información del mensaje que se envía en cada uno de los bytes, Type puede ser “s” o “E”, dependiendo si la trama de datos es estándar o extendida respectivamente, Dir se refiere a la recepción (Rx) o envío (Tx) de mensajes ,Rate es la frecuencia del envío de mensajes y el último apartado es número de tramas que se han enviado.

Después de la lectura de los mensajes se inicia con la búsqueda de los identificadores de mensaje, para lo cual se tomó como referencia el identificador de Stefan [50] correspondiente a RPM de un panel de instrumentos idéntico al utilizado en este trabajo,

considerando este identificador también se utilizan los identificadores de [13], [14], [16] y el trabajo de [15] debido a que los IDs encontrados pertenecen a diferentes vehículos de la marca Volkswagen y el identificador de RPM es el mismo en las 5 referencias. Otra razón por la cual se utilizan los trabajos anteriormente mencionados es porque la base de datos de los identificadores de mensaje no se encuentra libres y cambia dependiendo de la marca y modelo.

El primer identificador de mensaje y la trama que se utilizó fue el empleado por Stefan en [50], en la figura 5.4 se muestra la imagen del mensaje enviado, se puede notar que el identificador es el 0X280 que corresponde a las Revoluciones por minuto (RPM), en cuanto a la trama de datos se encontró que el tercer byte es en el que se envía la información, en la figura se observa que se envía un valor de 31 RPM.

Para este identificador se encontró que al cambiar los valores de los demás bytes del mensaje este no tenía ningún cambio, incluso la longitud del mensaje podía limitarse a 4 bytes sin afectar el funcionamiento.

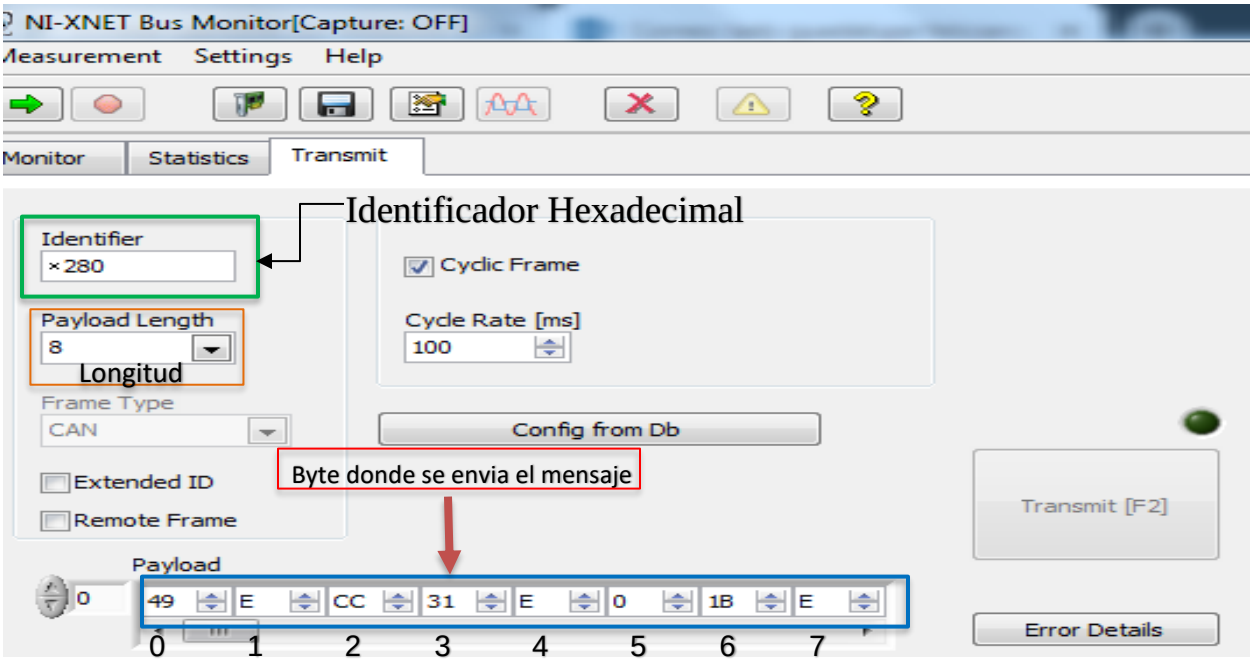


Figura 5.4 Mensaje enviado al panel de instrumentos

Posteriormente se emplearon los IDs de [7] puesto que se utiliza el mismo clúster, pero los Identificadores de mensaje son de un VW Passat B6, sin embargo, algunos de estos mensajes funcionaron en el clúster del Seat Ibiza con el que se contaba, esto puede ser debido a que Seat pertenece a Volkswagen. Los identificadores que coincidieron con el clúster fueron: 0x050, 0x1A0 y 0x5A0, esta información se puede afirmar con [6], donde Volkswagen mencionan los mismos identificadores sin mencionar el vehículo al que pertenecen.

El primer identificador corresponde a las bolsas de aire (Airbag), su longitud es de 4 bytes, durante las pruebas se encontró que el byte de la posición 0 y byte 2 son los encargados de llevar la información. El byte 0 informa del estado de la bolsa de aire, es decir, es el responsable del testigo en el cuadro de instrumentos y su comportamiento es el siguiente: el testigo permanece encendido si hay alguna avería esto le indicará al conductor que es necesario que acuda al taller para que revise el sistema o realizará la función de autodiagnóstico cada vez que se conecta el encendido, se enciende el testigo de bolsa de aire durante algunos segundo de esta manera diagnostica el funcionamiento del sistema de bolsas y después de 4 segundo se apaga el testigo.

El byte 2 corresponde al testigo de cinturón de seguridad, este testigo se mantiene encendido cuando algún ocupante no lleva puesto el cinturón de seguridad, es importante llevarlo puesto ya que en caso de un accidente absorbe parte de la energía cinética, es necesario su uso, aunque el vehículo cuente con bolsas de aire ya que los airbags se disparan en algunos casos de colisión, por ejemplo, los airbags delanteros solo se disparan en algunos casos de colisión frontal.

Debido a que las bolsas de aire y el cinturón de seguridad trabajan juntas en caso de un choque, es importante que se encuentren comunicadas y la mejor manera es estando en el mismo nodo, en el apartado de apéndice se encuentra la tabla de valores que debe tomar cada byte para indicar una avería o el buen funcionamiento de los sistemas de seguridad.

El segundo ID (0x1A0) se encarga de indicar el buen funcionamiento del ABS, este identificador es de 8 bytes de longitud y lleva la información en el byte 7, 0x1A0 también envía información del freno de mano y ESP.

El tercer identificador se encarga de la velocidad del vehículo, tiene 8 bytes de longitud, la información de velocidad se envía en el byte 2 y en el tercero se transmite información del estado de la presión de los neumáticos, nivel de líquido de freno insuficiente, ESP, entre otros. A pesar de que solo se envía información en los 4 primeros bytes es necesario que la longitud del mensaje sea de 8 bytes, ya que si la longitud es limitada a 4 el mensaje no es enviado.

En [15] se realiza un simulador de automóvil con un tablero de instrumentos de un VW Polo 6R, se utiliza esta referencia debido a que el Volkswagen Polo y el SEAT Ibiza, comparten plataforma (MQB A0), así como motorizaciones y tecnologías, son coches de tamaño casi idéntico. Asimismo, comparte cientos de componentes, tanto a nivel mecánico como mandos a la vista y en cuanto al clúster es el mismo, por lo cual se decidió probar con los identificadores que se utilizan en el trabajo. Los identificadores correspondientes a RPM, Velocidad, ABS y Airbag son idénticos, además de esto se presentan más identificadores de los cuales los siguientes funcionan con el clúster de instrumentos: 0x470,

0x3D0, estos identificadores de mensaje también concuerdan con los encontrados en [16] solo que la longitud de estos es diferente.

El primer ID 470 corresponde a las luces, con una longitud de 8 bytes, en el byte 0 envía información de las luces direccionales (izquierda, derecha, intermitentes), también se encarga del testigo de batería en caso de que esta tenga algún problema , en el byte 1 informa en caso de que la cajuela, una puerta o ambas estén abiertas, el byte 2 cambia la intensidad de luminosidad del tablero tomando el valor 64 como el 100% de intensidad, el byte 3 no envía datos, el byte 4 notifica problema en una bombilla, el byte 5 y 6 no remiten nada, en el byte 7 se comunica el encendido de la luz antiniebla, luz de carretera o ambas.

El identificador 0x3DO enciende el testigo cuando hay un problema en la dirección, este envía datos únicamente en el byte 1.

Posteriormente se localizaron más identificadores que trabajan con el clúster los cuales son: 0x288, 0x480, 0x540, pertenecen al testigo de control crucero, EPC y pisar el pedal de freno respectivamente, todos tienen longitud de 8 bytes, en las tablas localizadas en el apéndice se describen cada uno de los bytes. Durante las pruebas realizadas era necesario enviar primero el mensaje de RPM y posteriormente 0X288 o 0X480 para encender el testigo, esto significa que comparten el mismo sensor, por lo cual están comunicadas y necesitan el valor de las revoluciones por minuto antes de activarse.

El control crucero es un sistema diseñado para automatizar y mantener la velocidad del automóvil constante. El conductor debe establecer la velocidad del vehículo de forma manual y usar el control para establecer la velocidad actual como velocidad de crucero. El control de velocidad puede calcular la velocidad a partir del velocímetro, sensor de velocidad (situado en las ruedas) o a partir de las revoluciones por minuto del motor, en este caso se deduce que es a partir de las RPM.

Por otra parte, el EPC muestra al conductor que existe una anomalía en la gestión del motor, el sistema electrónico de aceleración (EPC) controla la ignición y el manejo del motor, y se encarga de regular la mariposa de aceleración de acuerdo a la señal que envía el sensor en el pedal, por lo tanto, en caso de existir un problema con alguno de estos se mostrará en el clúster el testigo del sistema electrónico de aceleración. Este sistema además de encender la luz, también limita las RPM del motor, por estas razones se afirma que está relacionado con las Revoluciones por minuto.

El identificador 0x540, con una longitud de 8 bytes, establece en su byte 6 que se debe pisar el pedal de freno para poder arrancar el motor o cambiar de posición la palanca de cambios (en caso de transmisión automática).

Durante la averiguación de los identificadores de mensaje se encontró que el mensaje enviado por el ID de velocidad no realiza ningún cambio sobre la aguja del clúster de

instrumentos, a menos que se enviará un mensaje del sistema ABS antes. Este también es un problema durante el desarrollo del trabajo en [15].

En la tabla 8 se resumen los identificadores localizados con los que trabaja el panel de instrumentos 6J0 920801X:

Tabla 8 Identificadores de mensaje localizados.

Identificador de Mensaje CAN	Byte donde se envía la información	Descripción
0X280	Tercer Byte	RPM
0X050	Byte cero Segundo Byte	Bolsa de aire Cinturón de seguridad
0X1A0	Séptimo Byte	ABS
0X5A0	Segundo Byte	Velocidad
0X470	Byte Cero, Primer Byte, segundo Byte, Cuarto Byte, Séptimo Byte	Varios indicadores de luces
0X3D0	Primer Byte	Dirección asistida
0X288	Segundo Byte	Control crucero
0X480	Primer byte	EPC
0X540	Sexto Byte	Pisar el pedal de freno

5.2 Desarrollo de la Red CAN

Para la construcción de la Red CAN de tres nodos primero se desarrolla un nodo CAN, para esto se utiliza una placa CAN Bus Shield conectada a un Arduino UNO, estas dos placas intercambian información mediante el protocolo SPI y para hacerlo es necesario implementar al entorno de desarrollo (IDE de Arduino) una librería especial para la comunicación entre estas placas, esta librería es SPI.h. La velocidad de transmisión de datos se posicionó en 500 Kbps debido a que el clúster de instrumentos utilizado trabaja a esta velocidad, para poder definir la velocidad se necesita de una segunda librería (mcp_can.h) la cual también ayudará en la lectura, envió, comprobación e inicio del Bus.

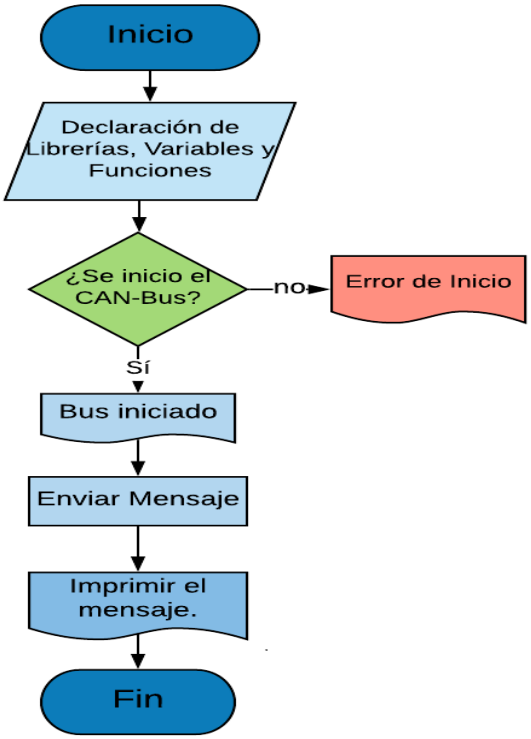


Figura 5.5 Diagrama del envió de mensaje de un Nodo CAN

Para comprobar la comunicación entre las placas se realizó un programa del envió de un dato. En la figura 5.5 se muestra un diagrama de flujo donde se representa el comportamiento de dicho programa. En el sketch de Arduino se implementan las librerías mcp_can.h y SPI.h, se definen las variables, la longitud del mensaje, el pin 10 para la comunicación entre la placa Arduino y CAN Bus Shield, la velocidad de datos para la transmisión de datos en serie.

Posteriormente se inicia el Bus a 500 kbps, en caso de que este no comience se muestra un mensaje de error, de lo contrario se confirma la inicialización, se envía el mensaje y se imprime el contenido de este.

Durante esta prueba se envió un mensaje de Revoluciones por minuto (RPM), cuyo identificador de mensaje en sistema hexadecimal es 0x280. Para mostrar que el único byte donde se envía la información es en el tercero y que los otros bytes no influyen en el mensaje se envió el valor 1 en los 7 bytes restantes.

En la figura 5.6 se muestra la lectura a través del Monitor Serial de Arduino de la trama de datos enviada por el nodo, se observa el identificador de mensaje y la información que se envía en cada uno de los bytes.

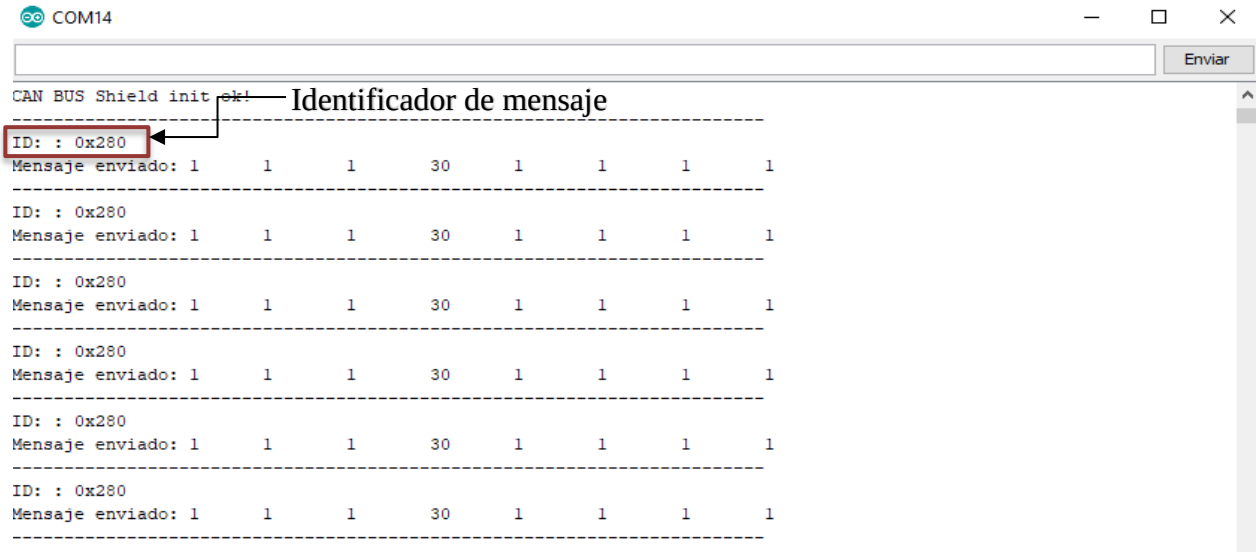


Figura 5.6 Lectura de datos del Bus

Después de comprobar la comunicación de las placas mediante el Monitor Serial de Arduino se verificó empleando el osciloscopio, en la figura 5.7 se muestra la lectura de CAN_H de la señal. Se observa el identificador de mensaje, los datos enviados y la verificación de redundancia cíclica (CRC, por sus siglas en inglés). Los datos se visualizan en sistema hexadecimal, razón por la cual el tercer byte es 1E (valor 30 en sistema decimal). También se señala la velocidad de transmisión de datos, la cual es de 500 kbps.

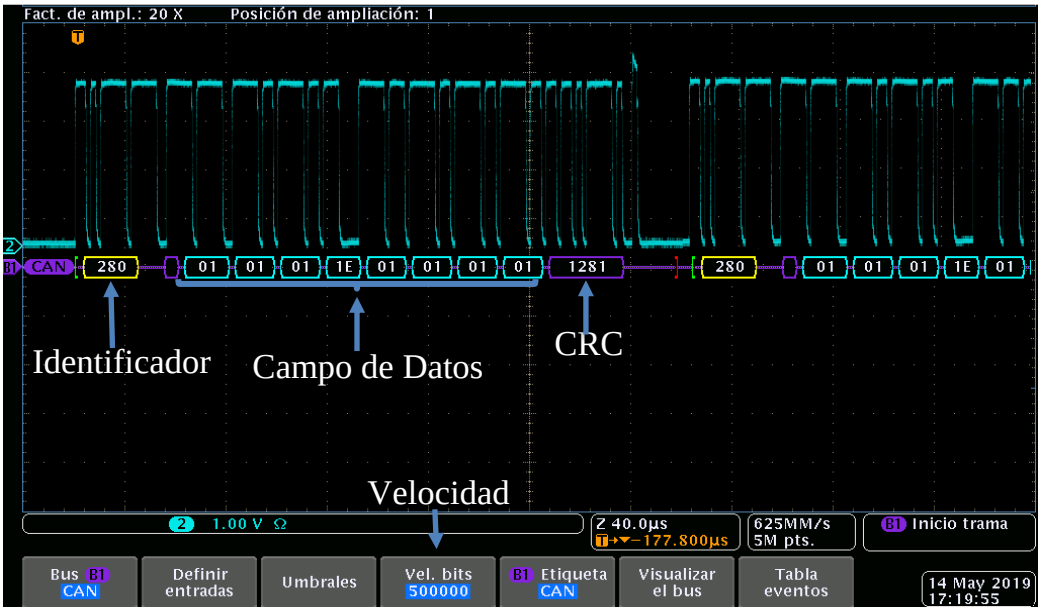


Figura 5.7 Lectura de trama de datos

Mediante la lectura en el osciloscopio y el Monitor Serial se comprueba la comunicación entre las placas y él envió de información. Así como la inicialización del Bus a 500 kbps.

5.3 Envío y recepción de datos e implementación de sensores al Nodo

Los nodos deben ser capaces de enviar y recibir información, leer los datos recibidos tanto de otros nodos como de sensores que estén conectados a estos y enviar información de sus sensores a los demás nodos. Para realizar estas operaciones y trabajar de una forma más ordenada fue necesario implementar dos funciones, una para realizar la recepción y otra para él envió de información. En la figura 5.8 se representa gráficamente el proceso.

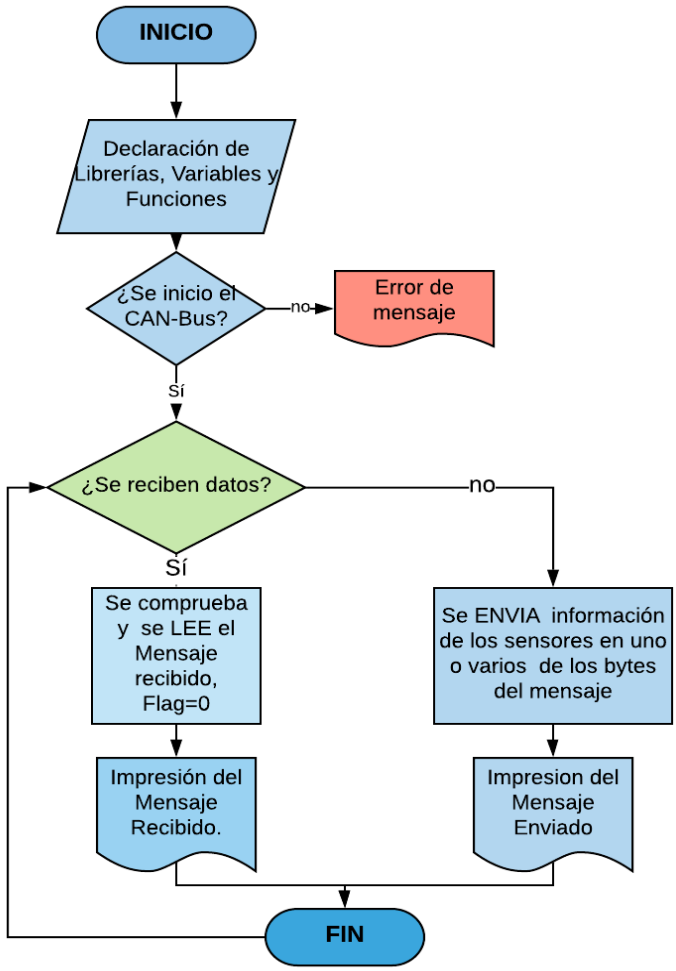


Figura 5.8 Diagrama del envío y recepción de Mensajes del nodo CAN

El programa inicia con la declaración de las variables, las librerías y funciones, para posteriormente comprobar el inicio del CAN Bus a la velocidad de 500 kbps, en caso de que el CAN no se inicialice correctamente mostrará en el Monitor Serial un mensaje de fallo, de lo contrario se mostrará el mensaje de inicio. Para detectar las tramas de datos de entrada se establece una rutina de interrupción, para la cual se determina “0” como el pin que recibirá la señal de disparo, seguido de la función (MCP2515_ISR) que se llama cuando sucede la interrupción, por último, el modo en el que sucederá la interrupción será FALLING, para que se dispare cuando el pin pase de un nivel alto a un nivel bajo.

En el bucle se verifica si se han recibido datos, de ser esto cierto se pone la bandera de recepción (flag) en cero y se verifica que haya datos para leer, se recibe la información y se muestra en el Monitor Serial. En caso de no recibir datos en el bucle, el nodo envía información de los sensores en uno o varios de los bytes del mensaje, este mensaje se imprime en el Monitor Serial. En el apartado de apéndice se muestra el código completo.

Para la comprobación de este código se realizó la comunicación entre un nodo y el clúster de instrumentos, el clúster envía mensajes propios y recibe información externa que muestra mediante sus testigos o medidores, el nodo por su parte enviará información proveniente de un dispositivo externo, para este caso se utilizó la palanca de luces del automóvil, un potenciómetro para simular la manija de luces antiniebla y luces altas, también se utiliza un push-button para el encendido del testigo de las luces intermitentes (luces de emergencia), se realiza con la palanca debido a que se tiene el identificador correspondiente al sistema de luces. En la figura 5.9 se muestra la conexión del clúster, el nodo y la palanca de luces. En la parte superior derecha se observa el conector de 32 pines del clúster de instrumentos, se utilizan los pines 31 y 32 para alimentación donde el pin 31

corresponde al borne 15 y el pin 32 al borne 30 ambas van conectadas a 12.5 voltios y el pin 16 al negativo de batería. Los pines 29 y 28 corresponden a CAN_L Y CAN_H respectivamente y están conectados mediante una resistencia de 120 ohm al nodo. En la parte inferior derecha se muestra el conector de la palanca de luces y su conexión al nodo. En el apartado de apéndice se describen a detalle cada uno de los pines del conector del clúster y de la palanca de luces.

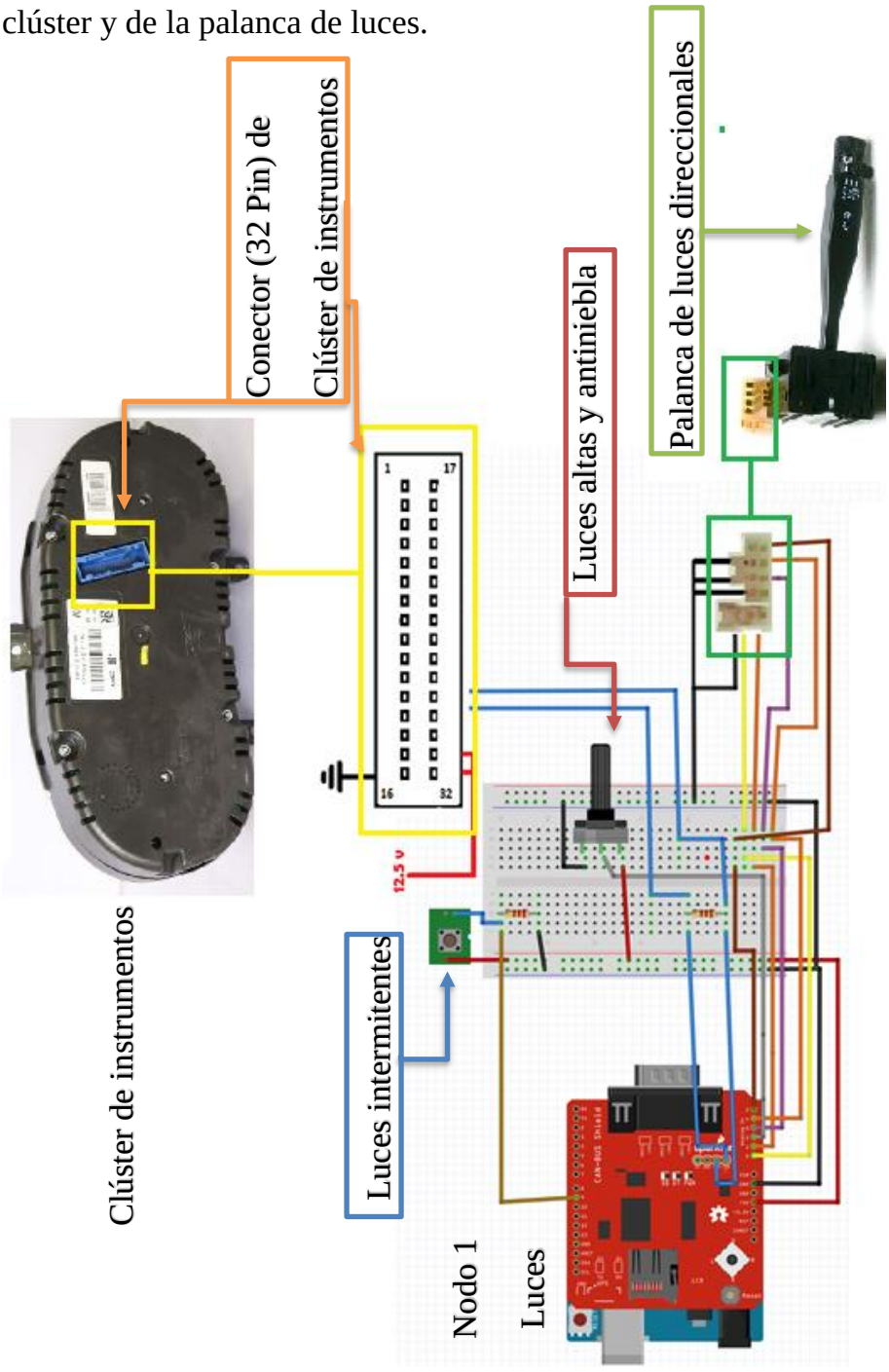


Figura 5.9 Conexión del panel de instrumentos al nodo de luces

En la figura 5.10 se muestran los mensajes recibidos por el nodo 1 del panel de instrumentos y el mensaje que el nodo de luces envía se resalta en rojo. También se numeran los 8 bytes del mensaje y se describe la información que se envía en cada uno de ellos según el valor que se muestra en la figura 5.10. La tabla del identificador 470 correspondiente a luces y los valores de los bytes de muestra completa en el apartado de apéndice.

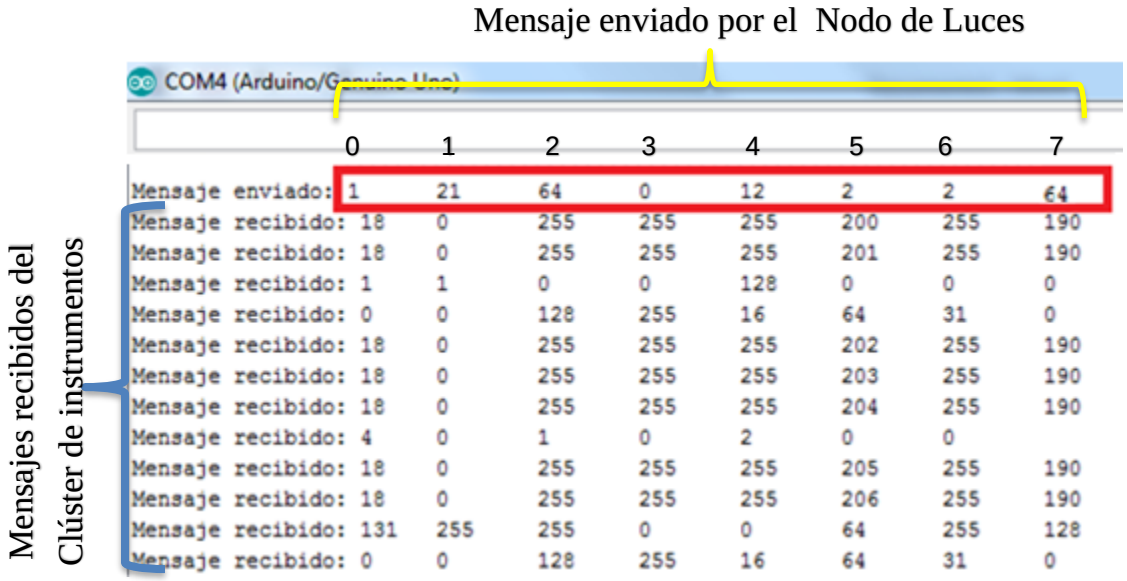


Figura 5.10 Lectura del mensaje enviado correspondiente a luces

El nodo de luces envía un mensaje de 8 bytes, donde el byte 0 y el byte 2 son controlados mediante la palanca de luces y el byte 7 por un potenciómetro. La información enviada en cada uno de los bytes del mensaje durante esta prueba se describe a continuación:

El byte 0 corresponde a las luces direccionales, en este caso el número uno que se muestra en el mensaje significa que la luz izquierda ha sido encendida mediante la palanca de luces y en el clúster se enciende la luz correspondiente a esta indicación.

Al byte 1 se asignó el valor de 21 para que el cuadro de instrumentos presente el testigo de puerta abierta, este dato se asigna manualmente en el código.

El byte 2 se utilizó para mostrar en el clúster la indicación de luces de posición y luces bajas, al no tener testigos para estas se utiliza la intensidad de iluminación del cuadro para indicarle al conductor el encendido de las luces, en la figura 5.10 se observa que la información enviada por la palanca fue el encendido de las luces bajas, el código se programó para que cuando se enciendan las luces de posición se envié 32 como dato y 64 cuando se elijan las luces bajas, estos dos valores corresponden al 50 y 100% de intensidad de iluminación del clúster.

El byte 3 no se utiliza para enviar ninguna información por lo cual se envía un cero.

El byte 4 muestra un valor que corresponde al daño de una bombilla.

El byte 5 y 6 no se utilizan para enviar ningún dato, en la figura 5.10 se muestra el valor 2 en estos bytes con la intención de mostrar que no afectan el mensaje.

El byte 7 es utilizado para enviar información del encendido de luz antiniebla trasera y luces altas, el mensaje que se muestra en esta imagen representa el encendido de las luces altas, debido a que la palanca no cuenta con la función de luces antiniebla se implementó un potenciómetro para realizar esta indicación. En el apartado de apéndice se muestra la tabla de los diferentes valores que puede tomar cada uno de los bytes del mensaje dependiendo de la indicación que se desee mostrar.

En la figura 5.11 se muestra el encendido de los testigos del clúster de instrumentos correspondientes al mensaje que muestra en la figura 5.10 y se comprueba que la respuesta al mensaje recibido es inmediata, ya que los testigos se encienden al mover la palanca.

Figura 5.11 Encendido de testigos del clúster de instrumentos

5.4 Construcción de Red CAN Bus y conexión del clúster de instrumentos

Para la construcción de la Red se seleccionaron 3 de los identificadores de mensaje que se localizaron para este panel de instrumentos, estos IDs son velocidad, luces y RPM. Para la implementación de la Red CAN de 3 nodos se agregaron 2 nodos más. De los dos nodos adicionales, el primero es encargado de las Revoluciones por minuto del motor, a este nodo se le cargo un programa similar al de la palanca de luces, cambiando el identificador de mensaje, la trama de datos y la lectura del sensor. El segundo nodo adicional se encarga de la velocidad del automóvil. Los identificadores de mensaje que se le cargaron a cada uno de los nodos se muestran en la tabla 9.

Tabla 9 Implementación de nodos y descripción del mensaje a enviar

Número de nodo	Mensaje	Identificador
Nodo 1	Luces	470
Nodo 2	Revoluciones por minuto	280
Nodo 3	Velocidad	5A0

Antes de conectar el panel de instrumentos a la red se realizaron varias pruebas sobre la trama del mensaje enviado, la comunicación entre los tres nodos y envío de mensajes de acuerdo a la prioridad del identificador del nodo.

Prueba 1. Decodificación de un mensaje enviado.

La primera prueba fue la decodificación de uno de los mensajes enviados, para esta prueba se seleccionó el mensaje enviado por el nodo 2 que corresponde a las revoluciones por minuto (ID: 280), en la figura 5.12 se muestra la decodificación de la señal CAN_H del nodo utilizando un osciloscopio con analizador de protocolo CAN. En la trama de datos de la figura 5.12 se muestra la señal CAN_H acompañada en la parte inferior del análisis realizado por el osciloscopio, este análisis se enfoca en 4 campos importantes de la trama los cuales son: inicio de trama, Campo de control, campo de datos y campo de comprobación de redundancia cíclica, dichos marcos se muestran en sistema hexadecimal y binario para poder realizar el análisis de la señal. Posteriormente se señaló en la figura los 7 campos que forman la trama de datos.

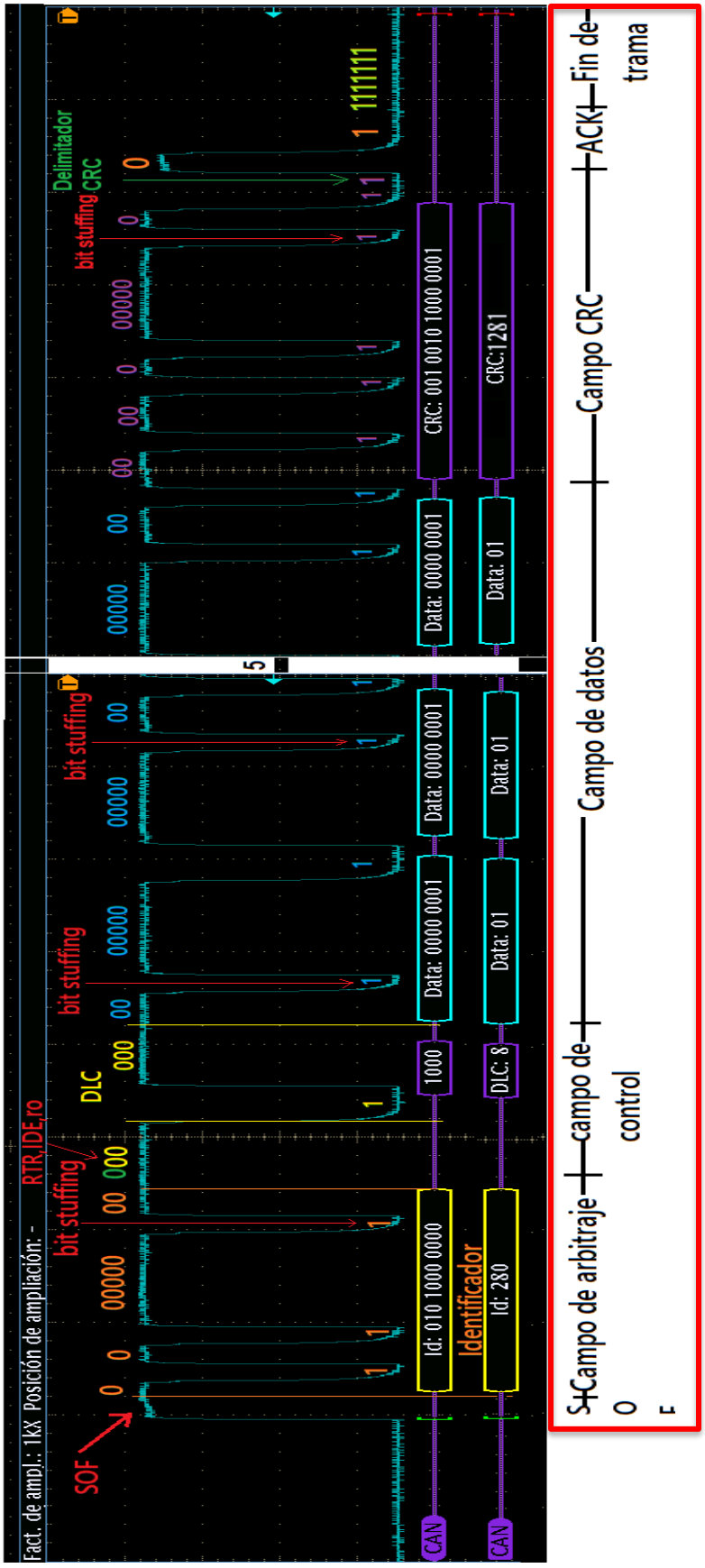


Figura 5.12 Decodificación de una trama de datos CAN.

El primer campo es el inicio de trama (SOF, por sus siglas en inglés) es un bit dominante (0) que denota el inicio de trama y sirve para la sincronización de los nodos de la red.

En el segundo campo se encuentra el identificador de mensaje donde al comparar la señal con el valor en sistema binario se observa que se ha agregado un bit de relleno (bit stuffing), este bit es enviado por el nodo remitente quien transmite el bit complementario después de cinco bits homogéneos. El bit de relleno se utiliza como mecanismo de re sincronización para mantener el sincronismo hasta el final de la transmisión del mensaje (comienza con la transmisión del SOF y termina con la transmisión del último bit de la secuencia CRC), razón por la cual se puede observar que se agrega 3 veces más en la trama de datos. Este bit no afecta la trama de datos ya que es eliminado en el nodo receptor.

Después se muestra el campo de control formado por los bits RTR, IDE, ro donde todos son dominantes ya que el primero denota que se trata de una trama de datos, el segundo que es una trama estándar (identificador de 11 bits) y el bit r0 (bit reservado) que tiende a ser dominante. En este mismo campo se localiza la longitud del mensaje (DLC= 8) en la parte inferior se observa el dato en sistema hexadecimal y binario.

Posteriormente se localiza el campo de datos, donde solo se observan 3 bytes del mensaje ya que debido a la longitud se decidió recortar la imagen para poder mostrar todos los campos de la trama, en este campo también se observa la implementación de bits de relleno.

Seguidamente se encuentra el campo de comprobación de redundancia cíclica, el campo de comprobación y el final de trama. En la figura 5.12 también se puede verificar que se está enviado el mensaje a una velocidad de 500 kbps ya que cada bit tiene una longitud de 2 microsegundos, transmitiendo la trama de datos en aproximadamente 0.25 ms debido a que es un mensaje de 8 bytes [4]. Después de decodificar la trama de datos con el fin de verificar que se cumpliera con los campos de la trama, que la trama no se distorsionara y se implementaran los bits de relleno de forma correcta, se puede confiar en que la trama de datos se está transmitiendo de forma adecuada.

Prueba 2. Lectura de mensajes en la Red usando Osciloscopio con analizador CAN

Posteriormente se realizó una prueba de la lectura del envío de mensajes de los tres nodos de la red, esto para probar que los mensajes se enviaran de acuerdo a la prioridad del identificador de cada nodo y que todas las tramas de datos se estuvieran enviando. En la figura 5.13 se muestran las tres tramas de datos utilizando el osciloscopio.

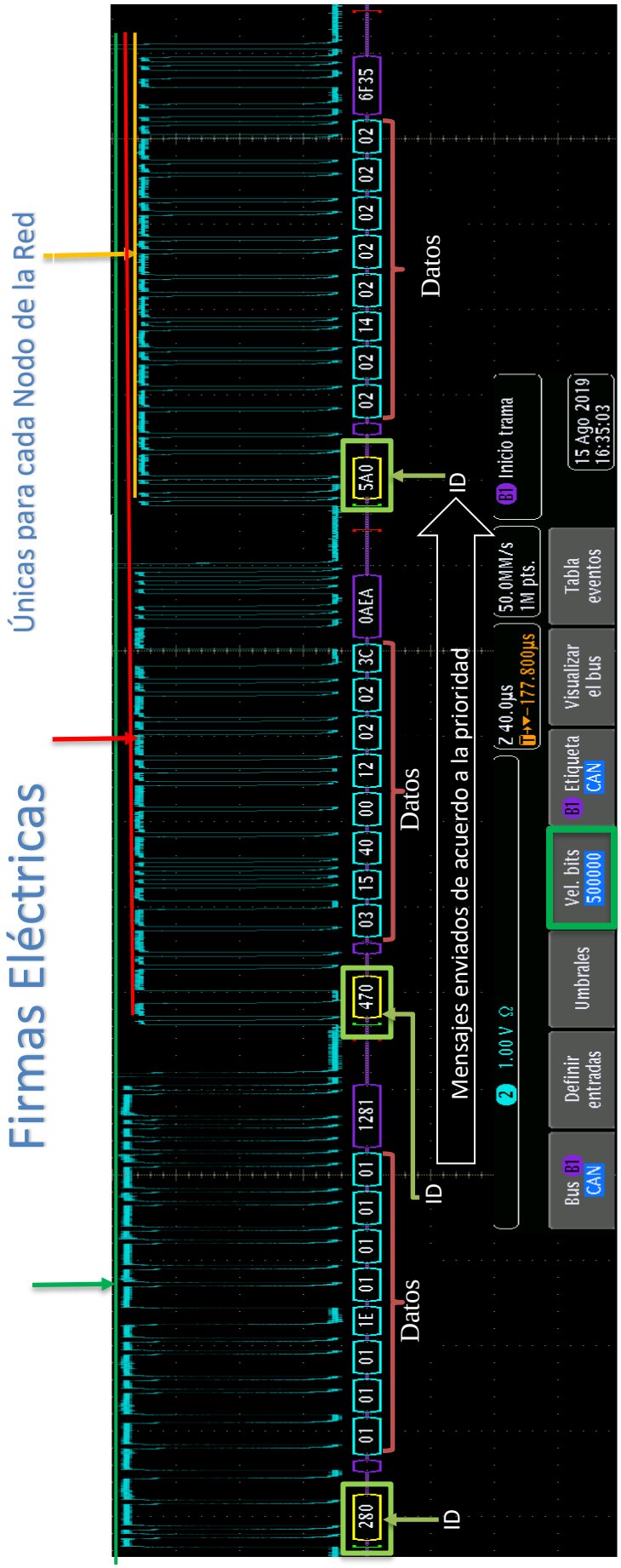


Figura 5.13 Lectura de las tramas de datos enviadas por los 3 nodos de la Red

En esta figura 5.13 se observan las tres tramas de datos de los nodos de la Red CAN, en ellas se puede verificar el envío de los mensajes de acuerdo a la prioridad. También se muestran los identificadores de mensaje, los datos enviados y el campo de comprobación de redundancia cíclica.

En la parte superior de las tramas de datos se observan 3 líneas punteadas (verde, naranja y amarilla) que demuestran que las tres tramas de datos tienen diferentes amplitudes, esto se debe según [51] a las firmas eléctricas de los mensajes enviados por un nodo CAN, esto sucede debido a la composición física del Bus (por ejemplo posición del nodo y distancia al Bus). Una firma eléctrica es única para cada nodo. Y si se deseara obtener una firma única para un mensaje CAN de un nodo de transmisión, las mediciones deben tomarse desde el campo de datos de la trama CAN, que es cuando solo un nodo está generando los datos CAN.

Prueba 3. Lectura de mensajes en la Red empleando Bus Monitor

Posteriormente se realizó una prueba utilizando el NI-XNET Bus Monitor, para probar que los nodos que se enviaran los mensajes de acuerdo a su prioridad. En la figura 5.14 se muestran los resultados, donde se comprueba el correcto envío de mensajes de acuerdo al orden de importancia tomando en cuenta la marca de tiempo (Cuadro naranja). También se comprueba que todos los nodos han enviado la misma cantidad de mensajes (Cuadro verde) por lo cual se comprueba que los tres nodos han estado enviando mensajes al Bus.

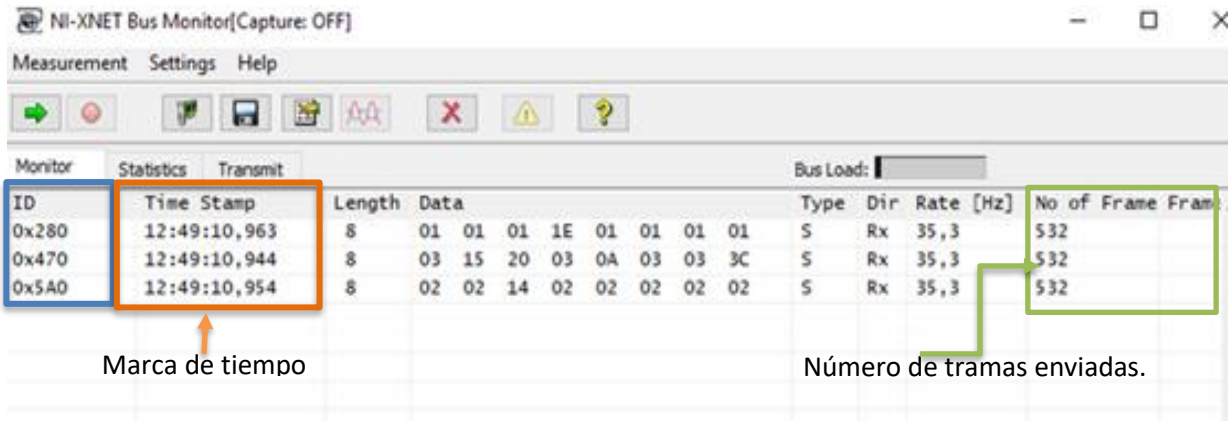


Figura 5.14 Lectura mediante NI-XNET Bus Monitor de mensajes enviados por los nodos de la Red CAN

Prueba 4. Lectura de mensajes de la Red empleando LabVIEW

Para poder visualizar el orden de una mayor cantidad de tramas de datos se empleó un ejemplo de un VI para leer datos de red, a este ejemplo se le agrego una tabla adicional para la lectura de datos. En la figura 5.15 se muestra el diagrama de bloques de este VI y se puede ver que este programa crea una sesión NI-XNET en tiempo de ejecución, la sesión

es la conexión entre los productos de hardware can con la red externa. Después se encuentra el bloque frame in stream el cual es encargado de leer todos los cuadros recibidos de la red utilizando una sola secuencia, normalmente se utiliza para analizar y/o registrar todo el tráfico de cuadros en la red [52]. La interfaz se configura a 500 kbps. Para la lectura de las tramas se emplea Frame CAN debido a que es la forma recomendada de leer los datos para este modo[53]. A programa se le implementó una tabla más para la lectura de los datos y poder visualizar más a detalle la secuencia de las tramas de datos que se reciben.

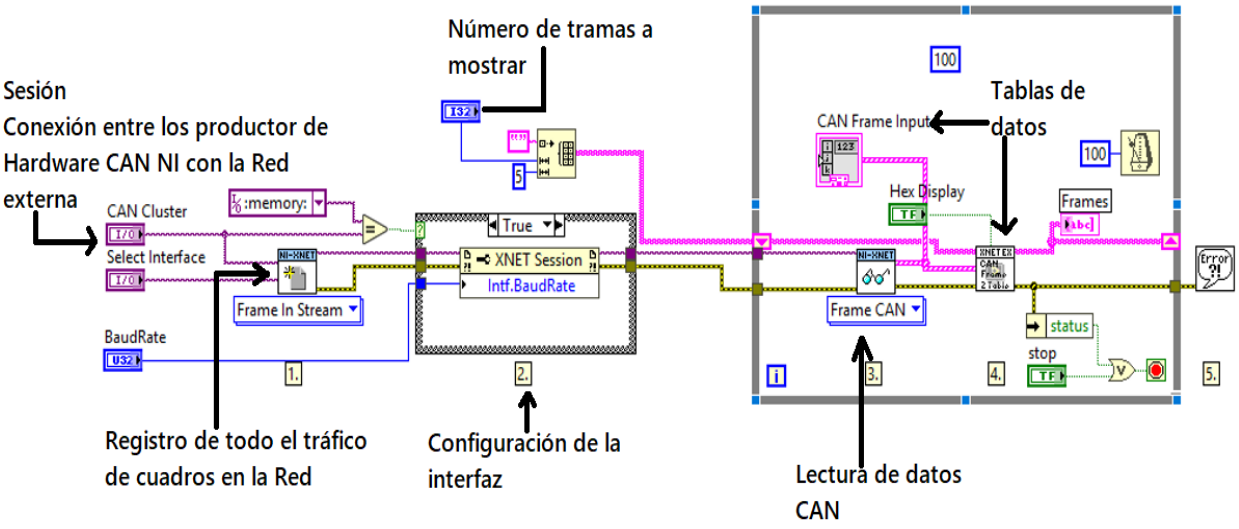


Figura 5.15 Programa LabVIEW para la lectura de las tramas de datos

Para probar el programa se realizó la lectura de las tramas de datos de los tres nodos de la red, en la figura 5.16 se observan los resultados. Los datos que se muestran son: el tiempo de muestra, los datos enviados, el identificador del nodo, el tipo de trama y si se trata de una trama de dato estándar o extendido (Rectángulo naranja). Además de que se pueden implementar el número de muestras para visualizar o cambiar la velocidad del Bus, por lo que puede ser utilizado en la lectura de tramas a diferente velocidad. En nuestra lectura podemos visualizar que los mensajes siguen su prioridad al enviar mensajes al Bus. Es importante mencionar que para realizar la lectura de las tramas de datos se necesita de emplear el sistema PXI que se mencionó en el capítulo 3 sección 4, donde se definen los tres elementos del sistema, los cuales hacen posible realizar la interfaz. Los elementos del sistema empleados en este trabajo son el Chasis PXI-1042, el controlador NI PXI-8101 y el Módulo de interfaz CAN PXI-8513. Ya que de esta forma se puede conectar el controlador con la computadora y emplear el módulo CAN para proporcionar la comunicación con la red externa. El beneficio que se obtiene al emplear las interfaces NI-XNET es que ayudan a desarrollar aplicaciones para crear prototipos, simular y probar redes como CAN, LIN y FlexRay de forma más rápida y sencilla en NI LabVIEW. Las interfaces NI-XNET están diseñadas para funcionar en un entorno de tiempo real con lo

que hace que las simulaciones sean más precisas y que los sistemas de prueba sean más confiables [54].

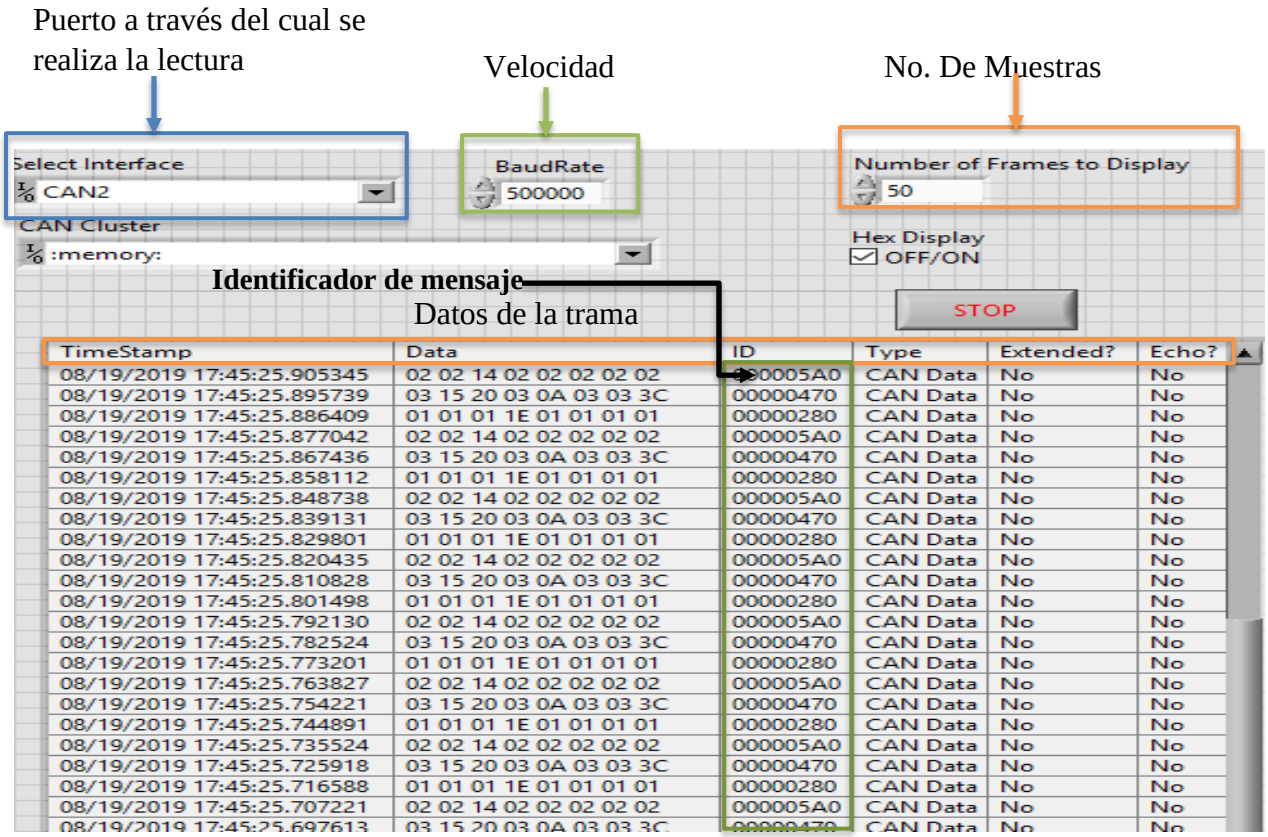


Figura 5.16 Lectura de tramas de datos de la Red empleado programa LabVIEW

Prueba 5. Comportamiento de la Red ante la inserción de uno nodo de mayor prioridad.

Después se realizó una prueba para observar que sucedía cuando a dos nodos que se encuentran en la red se les implementaba otro nodo de mayor prioridad (identificador más bajo). Para realizar esta prueba fue necesario incrementar la cantidad de muestras a 1000 para poder visualizar los cambios debido a la velocidad con la que se envían los mensajes. En la figura 5.18 podemos ver los resultados obtenidos y se observa que los dos nodos que se encuentran en la red envían sus mensajes alternamente y en el momento que se inserta un tercer nodo a la red este envía tres mensajes consecutivos, esto podría deberse a que es un nodo de mayor prioridad y gana el arbitraje consecutivamente (figura 5.17) pero después de esto los 3 nodos empiezan a enviar mensajes alternamente de acuerdo a su prioridad.

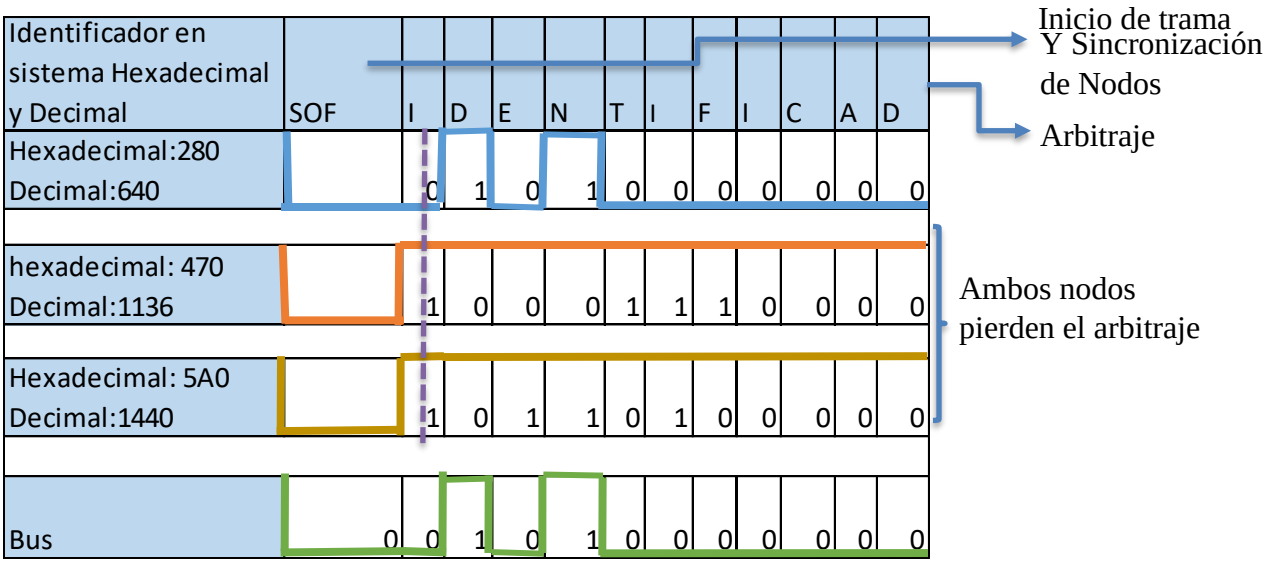


Figura 5.17 Arbitraje de los tres nodos de la Red

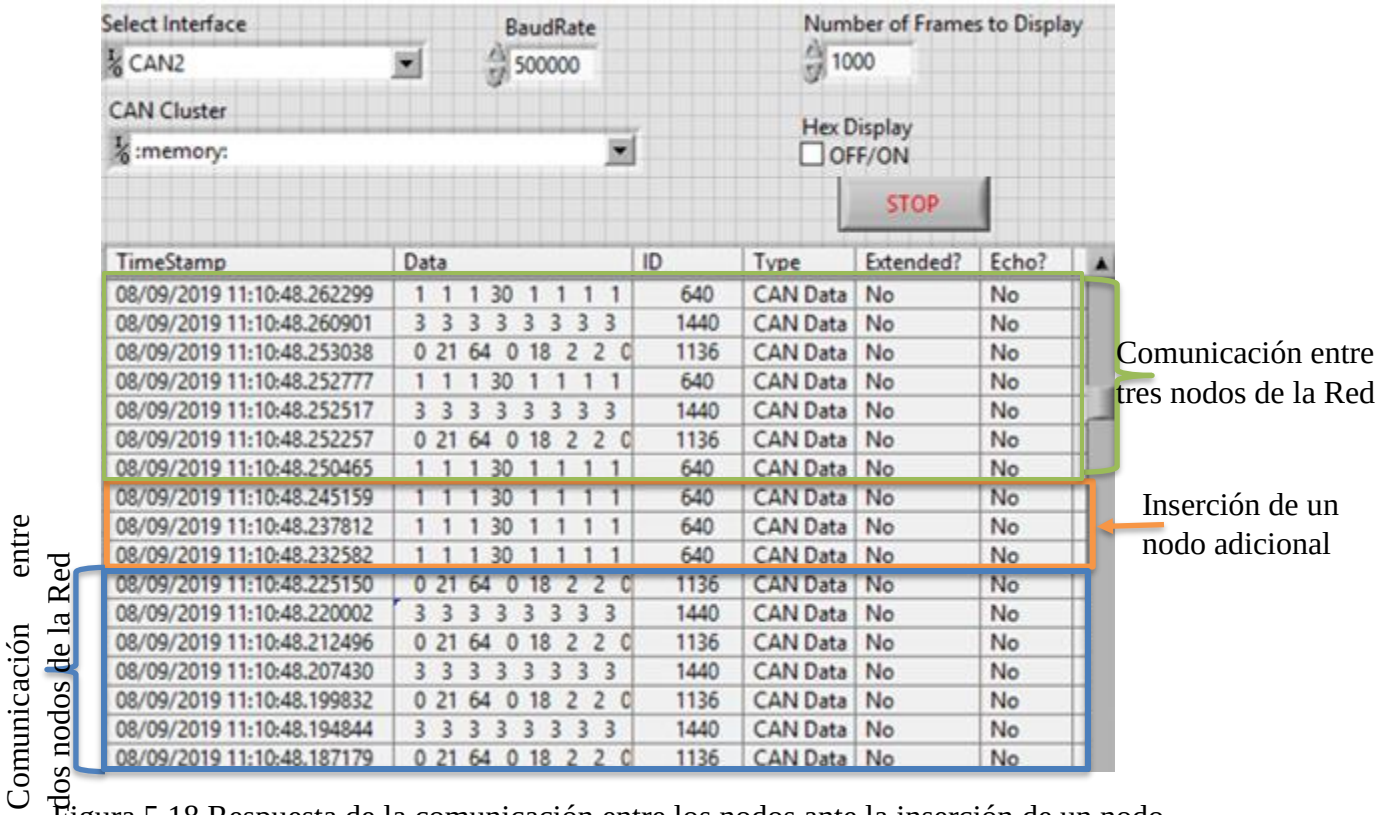


Figura 5.18 Respuesta de la comunicación entre los nodos ante la inserción de un nodo

Después de realizar estas pruebas se procedió a conectar el panel de instrumentos a la Red CAN. En la figura 5.19 se observa el diagrama de la conexión de los tres nodos y el Clúster de instrumentos.

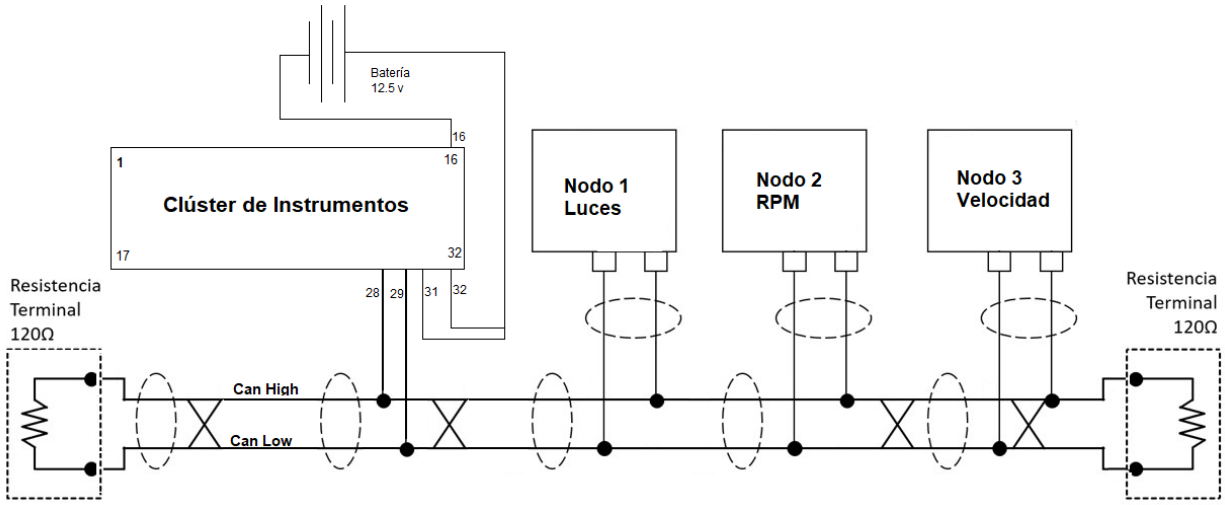


Figura 5.19 Diagrama de Conexión del clúster de instrumentos y la Red CAN de 3 nodos.

En las figuras 5.20, 5.21, 5.22 se pueden ver tres diagramas ilustrativos de la conexión del clúster de instrumentos a la Red CAN. En la figura 5.20 se muestran los 3 nodos con sus sensores y el Clúster conectados al mismo Bus. En la figura 5.21 se observa que al nodo de luces se implementó como sensor una palanca de luces direccionales, un push-button para la selección de luces intermitentes y un potenciómetro para simular el encendido de luces antiniebla y luces altas. Al nodo de velocidad se implementó un potenciómetro debido a que no se cuenta con un pedal de acelerador y al igual que al nodo de RPM. En la figura 5.21 se observa la conexión de todos los elementos al mismo Bus y resistencias terminadoras de 120 Ohm. En la figura 5.22 se muestra un diagrama eléctrico automotriz.

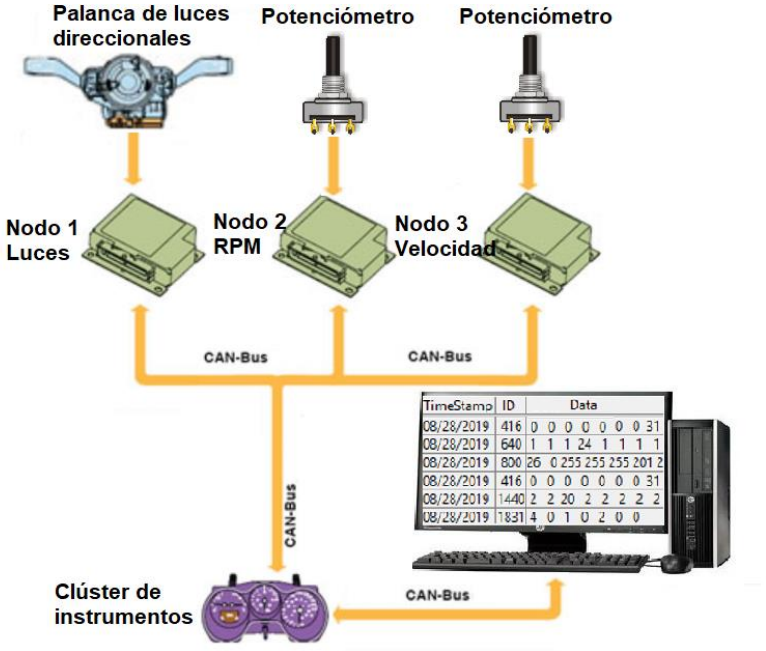


Figura 5.20 Conexión de los 3 Nodos y el Clúster de instrumentos al Bus.

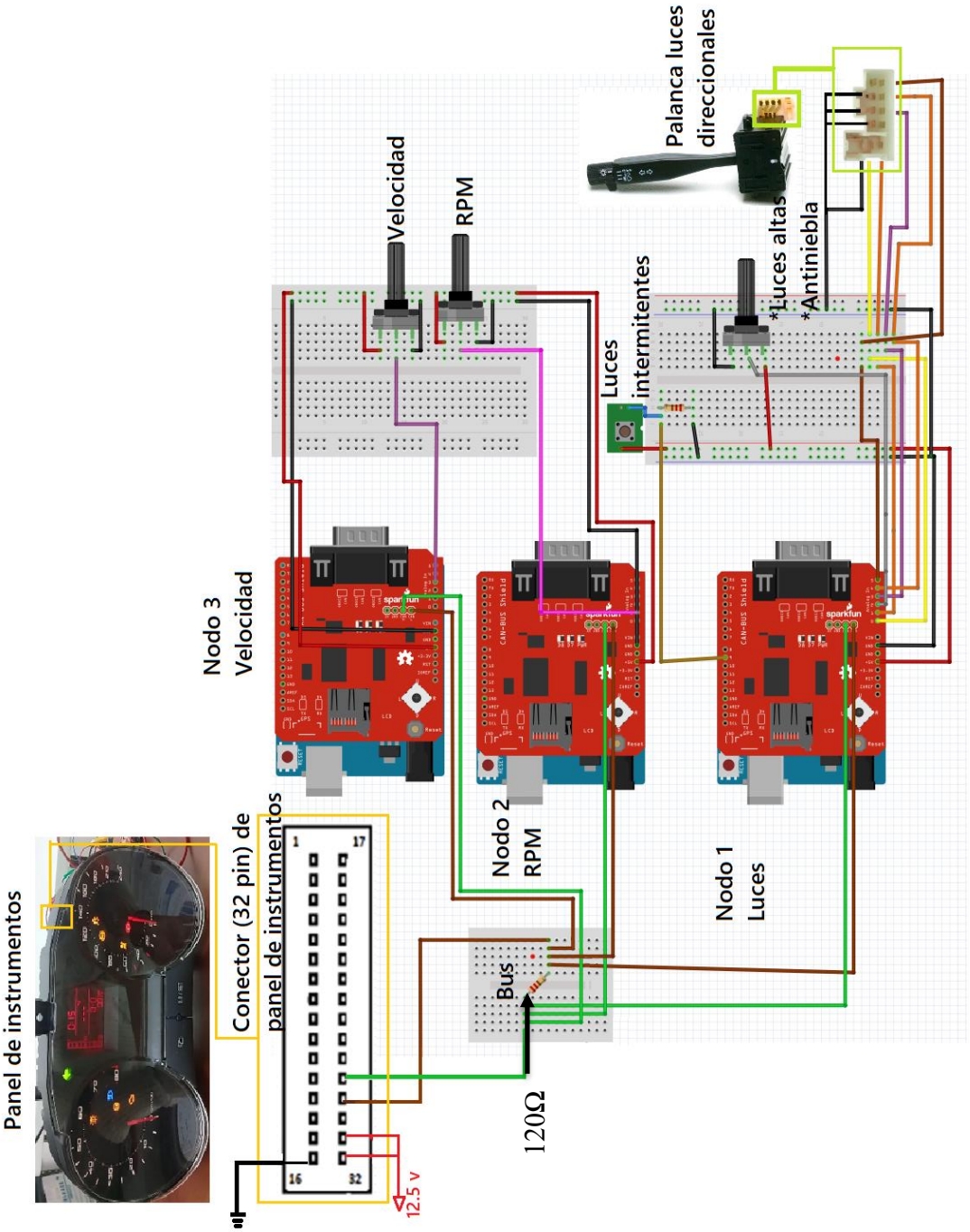


Figura 5.21 Conexión de la Red CAN y el clúster de instrumentos.

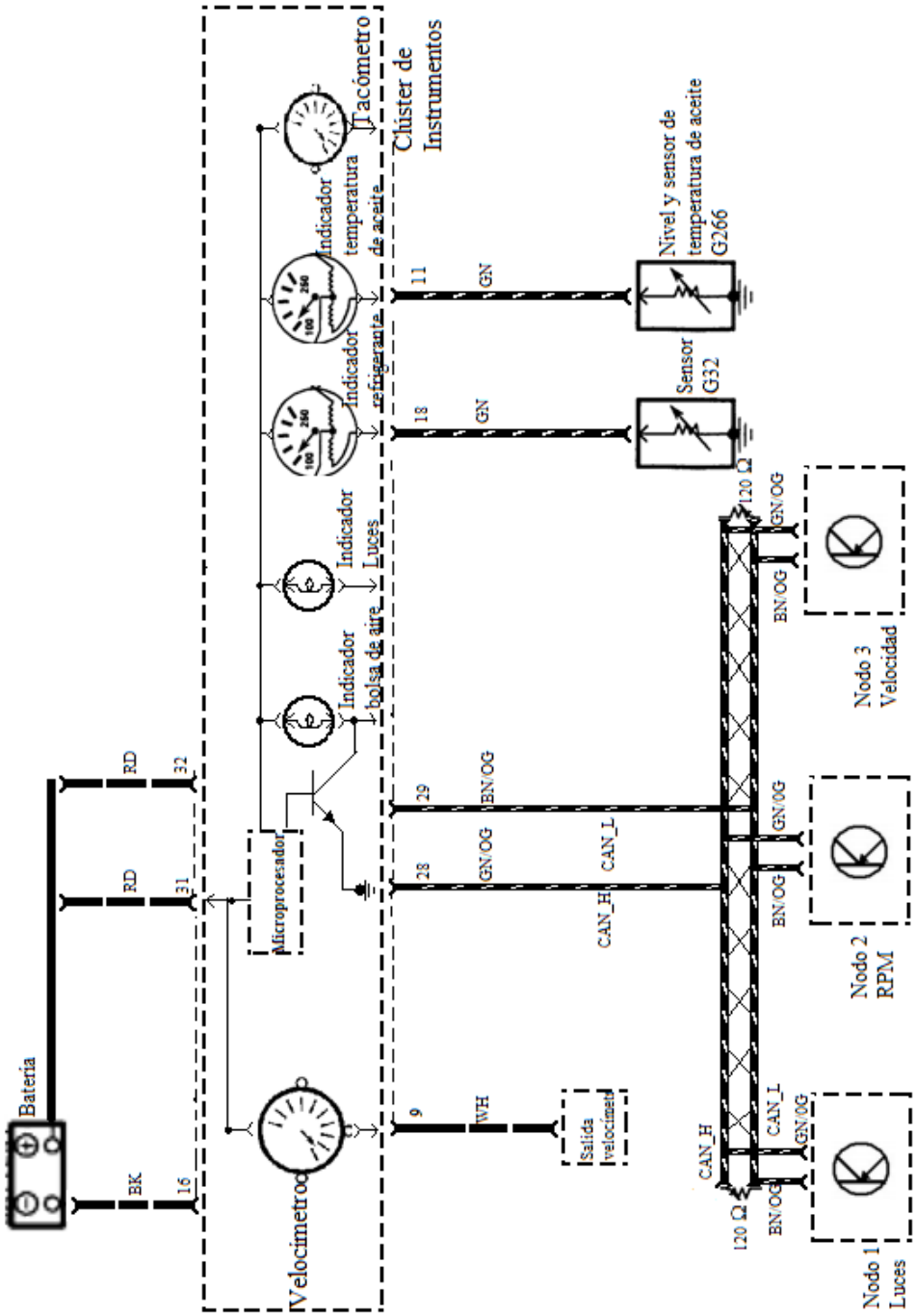


Figura 5.22 Diagrama eléctrico automotriz de la conexión del clúster de instrumentos y la Red CAN de tres Nodos

CAPÍTULO 6

Resultados y Conclusiones

6.1 Resultados de comunicación y verificación de envío conforme a la prioridad

Para probar la comunicación entre los nodos y la realización de la misma de acuerdo a la prioridad de los mensajes se efectuaron diferentes tipos de pruebas y para realizarlas se emplean:

- El osciloscopio con analizador de protocolo CAN
- El programa realizado en LabVIEW con interfaz CAN de NI-XNET
- El Bus Monitor NI-XNET.

Las pruebas que se realizan con el osciloscopio y con LabVIEW son utilizadas para observar la interacción entre los nodos y el clúster de instrumentos, para probar la comunicación de acuerdo a la prioridad y la respuesta ante el cambio de lectura de los valores de los sensores. Con estas dos pruebas se puede observar el orden en que se envían los mensajes a la red.

Por otra parte, se emplea el Bus Monitor NI-XNET para ver la cantidad de tramas que cada nodo ha enviado a la red y el historial de carga en el Bus.

La primera prueba se realiza utilizando el programa realizado en LabVIEW, en la figura 6.1 se muestra la conexión de la red y el panel de instrumentos al sistema PXI y este a su vez conectado al CPU de una computadora para la lectura de las tramas de datos mediante LabVIEW. En la figura 6.1 se observa la conexión de los sensores a cada uno de los nodos, también se observa la conexión de los nodos, el panel de instrumentos y el sistema PXI al Bus donde los cables de los canales CAN_H y CAN_L son verde y marrón respectivamente. Los colores del Bus se ilustran de estos colores debido a que en el manual del vehículo Seat [4] especifica que estos son los colores de los canales CAN_H y CAN_L del Bus tracción, el cual envía la información a 500 kbps razón por la cual se selecciona estos colores de cable. También se puede ver que la conexión del sistema PXI al Bus es mediante un cable serial DB9 en los pines 7 y 2 para CAN_H y CAN_L respectivamente. Finalmente se puede ver la conexión Ethernet entre el sistema PXI y el CPU para la visualización de datos.

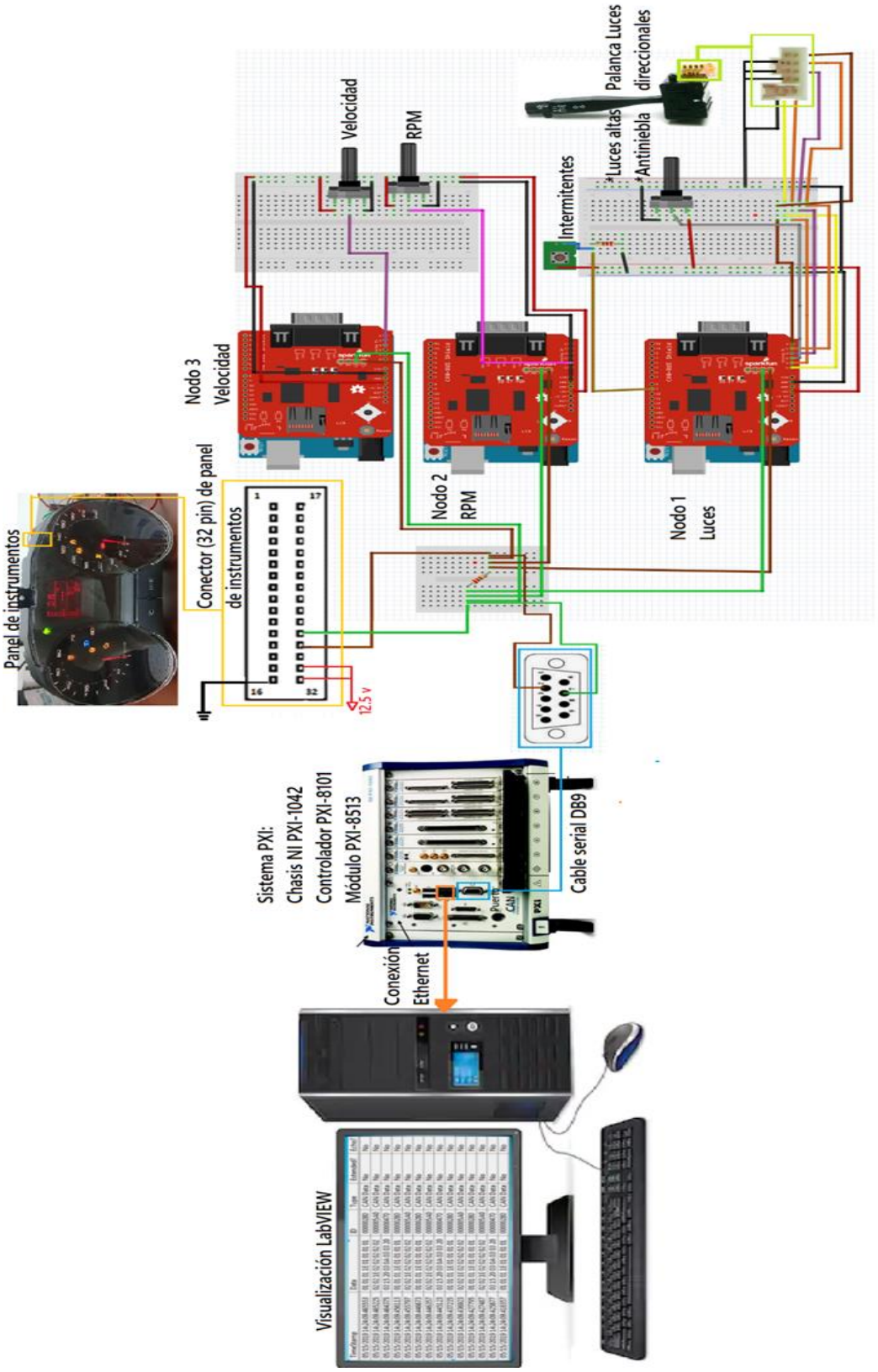
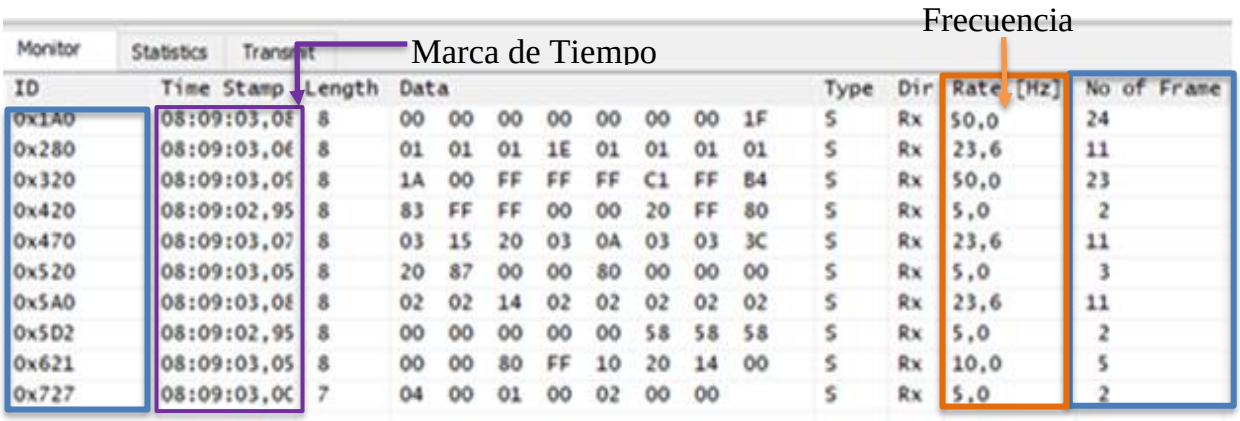


Figura 6.1 Conexión de la Red al sistema PXI

Empleando la conexión mostrada en la figura 6.1 se pueden realizar las pruebas empleando el Bus Monitor NI-XNET y el programa realizado en LabVIEW con interfaz CAN.

Prueba 1. Lectura de mensajes en el CAN Bus empleando Bus Monitor

Esta prueba es realizada con el Bus Monitor, la respuesta se muestra en la figura 6.2 donde se pueden ver los identificadores de mensaje y el número de tramas que han enviado.



ID	Time Stamp	Length	Data								Type	Dir	Rate [Hz]	No of Frame
0x1A0	08:09:03,0[illegible]	8	00	00	00	00	00	00	00	1F	S	Rx	50,0	24
0x280	08:09:03,0[illegible]	8	01	01	01	1E	01	01	01	01	S	Rx	23,6	11
0x320	08:09:03,0[illegible]	8	1A	00	FF	FF	FF	C1	FF	B4	S	Rx	50,0	23
0x420	08:09:02,95	8	83	FF	FF	00	00	20	FF	80	S	Rx	5,0	2
0x470	08:09:03,07	8	03	15	20	03	0A	03	03	3C	S	Rx	23,6	11
0x520	08:09:03,05	8	20	87	00	00	80	00	00	00	S	Rx	5,0	3
0x5A0	08:09:03,0[illegible]	8	02	02	14	02	02	02	02	02	S	Rx	23,6	11
0x5D2	08:09:02,95	8	00	00	00	00	00	58	58	58	S	Rx	5,0	2
0x621	08:09:03,05	8	00	00	80	FF	10	20	14	00	S	Rx	10,0	5
0x727	08:09:03,0[illegible]	7	04	00	01	00	02	00	00		S	Rx	5,0	2

Figura 6.2 Lectura de tramas de datos en el Bus utilizando el Bus Monitor.

En la figura 6.2 se resaltan 4 de los apartados de la opción Monitor: Identificadores de mensaje, Marca de tiempo, Frecuencia y Número de tramas.

El primer apartado muestra los identificadores de mensaje ordenados de acuerdo a su prioridad.

El segundo apartado es la marca de tiempo (time Stamp, por su traducción al inglés) en este apartado podemos observar el orden en el que se fueron enviando los mensajes ya que el valor que se muestra en ese apartado es el tiempo absoluto en que la interfaz XNET recibió la trama (fin de la trama)[49].

El tercer apartado es la frecuencia (número de veces por segundo que se repite una trama) en la figura 6.2 se observa que las frecuencias son diferentes dependiendo del identificador de mensaje, esto se debe a que en una comunicación en tiempo real la necesidad de enviar un mensaje a los diferentes nodos que forman la red puede variar dependiendo de la prioridad de los mensajes o de los cambios rápidos de los parámetros de una trama de datos. Por ejemplo, el ABS cambiara con mayor frecuencia que las luces, ya que durante esta prueba se encontró que la frecuencia de envío del ABS tenía variaciones de 50 a 60 Hz. Si comparamos la frecuencia con la que se repite la trama de datos encontradas con la frecuencia de los identificadores de mensaje de un VW Passat [14] de los cuales algunos coinciden con los localizados para el panel de instrumentos que se utiliza para este trabajo se puede observar que los identificadores que tienen mayor importancia tiene una frecuencia alta, tal es el caso de la velocidad, las Revoluciones por Minuto, ABS, Clúster de instrumentos (ID:320).

El último apartado muestra el número de tramas de datos que han enviado dependiendo del identificador de mensaje.

Prueba 2. Lectura de la carga en el Bus empleando Bus Monitor

Para visualizar el historial de carga del Bus en la figura 6.3 se observa la carga en la opción de estadística del Bus Monitor NI-XNET, se enmarca en un recuadro naranja el porcentaje de carga y numero de tramas que se han enviado antes y después de conectar la red con el panel de instrumentos. También se señala el número de tramas por segundo.

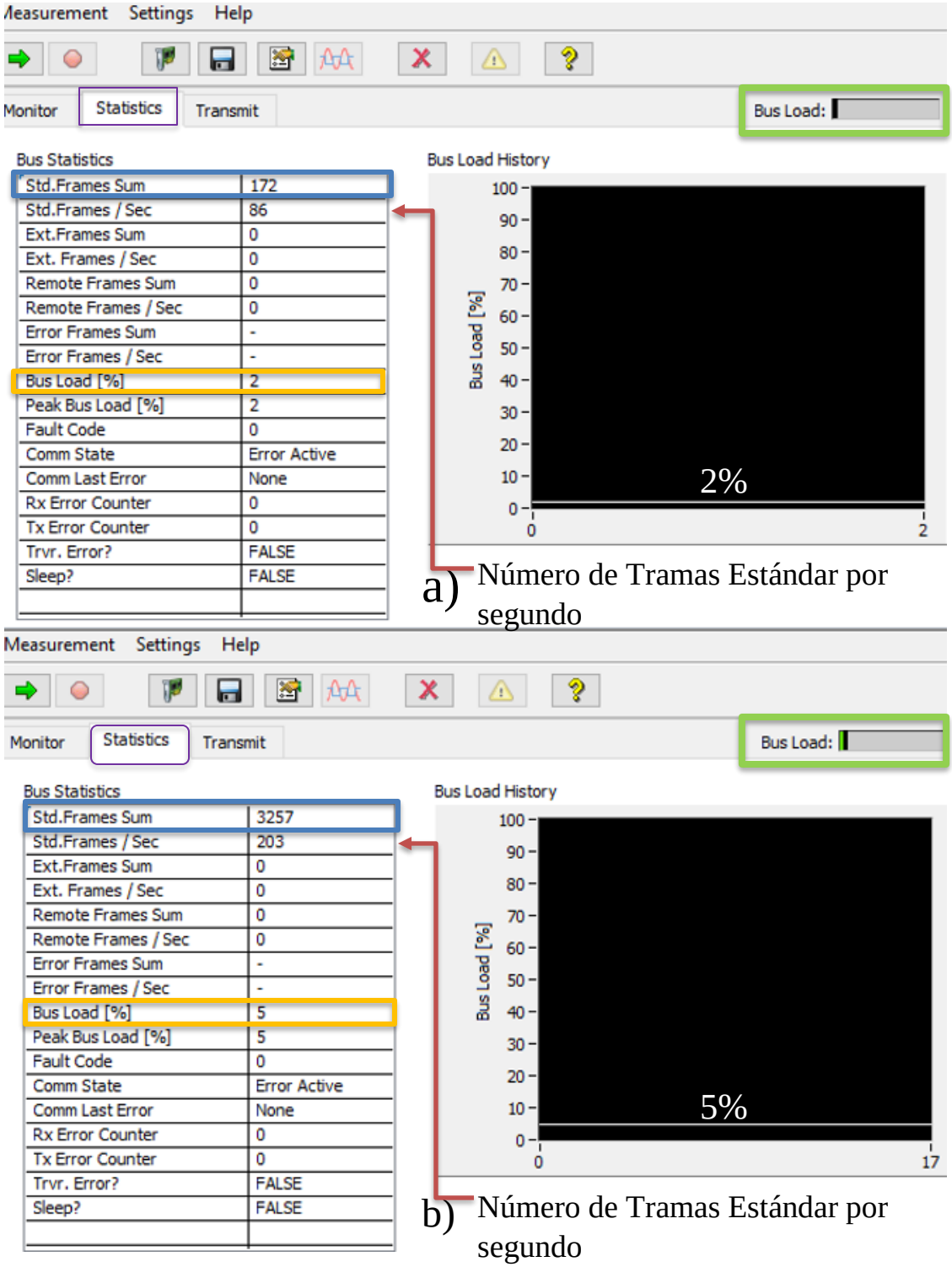
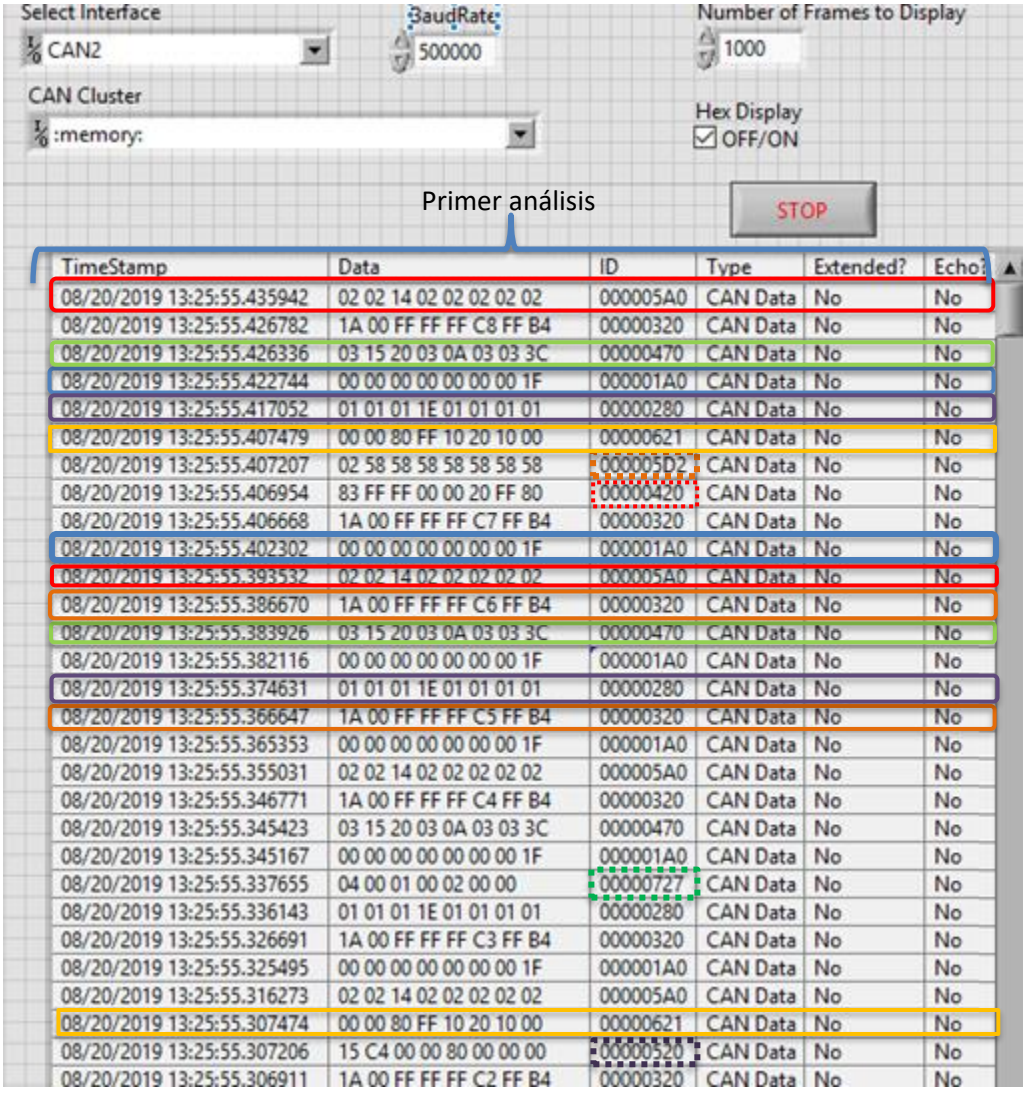


Figura 6.3 Carga en el Bus a) Carga solo con el clúster de instrumentos b) Carga con clúster de instrumentos y Red CAN

En la Figura 6.3 se observa que no se genera ninguna trama de error o sobrecarga, los contadores de error de transmisión y recepción no se incrementan al añadir los tres nodos de la red al Bus, por lo cual se afirma que los nodos y la Red se comunican de forma adecuada.

Prueba 3. Análisis de la lectura de la secuencia de las tramas empleando LabVIEW

Se realizó una prueba utilizando el programa realizado en LabVIEW para la lectura de la secuencia de las tramas y se realizan dos análisis de las tramas. En la figura 6.4 se muestran los resultados obtenidos y se realiza un primer análisis de las tramas para probar que las frecuencias de los identificadores sean las mismas que las obtenidas con los resultados de la prueba realizada con el Bus Monitor (Figura 6.2), para esto se resaltan en rectángulos del mismo color las tramas de datos de un mismo identificador de mensaje.



TimeStamp	Data	ID	Type	Extended?	Echo?
08/20/2019 13:25:55.435942	02 02 14 02 02 02 02 02	000005A0	CAN Data	No	No
08/20/2019 13:25:55.426782	1A 00 FF FF FF C8 FF B4	00000320	CAN Data	No	No
08/20/2019 13:25:55.426336	03 15 20 03 0A 03 03 3C	00000470	CAN Data	No	No
08/20/2019 13:25:55.422744	00 00 00 00 00 00 00 1F	000001A0	CAN Data	No	No
08/20/2019 13:25:55.417052	01 01 01 1E 01 01 01 01	00000280	CAN Data	No	No
08/20/2019 13:25:55.407479	00 00 80 FF 10 20 10 00	00000621	CAN Data	No	No
08/20/2019 13:25:55.407207	02 58 58 58 58 58 58 58	000005D2	CAN Data	No	No
08/20/2019 13:25:55.406954	83 FF FF 00 00 20 FF 80	00000420	CAN Data	No	No
08/20/2019 13:25:55.406668	1A 00 FF FF FF C7 FF B4	00000320	CAN Data	No	No
08/20/2019 13:25:55.402302	00 00 00 00 00 00 00 1F	000001A0	CAN Data	No	No
08/20/2019 13:25:55.393532	02 02 14 02 02 02 02 02	000005A0	CAN Data	No	No
08/20/2019 13:25:55.386670	1A 00 FF FF FF C6 FF B4	00000320	CAN Data	No	No
08/20/2019 13:25:55.383926	03 15 20 03 0A 03 03 3C	00000470	CAN Data	No	No
08/20/2019 13:25:55.382116	00 00 00 00 00 00 00 1F	000001A0	CAN Data	No	No
08/20/2019 13:25:55.374631	01 01 01 1E 01 01 01 01	00000280	CAN Data	No	No
08/20/2019 13:25:55.366647	1A 00 FF FF FF C5 FF B4	00000320	CAN Data	No	No
08/20/2019 13:25:55.365353	00 00 00 00 00 00 00 1F	000001A0	CAN Data	No	No
08/20/2019 13:25:55.355031	02 02 14 02 02 02 02 02	000005A0	CAN Data	No	No
08/20/2019 13:25:55.346771	1A 00 FF FF FF C4 FF B4	00000320	CAN Data	No	No
08/20/2019 13:25:55.345423	03 15 20 03 0A 03 03 3C	00000470	CAN Data	No	No
08/20/2019 13:25:55.345167	00 00 00 00 00 00 00 1F	000001A0	CAN Data	No	No
08/20/2019 13:25:55.337655	04 00 01 00 02 00 00	00000727	CAN Data	No	No
08/20/2019 13:25:55.336143	01 01 01 1E 01 01 01 01	00000280	CAN Data	No	No
08/20/2019 13:25:55.326691	1A 00 FF FF FF C3 FF B4	00000320	CAN Data	No	No
08/20/2019 13:25:55.325495	00 00 00 00 00 00 00 1F	000001A0	CAN Data	No	No
08/20/2019 13:25:55.316273	02 02 14 02 02 02 02 02	000005A0	CAN Data	No	No
08/20/2019 13:25:55.307474	00 00 80 FF 10 20 10 00	00000621	CAN Data	No	No
08/20/2019 13:25:55.307206	15 C4 00 00 80 00 00 00	00000520	CAN Data	No	No
08/20/2019 13:25:55.306911	1A 00 FF FF FF C2 FF B4	00000320	CAN Data	No	No

Figura 6.4 Lectura de las tramas de datos en el Bus.

En la tabla 10 se muestra el primer análisis realizado, la frecuencia obtenida en esta prueba se compara con la adquirida en la prueba realizada con el Bus Monitor, esto con la intensión de mostrar el comportamiento de los mensajes en la red.

Tabla 10 Análisis de las tramas en el Bus

Identificador	Frecuencia Hz (LabVIEW)	Frecuencia Hz (Bus Monitor)
1A0	50	50
280	23.6	23.6
320	50	50
420	5	5
470	23.6	23.6
520	5	5
5A0	23.6	23.6
621	10	10
727	5	5

En la tabla 10 se observa la frecuencia con la que se envían los mensajes al Bus, si se obtiene el tiempo que existe entre tramas del mismo identificador se puede corroborar la información que se muestra en [4] donde se describe que los nodos transmiten continuamente su mensaje entre cada 7 a 20 ms, cumplido cada periodo un nodo de mayor importancia intenta realizar el envío de una trama de datos al Bus.

En la tabla 10 también se muestra la comparación de la frecuencia obtenida utilizando el programa en LabVIEW y el Bus Monitor, se puede observar que esta es la misma y que el tiempo entre las tramas del mismo identificador es similar en el Bus. Un factor que influye en que las pruebas sean más precisas es el motor de acceso directo a memoria (DMA, por sus siglas en inglés) el cual es impulsado por el dispositivo NI-XNET que permite mover tramas CAN entre la interfaz y el programa del usuario sin interrupciones de CPU, de esta manera se disminuye la latencia del mensaje y se libera el tiempo del procesador principal.

El segundo análisis es realizado sobre la misma lectura de tramas de datos, pero a diferencia de la anterior esta se enfoca en la secuencia de envío de mensajes y en el cumplimiento de la prioridad de las tramas.

A simple vista las tramas de datos no se están enviando de acuerdo a su prioridad, ya existen tramas de datos que se encuentran con mayor frecuencia en el Bus (como se mostró en la tabla 10), se debe de tomar en cuenta que estos mensajes se envían con mayor frecuencia debido a la importancia del mensaje que transfieren, al rápido cambio de los datos y al nivel de seguridad que le proporcionan al conductor, tal es el caso del ABS (Identificador 1A0).

Para este análisis se señalaron los identificadores de mensaje en recuadros de acuerdo a la frecuencia con la que entran al Bus.

En la figura 6.5 se muestran las tramas y 3 diferentes recuadros que demarcan las tramas de acuerdo a su frecuencia en el Bus, la división es la siguiente:

Recuadro verde: Tramas con frecuencia de 50 HZ

Recuadro negro: Tramas con frecuencia de 23.6 Hz

Recuadro naranja: Tramas con frecuencia de 10 y 5 Hz

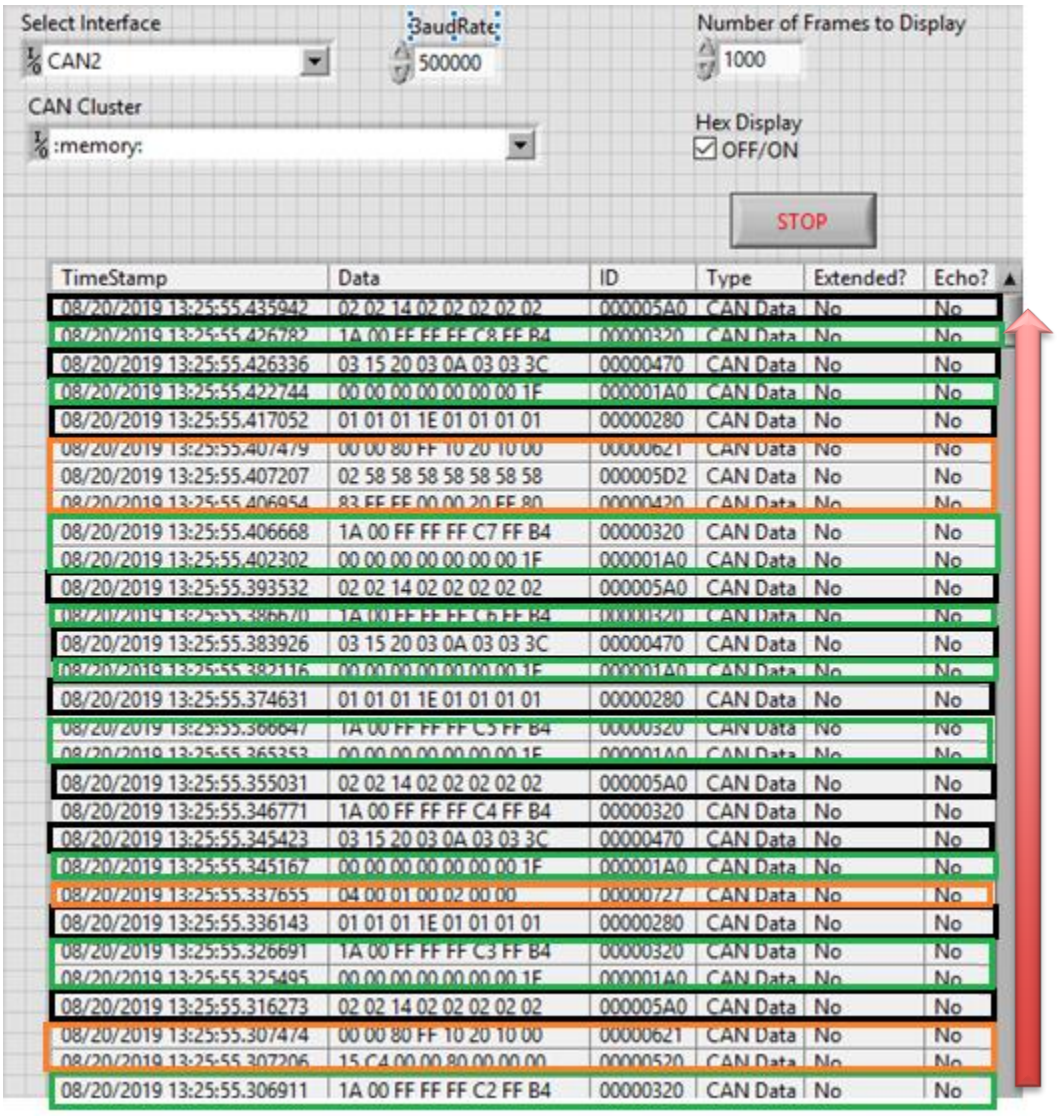


Figura 6.5 Análisis de tramas de datos para verificación de prioridad de mensajes

Después de marcar las tramas de acuerdo a la frecuencia de envío y leerlas en el sentido de la flecha azul se observa que las tramas de datos se están enviando de acuerdo a su prioridad.

Por ejemplo, si se leen las tramas de datos de los identificadores con frecuencia de 23.6 Hz se tiene la siguiente secuencia: 5A0, 280, 470,5A0, 280, 470,5A0, el orden en el cual se envían los mensajes es de acuerdo a la prioridad de cada uno de los identificadores. Para las tramas de datos con frecuencias de 50 Hz y las tramas a 10 y 5 Hz, también se cumple el envío de mensajes de acuerdo a la prioridad. En la figura 6.5 la razón de que la secuencia inicie en 5A0 se debe al fragmento que se tomó como muestra, ya que solo se empleó una fracción (130 milisegundos) de la prueba total.

Prueba 4. Lectura del cambio del valor del mensaje de un nodo

Esta prueba se realizó cambiando el valor de las revoluciones por minuto del panel de instrumentos para verificar que la comunicación y los cambios se realicen rápidamente. En la figura 6.6 se muestran los resultados.

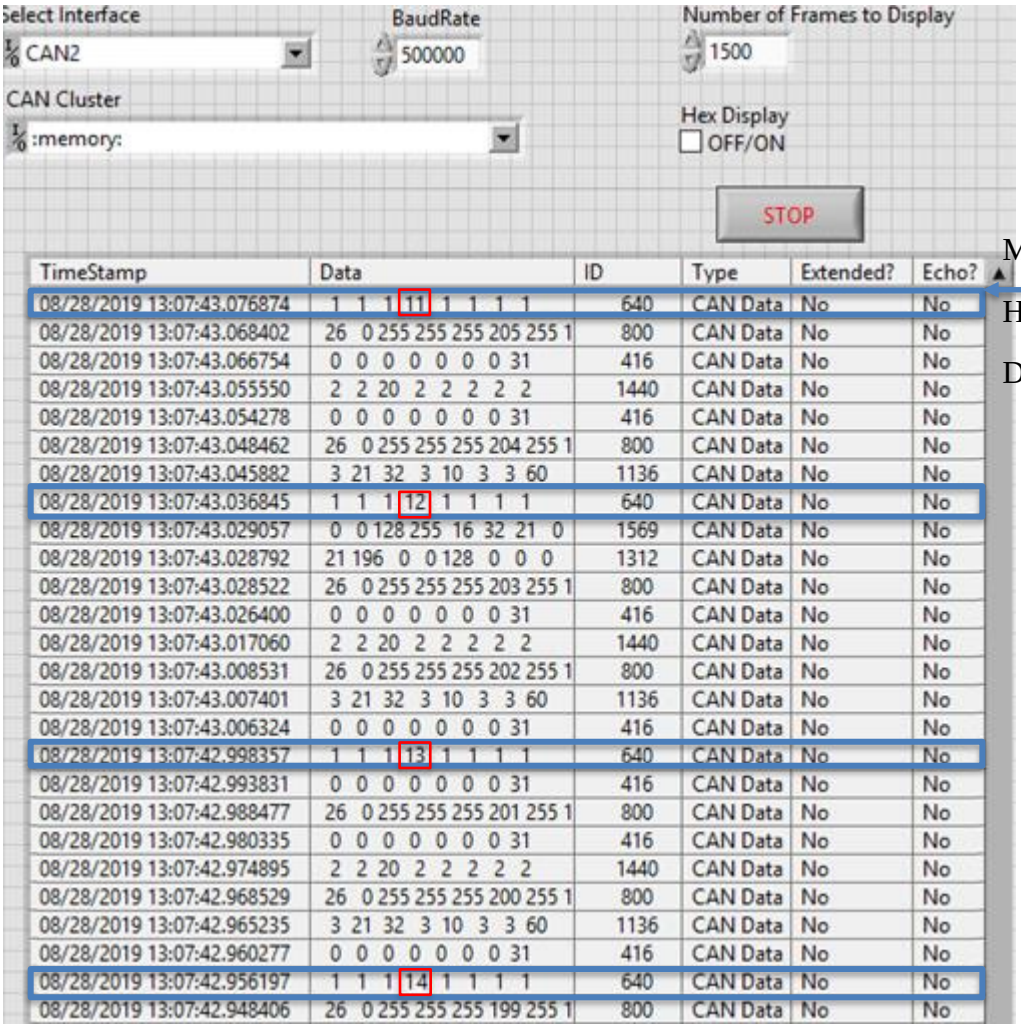


Figura 6.6 Respuesta ante el cambio de valor del mensaje que se transmite.

En la figura 6.6 no se observa el identificador de revoluciones por minuto como 280 en su lugar se muestra el valor 640, esto se debe a que los datos durante esta prueba fueron tomados sin activar la casilla para mostrar los valores en sistema hexadecimal. En la figura 6.6 se muestra el cambio del valor del tercer byte de la trama de datos, también se observa que se sigue conservando la prioridad de los mensajes y que estos conservan la frecuencia con la que intervienen en el Bus.

Prueba 5. Tabla de eventos de mensajes en el Bus. Finalmente se realizó una prueba más empleando el osciloscopio para corroborar los tiempos entre tramas y el envío de mensajes de acuerdo a su prioridad. Para realizar esta prueba se realiza la conexión que se muestra en la figura 6.7.

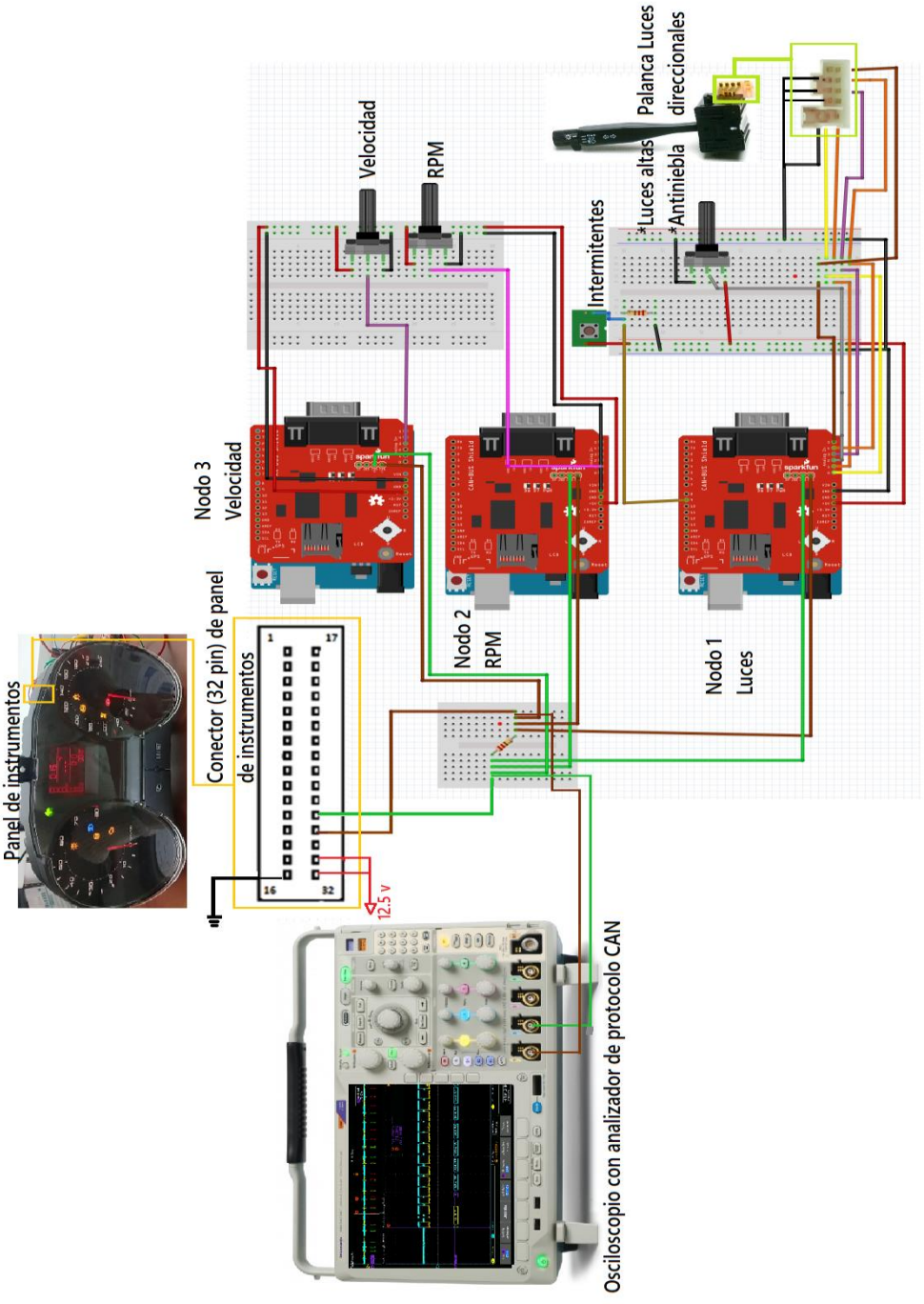


Figura 6.7 Conexión para lectura de tramas de datos empleando osciloscopio con analizador CAN

Para probar el envío de mensajes de acuerdo a la prioridad y su intervención en el Bus según su frecuencia se realizan dos análisis de una tabla de eventos, para capturar las tramas se configuro en el osciloscopio una velocidad de 500 kbps ya que es la velocidad con la que se envían los mensajes al Bus y el trigger se configuro para que se realicen las capturas cada Inicio de trama (Start of frame, por su traducción al inglés). En la figura 6.8 se observa una tabla de eventos que contiene los mensajes que se leen del Bus, usando esta la figura 6.8 se realiza el primer análisis enfocado en la comprobación de la presencia de mensajes en el Bus conforme a su frecuencia. Para obtener la frecuencia de los mensajes se toman los tiempos entre tramas del mismo identificador y se enmarcan en cuadros de color idéntico. En la tabla 11 se comparan las frecuencias encontradas con el osciloscopio y las obtenidas empleando el NI-XNET Bus Monitor (figura 6.2), así se prueba mediante dos diferentes herramientas la frecuencia de las tramas.

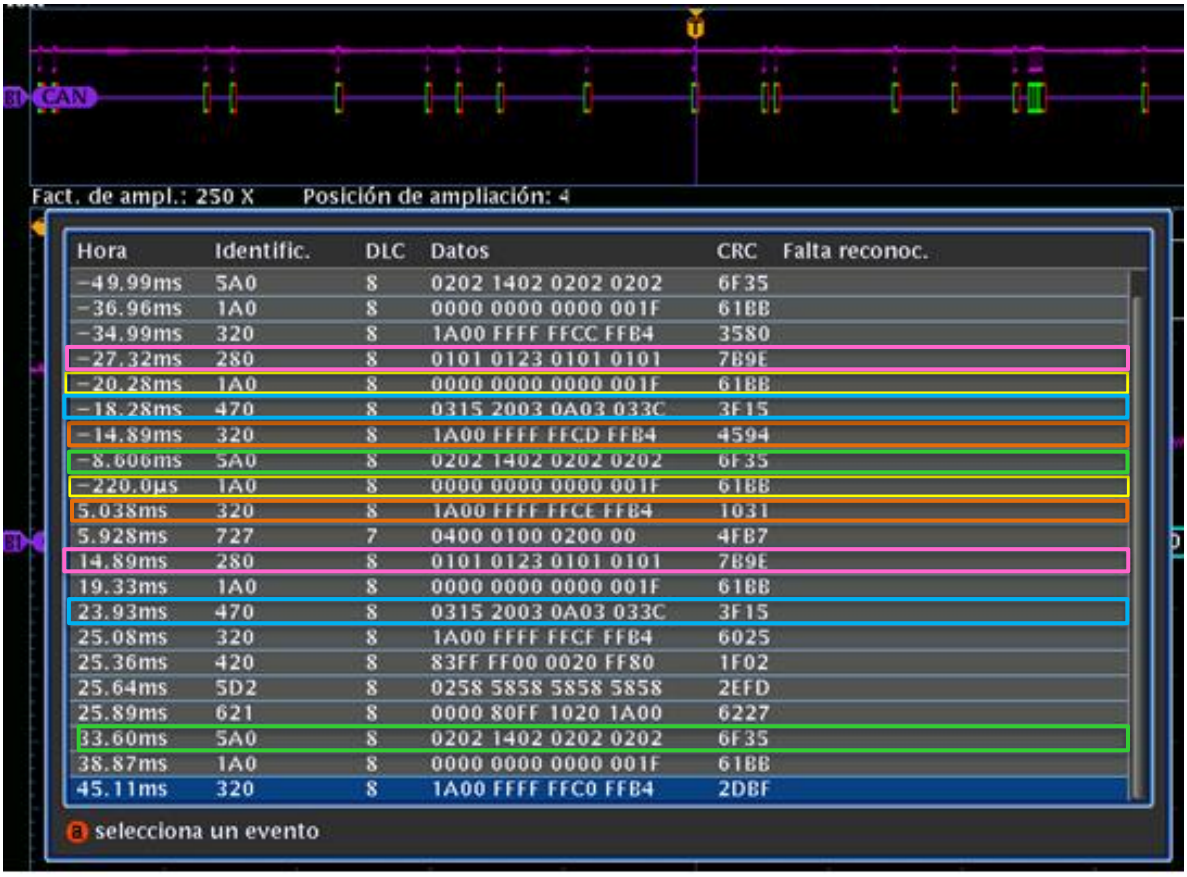


Figura 6.8 Lectura de mensajes en el Bus empleando tabla de eventos.

Tabla 11 Comparación de Frecuencias de intervención de mensajes en el Bus

Identificador	*Frecuencia con osciloscopio (Hz)*	Frecuencia con Bus Monitor (Hz)
1A0	50	50
280	23.68	23.6
320	50	50
470	23.68	23.6
5A0	23.68	23.6

En la tabla anterior que observa que las frecuencias de los mensajes con identificador 1A0 y 320 encontradas con el osciloscopio y con LabVIEW son las misma. Por otro lado, las frecuencias y tiempo entre tramas de los identificadores: 280, 470 y 5A0 tienen una ligera diferencia de 0.08 Hz. Tomando en cuenta que se utilizaron los mismos nodos, una razón por la cual el envío de mensajes difiere en los tiempos es por la distancia de los nodos al Bus. Otra razón por la cual el tiempo varia es por la inserción de bits de relleno en las tramas y es que dependiendo del patrón de bits de un mensaje este puede variar en tamaño entre 0 y 24 bits de relleno, se debe tomar en cuenta que el tiempo más largo empleado para transmitir un mensaje de ocho bytes es 130 microsegundos. También se debe tomar en cuenta que la puesta de un mensaje puede ocurrir con jitter (variabilidad temporal durante el envío de señales digitales), lo cual puede afectar tanto la amplitud como la frecuencia del mensaje[54], [55].Sin embargo estas ligeras variaciones en el tiempo apenas representan el 0.2% de diferencia entre las medidas de los dos métodos y no afecta la comunicación entre los nodos ya que estos siguen enviando mensajes al Bus en el mismo orden que el mostrado en la figura 6.4.

El segundo análisis se encarga de verificar la comunicación entre los nodos del Bus de acuerdo a la prioridad de los mensajes, en la figura 6.9 se muestra una secuencia de mensajes que se encuentran enmarcados en recuadros de color dependiendo de la frecuencia con la que se encuentran en el Bus, como se describe a continuación:

- Recuadros de color verde enmarcan mensajes que se envían con frecuencia de 50 Hz
- Los recuadros rosas enmarcan mensajes que se envían a una frecuencia de 23.6 Hz
- Los que se encuentran señalados con una flecha amarilla se refieren a mensajes con una frecuencia de 10 o 5 Hz.

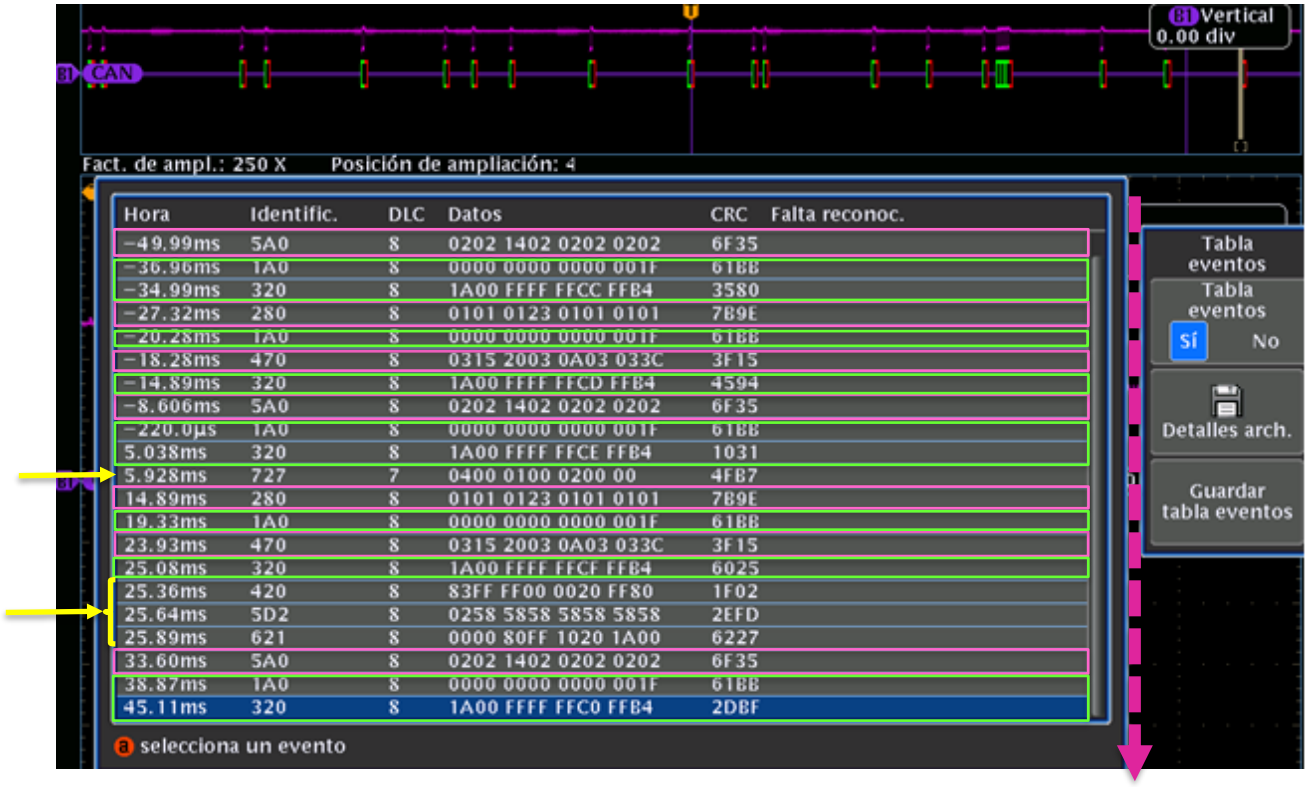


Figura 6.9 Análisis de envió de mensajes de acuerdo a la prioridad.

En la figura 6.9 se puede leer de arriba hacia abajo el orden en el que se enviaron los mensajes, los mensajes que más predominan el Bus son los correspondientes a los identificadores 320 y 1A0 estos tienen una frecuencia de 50 Hz y se encargan de informar del estado del panel de instrumentos y el ABS respectivamente, razón por la cual deben estar con mayor frecuencia en el Bus. Los identificadores 280, 470,5A0 con frecuencia de 23.6 Hz se envían en el Bus en el orden escrito. Finalmente se pueden observar 4 identificadores señalados con dos flechas amarillas los cuales solo aparecen una vez en la figura 6.9 debido a la baja frecuencia con la que se envían al Bus.

Después de la lectura de las ramas de datos mediante dos diferentes métodos (LabVIEW y osciloscopio), de la verificación de los mensajes de acuerdo a su prioridad y frecuencia y de la comparación de los resultados se puede afirmar que la Red realizada puede controlar los parámetros del panel de instrumentos y enviar mensajes al Bus de acuerdo a la prioridad de estos.

6.2 Discusiones.

- Durante la realización de este trabajo se encontraron algunos problemas que tuvieron que resolverse para poder cumplir con el objetivo general que es implementar una Red CAN a un clúster de instrumentos para el control de sus parámetros mediante sus identificadores de mensaje. El primer problema con el que se encontró fue al momento de conectar el clúster de instrumentos a alimentación, ya que no se conocían estos pines. Cuando se adquirió el clúster solo fue entregada una hoja de garantía y no se entregó una hoja de datos, por lo cual no se tenía conocimiento sobre a que pertenecía cada uno de los 32 pines del clúster. La forma en que se resolvió fue buscando trabajos que emplearan un clúster de instrumentos similar y fue así que se encontró que el vehículo Seat Ibiza 6J y VW Polo 6R comparte la mayoría de sus componentes entre ellos el clúster de instrumentos y a partir de un diagrama de conexión del clúster del Polo 6R fue como se localizaron los pines de conexión del clúster empleado en este trabajo, de esta manera también se pudieron localizar los pines que pertenecen a CAN_H y CAN_L.
- Un problema aun mayor fue la localización de los identificadores de mensaje del clúster de instrumentos y es que un identificador con una longitud de 11 bits da la posibilidad de enviar hasta 2032 diferentes IDs, por lo cual la localización de los identificadores de mensaje se volvió una tarea muy tardada. Además, se debe tomar en cuenta que los fabricantes automovilísticos no publican sus identificadores de mensaje y que algunos como VW, BMW, Ford tienen identificadores propios y no están disponibles al público por motivos de seguridad. Aunque existen algunos identificadores de mensaje estandarizados e incluso se interpretan los datos enviados por los módulos en la norma SAE J1939, este es un protocolo para aplicaciones de camiones pesados por lo cual no fue posible emplear estos IDs. La forma en que se resolvió este problema fue buscando identificadores de mensaje de trabajos que se encontraran en la Red y se encontró un trabajo en donde se empleaba el mismo clúster de instrumentos que el empleado en este trabajo, pero solo daba a conocer uno de los identificadores de mensaje (280) con el cual el clúster trabaja. Este identificador represento una pista muy grande ya que basándose en el valor de este identificador y al mensaje que representaba se pudieron descartar algunos identificadores de la base de datos que se tenía y solo se tomaron como referencia y para la realización de pruebas los identificadores de trabajos donde se tuviera el identificador 280 perteneciente a Revoluciones por minuto. De esta forma se localizaron 9 identificadores de mensaje. Pero es importante mencionar que los identificadores de mensaje que se localizaron solo funcionan con el clúster de instrumentos 6J0 920801X, ya que los IDs varían dependiendo de la marca, el vehículo, el modelo y el año[2]. Esta última información se pudo corroborar ya que todos los identificadores de mensaje que se emplearon como referencia para

localizar los IDs con los que trabaja el clúster pertenecían a vehículos de la marca Volkswagen. Aunque se encontraron identificadores de mensaje que coinciden y que no pertenecen al mismo modelo de vehículo, por ejemplo, Seat, Passat y Polo, pero en algunos casos cambiaba el byte en el que se enviaba la información. También se comprobó que los IDs cambian dependiendo del año ya que se probaron IDs de un Vehículo similar, pero de años posteriores y estos no coincidieron.

- Posteriormente a la localización de los IDs se realizaron algunas pruebas en donde se encontró que si un nodo de mayor prioridad es insertado en un Bus donde se encuentran comunicándose nodos de menor prioridad a este, el ultimo nodo que se encontraba enviando su mensaje termina de enviarlo y posteriormente el nodo de mayor prioridad gana varias veces el acceso al Bus, pero después de esto los nodos que se encuentran en el Bus vuelven a comunicarse de acuerdo a la prioridad. También se observó que los nodos transmiten información por un proceso cíclico y para verificar su comunicación se deben realizar pruebas de acuerdo a la frecuencia con la que los mensajes entran al Bus. De igual forma se observó que ante el cambio de uno de los valores de los bytes del nodo este sigue entrando al Bus con la misma frecuencia y los demás nodos envían mensaje de la misma forma.
- En cuanto a las tramas de datos se observó que los nodos cuentan con firmas eléctricas las cuales hacen que cambien su amplitud, además que la firma eléctrica es única para cada nodo y depende de la distancia de este al Bus y de su posición. También se verifico el uso de los métodos para detección de error que se emplean a nivel de bit y a nivel de mensaje.
- Los dos equipos empleados para obtener las tramas de datos del CAN Bus (Sistema PXI y Osciloscopio) fueron de gran utilidad, ya que empleando el sistema PXI se pudieron obtener las frecuencias de los mensajes, leer los datos, observar la introducción de los mensajes e incluso enviar mensajes al Bus. El osciloscopio dio la posibilidad de observar si existía algún error o ruido en la señal, verificar la implementación de métodos de detección de errores en las tramas de datos, observar las firmas eléctricas, obtener las frecuencias de los mensajes. Ambos métodos coincidieron en los resultados, siendo una forma de corroborar el correcto envío de mensajes en el Bus.
- Respecto a la frecuencia con la cual se introducen los mensajes al Bus se debe mencionar que no se determinó en la programación, sino que el clúster de instrumentos es quien se encarga de determinar el número de veces que estos mensajes entran al Bus.

6.3 Conclusiones

El protocolo de comunicación CAN se ha implementado en la mayoría de los vehículos actuales debido al aumento de sistemas electrónicos y funciones nuevas, por lo que se realizó este trabajo para ser una herramienta de aprendizaje interactivo que apoye a los alumnos de la carrera de Ingenieria Sistemas Automotrices al aprendizaje de dicho protocolo de comunicación. Para la realización de este trabajo el estudio profundo del protocolo de comunicación CAN fue crucial para el análisis de las respuestas de las pruebas realizadas.

Se estableció una metodología para la identificación de algunos de los testigos del clúster de instrumentos. El medio para el envío de mensajes que se empleó en esta metodología fue el NI-XNET Bus Monitor, el cual facilito el envío de identificadores y localización de los bytes en los cuales se envía la información al Bus. De esta forma se pudieron localizar 9 de los identificadores de mensaje del clúster de instrumentos y los bytes en los que se envía la información del mensaje dejando tablas de los identificadores y valores que deben tomar para el encendido de algún testigo. Durante la obtención de los identificadores de mensajes se encontró que el mensaje de velocidad necesita que antes se envié en el Bus el mensaje del estado del ABS para que la aguja del velocímetro se mueva, esto puede ser debido a que el velocímetro indica la velocidad del vehículo calculando las señales de los sensores de velocidad de la rueda a través del nodo de ABS [56].El sensor de velocidad del vehículo es sólo uno de los muchos sensores en el vehículo moderno y puede proporcionar información sobre la velocidad del vehículo a muchos sistemas, estos pueden incluir el módulo de control del motor (ECM), el módulo de control de crucero (CCM), el módulo del sistema de frenos antibloqueo (ABS) y el módulo del grupo de instrumentos (ICM), por nombrar alguno y esto lo hace a través de líneas de datos del nodo de ABS[57]. Sin embargo, el mensaje de ABS y Velocidad no pueden estar en el mismo nodo porque ocasionan que el clúster emita una alarma permanentemente, por esta razón se separaron estos mensajes. También se identificó que el Identificador 5A0 referente al parámetro de velocidad es un poco inestable, esta situación es similar a la que se encontró en [15]. Para los identificadores de control crucero y EPC también se encontró que requieren del mensaje de las revoluciones por minuto para el encendido del testigo, esto debido a que estos nodos comparten sensor, una de las características del protocolo de comunicación CAN. Se debe mencionar que los identificadores de mensaje solo funcionan con el clúster de instrumentos empleado en este trabajo, ya que los IDs cambian de acuerdo a la marca, el modelo y año del vehículo.

La localización de los parámetros represento una dificultad y un retardo para el avance del trabajo ya que los fabricantes de automóviles protegen el acceso a sus identificadores y la información que se envía en ellos, esto como medio de seguridad de los datos de la Red

CAN, ya que de no ser así cualquier persona con acceso físico al Bus de datos podría enviar mensajes falsos a nodos encargados de operaciones criticas afectando la seguridad del conductor. Para evitar estos problemas los fabricantes no publican sus identificadores reales utilizados para la comunicación del vehículo, tales como VW,GM, BMW, Ford, Honda, etc. ya que los identificadores son propiedad y esta información no está a disposición del público por los fabricantes de automóviles[58].

Se implementó un nodo CAN y se desarrolló un código en la IDE de Arduino para el envío y recepción de mensajes, gracias a esto fue posible efectuar la comunicación de los nodos en el Bus los cuales pudieron realizar envío y lectura de mensajes.

Se desarrolló una Red CAN integrada por tres nodos CAN y se integró a cada uno de los nodos un sensor a excepción del nodo de luces al cual se le implementaron dos.

Se evaluó el envío de mensajes de acuerdo a su prioridad y se realizaron una serie de pruebas para la comprobación de la comunicación. Se realizó una exitosa comunicación entre el clúster de instrumentos y la red de 3 nodos, debido a la rápida respuesta del clúster y los nodos ante la lectura de las señales de los sensores con los cuales se logró controlar los parámetros de un clúster de instrumentos automotriz. Durante las pruebas realizadas sobre la comunicación de los nodos y la intervención de estos en el Bus de acuerdo a la prioridad se encontró que algunos mensajes se hallaban con mayor frecuencia que otros debido a que en la comunicación en tiempo real en un vehículo es necesario informar con mayor frecuencia los mensajes que tienen mayor prioridad o variaciones significativas tal es el caso del ABS. Se debe mencionar que la frecuencia con la que los mensajes entran al Bus es determinada por el Clúster de Instrumentos. Se observó que existen nodos que son más importantes que otros, pero aun cuando algunos mensajes se envían con mayor frecuencia se sigue conservando la introducción al Bus de acuerdo la prioridad de los mensajes. Es por esta razón que los automóviles tienden a tener más de un Bus (confort, tracción, etc.), para enviar mensajes a diferentes velocidades dependiendo de la necesidad de información del mensaje en el Bus. También se detectó que durante la comunicación de los nodos pueden existir variaciones en el tiempo de envío de los mensajes, estos retardos pueden ser causados por la inserción de bits de relleno a la trama de datos, la cantidad máxima de bits de relleno sería 24 para la transmisión de una trama de datos en formato estándar cuando el campo de datos incluye ocho bytes, sin alterar el mensaje ya que estos bits son eliminados por el receptor. También pueden existir variaciones en la frecuencia debido a una variación en el tiempo (jitter).

Después de la evaluación de la comunicación de los nodos en la red de acuerdo a su prioridad y la obtención de los identificadores de mensaje con los cuales trabaja el clúster de instrumentos empleado, se puede decir que se cumplió con los objetivos planteados para este trabajo de tesis. Además de que la Red CAN que se implementó para este trabajo tiene la característica de ser flexible y permitir incluir nuevos nodos en cualquier momento, aun

cuando la red se encuentre funcionando sin tener que hacer cambios en los nodos que ya se encuentran en la Red. La ventaja de este protocolo de comunicación es que cada uno de los mensajes de los nodos representa una señal que informa del estado de un sensor, razón por la cual los vehículos incluyen el protocolo CAN.

6.4 Trabajos futuros

Esta tesis representa una base para la realización de más trabajos enfocados en el protocolo de comunicación CAN, el cual es uno de los más empleados en la industria automotriz. Con esta herramienta se pueden realizar más trabajos de investigación sobre este protocolo además de aumentar la cantidad de elementos de la misma. Al contar con algunos de los identificadores de mensaje del panel de instrumentos y el mensaje que se envía en cada uno de los bytes del campo de datos, es posible implementar más nodos para acrecentar la red, además de agregar más sensores y actuadores. Implementar más dispositivos para simular un entorno vehicular, observar y analizar el comportamiento del contenido del Bus. Para acrecentar aún más la Red se puede tener como trabajo futuro la implementación de un Bus de menor velocidad para enviar información de identificadores correspondientes al CAN confort. Este trabajo también puede emplearse para la implementación de sistemas de simulación de manejo empleando el clúster de instrumentos que se ha podido controlar en este trabajo de tesis.

Bibliografía

[1] J. Hernandez, “Vulnerability assessment of Electronic Control Unit (ECU) of automotive system through OBD-II port and CAN bus,” INSTITUTO POLITÉCNICO NACIONAL, 2017.

[2] O. Sánchez Vela, Luis Gerardo, Molano Clemente, Martín Jonathan, Fabela Gallegos, Manuel de Jesús, Martínez Madrid, Miguel, Hernández Jiménez, José Ricardo, Vázquez Vega, David, Flores Centeno, “Revisión documental del protocolo CAN como herramienta de comunicación y aplicación en vehículos,” no. 474, 2016.

[3] Robert Bosch GmbH (Ed.), *Bosch Automotive Electrics and Automotive Electronics : Systems and Components, Networking and Hybrid Drive*. 2014.

[4] Seat, “CAN-Bus Información para el instructor,” vol. 1, p. 30.

[5] C. Smith, *The Car Hacker’s Handbook (2016).pdf*. .

[6] C. Miller and C. Valasek, “Adventures in Automotive Networks and Control Units,” *IOActive Tech. White Pap.*, pp. 1–99, 2013.

[7] Yingping Huang, A. Mouzakitis, R. McMurran, G. Dhadyalla, and R. P. Jones, “Design validation testing of vehicle instrument cluster using machine vision and hardware-in-the-loop,” in *2008 IEEE International Conference on Vehicular Electronics and Safety*, 2008, pp. 265–270.

[8] S. Nimara, D. B. Popa, and R. Bogdan, “Automotive instrument cluster screen content validation,” in *2017 25th Telecommunication Forum (TELFOR)*, 2017, pp. 1–4.

[9] Continental Automotive GmbH, “Modular instrument clusters Continental Instrument Clusters – - All necessary infor - mation in premium quality .,” Alemania, 2016.

[10] A. Patterson, “Security and Safety in Embedded Applications Use Case : Instrument Cluster,” 2016.

[11] Jiejie Dai and Hui Song, “Design and realization of CAN bus vehicle instrument cluster based on μ C/OS-II,” in *2011 Second International Conference on Mechanic Automation and Control Engineering*, 2011, pp. 1159–1161.

[12] A. García, “Diseño de una red CAN bus con Arduino,” Universidad Publica de Navarra, 2015.

[13] Volkswagen, “Data Exchange On The CAN Bus I,” Wolfsburg, 2003.

[14] S. Nürnberger and C. Rossow, "– vatiCAN – Vetted, Authenticated CAN Bus," pp. 106–124, 2016.

[15] L. Bataille, "Volkswagen CAN BUS Gaming," 2015. [Online]. Available: https://hackaday.io/project/6288-volkswagen-can-bus-gaming. [Accessed: 05-Apr-2019].

[16] R. Steinhilper, S. Freiberger, M. Albrecht, J. Käufl, E. Binder, and C. Brückner, "Reverse Engineering Technologies for Remanufacturing of Automotive Systems Communicating via CAN Bus," *Glocalized Solut. Sustain. Manuf. - Proc. 18th CIRP Int. Conf. Life Cycle Eng.*, pp. 90–95, 2011.

[17] A. Soriano *et al.*, "BUS CAN qué es CAN y cómo funciona el modulo del microcontrolador dsPIC30F4013," Universidad Politecnica de Valencia, Valencia, 2005.

[18] M. Grusin, "Serial Peripheral Interface (SPI)," 2003. [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all. [Accessed: 26-Mar-2019].

[19] Microchip, "MCP2551," 2010.

[20] J. Beltrán Zambrano, "Desarrollo de un simulador electrónico de una ECU y su diagnóstico sobre CAN y OBD-II," 2015.

[21] Joan Antoni Ros Marín and Óscar Barrera Doblado, *Sistemas eléctricos y de seguridad y confortabilidad*. Paraninfo, 2010.

[22] H. Flores, "Automatización de pruebas de pérdida de comunicación can/lin, reseteo y desactivación/activación en sistemas automotrices embebido," ciateq, Jalisco, 2018.

[23] J. Pacheco, "Monitoreo de hábitos de manejo por medio de una red CAN automotriz," UDLAP, 2011.

[24] J. Andres, B. Barrera, U. Tecnológica, D. E. Pereira, F. De, and T. Pereira, "Estudio del Protocolo de Comunicación serial BUS CAN y la aplicación en la industria de vehículos de transporte BUSSCAR DE COLOMBIA," Universidad Tecnológica de Pereira, Pereira, 2017.

[25] National Instruments, "Normas de la Capa Física de CAN: Alta-Velocidad vs. Baja-Velocidad/Tolerante a Fallas CAN - National Instruments," 2002. [Online]. Available: http://digital.ni.com/public.nsf/allkb/9E97DC2B8C3205448625797B00571E04. [Accessed: 13-Mar-2019].

[26] N. Antonio, "Comunicaciones industriales y en Tiempo Real," Madrid, 2011.

[27] W. Voss, *A Comprehensible Guide to J1939*, 1st ed. Greenfield: Copperhill

Technologies Corporation, 2008.

[28] H. Kaschel and E. Pinto, “Analisis protocolar del bus de campo can,” Santiago, 2010.

[29] D. Encinas and P. Meilan, “Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores,” in *XV Congreso Argentino de Ciencias de la Computación*, 2009, p. 9.

[30] ISO, “ISO 11898-1,” Ginebra, 2003.

[31] ISO, “ISO 11898-2,” Ginebra, 2016.

[32] ISO, “ISO 11898-3,” Ginebra, 2006.

[33] ISO, “ISO 11898-4,” Ginebra, 2004.

[34] ISO, “IS0 11898-5,” Ginebra, 2007.

[35] L. Molin and G. Mucchiano, “Fully automated test environment for the Scania instrument cluster,” p. 59, 2011.

[36] S. T. Bodybuilder, “Bodywork information in the instrument cluster , ICL Functions Bodywork information in the instrument cluster , ICL,” vol. 1, no. 1, pp. 1–37, 2016.

[37] D. M. Agarwal and A. P. Chowdhury, “The basics of automotive cluster device architectures and applications , part I,” 2014.

[38] M. Buczaj and A. Sumorek, “SIMULATION OF MOTOR VEHICLES INSTRUMENT PANELS IN LABVIEW ENVIRONMENT,” 2007.

[39] Mouser Electronics, “Automotive Application - Instrument Cluster | Mouser.” [Online]. Available: https://www.mouser.mx/applications/automotive-instrument/. [Accessed: 04-Mar-2019].

[40] National Instruments, “Qué es PXI,” 2011. [Online]. Available: http://www.ni.com/tutorial/4811/es/.

[41] National Instruments, “PXI-1042 - National Instruments.” [Online]. Available: http://www.ni.com/es-mx/support/model.pxi-1042.html. [Accessed: 02-Apr-2019].

[42] National Instruments, “PXI-8101 - National Instruments.” [Online]. Available: http://www.ni.com/es-mx/support/model.pxi-8101.html. [Accessed: 02-Apr-2019].

[43] National Instruments, “PXI-8513 - National Instruments.” [Online]. Available: http://www.ni.com/es-mx/support/model.pxi-8513.html. [Accessed: 02-Apr-2019].

[44] National Instruments, “Módulo de Interfaz CAN,” *2011*. [Online]. Available: https://www.ni.com/es-mx/shop/select/c-series-can-interface-module.

[45] Arduino, "¿Que es Arduino?" [Online]. Available: http://arduino.cl/que-es-arduino/. [Accessed: 22-Mar-2019].

[46] Sparkfun, "CAN-BUS Shield - DEV-13262 - SparkFun Electronics." [Online]. Available: https://www.sparkfun.com/products/13262. [Accessed: 22-Mar-2019].

[47] Microchip, "MCP2515 NOTES," 2003.

[48] P. Richards, "A CAN Physical Layer Discussion," *Technology*, pp. 1–12, 2002.

[49] National Instruments, *NI-XNET Hardware and Software Manual*, no. July. Austin: National Instruments Corporate Headquarters, 2016.

[50] S. Nurnberger, "Controlling an Instrument Cluster with an Arduino - YouTube," *Automotive Security*, 2015. [Online]. Available: https://www.youtube.com/watch?v=4SgW64d_fbE. [Accessed: 05-Apr-2019].

[51] C. Quigley, D. Charles, and R. Mclaughlin, "The Use of CAN Bus Message Electrical Signatures for Automotive Reverse Engineering," no. Can Id, p. 10, 2018.

[52] National Instruments, "Modo de flujo de entrada de cuadro," 2018. [Online]. Available: http://zone.ni.com/reference/en-XX/help/372841U-01/nixnet/modeframeinputstream/.

[53] National Instruments, "XNET Read (Frame CAN).vi," 2018. [Online]. Available: http://zone.ni.com/reference/en-XX/help/372841U-01/nixnet/xnetreadframecanvi/.

[54] K. W. Tindell, H. Hansson, and A. J. Wellings, "Analysing real-time communications: Controller area network (CAN)," *Proc. - Real-Time Syst. Symp.*, no. January 1995, pp. 259–263, 1994.

[55] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, no. 3, pp. 239–272, 2007.

[56] SsangYong Motor, "Manual de usuario Ssangyong," 2006.

[57] Hella, "SENSORES ABS : PRINCIPIOS BÁSICOS," 2009. [Online]. Available: https://www.hella.com/techworld/uk/Technical/Sensors-and-actuators/Check-change-ABS-sensor-4074/.

[58] J. Staggs, "How to Hack Your Mini Cooper: Reverse Engineering CAN Messages on Passenger Automobiles," *Defcon21*, 2013.

Apéndice A. Tablas de Identificadores de Mensaje localizados en este trabajo.

Tabla A.1 Velocidad del vehículo (ID: 5A0)

ID 5A0 VELOCIDAD DEL VEHÍCULO		
BYTE 0	SIN FUNCIÓN	
BYTE 1	SIN FUNCIÓN	
BYTE 2	EL VALOR QUE SE QUIERE MOSTRAR ENTRE TRES	EJEMPLO: SE DESEA MOSTRAR 120, POR LO TANTO, SE ESCRIBE 40
BYTE 3	ABS	1,11,21
	ESP	2,12,22
	ABS + ESP	3,13,23
	FRENO DE MANO/NIVEL DE LÍQUIDO DE FRENO INSUFICIENTE	4,14
	ABS + FRENO DE MANO	5,15
	ESPY FRENO DE MANO	6,16
	ABS, FRENO DE MANO Y ESP	7,17
	PRESIÓN EN LOS NEUMÁTICOS	8,18
	ABS + PRESIÓN EN LOS NEUMÁTICOS	9,19
	NADA	10,20
BYTE 4	SIN FUNCIÓN	
BYTE 5	SIN FUNCIÓN	
BYTE 6	SIN FUNCIÓN	
BYTE 7	SIN FUNCIÓN	

Tabla A.2 Bolsa de aire (ID: 050)

ID 050 BOLSA DE AIRE		
BYTE 0	TESTIGO BOLSA DE AIRE APAGADA	0,4,2
	TESTIGO DE BOLSA DE AIRE ENCENDIDO	1,3,5
BYTE 1	SIN FUNCIONAMIENTO	
BYTE 2	TESTIGO DE CINTURÓN DE SEGURIDAD ENCENDIDO	4-7, c-f,14-17
	TESTIGO DE CINTURÓN DE SEGURIDAD APAGADO	0-3,8-b,18-
BYTE 3	SIN FUNCIONAMIENTO	

Apéndice A. Tabla de Identificadores de Mensaje localizados en este trabajo

Tabla A.3 ABS (ID: 1A0)

ID 1A0 ABS		
BYTE 0	SIN FUNCIÓN	
BYTE 1	SIN FUNCIÓN	
BYTE 2	SIN FUNCIÓN	
BYTE 3	SIN FUNCIÓN	
BYTE 4	SIN FUNCIÓN	
BYTE 5	SIN FUNCIÓN	
BYTE 6	SIN FUNCIÓN	
BYTE 7	ABS + FRENO DE MANO, AL ENCENDER Y APAGAR	AA-AF, CA-CF, EA-EF,0-9,20-29,40-49,60-69,80-89
	ABS + FRENO DE MANO AL ENCENDER Y AL PAGAR ABS + ESP FRENO DE MANO	BA-BF, DA-DF, FA-FF,10-19,30-39,50-59,70-79,90-99

Tabla A.4 Dirección asistida (ID: 3D0)

ID 3D0 DIRECCION ASISTIDA		
BYTE 0	SIN FUNCION	
BYTE 1	TESTIGO ENCENDIDO	1,3,5,7,9,11,
	TESTIGO APAGADO	0,2,4,6,8,10
BYTE 2	SIN FUNCION	
BYTE 3	SIN FUNCION	
BYTE 4	SIN FUNCION	
BYTE 5	SIN FUNCION	
BYTE 6	SIN FUNCION	
BYTE 7	SIN FUNCION	

Apéndice A. Tabla de Identificadores de Mensaje localizados en este trabajo

Tabla A.5 Luces (ID: 470)

ID 470 LUCES		
BYTE 0	LUZ IZQUIERDA	1,5,9,11, D
	LUZ DERECHA	2,6,12, A, E
	AMBAS LUCES (intermitentes)	3,7,13, B, F
	NADA	4,8,10,14, C,20,30,40,50
	BATERIA Y LUZ DERECHA	AA, AE, BA
	BATERIA Y LUZ IZQUIERDA	AD, B1, BD, DD
	BATERÍA Y AMBAS	AB, AF, BB
	BATERÍA	AC, BC, CC
BYTE 1	CAJUELA	20,30,60,70, A0, B0, F0
	PUERTA ABIERTA	1-F,11-1F,4A,51-5F,81-8F,91-9F, C1-CF, D1,21
	CAJUELA Y PUERTA ABIERTA	22-2F,31-39,61-6F,7-7FA1-AF, B1-BF, F1-FF
	NADA	10,50,80,90, C0, D0
BYTE 2	LUZ DE FONDO APAGADA	0
	LUZ DE FONDO ENCENDIDA	1-64,64 es el 100% de brillo.
BYTE 3	SIN FUNCIONAMIENTO	
BYTE 4	SONIDO	2,6, E, AF, CA
	PROBLEMA EN UNA BOMBILLA	10,11,13,14,15,18,19,30,32,33,34,35,37,38, B1, B4, FA
	SONIDO Y PROBLEMA EN BOMBILLA	12,16,17,36, BA, B6
	NADA	20-29, A, B, C, D, F, A1, A2, A3, A5
BYTE 5	SIN FUNCIONAMIENTO	
BYTE 6	SIN FUNCIONAMIENTO	
BYTE 7	LUZ ANTINIEBLA TRASERA	20-38, AA-BF,2A-2F
	LUZ DE CARRETERA ENCENDIDA	40-59, CA-DF,64
	LUZ ANTINIEBLA TRASERA Y LUZ DE CARRETERA ENCENDIDA	60-79, EA-FF

Tabla A.6 Revoluciones por minuto (ID: 280)

ID 280 RPM	
BYTE 0	SIN FUNCIÓN
BYTE 1	SIN FUNCIÓN
BYTE 2	SIN FUNCIÓN
BYTE 3	VALOR DE RPM
BYTE 4	SIN FUNCIÓN
BYTE 5	SIN FUNCIÓN
BYTE 6	SIN FUNCIÓN

BYTE 7	SIN FUNCIÓN

Tabla A.7 Control crucero (ID: 288)

ID 288 CONTROL CRUCERO		
BYTE 0	SIN FUNCIÓN	
BYTE 1	SIN FUNCIÓN	
BYTE 2	TESTIGO ENCENDIDO	40-9F
	TESTIGO APAGADO	0-3F
BYTE 3	SIN FUNCIÓN	
BYTE 4	SIN FUNCIÓN	
BYTE 5	SIN FUNCIÓN	
BYTE 6	SIN FUNCIÓN	
BYTE 7	SIN FUNCIÓN	

Tabla A.8 Pisar el pedal de freno (ID: 540)

ID 540 Pisar el pedal de freno		
BYTE 0	SIN FUNCIÓN	
BYTE 1	SIN FUNCIÓN	
BYTE 2	SIN FUNCIÓN	
BYTE 3	SIN FUNCIÓN	
BYTE 4	SIN FUNCIÓN	
BYTE 5	SIN FUNCIÓN	
BYTE 6	SIN FUNCION	1,2,3,23
	STOP GETRIEBE ZU HEISS/detener el engranaje demasiado caliente	4,5,6,7
	PISAR EL PEDAL DE FRENO	10,11,12,13,18
	PISAR EL PEDAL DE FRENO + STOP GETRIEBE ZU HEISS	14,15,16,17
BYTE 7	SIN FUNCIÓN	

Apéndice B. Identificadores de Mensaje de Referencia

Tabla B.1 Identificadores de Mensaje de la referencia [14]

Function	CAN ID	Every	Frequ.	Length	Bytes/s	Exemplary Payload	vatiCAN
Airbag	050	20 ms	50 Hz	4 bytes	200	E0:F0:00:FF	⊘
Steering	0C2	20 ms	50 Hz	8 bytes	400	F0:00:00:00:80:40:00:CF	⊘
Electronic Power Steering (EPS)	0D0	50 ms	20 Hz	6 bytes	120	D7:C0:61:08:5E:20	⊘
ABS1	1A0	10 ms	100 Hz	8 bytes	800	00:00:00:00:FE:FE:00:00	⊘
ABS2	4A0	10 ms	100 Hz	8 bytes	800	00:00:00:00:FE:FE:00:00	⊘
Brakes	1AC	20 ms	50 Hz	8 bytes	400	00:80:7F:7F:69:A1:00:C2	☑
ETC / Engine RPM	280	20 ms	50 Hz	8 bytes	400	49:00:20:20:00:FA:36:00	☑
Engine Status	35B	100 ms	10 Hz	8 bytes	80	0F:00:00:B8:28:19:02:96	⊘
Coolant	288	20 ms	50 Hz	8 bytes	400	43:78:00:04:00:56:00:00	⊘
Instrument Cluster	320	20 ms	50 Hz	8 bytes	400	02:00:00:ff:ff:cd:ff:96	⊘
Vehicle Speed	5A0	100 ms	10 Hz	8 bytes	80	00:00:00:5B:B6	☑
Instrument Cluster	621	100 ms	10 Hz	7 bytes	70	04:00:01:00:02:00:00	⊘
Instrument Cluster	727	100 ms	10 Hz	8 bytes	80	02:00:00:ff:ff:c6:ff:9e	⊘

Tabla B.2 Identificadores de Mensaje referencia [16]

Time	Diff time	ID	DLC	Data
25.446944	0.020036	50	4	00 02 10 12
25.447194	0.007198	1A0	8	00 41 00 00 FE FE 00 03
25.442988	0.010980	280	8	48 26 40 39 27 2F 2C 00
25.443222	0.010844	288	8	D0 9C 10 00 00 49 D6 14
25.425994	0.023989	320	8	05 02 15 00 00 00 00 00
25.443468	0.010688	380	8	00 66 2F 00 00 00 00 1B
25.433546	0.019980	3D0	2	05 00
25.301656	0.200174	420	8	81 00 00 6E 9C 00 FF 00
25.421938	0.048553	470	5	00 11 00 FF 00
25.443710	0.020430	480	8	EC 00 F3 34 00 00 00 2B
25.443940	0.010680	488	8	6D 27 26 7A A6 00 00 B0
25.440248	0.013756	4A0	8	00 00 00 00 00 00 00 00
25.301278	0.200038	520	8	40 42 02 C0 0B FB EE 01
25.373787	0.100495	570	4	87 A0 B6 00
25.044520	1.000000	580	8	8E 00 00 1E 0B 2F 38 8C
25.444182	0.020430	588	8	08 5D 7A 82 00 00 00 00
25.440490	0.013756	5A0	8	7F 00 00 77 7E 00 10 2B
25.372967	0.100273	5D0	6	80 02 40 80 00 41
25.374279	0.101339	5D8	8	55 0C 00 00 00 00 00 00
25.314309	0.199940	5DE	5	00 00 00 05 00

Tabla B.3 Identificadores de Mensaje referencia [13]

Identifier	Binär	Hex
Motor_1	010_1000_0000	280
Freno_1	010_1010_0000	1A0
Cuadro instr._1	011_0010_0000	320
Ángulo viraje_1	**000_1100_0000**	**0C2**
Cambio_1	100_0100_0000	440

Identificadores posibles en el CAN tracción *238_027b*

Apéndice C. Pines del Clúster de Instrumentos

PIN	FUNCIÓN
1	Medidor de combustible del emisor (completo)
2	Medidor de combustible del emisor (vacío)
3	Libre
4	Libre
5	Libre
6	Libre
7	bobina lectora de inmovilizador -D2- (1)
8	bobina lectora de inmovilizador -D2- (2)
9	velocímetro, salida
10	Libre
11	Nivel de aceite y sensor de temperatura del aceite -G266-
12	Libre
13	luz antiniebla trasera de advertencia
14	luz de advertencia de haz principal
15	Sensor de desgaste de las pastillas de freno delantero izquierdo -G34- / Sensor de desgaste de las pastillas de freno trasero derecho -G37
16	Terminal 31
17	limpiaparabrisas nivel de fluido emisor -G33-
18	Indicador de escasez de refrigerante emisor -G32-
19	Sensor de temperatura ambiente -G17-
20	Terminal 31, Sensor tierra
21	Botón de llamada de pantalla multifunción -E86- (MFI +)
22	Botón de llamada de pantalla multifunción -E86- (MFI -)
23	interruptor de memoria de pantalla multifunción -E109- (MFI-R)
24	Libre
25	interruptor de advertencia de freno de mano -F9-
26	Contacto de aviso de nivel de líquido de frenos -F34-
27	interruptor de presión de aceite -F1-
28	interfaz de diagnóstico para Bus de datos CAN, alto (CAN_H)
29	interfaz de diagnóstico para Bus de datos CAN, bajo (CAN_L)
30	Libre
31	alimentación de tensión, terminal 15
32	alimentación de tensión, terminal 30

Apéndice D. Palanca de Luces direccionales y Conector DB9

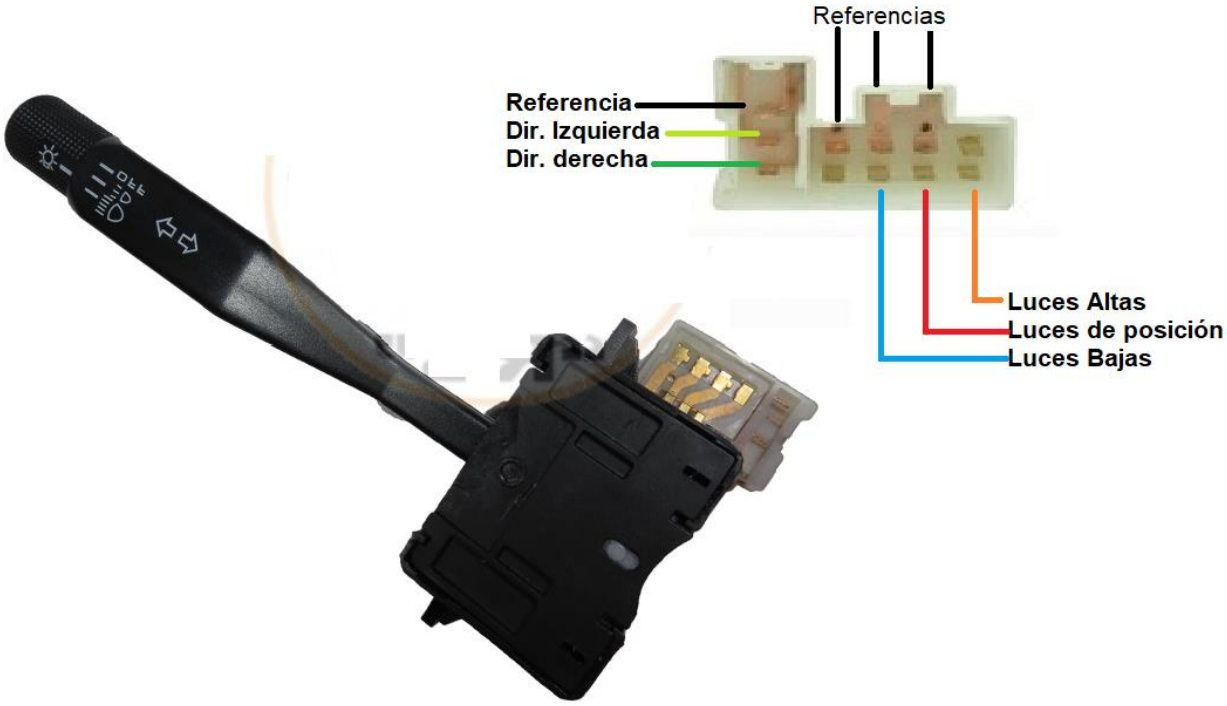
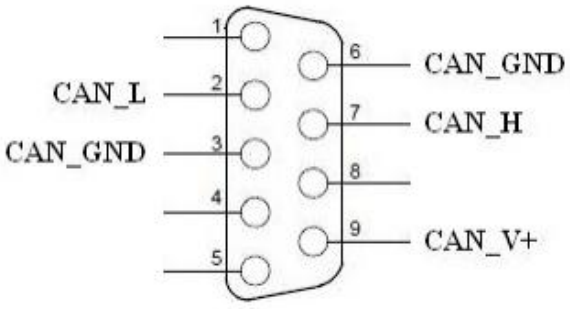


Figura D.1 Palanca de Luces direccionales utilizada en este trabajo

Pines del Conector DB9



Pin	Signals	Description
1		
2	CAN_L	Low dominant level
3	CAN_GND	Ground
4		
5		
6	CAN_GND	Ground(optional)
7	CAN_H	High dominant level
8		
9	CAN_V+	Power (optional)

Figura D.2 Conector DB9

Apéndice E. Colores de Cables CAN

Algunos de los vehículos de la marca Volkswagen disponen de 3 redes CAN: tracción, confort (también llamado de conveniencia) e infotainment. Esto se debe a las diferentes demandas respecto a la frecuencia de repetición de las señales. Los cables se encuentran en el arnés de cableado y tiene un color definido de acuerdo al tipo de Red CAN [4], [13].

Por otro lado, los vehículos de la Seat cuentan con 2 redes CAN: tracción y confort. Sin embargo, comparten algunas características en los códigos de colores CAN, esto puede deberse a que Seat es una marca subsidiaria de Volkswagen. Dentro del consorcio se han definido las características de los cables de comunicación CAN del vehículo Seat de acuerdo al tipo de Red CAN, en la tabla F.1 se puede ver el código de colores de cables [4].

Tabla F.1 Código de colores CAN en vehículo Seat [4].

Área de control	Sección en mm^2	Colores	Paso del trenzado
Confort	0.35-0.5	H= Amarillo L= Negro	20 mm.
Motopropulsor-tren de rodaje	0.35-0.5	H= Naranja/verde L= Naranja/marrón	20 mm.

Los colores de cable del CAN tracción (tren de rodaje) son los que se emplearon para los diagramas y conexión en este trabajo, debido a que se enviaron mensajes a una velocidad de 500 kbps.

Apéndice F. Códigos de Nodos CAN.

Código encargado del mensaje del nodo 1 con identificador 470 “Luces”

```
#include <SPI.h> // permite comunicarse a la placa Shield con Arduino
#include <mcp_can.h> //permite comuicarse con el Bus

// Para tener un mayor orden se crean 3 funciones.
void MCP2515_ISR(); // Interrupción
void Enviar();  // Función para enviar mensajes
void Leer();   // Función para lectura de mensajes
// Declaración de variables
unsigned char flagRecv = 0; // Bandera de recepción del mensaje
unsigned char lon = 8;   // Longitud de la trama de datos(0-8)
unsigned char buf[8];       // Almacenamiento de Mensaje
const int SPI_CS_PIN = 10;  // Pin para la comunicación SPI
MCP_CAN CAN(SPI_CS_PIN);    // Establece el pin para la comunicación
void setup()
{ Serial.begin(115200);
while (CAN_OK != CAN.begin(CAN_500KBPS))// Velocidad del bus de 500kbps
    {Serial.println("Fallo de inicio de CAN BUS Shield");
        delay(100);}
     Serial.println(" Inicio CAN BUS Shield!");
    attachInterrupt(0, MCP2515_ISR, FALLING); }//Se establece interrupc
void MCP2515_ISR()  // la funcion de la ISR se limita a activar un flag
{ flagRecv = 1; }
void loop()
{ if(flagRecv) // si flagRecv=1
    { flagRecv = 0;
        while (CAN_MSGAVAIL == CAN.checkReceive())
        { Leer(); } }
   else
// Funcion para Enviar mensaje del Nodo
void Enviar(){ direccionales(); cuartos(); altas(); //Funciones para leer los sensores
   unsigned char datos[8]={direccionales(),21,cuartos(),0,18,2,2,altas()};
   // Envio de mensaje
    CAN.sendMsgBuf(0x470,0,8,datos);// Enviar datos
  // Impresión de datos
  Serial.println("-----------------------------------------------------");
   Serial.print("ID: ");
    unsigned long canId = CAN.getCanId();
    Serial.print(": 0x");
    Serial.println(canId, HEX);
    Serial.print("Mensaje enviado: ");
    for (int i = 0; i<8; i++){
      Serial.print(datos[i]); Serial.print("\t");}
    Serial.println();}
void Leer(){  CAN.readMsgBuf(&lon, buf);  // Lectura del mensaje
     Serial.print("ID: ");
     unsigned long canId = CAN.getCanId();
     Serial.print(": 0x");
```

```
      Serial.println(canId, HEX);
      Serial.print("Mensaje recibido: ");
      for(int i = 0; i<lon; i++)
      {Serial.print(buf[i]);Serial.print("\t");}
      Serial.println();}

// Funciones para la lectura de los sensores

int direccionales()
{const int analogpin= A1;
const int pin2 = A0;
const int intermitentes= 9;
int izquierda= analogRead(pin2);
int derecha=analogRead(analogpin);
 pinMode(9, INPUT);
int valor;
if (derecha==0)
{valor=2;}
  else if (izquierda==0)
  {valor=1;}
    else if (digitalRead(9) == HIGH)
    { valor=3;}
    else {valor=0;}
      return(valor);}

int cuartos() // cuartos(posición) y luces bajas
{const int pin3=A3;
const int pin4= A4;
int cuartos=analogRead(pin4);
int bajas=analogRead(pin3);
 int vali;
if ((cuartos==0) && (bajas==0))
{vali=64;}
else if (cuartos==0)
{vali=32;}
else
{vali=0;}
return (vali);}

int altas() //luces altas y antiniebla
{const int analogp= A2;
int potv=analogRead(analogp);
const int valor= map(potv,0,1025,0,4);
int valor2;
if (valor==2)
{ valor2=40;}
  else if (valor==3)
  {valor2=20;}
    else{valor2=0;}
  return (valor2);}
```

Código Nodo 2 identificador 280 “Revoluciones por minuto”

```
#include <SPI.h> // Libreria permite comunicarse a la placa Shield con Arduino
#include <mcp_can.h>  //Libreria que permite comuicarse con el Bus

// Para tener un mayor orden se crean 3 funciones.
void MCP2515_ISR(); // Interrupción
void Enviar();  // Función para enviar mensajes
void Leer();   // Función para lectura de mensajes
// Declaración de variables
unsigned char flagRecv = 0; // Bandera de recepción del mensaje
unsigned char lon = 8;      // Longitud de la trama de datos (0-8 numero de byte
unsigned char buf[8];      // Almacenamiento de Mensaje
const int SPI_CS_PIN = 10;  // Pin para la comunicación de la placa shield con A
MCP_CAN CAN(SPI_CS_PIN);    // Establece el pin para la comunicación serie
void setup()

{Serial.begin(115200);
while (CAN_OK != CAN.begin(CAN_500KBPS))  // Velocidad del bus de 500kbps
 { Serial.println("Fallo de inicio de CAN BUS Shield");
   delay(100); }
    Serial.println(" Inicio CAN BUS Shield!");
    attachInterrupt(0, MCP2515_ISR, FALLING);}

void MCP2515_ISR()  //Frecuentemente la funcion de la ISR se limitará a activar
{flagRecv = 1; } // Durante la rutina de interrupción se alza la bandera de rece
void loop()
{ if(flagRecv) // si flagRecv=1
    {   flagRecv = 0;     // Se resetea la bandera de recepción
        while (CAN_MSGAVAIL == CAN.checkReceive())    // se comprueba si hay un
        { Leer(); }  // Si se comprueba que hay un mensaje se llama a la función
        }
        else
          Enviar();  }  // De lo contrario se envia el mensaje del Nodo
// Funcion para Enviar mensaje del Nodo
void Enviar(){
 Revoluciones();
  unsigned char datos[8]={1,1,1,Revoluciones(),1,1,1,1};
  CAN.sendMsgBuf(0x280,0,8,datos);
  Serial.println("------------------------datos-----------------------------");
    Serial.print("Mensaje enviado: ");
    for (int i = 0; i<8; i++){
      Serial.print(datos[i]); Serial.print("\t"); }
    Serial.println();}
// Función para la lectura de Mensaje del Bus
void Leer(){
  CAN.readMsgBuf(&lon, buf);   // Lectura del mensaje
    // Impresión del mensaje recibido
   Serial.print("Mensaje recibido: ");
    for(int i = 0; i<lon; i++)
      {Serial.print(buf[i]);Serial.print("\t");}
      Serial.println();}
```

```
// Lectura del sensor
int Revoluciones() //luces altas y antiniebla
{const int analogp= A2;
int potv=analogRead(analogp);
const int valor= map(potv,0,1025,0,80);
  return (valor);}
```

Código para enviar un mensaje (este ejemplo tiene el ID: 280 correspondiente a "RPM")

```
// Codigo empleado para el envio de un mensaje en el
//apartado 5.2 implementación de un Nodo CAN
#include <SPI.h>
#include <mcp_can.h>

const int SPI_CS_PIN = 10;
MCP_CAN CAN(SPI_CS_PIN);
void setup()
{Serial.begin(115200);
    while (CAN_OK != CAN.begin(CAN_500KBPS))
    {Serial.println("CAN BUS Shield init fail");
        delay(100);}
    Serial.println("CAN BUS Shield init ok!");}
  void loop()
{//unsigned char datos[8]={0x49,0x0E,0xCC,0x0D,0x0E,0x00,0x1B,0x0E};
  unsigned char datos[8]={1,1,1,30,1,1,1,1};
    CAN.sendMsgBuf(0x280,0,8,datos);
  // Impresión de datos
  Serial.println("-----------------------------------------------");
    Serial.print("ID: ");
       unsigned long canId = CAN.getCanId();
       Serial.print(": 0x");
        Serial.println(canId, HEX);
    Serial.print("Mensaje enviado: ");
    for (int i = 0; i<8; i++){
      Serial.print(datos[i]); Serial.print("\t"); }
    Serial.println();  }
// END FILE
```

Código Para envío y recepción de mensajes sin emplear sensores

```
#include <SPI.h> //Permite comunicarse a la placa Shield con Arduino
#include <mcp_can.h>//Libreria que permite comuicarse con el Bus

// Para tener un mayor orden se crean 3 funciones.
void MCP2515_ISR(); // Interrupción
void Enviar();  // Función para enviar mensajes
void Leer();   // Función para lectura de mensajes
// Declaración de variables
unsigned char flagRecv = 0; // Bandera de recepción del mensaje
unsigned char lon = 8; // Longitud de la trama de datos (0-8 byte)
unsigned char buf[8];  // Almacenamiento de Mensaje

const int SPI_CS_PIN = 10;  // Pin para la comunicación SPI
MCP_CAN CAN(SPI_CS_PIN);    // Establece el pin para la comunicación
void setup()
{Serial.begin(115200);
while (CAN_OK != CAN.begin(CAN_500KBPS)) //Velocidad del bus 500kbps
 { Serial.println("Fallo de inicio de CAN BUS Shield");
   delay(100); }
    Serial.println(" Inicio CAN BUS Shield!");
    attachInterrupt(0, MCP2515_ISR, FALLING);}

void MCP2515_ISR()  //la funcion se limita a activar flag
{flagRecv = 1; }
void loop()
{ if(flagRecv) // si flagRecv=1
    {   flagRecv = 0; // Se resetea la bandera de recepción
        while (CAN_MSGAVAIL == CAN.checkReceive()) // se comprueba si hay un mensaje
        { Leer(); }  // Si se comprueba que hay un mensaje se llama a la función leer
        }
        else
          Enviar();  }  // De lo contrario se envia el mensaje del Nodo
// Funcion para Enviar mensaje del Nodo
void Enviar(){
  unsigned char datos[8]={Byte0,Byte1,Byte2,Byte3,Byte4,Byte5,Byte6,Byte7};
  CAN.sendMsgBuf(0xIDENTIFICADOR,0,8,datos);
  Serial.println("-----------------------datos----------------------------");
    Serial.print("Mensaje enviado: ");
    for (int i = 0; i<8; i++){
      Serial.print(datos[i]); Serial.print("\t"); }
    Serial.println();}


// Función para la lectura de Mensaje del Bus
void Leer(){
  CAN.readMsgBuf(&lon, buf);   // Lectura del mensaje
    // Impresión del mensaje recibido
   Serial.print("Mensaje recibido: ");
    for(int i = 0; i<lon; i++)
      {Serial.print(buf[i]);Serial.print("\t");}
      Serial.println();}
```

Apéndice F. Códigos de nodos CAN

Código para practica 3

```
#include <SPI.h> //permite comunicarse a la placa Shield con Arduino
#include <mcp_can.h>  //Libreria que permite comuicarse con el Bus

// Para tener un mayor orden se crean 3 funciones.
void MCP2515_ISR(); // Interrupción
void Enviar();  // Función para enviar mensajes
void Leer();   // Función para lectura de mensajes

// Declaración de variables
unsigned char flagRecv = 0; // Bandera de recepción
unsigned char lon = 8;  // Longitud de trama de datos(0-8 byte)
unsigned char buf[8];       // Almacenamiento de Mensaje
const int SPI_CS_PIN = 10;  // Pin para la comunicación SPI
MCP_CAN CAN(SPI_CS_PIN); // Establece el pin para la comunicación

void setup()
{Serial.begin(115200);
  while (CAN_OK != CAN.begin(CAN_500KBPS))  // Velocidad del bus de 500kbps
    {Serial.println("Fallo de inicio de CAN BUS Shield");
        delay(100);}
    Serial.println(" Inicio CAN BUS Shield!");
    attachInterrupt(0, MCP2515_ISR, FALLING); } // Se establece una interrup
void MCP2515_ISR()
{   flagRecv = 1; }
void loop()
{if(flagRecv) // si flagRecv=1
 { flagRecv = 0; // Se resetea la bandera de recepción

  while (CAN_MSGAVAIL == CAN.checkReceive())  // se comprueba si hay un mensaje
        { Leer();}
    }
    else
    Enviar(); }  // De lo contrario se envia el mensaje del Nodo

// Funcion para Enviar mensaje del Nodo
void Enviar(){

   Direccionales();
 unsigned char datos[8]={Direccionales(),21,32,0,18,0,0,60};
    // Envio de mensaje
     CAN.sendMsgBuf(0x470,0,8,datos);//Enviar datos
   // Impresión de datos
   Serial.println("---------------------------------------");
     Serial.print("ID: ");
     unsigned long canId = CAN.getCanId();
     Serial.print(": 0x");
     Serial.println(canId, HEX);
     Serial.print("Mensaje enviado: ");
     for (int i = 0; i<8; i++){
     Serial.print(datos[i]); Serial.print("\t");}
     Serial.println();}
```

Apéndice F. Códigos de nodos CAN

```
// Función para la lectura de Mensaje del Bus
void Leer(){ CAN.readMsgBuf(&lon, buf);    // Lectura del mensaje
      Serial.print("ID: ");
      unsigned long canId = CAN.getCanId();
      Serial.print(": 0x");
      Serial.println(canId, HEX);
      Serial.print("Mensaje recibido: ");
      for(int i = 0; i<lon; i++)
      {Serial.print(buf[i]);Serial.print("\t");}
      Serial.println();}

//Función para lectura de sensor
int Direccionales()

{const int Lderecha=A1;
  const int FocoDer=8;
  const int Lizquierda=A0;
  const int FocoIzq=7;
  const int intermitentes=9;
  int izquierda= analogRead(Lizquie
  int derecha=analogRead(Lderecha);
  pinMode(7,INPUT);
   pinMode(8,OUTPUT);
    pinMode(7,OUTPUT);
    int valor;
    if (derecha==0)
    {valor=2;
    digitalWrite(FocoDer,HIGH);}
    else if (izquierda==0)
    {valor=1;
    digitalWrite(FocoIzq,HIGH);}
  else if (digitalRead(9)==HIGH)
    {valor=3;
    digitalWrite(FocoIzq,HIGH);
    digitalWrite(FocoDer,HIGH);}
    else{valor=0;}
    return(valor);}
```

Apéndice G. Prácticas propuestas

Practica1

Lectura del CAN Bus del clúster de instrumentos

El clúster de instrumentos (CI) representa el nodo más importante de un automóvil moderno con respecto a la interfaz con el conductor. El clúster de instrumentos proporciona información esencial al conductor, como velocímetro, señales de advertencia, revoluciones por minutos, pero para mostrar esta información el CI debe recibir información de los demás nodos de la red encargados de estas tareas. El CI también envía mensajes referentes a su estado y esta práctica es para localizar los mensajes enviados por el clúster de instrumentos.

Para la efectuar esta práctica se necesitan los siguientes elementos:

- Clúster de instrumentos 6J0 920801X
- Osciloscopio con analizador de protocolo CAN

Procedimiento:

1.-Para visualizar los mensajes que el CI envía al Bus primero se debe conectar este a alimentación empleando los pines que se muestran en la tabla G.1, la conexión se muestra en la figura G.1.

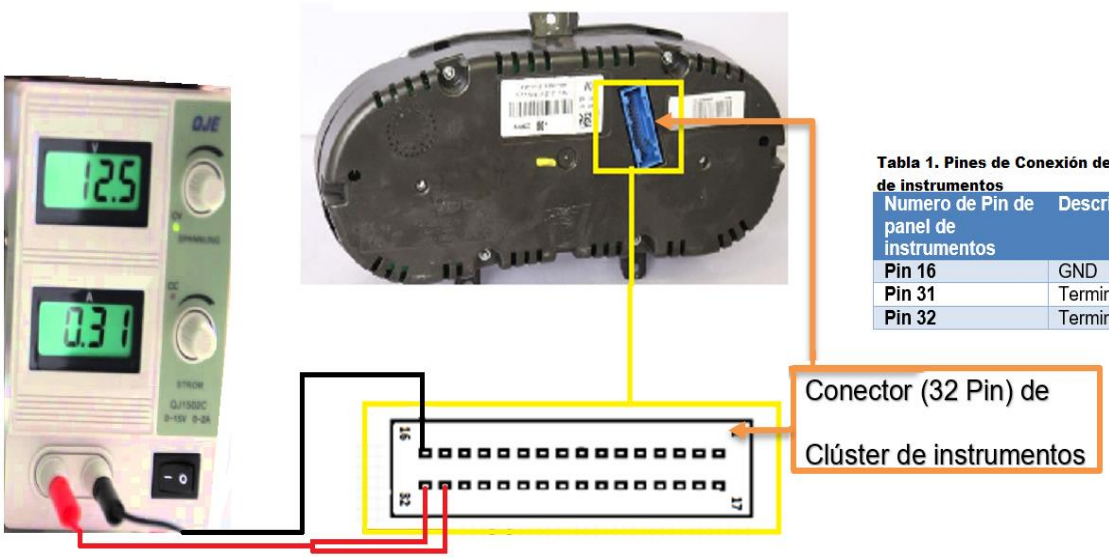


Tabla 1. Pines de Conexión del Clúster de instrumentos

Numero de Pin de panel de instrumentos	Descripción
Pin 16	GND
Pin 31	Terminal 15
Pin 32	Terminal 30

Figura G.1 Conexión del Clúster de instrumentos a alimentación.

2. Posteriormente se debe realizar la lectura de los mensajes del clúster de instrumentos empleando el Osciloscopio con analizador de protocolo CAN, los pines de los canales CAN_ H y CAN_L son el pin 28 y 29 respectivamente. La conexión del CI al Osciloscopio se muestra en la figura G.2

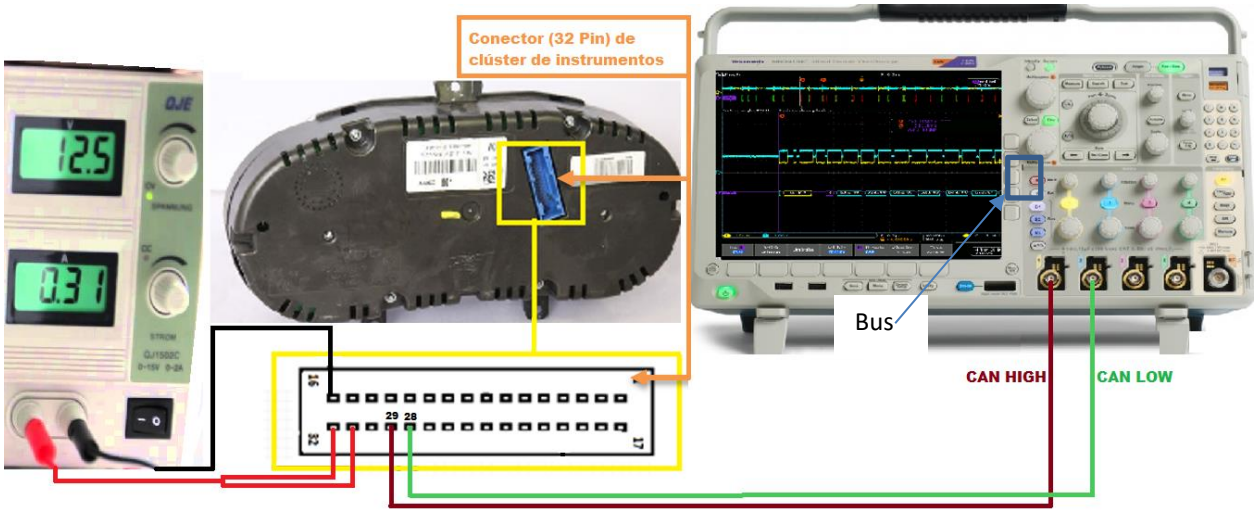


Figura G.2 Conexión de CAN HIGH y CAN LOW al Osciloscopio

3.- Después de realizar la conexión anterior se puede realizar la decodificación de las tramas de mensaje que el CI envía al Bus, se deben ajustar las escalas tanto verticales como horizontales y colocar la longitud de registro a 1M para tener mayor resolución. Posteriormente se selecciona el Bus B1 (ver figura G.2), al realizar esto se muestran las opciones de menú que se señalan en la figura G.3, donde se tiene que definir:

- El tipo de Bus: Bus CAN
- La entrada: El canal (Channel) a través del cual se real la lectura de la señal, en este mismo apartado también se selecciona la señal CAN_H o CAN_L que se va a decodificar.
- La velocidad: Se selecciona la velocidad a la cual se envían los datos al Bus, en este caso la velocidad es de 500 kbps, ya que esta es la velocidad a la que el clúster de instrumentos trabaja.

Figura G.3 Decodificación de una trama de datos CAN

4. Empleando la opción del menú “visualizar el Bus” que se muestra en la figura 3 se selecciona el sistema hexadecimal para mostrar en este formato los datos de la trama. Empleando las tramas de datos localizadas llenar la tabla G.2 con los valores de manera ascendente de los 9 identificadores que envía el clúster de instrumentos.

Nota: Se puede ayudar de la opción de menú “tabla de eventos” para visualizar las tramas de datos.

Tabla G.2 Tramas de datos enviadas por el Clúster de instrumentos

Número de Identificador	**Identificador de Mensaje en sistema Hexadecimal**	**Longitud de datos (DLC)**	**Frecuencia con la que entran al Bus.** $\boldsymbol{f = 1/t}$
1	320	8	50 Hz
2		8	
3		8	
4		8	
5		8	
6		2	
7		8	
8		7	
9		7	

5.- ¿Cuáles son los identificadores de mensaje que se introducen con más frecuencia en el Bus?

6.- Localizar los 7 campos de una de las tramas de datos CAN del Clúster de instrumentos empleando el osciloscopio con analizador CAN y llenar los espacios faltantes de la figura G.4. Debido a la corta longitud del apartado “campo de datos” este puede ser llenado con solo uno de los bytes de información del mensaje. Se deben de ignorar los bits de relleno al momento de llenar la figura 4. Recordar que se inserta un bit de relleno después de 5 bits del mismo valor y que en el campo fin de trama no se insertan bits de relleno.

Los siete campos que se deben localizar son:

1.-Inicio de trama

2.- Campo de arbitraje

3.- Campo de control

4.- Campo de datos

5.- Campo CRC

6.-Campo de reconocimiento

7.- Final de trama

Nota: Esta prueba se realiza aplicando zoom a la trama de datos para localizar los bits de cada campo.

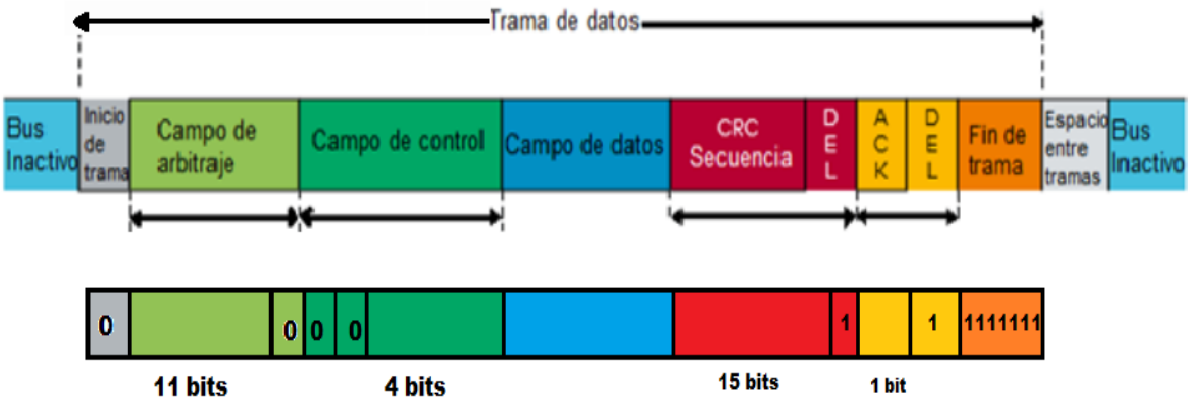


Figura G.4 Siete campos de la Trama de datos CAN

Responda las siguientes preguntas de acuerdo a la información que encontró anteriormente.

7. ¿Identifico bits de relleno?

8.- ¿Cuantos bits de relleno localizo?

9.- ¿Para qué sirven los bits de relleno?

10.- Mencione 3 de los métodos de detección de errores que detecto al llenar la figura 4.

Practica 2

Control de uno de los parámetros del clúster de instrumentos

El clúster de instrumentos (CI) proporciona al conductor información esencial sobre el estado del vehículo, cuantos más datos recibe el conductor, más completo es su conocimiento de la condición del vehículo. El CI muestra información del medidores y testigos como el velocímetro, señales de advertencia (ABS, líquido de frenos bajo, airbag) o testigos informativos como el encendido de una luz. Esta práctica es para controlar uno de los parámetros del clúster de instrumentos. Se debe elegir el identificador y valores de los Bytes que se deseen enviar. Puede escoger entre los identificadores de mensaje que se encuentran en las tablas del apéndice A “Tablas de Identificadores de Mensaje encontrados en este trabajo”

Para la efectuar esta práctica se necesitan los siguientes elementos:

- Clúster de instrumentos 6J0 920801X
- Fuente de Voltaje (12.5 v) o Batería
- Nodo CAN (Placa Arduino y Placa CAN Bus Shield)
- Cables para la Conexión
- 1 Resistencia de 120 Ohm

Procedimiento:

1.- Para controlar uno de los parámetros del clúster de instrumentos se simula un nodo de la Red CAN empleando la placa Arduino y CAN Bus Shield, se utilizan estas dos placas debido a que el nodo CAN está formado por un microcontrolador, controlador y transceptor CAN. La placa Arduino se emplea como microcontrolador y se realiza en su IDE un código para el Envío y recepción de mensajes. La placa CAN Bus Shield cuenta con el controlador y transceptor. Para la construcción de un nodo CAN se conecta la placa CAN Bus Shield a un Arduino UNO como se muestra en la figura G.5.

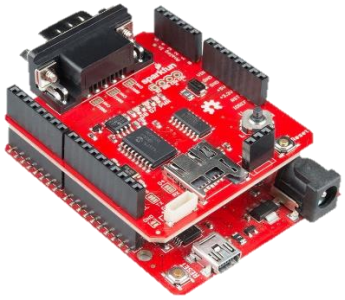


Figura G.5. Conexión de la placa Arduino y CAN Bus Shield

2.- Posteriormente para poder recibir y enviar mensajes al clúster de instrumentos y controlar los parámetros del clúster utilice el código “Código para envió y recepción de mensajes sin emplear

sensores" que se encuentra en el apartado de apéndice F. Las placas se comunican por protocolo SPI por lo cual se debe implementar la librería SPI.h y para realizar él envió y lectura de mensajes se utiliza la librería mcp_can.h. Después de realizar esto lo único que se tiene que hacer para él envió de un mensaje y encendido de los testigos es en el apartado de la función para enviar mensajes cambiar el valor de cada uno de los bytes para encender el testigo que se desee (Ver figura G.6.a). En el apartado de cada uno de los 8 bytes del recuadro naranja se debe colocar el valor que debe tomar cada byte para encender el testigo que se quiera mostrar en el clúster de instrumentos para esto se deben basar en la tabla de información del nodo de su elección que se muestra en el apéndice A "Tablas de Identificadores de Mensaje encontrados en este trabajo". En el apartado del recuadro verde se debe poner el identificador de mensaje que se desee enviar. En la figura G.6.b se muestra un ejemplo de que valores puede tomar cada byte para el encendido de luces del clúster de instrumentos. En la figura G.6.b también se describe a que testigo pertenece cada uno de los valores que se agregó.

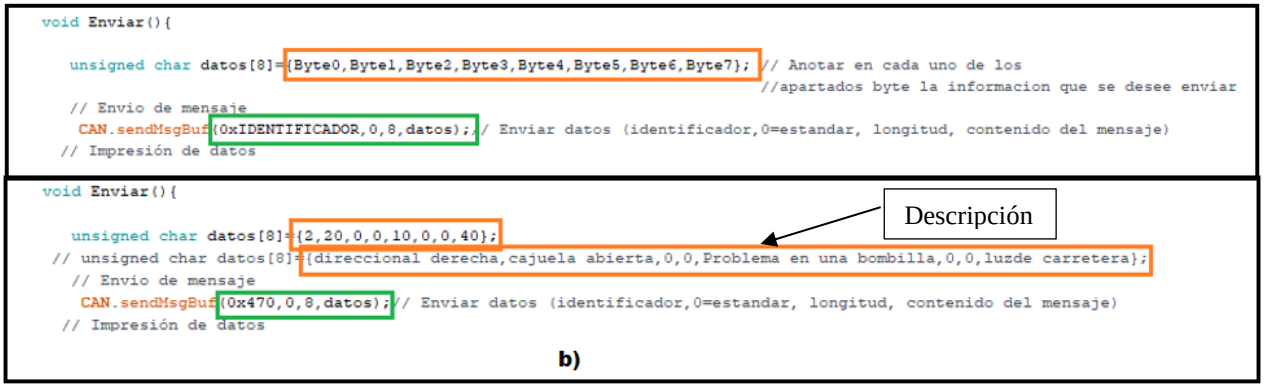


Figura G.6. Sección del código para envió de mensaje al clúster de instrumentos. a) apartados en donde se debe agregar información del mensaje a enviar b) ejemplo del envió del mensaje de luces (ID: 470) al clúster de instrumentos y valor de cada uno de sus identificadores.

3.- Después de cambiar el valor del identificador y los valores de los bytes del mensaje se carga el programa en la placa Arduino y se realiza la conexión del nodo CAN con el clúster de instrumentos como se muestra en la figura G.7. Los pines CAN_H y CAN_L del clúster de instrumentos se muestran en la tabla G.3.

Tabla G.3. Pines del Bus del clúster de instrumentos

Señal	Número del Pin del clúster de instrumentos
CAN H	Pin 28
CAN L	Pin 29

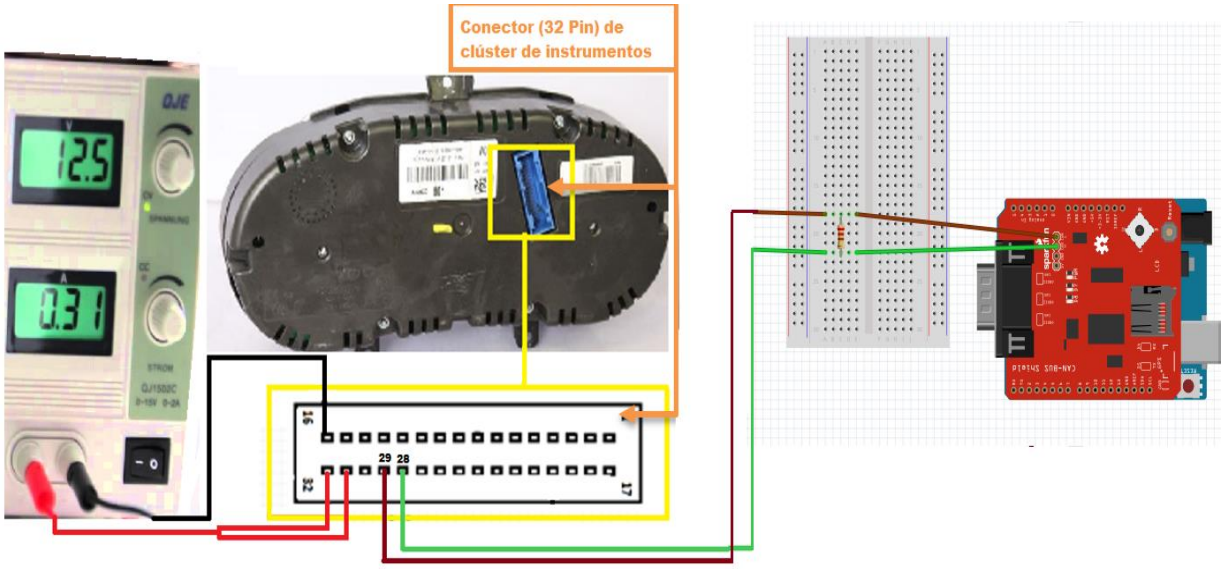


Figura G.7. Conexión del Clúster de instrumentos y Nodo CAN

4. Empleando el Monitor Serial de Arduino lea los mensajes recibidos y el mensaje enviado por el nodo CAN y llene la tabla G.4 con al menos 5 de los valores de los identificadores localizados y su longitud de datos (Número de bytes leídos en el Monitor Serial).

Tabla G.4. Identificadores de mensaje localizados.

Número de Identificador localizado	**Identificador de Mensaje en sistema Hexadecimal**	**Longitud de datos (DLC)**
1		
2		
3		
4		
5		

Practica 3

Implementación de un sensor y actuador al clúster de instrumentos.

Los nodos se encargan de diversas tareas del vehículo y para realizarlas emplean sensores y actuadores. El sensor capta la información, se la envía al microcontrolador quien la procesa, la compara con la información almacenada y dependiendo de esta realizar una acción con su actuador. De esta forma es como los sensores y actuadores de un nodo CAN ayudan en la realización de las tareas de las cuales están encargados (Ver figura G.8). Esta práctica es para implementar un sensor y un actuador al nodo CAN y controlar el estado (encendido, apagado, movimiento) de alguno de los testigos o medidor del clúster de instrumentos.

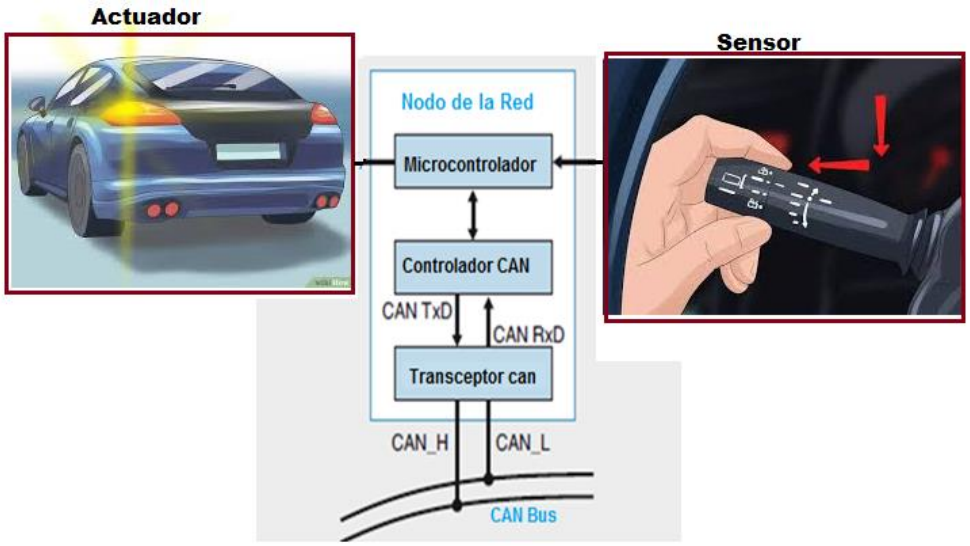


Figura G.8. Sensores y actuadores en el nodo CAN.

Para la efectuar esta práctica se necesitan los siguientes elementos:

- Clúster de instrumentos 6J0 920801X
- Fuente de Voltaje (12.5 v) o Batería
- Nodo CAN (Placa Arduino y Placa CAN Bus Shield)
- Cables para la Conexión
- 1 Resistencia de 120 Ohm
- Sensor (Puede emplear potenciómetros, push-button, sensor de distancia, palanca de luces direccionales, etc.)
- Actuador (Led, Motor 5v, buzzer, etc.)

1.- Para controlar un parámetro del clúster de instrumentos empleando la información de un sensor, primero se debe implementar un nodo CAN para el este se emplea la placa Arduino y CAN Bus Shield, se utilizan estas dos placas debido a que el nodo CAN está formado por un microcontrolador, controlador y transceptor CAN. La placa Arduino se emplea como microcontrolador y se realiza el código para envió y recepción de mensajes en la IDE de Arduino, ahí mismo se programa la acción que realizara el actuador ante la señal del sensor. La placa Shield cuenta con el controlador y transceptor. Para la construcción de un nodo CAN se conecta la placa CAN Bus Shield a un Arduino UNO como se muestra en la figura G.9.

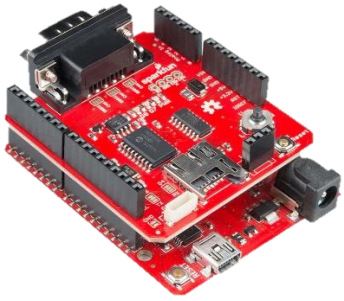


Figura G.9. Conexión de la placa Arduino y CAN Bus Shield

2.- Posteriormente para recibir y enviar mensajes al clúster de instrumentos y controlar el encendido de un testigo o movimiento de un medidor utilice el código "código para practica 3" que aparece en el apartado de apéndice F, este código cuenta con 3 funciones: enviar, leer y una interrupción para la detección de mensajes en el Bus, con este código usted podrá realizar el encendido de la luz derecha, izquierda o ambas. El encendido de los testigos del clúster sucede ya que en el código se programa que la información que el sensor detecte se envié en el mensaje de datos por esta razón se llama a la función sensor en el apartado leer mensaje del código (figura G.10), la acción que el actuador realizara se programa en "función para leer el sensor". En la tabla G.5 se muestran los pines de conexión de clúster de instrumentos y en la figura G.11 puede ver la conexión del clúster, el nodo, el sensor y actuador.

```
// Funcion para Enviar mensaje del Nodo
void Enviar(){
direccionales();  Se manda a llamar a la funcion para lectura del sensor direccionales
unsigned char datos[8]={direccionales(),21,32,0,18,2,2,60};
                        se envia la información del sensor en el byte 0 del mensaje.
// Envio de mensaje
 CAN.sendMsgBuf(0x470,0,8,datos);// Enviar datos (identificador,0=estandar, longitud, contenido del mensaje)
```

Figura G.10. Envió de mensaje con lectura del sensor

Tabla 1. Pines de conexión del clúster de instrumentos.

Canal	Pin del clúster de instrumentos
CAN_H	28
CAN_L	29

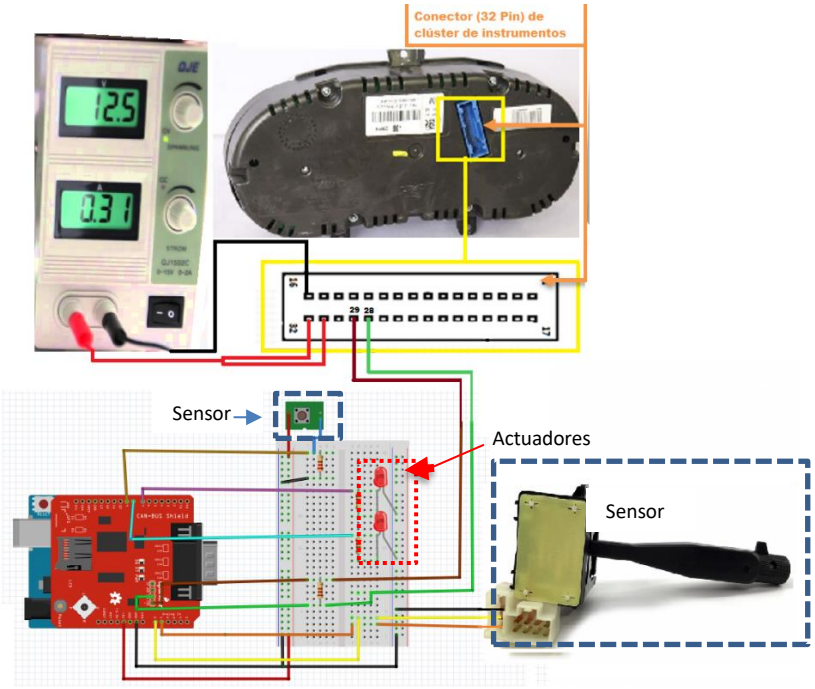


Figura. G.11 Conexión del clúster de instrumentos, nodo de luces, sensor y actuadores.

5.- Es momento de que Implemente más sensores al nodo. El mensaje de luces puede enviar información en 5 de sus 8 bytes por lo que puede agregar más sensores al código y enviar más mensaje, agregando una o más funciones para la lectura del sensor. Puede encender luces, disminuir su intensidad (simulando luces bajas o cuarto), etc. También puede cambiar el mensaje que desee enviar, lo único que tiene que hacer es en la función Enviar colocar el valor del identificador de mensaje que elija y de acuerdo con el ID y la tabla del identificador (Localizada en el apéndice A) colocar la lectura del sensor en uno de los 8 bytes de la trama de datos. En el apéndice F se encuentran dos códigos en los cuales se puede basar, uno es para enviar un mensaje de luces (ID: 470) y otro para Revoluciones por minuto (ID: 280).

4) Manual para acceder al bus Monitor y al Programa de LabVIEW

Nota: Para acceder al Bus Monitor y Programa de LabVIEW se debe conectar el sistema PXI (con software instalado) al CPU mediante un cable Ethernet

4.1 Acceso al Bus Monitor

1. Abrir el software NI-XNET y seleccionar de acuerdo a la numeración las opciones que se muestran en la figura G.12, de esta manera podrá acceder al Bus. El paso 5 se selecciona de acuerdo al puerto del Módulo PXI-8513 donde esté conectado el cable DB9.

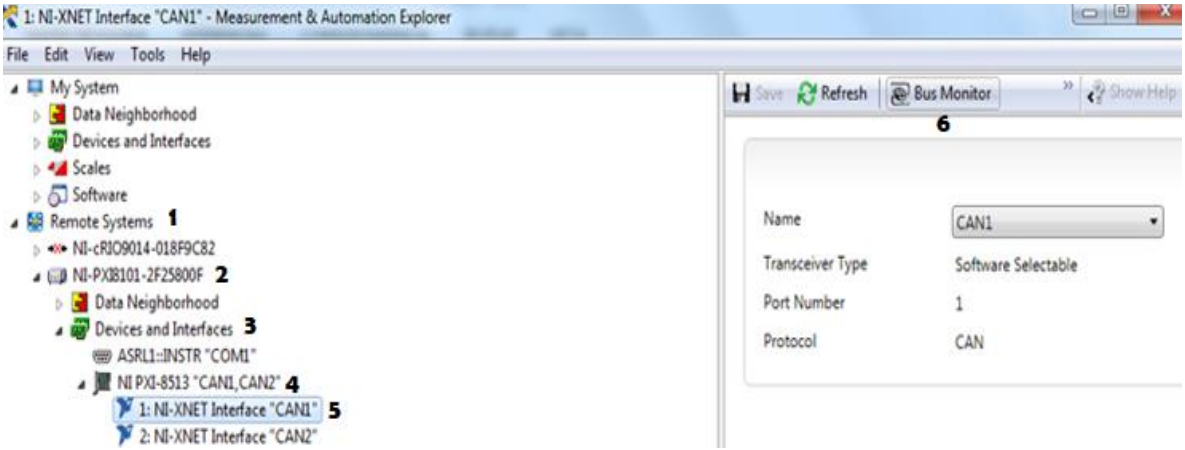


Figura G.12. Pasos para acceder al Bus Monitor.

2. Configurar la velocidad a la cual se van a leer y enviar los mensajes, seguir los pasó que se muestran en la figura G.13. En Baud Rate seleccione la velocidad.

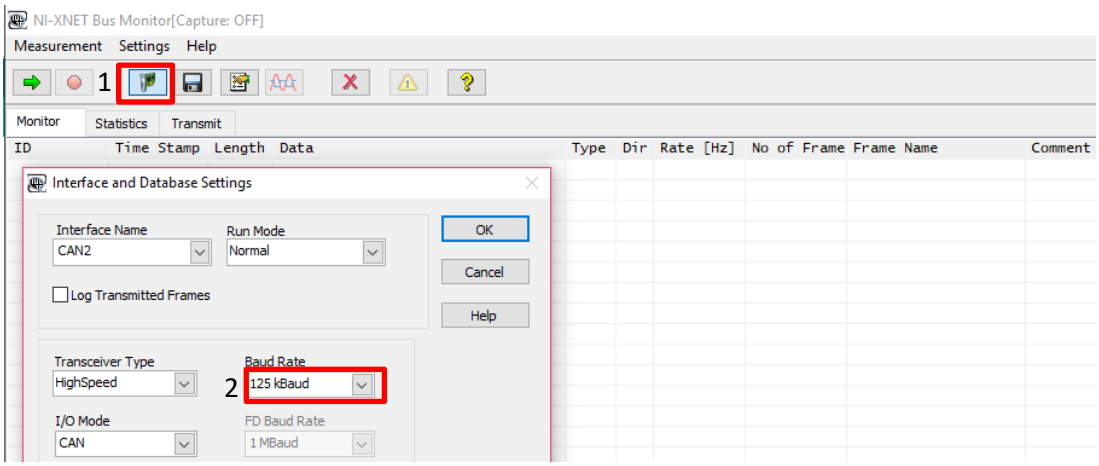


Figura G.13. Configuración del Bus

3.- Después de configurar el Bus Monitor se pueden utilizar los tres apartados de menú Monitor, Estadística y Enviar. Monitor ayuda a ver los mensajes que se están transmitiendo en el Bus. Estadística muestra la carga del Bus, tipo de mensaje, mensajes por segundo y Contadores de error. Transmitir sirve para enviar mensajes al Bus. En la figura G.14 se muestra este último y sus

apartados. En el recuadro naranja está el botón de inicio y stop, en el rectángulo verde las 3 opciones de menú del Bus Monitor. En rectángulos azules se muestran los apartados que se pueden configurar para enviar una trama al Bus. El cuadro rojo es el botón para comenzar a transmitir mensajes al Bus.

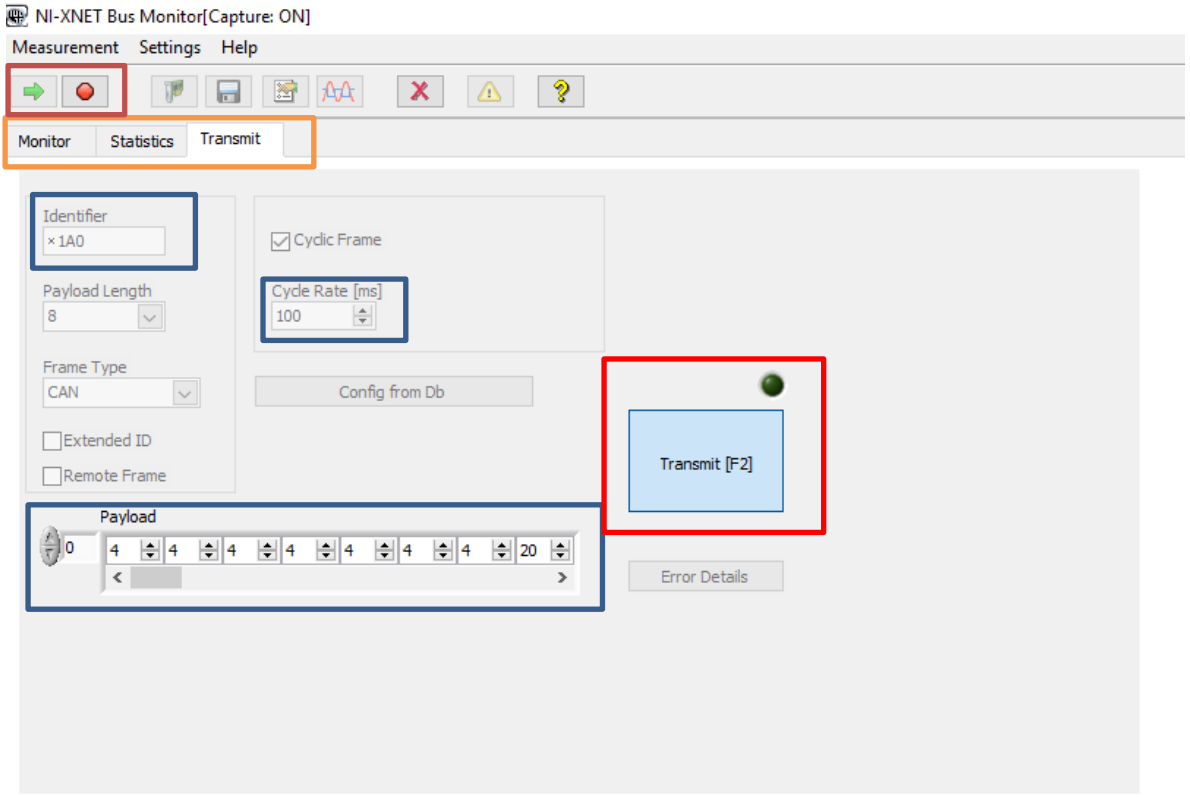


Figura G.14. Elementos de la opción transmitir del Bus Monitor

En la figura G.15 se pueden ver las opciones de menú Monitor y estadística.

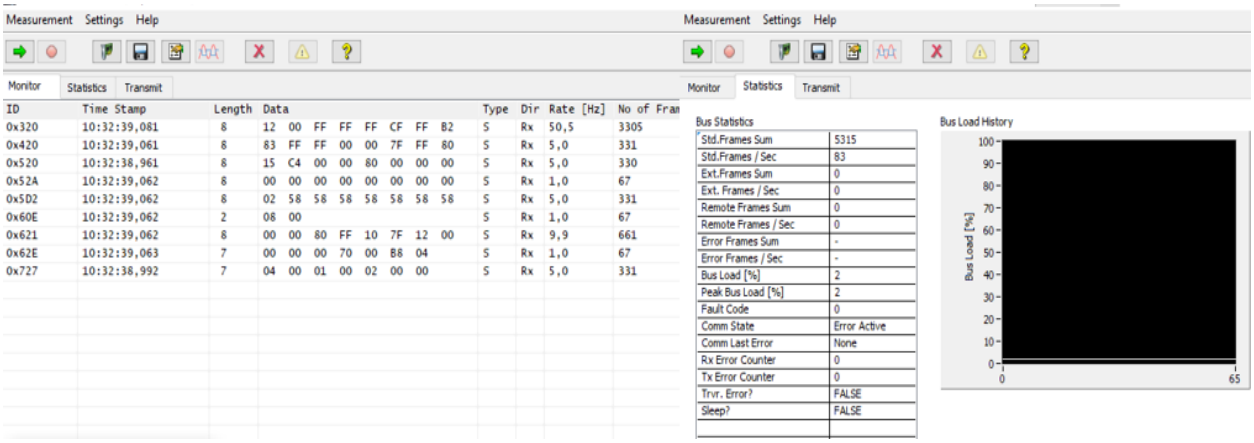


Figura G.15. Opciones Monitor y Estadísticas del Bus Monitor

4.2 Programa de LabVIEW para lectura de Mensajes en el Bus

1.- Abrir el proyecto Tramas, clic derecho sobre el controlador NI-PXI8101 y seleccionar conectar. Cuando se encuentre conectado debe de cambiar la intensidad del punto verde como se muestra en la figura G.16. Después de conectar abrir el programa ladyprueba.vi que se remarca en el rectángulo rojo.

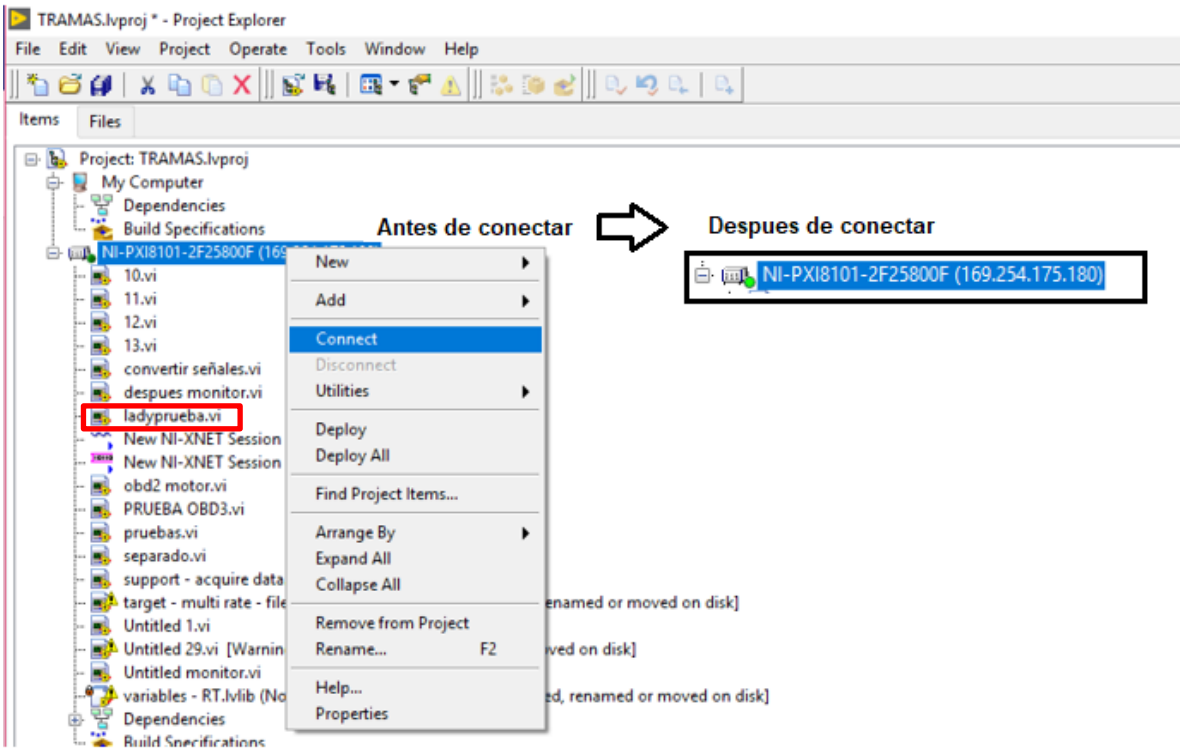


Figura G.16. Conexión del controlador e inicio del programa para lectura de tramas CAN.

2.- En la figura G.17 se muestra la ventana del panel frontal del programa de LabVIEW (VI) en esta interfaz de usuario debe configurar la velocidad de lectura de datos, el número de tramas a mostrar y el puerto (rectángulos rojos) a través del cual se realiza la lectura. Finalmente corra el programa y visualice los tiempos, datos, identificador y tipo de trama de datos (rectángulo azul).

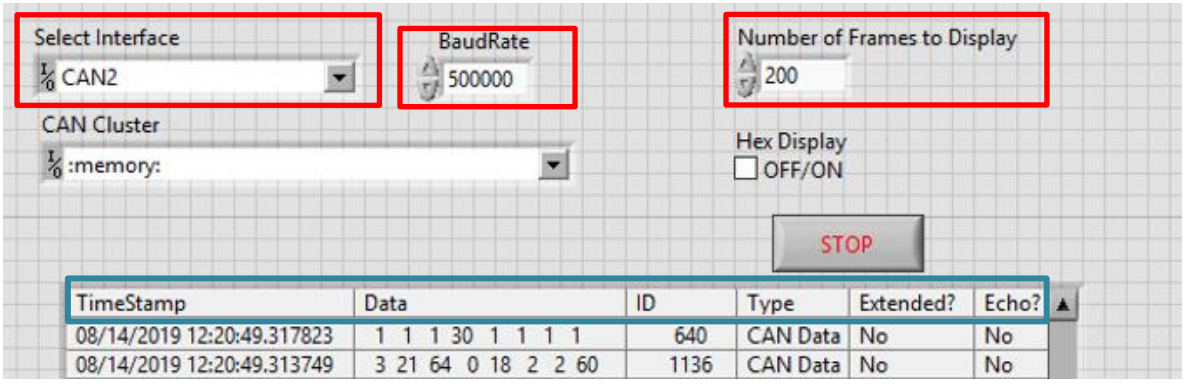


Figura G.17. Configuración de parámetros para lectura de Mensajes en el Bus.